UNIVERSIDADE DE LISBOA
INSTITUTO SUPERIOR TÉCNICO

Distributed Ledger Interoperability Security

Rafael André Pestana Belchior

Supervisor: Doctor André Ferreira Ferrão Couto e Vasconcelos
Co-Supervisor: Doctor Miguel Nuno Dias Alves Pupo Correia

Thesis approved in public session to obtain the PhD Degree in Computer Science and Engineering

Jury final classification: Pass with Distinction and Honour

2024

UNIVERSIDADE DE LISBOA
INSTITUTO SUPERIOR TÉCNICO

Distributed Ledger Interoperability Security

Rafael André Pestana Belchior

Supervisor: Doctor André Ferreira Ferrão Couto e Vasconcelos
Co-Supervisor: Doctor Miguel Nuno Dias Alves Pupo Correia

Thesis approved in public session to obtain the PhD Degree in Computer Science and Engineering

Jury final classification: Pass with Distinction and Honour

Jury

Chairperson: Doctor Maria Inês Camarate de Campos Lynce de Faria, Instituto Superior Técnico, Universidade de Lisboa

Members of the committee:

Doctor William John Knottenbelt,
Department of Computing, Faculty of Engineering, Imperial College London, United Kingdom
Doctor Carlos José Corredoura Serrão,
Escola de Tecnologias e Arquitetura, ISCTE-Instituto Universitário de Lisboa
Doctor João Fernando Peixoto Ferreira,
Instituto Superior Técnico, Universidade de Lisboa
Doutor André Ferreira Ferrão Couto e Vasconcelos,
Instituto Superior Técnico, Universidade de Lisboa

2024

Abstract

Blockchain interoperability conflates the need for *blockchains* to communicate with third-party systems and other distributed ledgers via *interoperability mechanisms* (IMs). Blockchains increasingly rely on exchanging data and value across network boundaries in a more mature and interconnected set of ecosystems. However, interoperability comes with challenges. Through a systematic literature review, we identified the main challenges impacting the field: 1) the lack of a common conceptual model for blockchain interoperability, and thus a lack of evaluation frameworks for IMs; 2) the absence of organizational interoperability in most IMs; and 3) ineffective methods for blockchain interoperability security.

Our contributions can be aggregated into three main groups. The first group of contributions delivers a conceptual model, evaluation framework, and decision models that allow researchers to reason about blockchain interoperability, compare solutions, and decide on the best IM given their requirements. The second group of contributions delivers a new paradigm for blockchain interoperability called the blockchain gateway paradigm, with our system called `Hermes`. This paradigm considers privacy and accountability in the interoperability processes across centralized and decentralized organizations. This is a suitable model for the enterprise and is a middle-ground between permissioned and permissionless infrastructure.

The last group of contributions addresses the prominent security challenge in the blockchain industry. We explore the potential *SNARKs* as a technology to verify computation succinctly, such that it reduces the attack surface for hackers. We propose a framework called `Harmonia` to build interoperable decentralized applications on top of this technology. As no technology is safe from malicious parties, we propose a new monitoring method for interoperable decentralized applications, `Hephaestus`. By generating a model of the interoperable application, we can analyze and monitor relevant states, allowing us to respond to attacks more effectively.

Keywords: blockchain, interoperability, cross-chain transaction, digital assets, decentralized protocol

Resumo

A interoperabilidade de blockchains aborda a necessidade de as mesmas comunicarem com sistemas centralizados e outros sistemas descentralizados, via *mecanismos de interoperabilidade* (MIs). No entanto, a interoperabilidade traz desafios. Através de uma revisão sistemática da literatura, identificamos os principais desafios que impactam a área: 1) a falta de um modelo conceptual para a interoperabilidade de blockchain e, portanto, a ausência de estruturas de avaliação para MIs de blockchain; 2) a ausência de interoperabilidade organizacional na maioria dos MIs; e 3) métodos ineficazes para a segurança da interoperabilidade de blockchain.

As nossas contribuições estão agrupadas em três principais grupos. O primeiro grupo de contribuições fornece um modelo conceptual, uma estrutura de avaliação e modelos de decisão que permitem aos investigadores raciocinar sobre a interoperabilidade de blockchain, comparar MIs e decidir sobre o melhor MI com base nos seus requisitos. O segundo grupo de contribuições introduz um novo paradigma para a interoperabilidade de blockchain chamado paradigma de *blockchain gateway*, instanciado com o nosso sistema *Hermes*. Este paradigma considera a privacidade e a responsabilização nos processos de interoperabilidade entre organizações centralizadas e descentralizadas. Este é um modelo adequado para empresas e representa um meio-termo entre blockchains que requerem e não requerem autorização para participar em redes descentralizadas.

O último grupo de contribuições aborda o desafio de segurança proeminente na indústria de blockchain. Exploramos o potencial das *SNARKs* como uma tecnologia para verificar computações de forma sucinta, reduzindo assim a superfície de ataque para potenciais atacantes. Propomos uma framework chamada *Harmonia* para construir aplicações descentralizados interoperáveis com base nessa tecnologia. Como nenhuma tecnologia está imune a ataques, propomos um novo método de monitorização para aplicações descentralizadas interoperáveis, *Hepheastus*. Ao gerar um modelo da aplicação interoperável, podemos analisar e monitorizar o seu estado a qualquer momento, permitindo-nos responder a ataques de forma mais eficaz.

Palavras-chave: tecnologia de cadeia de blocos, interoperabilidade, transferência entre blockchains, ativos digitais, protocolo descentralizado

"Saruman believes that it is only great power that can hold evil in check. But that is not what I have found. I have found it is the small things, everyday deeds of ordinary folk, that keeps the darkness at bay. Simple acts of kindness and love."

Gandalf in The Hobbit: An Unexpected Journey (2012)

Acknowledgments

The present document is the result of several years of work realized in unique circumstances, which I emphasize a nascent, vibrant, fast-paced, and sometimes overwhelming industry and the result of the collaboration, support, and mentorship of many people who humbly have taught me.

First and foremost, I express my deepest gratitude to my family for their unconditional support, allowing me the conditions for realizing this PhD thesis. A special thanks to my father, my mother, my siblings, and my lovely partner for all the love and all the direct and indirect help - if you see an amazingly pretty figure throughout my papers, it is possible it had her artistic hand on it. Friendship is one of my life pillars. I owe a big thanks to all my friends for life-changing experiences, being in Southeast Asia, in unlikely places in Europe, watching the shooting stars in Acadia, or grabbing a mini in a rather small town in Alentejo.

To the people responsible for this PhD thesis: my advisors, André and Miguel. Their integrity, trust, reliability, and safe space to debate ideas and discuss helped cement my research pillars. The freedom to manage my own research, attention, availability, and respect allowed me to do my work in the best possible conditions. I am very grateful our paths crossed. In fact, I could write much more here, but I do not want the acknowledgments to have ten pages. I am also thankful for the opportunity to supervise students, which became an important learning experience that I will take for my personal and professional life. A big thanks to my PhD buddy, André Augusto, for the friendship, long discussions on ideas, and coding sessions hacking out gateways.

Throughout the years, I have had the opportunity to collaborate with institutions and organizations close to the industry and promote open-source and blockchain education. My big thank you to the Hyperledger Foundation for supporting not one or two but eight grant proposals on diverse blockchain projects. A special thank you to Blockdaemon and its leadership for allowing me a solid, beneficial collaboration with the industry, where we developed new theories, principles, algorithms, software, and proofs of concept supporting much of the work presented in this thesis. A heartfelt acknowledgment to Jonas Pfannschmidt for providing a solid foundation to discuss ideas and make things happen. I thank people I consider mentors: André Vasconcelos, Miguel Correia, Thomas Hardjono, Rui Cruz, Rui Henriques, Peter Somgyvari, and Jonas Pfannschmidt. I also thank the SATP working group folks at the IETF for the learning and collaboration on blockchain gateways. I would also like to thank my co-authors and collaborators for the great teamwork that gave birth to academic papers, whitepapers, software, technical reports, blog posts, and more software. A special mention goes to Rui Henriques, Tomás Alves, and, of course, my supervisors, who greatly helped me improve this document.

I thank Fundação para a Ciência e Tecnologia (FCT) and the Portuguese taxpayers for the financial sponsorship of this work through a scholarship with reference UIDB/50021/2020 (INESC-ID) and 2020.06837.BD. This scholarship was an essential support that allowed the seamless realization of this thesis. Hopefully, the contributions of this thesis will help progress the state of the art of distributed ledgers in Portugal and further. I extend my gratitude to Fulbright Portugal for the opportunity to realize part of the research in this thesis at the Massachusetts Institute of Technology (MIT), in the

United States. This would not have been possible without the support of Prof. Dr. Alex Pentland and Dr. Thomas Hardjono. I thank my host institutions, INESC-ID and Instituto Superior Técnico (IST), who provided the knowledge, infrastructure, and conditions to realize the present work acquired in recent years. Several projects supported various items of my research throughout the years with funding from the European Commission: European Commission program H2020 under the grant agreement 822404 (project QualiChain), project BIG ERA Chair (grant agreement 952226), project DE4A (grant agreement 870635), and "Recovery and Resilience Plan - Component 5: Agendas Mobilizadoras para a Inovação Empresarial," through project nr.51 "BLOCKCHAIN.PT - Agenda Descentralizar Portugal com Blockchain," included in the NextGenerationEU funding program.

I thank the anonymous reviewers of the papers we submitted and more than 40 people who provided feedback on early versions of our papers. I show my gratitude to the jury members for their time and dedication to helping improve the quality of this document. I particularly thank my thesis committee members, Prof. Dr. Carlos Serrão and Prof. Dr. João Ferreira, for early feedback on this thesis.

To all of you and to the dear reader, I dedicate this thesis.

Rafael Belchior,
26 January 2024, Portugal
0xe260afcd14b4e70246c1f92bacabf778a897d4f6dec10b0c3ef8bdde9d25a509

Research Output

This dissertation led to the production of multiple publications in peer-reviewed journals and international conferences. This section presents the publications and a list of supervised MSc students. An updated list of publications can be found in my Google Scholar profile[1] or my webpage[2].

Main Publications

The work developed in this thesis can be partially found in the following publications. A high-level description of each paper is present on the author's webpage, at the academic page[3].

P1 R. Belchior, L. Riley, T. Hardjono, A. Vasconcelos, and M. Correia, **"Do You Need a Distributed Ledger Technology Interoperability Solution**?,**"** Distributed Ledger Technologies (ACM DLT), Sep. 2022.

P2 R. Belchior, L. Torres, J. Pfannschmid, A. Vasconcelos, and M. Correia, **"BUNGEE: Dependable Blockchain Views for Interoperability."** Distributed Ledger Technologies (ACM DLT), January. 2024.

P3 R. Belchior, A. Vasconcelos, M. Correia, and T. Hardjono, **"HERMES: Fault-Tolerant Middleware for Blockchain Interoperability**,**"** Future Generation Computer Systems, Mar. 2021.

P4 R. Belchior, D. Dimov, Z. Karadjov, J. Pfannschmidt, A. Vasconcelos, M. Correia, **Harmonia: Securing Cross-Chain Applications Using Zero-Knowledge Proofs**. Submitted to IEEE Transactions on Dependable and Secure Computing.

P5 R. Belchior, P. Somogyvari, J. Pfannschmid, A. Vasconcelos, and M. Correia, **"Hephaestus: Modelling, Analysis, and Performance Evaluation of Cross-Chain Transactions."**´, IEEE Transactions on Reliability, Dec. 2023.

[1] https://scholar.google.com/citations?user=B0O3stQAAAAJ&hl=en
[2] https://rafaelapb.github.io
[3] https://rafaelapb.github.io/academic

Other Publications

Other publications resulting from this doctoral research program not included in the thesis:

P6 R. Belchior, A. Vasconcelos, S. Guerreiro, and M. Correia, **"A Survey on Blockchain Interoperability: Past, Present, and Future Trends"**, ACM Computing Surveys, vol. 54, no. 8, pp. 1–41, May 2021.

P7 R. Belchior, A. Vasconcelos, and M. Correia, **"Towards Secure, Decentralized, and Automatic Audits with Blockchain"**, in European Conference on Information Systems, 2020.

P8 S. Rouhani, R. Belchior, R. S. Cruz, and R. Deters, **"Distributed attribute-based access control system using permissioned blockchain"**, World Wide Web, vol. 24, no. 5, pp. 1617–1644, 2021.

P9 R. Belchior, B. Putz, G. Pernul, M. Correia, A. Vasconcelos, and S. Guerreiro, **"SSIBAC: Self-Sovereign Identity Based Access Control"**, in The 3rd International Workshop on Blockchain Systems and Applications, IEEE, 2020.

P10 R. Belchior, S. Guerreiro, A. Vasconcelos, and M. Correia, **"A Survey on Business Process View Integration"**, Business Process Management Journal, Nov. 2021

P11 R. Belchior, M. Correia, A. Augusto, and T. Hardjono, **"SATP Gateway Crash Recovery Mechanism"**, Internet Engineering Task Force, Internet Draft draft-belchior-satp-gateway-recovery-00, Jul. 2023. Accessed: Oct. 08, 2023. [Online]. Available: https://datatracker.ietf.org/doc/draft-belchior-satp-gateway-recovery

P12 M. Hargreaves, T. Hardjono, and R. Belchior, **"Secure Asset Transfer Protocol (SATP)"**, Internet Engineering Task Force, Internet Draft draft-ietf-satp-core-02, Jul. 2023. Accessed: Oct. 08, 2023. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-satp-core

P13 S. Ghaemi, S. Rouhani, R. Belchior, R. S. Cruz, H. Khazaei, and P. Musilek, **"A Pub-Sub Architecture to Promote Blockchain Interoperability,"** Future Generation Computer Systems, Jan. 2021, Accessed: Feb. 28, 2021. [Online]. Available: http://arxiv.org/abs/2101.12331

P14 R. Belchior, A. Vasconcelos, M. Correia, and T. Hardjono, **"Enabling Cross-Jurisdiction Digital Asset Transfer,"** in IEEE International Conference on Services Computing, IEEE, 2021.

P15 C. Pedreira, R. Belchior, M. Matos, and A. Vasconcelos, **"Securing Cross-Chain Asset Transfers on Permissioned Blockchains."** BlockTEE. June 2022.

P16 R. Belchior, S. Scuri, I. Mihaiu, N. Nunes, and T. Hardjono, **"Towards a Common Standard Framework for Blockchain Interoperability - A Position Paper."** TechRxiv, 2023. [Online]. Available: https://www.techrxiv.org/articles/preprint/A_Framework_to_Evaluate_Blockchain_Interoperability_Solutions/17093039. Submitted to Blockchain: Research and Applications.

P17 A. Augusto, R. Belchior, T. Hardjono, A. Vasconcelos, and M. Correia, **"Multi-Party Cross-Chain Asset Transfers."** TechRxiv, Jun. 27, 2023. [Online]. Available: https://www.techrxiv.org/articles/preprint/Resilient_Gateway-Based_N-N_Cross-Chain_Asset_Transfers/20016815

P18 A. Augusto, R. Belchior, A. Vasconcelos, I. Kocsis, G. László, and M. Correia, **"CBDC bridging between Hyperledger Fabric and permissioned EVM-based blockchains."** TechRxiv, Mar. 27, 2023. [Online]. Available: https://www.techrxiv.org/articles/preprint/CBDC_bridging_between_Hyperledger_Fabric_and_permissioned_EVM-based_blockchains/21809430

P19 R. Belchior, J. Süßenguth, Q. Feng, T. Hardjono, A. Vasconcelos, and M. Correia, **"A Brief History of Blockchain Interoperability."** Sep. 2024. Communications of the ACM (CACM)

P20 H. Montgomery et al., **"Hyperledger Cactus Whitepaper,"** Hyperledger Foundation, 2020. [Online]. Available: https://github.com/hyperledger/cactus/blob/master/docs/whitepaper/whitepaper.md

P21 A. Augusto, R. Belchior, M. Correia, A. Vasconcelos, L. Zhang, and T. Hardjono, **"SoK: Security and Privacy of Blockchain Interoperability,"** 2024, IEEE Symposium on Security and Privacy

P22 S. Subramanian, A. Augusto, and R. Belchior, **"Benchmarking Bridge Aggregators,"** Preprints, preprint, Jan. 2024. doi: 10.36227/techrxiv.170492248.87354060/v1. Accessed: Jan. 21, 2024. [Online]. Available: https://www.techrxiv.org/users/687326/articles/697988-benchmarking-bridge-aggregators

Co-Supervised MSc Theses

During my Ph.D. program, I mentored several M.Sc. students. These students are listed below along with the title of their dissertation and their co-supervisors.

- Luís Abrunhosa. **Migrating Smart Contracts Across Heterogeneous Blockchains** - co-supervised with André Vasconcelos and João Ferreira (2020-2021).
- Catarina Pedreira. **Trustable Blockchain Interoperability: Incentivizing Public Escrow Parties** - co-supervised with André Vasconcelos and Miguel Matos (2020-2021).
- André Augusto. **Ichain: Gateway-based enterprise Blockchain interoperability** - co-supervised with André Vasconcelos and Thomas Hardjono (2021-2022).
- Afonso Marques. **AuditChain: Blockchain views applied to auditing** - co-supervised with André Vasconcelos and José Miranda (2021-2022).
- Mónica Gomez. **Blockchain Interoperability: A technical approach to interoperability for Blockchain-based systems** - co-supervised with André Vasconcelos and Miguel Correia (2022 - ongoing).

- Sebastião Mayor. **An Enterprise Architecture approach to Semantic interoperability for Blockchain-based systems** - co-supervised with André Vasconcelos and Miguel Correia (2022 - ongoing).
- Daniel Henriques. **VChain: Visualization and Analysis of Cross-Chain Transactions** - co-supervised with André Vasconcelos (2022 - ongoing).
- Eduardo Vasques. **Algorithms for Heterogeneous Blockchain Interoperability** - co-supervised with André Vasconcelos and Miguel Correia (2023 - ongoing).
- Carlos Amaro. **Implementing Semantic Blockchain Interoperability Across Heterogeneous Infrastructure** - co-supervised with André Vasconcelos and Miguel Correia (2023 - ongoing).
- Bruno Mateus. **An Incident Response Framework Applied to Blockchain Interoperability** - co-supervised with André Vasconcelos and Miguel Correia (2023 - ongoing).

Foreword

My supervisors kindly told me that it is not common for a doctoral student from Técnico Lisboa to write a foreword. However, I feel the need to share a bit of my path in the blockchain world, a highly competitive, quick-paced, innovative industry.

My academic path, which coincides with the beginning of my blockchain journey, started in 2017 and led to the conclusion of this thesis, in 2024. This doctoral program taught me some things, which I underline two. First, research is a truly collaborative experience, where each participant constructs and deconstructs ideas, argues, and reaches a consensus; second, writing an up-to-date thesis is not easy, as research quickly becomes outdated - as new developments are announced on a monthly basis. This is because the industry is a moving target, especially in the hot topic of blockchain interoperability and scalability. Thus, we had to do research while actively following developments in the industry. The Design Science Research Methodology helped us achieve that but bear in mind that by the time you read this, the created knowledge might already be obsolete. Nonetheless, we believe this document provides a solid theoretical basis that captures much of the complexity of blockchain interoperability theory and practice. Take this thesis as a snapshot of the area as of January 2024.

When I started working on interoperability, very little work on blockchain interoperability existed (somewhere around 20 academic papers): blockchain technology potential was barely understood by the wider community, and the need for interoperability was not seen as a priority. Since then, we have come a long way. While I believe our high-level paper "A Brief History of Blockchain Interoperability" [1] introduces the area and its recent developments in great detail, one thing I can say: in ten years, we evolved from not knowing what the goal of blockchain is, to having completely different blockchains connected (somewhat) effectively - a gigantic pace of development for this technology.

New challenges emerged, partially due to the technology becoming more complex and partially due to the capital inflow in blockchain, leading malicious actors to try to exploit them. Combined with privacy, effective monitoring, and incident response frameworks, security seems to be the most prominent challenge today. Hopefully, some of these problems will be solved by the time you read this. Now, there is a lot of work ahead.

We are going to begin to act, beginning today. To do what needs to be done. Let's get on with the job!

Contents

List of Tables

List of Figures

Notation

This thesis uses tables and figures to better organize the conveyed ideas.

Definitions. The introduction of critical concepts is framed by a coloured box. Exemplifying:

Definition 0: A definition is a passage describing, and possibly formalizing the meaning

Research Questions. Key premises for the thesis validation are framed in a box:

RQ42: This is an illustrative *research question*.

Framed text. Colored frames are either used to illustrate basic concepts (Basics label) or to expose implications of the contextual contents (Implications label) in order to guarantee that the main text remains concise. Implications explain the impact that an item has on society.

Basics: Further Information

Experts may choose to skip these frames.

Implications: Impact on Society

Examples of implications are new research areas propelled by certain research

1 Introduction

Blockchain is often mentioned in the context of digital money and Bitcoin. Bitcoin has been referred to as digital money and digital gold [2], considered a reinvention of a technology that was around long before the alphabet [3].

Basics: Blockchain

The term *blockchain* has at least two different meanings: a type of system and a type of data structure. In this dissertation, we use the term blockchain to denominate a class of distributed systems. A blockchain maintains a shared state, specifically a replicated data structure that we denominate *distributed ledger* [4]. This ledger is maintained by a set of machines called nodes (or peers or participants) with computational and storage resources. Nodes are not trusted individually to maintain the distributed ledger; they are trusted as a group due to their number and diversity [5]. A blockchain can also be considered a *deterministic state machine* that provides a certain service, given existing incentives that the network can reward.

Such systems provide *safety* and *liveness* [6], which in the distributed system research area jargon means that such systems do not allow bad behavior from participants (*bad things do not happen*), and desired behavior eventually is processed by the system (*good things happen*) [7]. How these properties are realized depends on the desirable decentralization level, the fundamental property of blockchains, and the implementation specifics. Please refer to Appendix B.1 for further pointers.

The first blockchain was part of the Bitcoin system and provided as service transactions of a cryptocurrency, a digital currency, also designated Bitcoin [8]. The service provided by Bitcoin is the execution of transactions of bitcoins, the biggest cryptocurrency in terms of market capitalization, around 816 billion USD (as of 21 January 2024). It is debated whether blockchains can provide the financial infrastructure for cryptocurrencies with fiat-similar properties, namely medium of exchange, unit of account, and store of value. Arguably, cryptocurrencies are still not ready to be widely adopted. For example, its high volatility makes it a suboptimal unit of account [9]. However, we refrain from entering these discussions in this thesis and we thus assume that there is potential for the adoption of cryptocurrencies as (at least) viable complements to fiat currency [10, 11, 12].

Implications: Blockchain as Financial Infrastructure

The impact of cryptocurrencies in the current financial infrastructure is the creation of new investment vehicles and investment platforms [13, 14], promoting better financial inclusion to people with an internet connection. Second, it promotes faster settlement for cross-border transactions [15] and scalable, low-fee micropayments [16, 17]. Finally, cryptocurrencies, as alternatives to using banks, create tensions that origin new laws, regulations, and a new economic status quo [18, 19].

Since the introduction of Bitcoin, many other blockchains appeared. A notable example is Ethereum [20, 21], a blockchain supporting the execution of smart contracts and fault-tolerant programs the network executes. This invention effectively adds programmability to monetary transactions, allowing arbitrary data and business logic to be decentralized and enabling a wide range of applications. The term distributed ledger technology appeared to cover blockchain-like technologies that aggregate transactions in structures other than blocks; and to convey programmability of transactions other than financial: via smart contracts (for more details on smart contracts, refer to Appendix B.2).

***Definition 1**: Blockchain and Distributed Ledger Technology (DLT).* According to ISO TC 307, a DLT is a "ledger that is shared across a set of DLT nodes and synchronized between the DLT nodes using a consensus mechanism". A blockchain is a "distributed ledger with confirmed blocks organized in an append-only, sequential chain using cryptographic links" [22].

Smart contracts are triggered by transactions, which are recorded on the ledger. Nodes that form the DLT network agree on the validity and ordering of transactions via a *consensus mechanism*. Typically, DLTs provide transparency, tamper-resistance, and auditability of ledger information, providing desirable features that can alleviate some of the problems of centralized technologies which we will study throughout this thesis.

On the other hand, blockchains allow to decentralize processes and distribute trust among a network of mutually non-trusting peers. This feature of blockchain is of utmost importance because it is a technical mechanism to enforce, for the first time in history, decentralization of power, resources, and responsibility. By doing this, one can alleviate the problems of centralization, such as single point of failure and proclivity for conducting fraud and implementing corruption [23]. Corruption comes in the form of dishonesty or criminal offense when parties entrusted with a position of authority acquire illicit benefits – affecting all areas of society. It then does not come as a surprise that corruption is present at every layer of society, given a variety of reasons: technological limitations regarding traceability, lack of transparency, lack of auditability, and lack of accountability. For example, using centralized systems to record sensitive operations may open doors to hide traces without ways to enforce accountability [24, 25]. This causes distrust among stakeholders in communities, enterprises, and governments alike, causing prejudice to possible synergies that can raise efficiency and improve the status quo.

Basics: Centralized versus Decentralized Systems

A *centralized system* can still be distributed (e.g., for scaling or fault-tolerance purposes), but its components are trusted and operate under the umbrella of one authority. More concretely, it is a system where the state consensus is decided by a single party or multiple parties under the same authority. A *decentralized system* is a distributed system

where various parties control different components of the distributed system, and no party is fully trusted by all. In our context, we can consider a decentralized system to be a system where the state consensus is decided by conflicting or competing multiple parties, where accountability (from an external viewer's point of view) of individual decisions is *assured*. Each party composing the system can vote autonomously and has different incentives from other parties.

In fact, centralization is a consequence of attempting to achieve better performance in decentralized systems. For instance, when cryptocurrencies emerged, there were no cryptocurrency exchanges. Users had to trade directly but did not trust each other to go first (known as the fair-exchange problem, it is not solvable without a trusted third party [26]). An easy solution was to rely on centralized exchanges, which are single points of failure [27], vulnerable to corruption, and consistent targets of hackers [28]. Many examples exist. The most recent and mediatic example is the FTX exchange collapse due to corruption [29], which caused billions of dollars in damages, including millions of dollars lost from Portuguese taxpayers [30]. Therefore, there is a trade-off between convenience/user experience, performance, security, and decentralization. While these tradeoffs are being better studied, the community is advancing on these fronts. For instance, bringing privacy to permissionless blockchains [31, 32]. Thus, blockchain is becoming a great vehicle to build a more transparent, robust, and secure infrastructure to promote cooperation in the coming century.

We envision a world where decentralized currencies can support a free-market economy of trust-minimized applications, that empower synergies between mutually untrusted parties, as the foundation for new technological infrastructure. We, therefore, lay a vision for blockchain technology in the long run, to which we expect this thesis to contribute "*a world where more decentralized, efficient, and secure money is supported by more transparent, accountable, and decentralized institutions*".

Is this vision being accomplished? It seems that the current socio-economic environment, including rapid digitization of information and processes, the rise of machine learning, and the ubiquitous access to the Internet [33] amplifies the need for human-human and human-machine interactions without a single point of failure that is *transparent, dependable, resilient*, and that operate at a global scale. This creates an unprecedented need for decentralization. As a consequence, blockchain technology is growing at a fast pace. The development of real-world applications shows real interest from both industry and academia [34, 35]. For instance, decentralized applications have been developed in a wide variety of blockchains in the areas of public administration [24, 25], access control [36, 37], identity [38], healthcare [39, 40], and many others [41]. In fact, dozens of distributed ledger technology and blockchain systems [42] give rise to hundreds of blockchains [43], which in turn have deployed around 9,000 cryptocurrencies (only in permissionless blockchains) [43], leading to a new complex financial infrastructure. It includes payment networks [44], blockchain-based central bank digital currencies [45, 46] and decentralized finance (DeFi) applications [47, 48] are already being leveraged by multiple players, such as (centralized and decentralized) hedge funds [49]. El Salvador adopted Bitcoin as a legal tender in June 2021. Several dozen projects on central bank digital currencies, including the Digital Pound consortium and the European Central Bank's Digital Euro [50] are displaying the increasing need for digitizing money [51]. Adoption seems inevitable as the world's financial ecosystems evolve [52].

Finally, research suggests that the market for applications using DLTs will grow, with many organizations stating that blockchain is a critical priority [53, 54], due to, for example, cost reduction.The inherent advantages of blockchain are propelling the institutional adoption of blockchain. Many blockchain ecosystems invest and promote projects that advance knowledge in cryptocurrencies [55, 56] and blockchain open-source research [57, 58], bringing more adoption to the space and knowledge that feeds back into economics, cryptography, and distributed systems research, to name a few. In Portugal, a recent, national-wide blockchain initiative (October 2023) involving almost 50 partners from the academia and the industry proposes to create around 30 products, services, or processes with high potential of exportation and scalability has been proposed, approved, and funded[1].

Thus, blockchain is slowly but steadily becoming an infrastructure for global value exchange and distributed computation [59]. However, blockchains have been created as standalone networks, as autonomy from most external systems was sufficient for the first applications. Although promising, blockchain technology has many open research questions and problems, including in standardization and governance [60, 1], security and privacy [61, 62, 63, 41], and interoperability [64, 65, 66, 67, 1]. The challenge we focus on in this dissertation is interoperability.

1.1 The Case for Blockchain Interoperability

There is considerable industry and academic interest in the applications of blockchain interoperability and its technical challenges. Furthermore, the industry has evolved to connect multiple decentralized infrastructures to transfer liquidity, capital efficiency, and data across domains. As of 21 January 2024, the total value locked in bridges is approximately 30B USD[2].

On the other hand, partially due to their practical applications, we can assist in a skyrocket of interest in this research area. Figure 1.1 depicts the number of search results that Google Scholar returned per year. In the 2014-2017 interval, only 22 documents were found. In 2018, 2019, 2020, 2021, 2022, and up to the end of 2023, the results were 120, 225, 263, 505, 722, and 867, respectively. We can see a steep increase in interest in this research area from 2020 onwards, showcasing the interest in the interoperability research area applied to DLT technologies. In this section, we explain the motivation of this research area.

The interoperability challenge is complex, with specific challenges and solutions depending on the area (for a detailed exposition of the interoperability of software systems versus in the context of this thesis, we refer the reader to Appendix B.3). In the case of blockchain, it stems from heterogeneous DLT infrastructure being created as standalone networks, as autonomy from most external systems was sufficient for the first applications, namely wealth transfer in a closed network. Connecting those blockchains and making them cooperate (i.e., achieving interoperability [68]) have a practical utility and importance [69, 70], as demonstrated by the wide interest in recent years. Interoperability allows serving multiple use cases and stakeholders who require different blockchain features and capabili-

[1] https://blockchain.void.pt/
[2] https://l2beat.com/bridges/summary

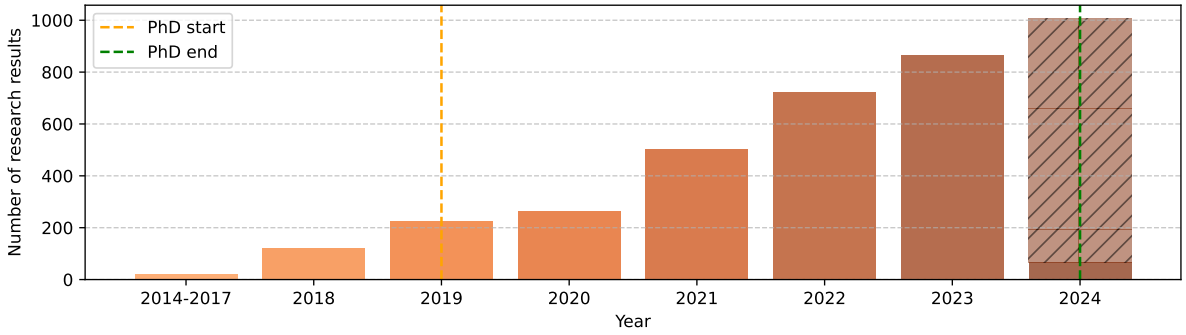


Figure 1.1: Number of search results in Google Scholar with keywords "blockchain interoperability" OR "cross-chain communication" from 2014 to 2024 as of 26^{th} January 2024 (67 results). The dashed lines in the shadowed box in "2024" represent the projected number of search results using linear interpolation of previous years.

ties [71] - e.g., the need for interoperation between blockchain-based state-controlled CBDCs does not conflate the same challenges as interoperating between decentralized finance applications hosted on public, permissionless blockchains. For an extensive list of use cases of blockchain interoperability, refer to Appendix D. Moreover, each blockchain has its security risks; as the technology is still maturing, the user base is relatively limited (e.g., compared to the World Wide Web). Consequently, there are uncovered bugs and security flaws [72]. Therefore, developers and researchers must choose between novelty and stability in choosing a blockchain, leading to a vast diversity of choices. This diversity leads to *fragmentation*, i.e., *data and value silos*.

Interoperability allows connecting these silos by enabling communication between systems to exchange data and assets (fungible and non-fungible), leading to a higher heterogeneity of solutions in the market, synergies between projects, and higher liquidity to end-users. Interoperability also promotes the avoidance of vendor lock-in, by allowing users and developers to migrate their infrastructure in case of need (e.g., blockchain becomes insecure or obsolete, more attractive features on new blockchains) [73, 74]. No blockchain should become a single point of failure, as it has happened before [75]. To study and develop secure and scalable interoperability techniques on blockchains that power decentralized applications (see Appendix D for a list of cross-chain decentralized applications), the field of blockchain interoperability appeared.

1.2 Thesis Scope

Studied since the 1980s [76, 77, 78, 79, 80], interoperability plays a major role in connecting information systems. In particular, this research area started gaining more notoriety with the emergence of the Internet [81]. The latter was created in a geo-political context (namely the Cold War) that required the creation of a resilient, dependable, scalable, manageable, and self-healing network that could sustain attacks from a powerful adversary. Effectively, the Internet architecture specified the number of properties that propelled it as a commercial success, enabling considerable economic growth [82]. Those properties are *survivability*, *diversity of services*, and *diversity of networks*.

***Definition 2**: Interoperability*. "Interoperability is the ability of two or more software components to cooperate despite differences in language, interface, and execution platform" [77]. Wegner established a bridge between the concept of interoperability and existing standards. As those authors were influenced by the standards existing at that time, authors nowadays are influenced by the Internet architecture and concepts, in what concerns blockchain interoperability [81, 61]. Thus, reflecting on the Internet's architecture is a good starting point to understand how blockchains can interoperate.

Non-surprisingly, these principles anchored in the Internet architecture are guiding the development of interoperability protocols and standards, with direct application to information systems and, as our object of study, blockchains [81, 83]. Given the history of the development of the Internet and computer networks in general, it does not come as a surprise that communities are pushing toward interoperability across blockchains (also referred to as *cross-chain interoperability*). As a consequence, the blockchain industry is settling on several multi-chain ecosystems connected by cross-chain solutions.

Despite recent progress connecting homogeneous blockchains, many unsolved challenges in DLT interoperability theory and practice are exacerbated by the lack of standardization among APIs, data models, security, privacy, and monitoring processes. Thus, integrating with different DLTs is an error-prone and tedious task [84]. This is one of the reasons why it is still difficult for centralized systems to exchange assets with blockchains, despite advances in developing higher-level APIs that simplify this process [83, 85, 86, 87].

Basics: Interoperability is not Binary

To provide better support for enterprise collaboration, synergies, and a richer ecosystem, integration processes should be verified and improved [88, 89]. Thus, integration is not a final step, but rather a continuous process that is also subject to change. Interoperability assessment tools exist for one to positioning systems in terms of how interoperable it is.

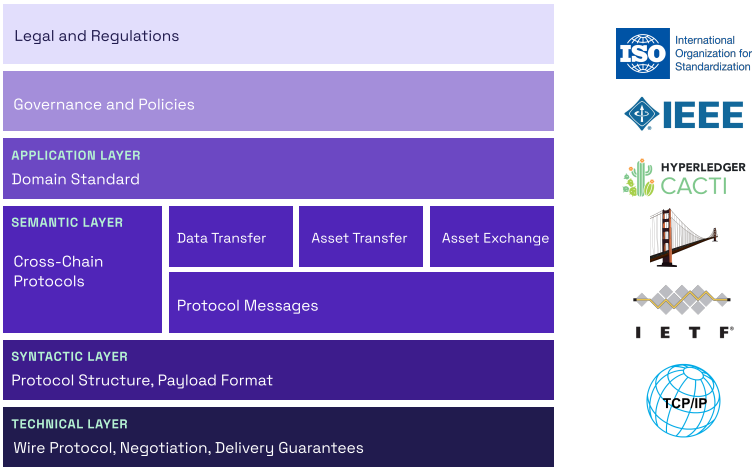


Figure 1.2: Blockchain interoperability layers [1]

We classify the different interoperability layers to understand the nature of these attacks better,

although various frameworks exist depending on the research field [90]. For blockchain interoperability, we consider a six-layer approach that captures its main challenges [1]. First, the technical layer focuses on data formats, communication protocols, interface specifications, and integration services ("bits and bytes"). It precedes the syntactic layer that defines protocol structure and payload formats. Semantic interoperability exists when systems can interpret exchanged information following a defined ontology. This translates into systems being able to exchange information and assets. Different applications can be built on top of the semantic layer (e.g., bridges on top of asset transfer protocols and functionalities).

Organizational interoperability comprises the set of processes, agreements, policies, and governance on interoperability across two or more organizations, realizing a specific use case. Legal interoperability assures organizations can cooperate under heterogeneous legal frameworks, policies, and strategies (legal and regulations). Figure 1.2 summarizes the existing standardization attempts across the different interoperability layers. Having clear interfaces between the different layers limits its development complexity and provides a separation of concerns that empowers developers to think more abstractly about the underlying layers and focus on application logic. Different standards are being built for each layer. For instance, at the IETF, the Secure Asset Transfer Protocol (SATP) working group [91] defines a protocol for digital asset transfers that focuses on the semantic layer (with the potential to influence the organizational layer and perhaps the legal layer). For simplicity, we refer to the four main layers: technical (includes syntactic), semantic (includes application), organizational, and legal.

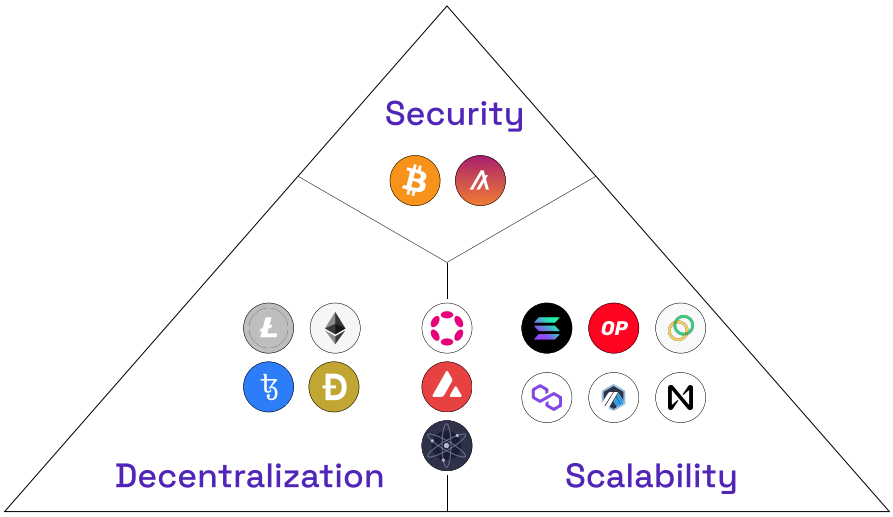


Figure 1.3: Classification of blockchains according to the blockchain trilemma. We position solutions in one of three buckets (only the bucket matters, and not the relative position of a blockchain to another within the bucket). Blockchain ecosystems, left to right, up to bottom: Bitcoin, Algorand, Litecoin, Ethereum, Polkadot, Solana, Optimism, Celo, Tezos, Dogecoin, Avalanche, Polygon, Arbitrum, NEAR, Cosmos.

Interoperability does not only conflate flexibility and application portability across different layers. It also has the potential to solve some of the biggest blockchain research challenges. Let us recall the blockchain trilemma (cf. Figure 1.3), postulated by one of Ethereum's founders [20], which

states that blockchains have an inherent trade-off between security, scalability, and decentralization. Being an equivalent of the CAP theorem[3] for blockchains, the core property chosen is typically security - implemented through consensus algorithms [93], crypto-economics [94], formal modeling, and results from distributed systems research (namely crash-fault tolerant and byzantine-fault tolerant algorithms [7, 95]). Typically, the more nodes involved in a peer-to-peer network, the harder it is to corrupt it, but the slower the consensus becomes (intuitively, more nodes, more messages exchanged, and, therefore, the higher the overall communication latency). Consequently, decentralization and security walk *manus in manu*. Nonetheless, we still have to solve the scalability part of the trilemma, via interoperability. In particular, interoperability promotes blockchain *scalability*, as it provides a way to offload transactions to other blockchains, e.g., via sharding [96] and sidechains (called layer twos [97]). On the one hand, interoperability is a requirement for scalability. On the other, it enables more functionality by force of synergies with external systems.

Blockchain interoperability is typically based on two prisms: *multi-chain interoperability*, and *cross-chain interoperability*. In multi-chain interoperability, instances of a *blockchain engine* [65] (aka *blockchain of blockchains*, e.g., Cosmos, Polkadot, Avalanche) communicate with each other through a trust anchor that is implemented in the protocol. Each instance of the blockchain engine has a built-in interoperability protocol and data format that other instances of that engine understand. Consider Polkadot's parachains: each parachain (application-specific blockchain) communicates with other parachains via XCMP, a built-in interoperability format [65]. Communications are anchored by the canonical blockchain (the Relay Chain [98] in the Polkadot network) and establish trust from one parachain to the world. In Cosmos, interoperable blockchain instances are called zones, which communicate via a protocol called Inter Blockchain Communication (IBC) [65]. The Cosmos developers suggest a main zone that performs the role of an aggregator, the Cosmos Hub [99]. A light-client interoperability mechanism anchors the multi-chain communication that processes cryptographic proofs [100].

Other blockchains that claim to have incredible scalability typically use a sharding system [101], where each shard is responsible for computing a subset of the overall transactions. The result is then communicated across shards. However, there is a problem with multi-chain interoperability. Polkadot's parachains can communicate with each other, but can they communicate with Cosmos or other blockchain engines? Not natively, because they follow a different protocol and have a different global state (i.e., are *heterogeneous*). Those are the boundaries of a blockchain network (otherwise, they would be considered the same system, i.e., *homogeneous*). That is, the cross-chain vision connects heterogeneous chains; in the multi-chain vision, a native cross-chain protocol connects homogeneous chains that utilize the same framework and typically are anchored in a common chain. This thesis focuses on cross-chain interoperability, which enables *hybrid blockchain applications*.

An example of a project enabling cross-chain interoperability is Hyperledger Cacti [102, 103, 45], the flagship interoperability project within the Hyperledger Foundation. Cacti contains, at its core, a set of *plugin connectors*, software components that allow Cacti nodes to read and write to different DLTs. Cacti nodes are API servers that translate requests from business logic plugins into DLT-specific

[3]The CAP theorem [92] states a trade-off between consistency, availability, and partition tolerance in distributed systems.

transactions. Business logic plugins allow realizing cross-chain transactions by enforcing a set of reads and writes constrained by rules. A cross-chain transaction is a set of transactions in different domains related by a set of rules [104]. A simple rule can be "when in DLT A my account burns X tokens, credit X tokens to my account on DLT B". For more details, we refer the reader to Appendix B.4.

***Definition 3**: Hybrid Blockchain Applications.* Cross-chain applications that utilize heterogeneous infrastructure, e.g., a private blockchain and a public blockchain. An example can be found in Hyperledger Cacti Carbon Emission working group [100].

Figure 1.4 showcases the high-level architecture of a general-purpose IM. In brackets, we illustrate the corresponding implementation of the concept within Hyperledger Cacti. The IM exposes its functionality via a set of APIs [a set of OpenAPI specifications]. For instance, in Cacti, core-api defines schemas, e.g., the format of plugins (business logic plugins, connectors), constants, structure of DLTs, consortia, authentication, among others. Each Cacti plugin exposes its own interfaces, specified with Open API.

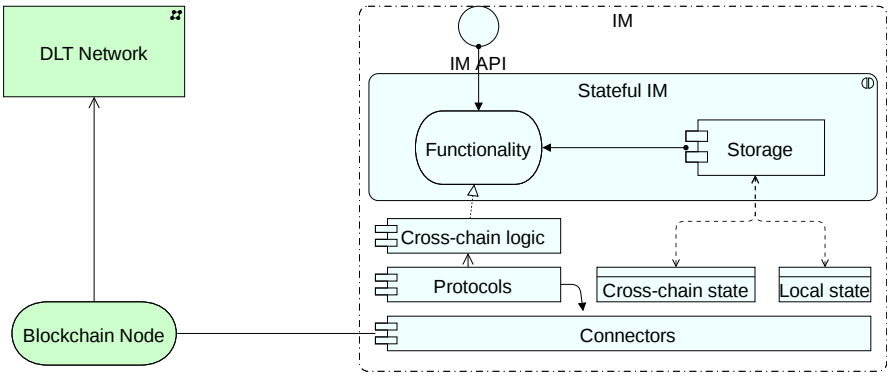


Figure 1.4: Representation of a general purpose blockchain interoperability solution in the Archimate modeling language [105].

Upon the IM receiving a request [Cacti node], it is redirected to the module implementing the functionality dealing with that request. Functionality is managed by cross-chain logic modules [business logic plugins]. Cross-chain logic modules process the request, translating it into protocol-specific transactions, that are sent to connectors [connector plugins], and dispatched to a blockchain node. This processing can be persisted in storage. Eventually, the interaction with target nodes (either DLT nodes or other IMs) receives a response, which is processed and optionally persisted, creating a cross-chain state. A detailed explanation is present in [100].

1.3 Problem Motivation

The vehicle for creating the hypothesis posed by this thesis is our survey on blockchain interoperability conducted in 2021 [65] (present in Appendix C), where we identified three interconnected problems. Note that Chapter 2 contains our work [4] that updates the 2021 survey by proposing a

simpler conceptual model and doing an informal survey of the available solutions, conducted around September 2022.

Problem 1: Lack of Classification and Evaluation Frameworks for Interoperability Mechanisms

Recent years have seen extensive work on IMs. Several surveys condensed that knowledge, focusing on public connectors (connecting public blockchains) [106, 64, 107, 108, 109], architecture for blockchains [61], and others [110, 111, 112, 113, 114]. Some surveys provide a systematic overview of the area, showcasing more modern interoperability solutions [65, 115].

However, the systematization of existing solutions is not coherent in the literature. This is due to the existence of inconsistent classification methodologies across surveys, making it challenging for researchers to evaluate available options systematically [90, 100]. What is a consistent classification framework that can improve past work and improve understanding when classifying IMs? What are the relevant theoretical contributions to the DLT interoperability research area, including the current capabilities, components, connection modes, interoperation modes, practical applications, and limitations of blockchain interoperability?

On the other hand, systematic assessments of IMs are also scarce [1]. Research typically does not follow a well-defined interoperability assessment framework and, therefore, assesses the solutions ad-hoc, prominently the technical specification [83] or performance [116]. The lack of evaluation methodologies makes it hard to systematically evaluate solutions, contributing to a more robust and unbiased comparison.

Problem 2: Absence of Organizational Interoperability

Both private organizations and governments are actively investigating and investing in blockchain-based digital assets by, for example, promoting new platforms for digital transactions [52]. A key challenge to enabling this digital economy is to safely connect different networks, enabling network effects among them [71, 61]. Although many blockchains, and in particular blockchain interoperability protocols solve this challenge at the technical and semantic layers, there is a lack of consortia implementing interoperability solutions, especially in the hybrid blockchain approach; there is also a lack of protocols capable of complying with local jurisdiction and laws (and thus acting at the organizational and legal layers). This hinders the development of enterprise-grade applications using blockchain, since public blockchains, private blockchains, and legacy systems cannot communicate seamlessly. This challenge is exacerbated by the lack of a common data format that can be used amongst heterogeneous infrastructure (DLT and non-DLT based). While technical and semantic interoperability are necessary conditions for organizational interoperability, they are not sufficient. We hypothesize that the lack of a privacy-preserving common data format for interoperability is the main barrier to the development of organizational interoperability, i.e., protocols implementing semantic interoperability currently do not provide the necessary technical requirements that are desirable for enterprises.

The lack of legal interoperability is also a prominent challenge in the field, which we do not address in this thesis.

Problem 3: Ineffective Methods for Securing IMs

To connect blockchains, we need to use cross-chain communication, a set of techniques allowing us to share data and transfer assets between blockchains [65]. This concept seems prone to security vulnerabilities [117, 118, 119], and it is indeed - more than $3B in losses happened only in blockchain bridges, the most popular cross-chain applications [104, 120, 121] (there are more than 110 bridges[4]). Cross-chain bridges conquered the rank of having the most devastating losses in terms of capital within DeFi applications. Due to the presence of a large attack vector, it has been pointed out by reputable people in the blockchain community that multi-chain is inherently more secure than cross-chain [122]. While the authors tend to agree that multi-chain does seem to lower the attack vector for interoperable applications, it is also the case that *there will not be a blockchain to rule them all*: design decisions need to be made, and some give priority to scalability while sacrificing decentralization (namely permissioned blockchains), while others focus on privacy [123]; others are application-specific [124, 125].

Blockchain security is critical due to its wide attack surface, which we emphasize the prime motivating factor: the existence of honeypots - the more value cross-chain bridges hold, the more incentive criminals will have to attack those systems [126]. The total value locked (TVL) in bridges peaked in March 2022, at around $30 billion worth of assets locked just in Ethereum (as the chain receiving the transferred assets) [127, 128], effectively reflecting the synergistic effects of free flow of capital, as now users can use their capital on multiple blockchains. As of January 2024, the TVL for canonical bridges is still significant, collecting around 28B USD [129], with considerable recent fluctuations, therefore reflecting the larger macroeconomic environment. In fact, as scalability solutions that come with blockchain adoption are used more and more, the higher the TVL increases, the more we will assist. Notwithstanding, security is paramount to secure all this TVL.

The condition that modifies the TVL the most is the execution of successful attacks by hackers. Some examples of recent mediatic attacks include the Wormhole bridge, where the attacker stole around $325M [130, 131], and the most significant on-chain attack in the cryptocurrencies history, the Axie Infinity's Ronin Bridge [132], which caused around $625M in losses. In February 2022, the Wormhole bridge was attacked and resulted in $320M in damage [133]. In June 2022, the Harmony bridge was hacked, resulting in $100 million in losses [134]. Although hackers were offered $1 million to return the funds to the community, they seem to have not complied [135]. In August 2022, the Nomad bridge collateral was stolen, resulting in the loss of $200M [136], despite the bridge being developed by an expert team and audited multiple times. Throughout 2023, several hacks were performed, causing at least 130M USD in damages, which we systematically analyzed and categorized [121]. The trend continues, as 2024 saw its first hack happen on the first day of the year, stealing 80M USD from users [137]. The grand total is now more than USD 3B in a mere three years [138, 139], leading to the loss of credibility and even to the closing of several companies in the space. With the increasing

[4] https://chainspot.io/

adoption of blockchain (see, for example, the first Bitcoin ETF approval [140]), it is clear that citizens will be more exposed to cryptocurrencies in the future and, therefore, exposed to blockchain bridges.

In order to protect the current and incoming users, the community still has a long way to go to implement secure bridges. As interoperability becomes general-purpose, and more data transfer use cases and hybrid (data transfers + asset transfers) will be developed, the more pressing security is.

1.4 Thesis Research Questions and Requirements

The **hypothesis** of this thesis, sustained by the author's vision, is that *IMs providing interoperability across the technical, semantic, and organizational layers can securely implement the requirements of both centralized and decentralized organizations.* Naturally, testing this hypothesis leads us to the how. First, how can IMs realize interoperability across the technical, semantic, and organizational layers for legacy systems and infrastructure that need to abide by the requirements of enterprises? Second, how can one add robustness and assure the safety of IMs connecting both legacy systems with blockchains, and native decentralized applications operating in different blockchains? Figure 1.5 lists the thesis structured view: problem space, solution space, and methodology to validate the target hypothesis. Each research question addresses its respective problem, posed in the previous section.

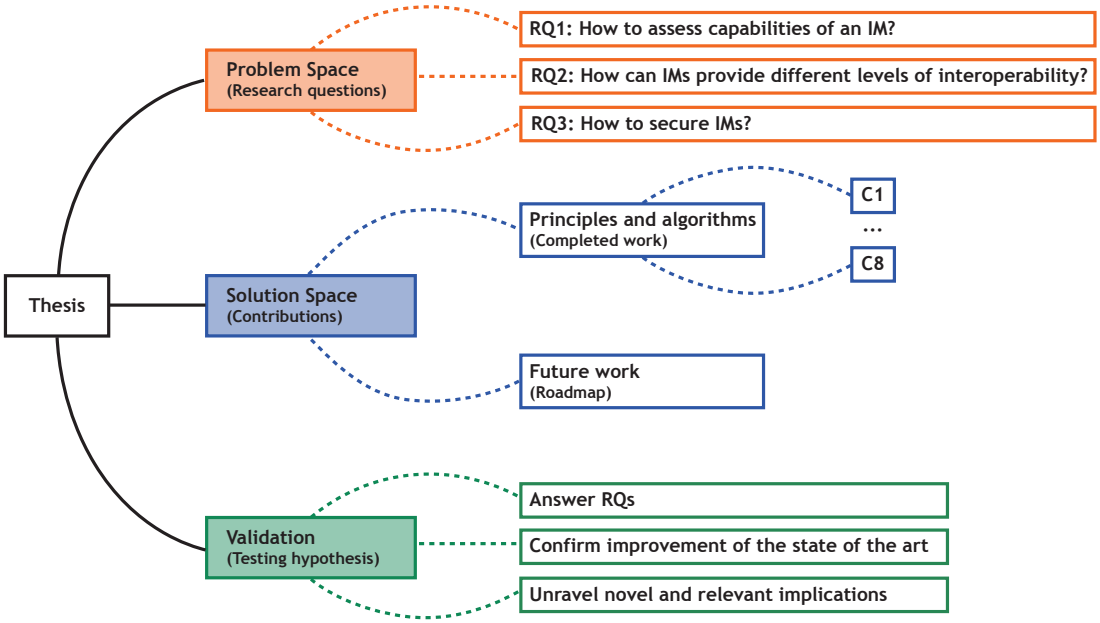


Figure 1.5: Structured view of the thesis concept. The structure view contains three parts. First, the problem space is where we define each research question RQ (slightly modified for the sake of space). Next is the solution space, defining each contribution, C, and future work directions. Finally, we demonstrate the thesis validation procedure by showcasing the axes that support the posed hypothesis.

Next, to validate the proposed hypothesis, we decompose its assertion according to an incremental set of research questions (RQ) and respective requirements (or sub-research questions). These research questions define the problem space.

RQ1: How to assess the interoperability capabilities of an interoperability mechanism?
Associated requirements:

- R1.1) Systematic classification of IMs;
- R1.2) Robust assessment of IMs;
- R1.3) Framework to choose an IM based on use case requirements.

Addresses Problem 1

To assess interoperability mechanisms, one needs to consider several key questions. First, what classes of solutions exist in the literature and their technical attributes (R1.1)? Provided an answer to R1.1, we can address the interoperability capacity system, by asking *can the system interoperate with other systems as is?*, and *is the system able to be changed to adapt to other systems?*. Assessing interoperability between systems can be done by asking *how well can a pair of systems interoperate?*, and *what are the current problems or barriers that prevent the systems from interoperating better?* (R1.2). Finally, measuring performance requires studying cross-chain latency, cross-chain throughput, and cross-chain costs associated with an IM (R1.3). This allows an end-user to choose an appropriate blockchain interoperability solution, both from the infrastructure and the functionality perspectives.

RQ2: How can IMs provide technical and semantic interoperability (and provide the basis for organizational interoperability)?
Associated requirements:

- R2.1) Privacy-preserving standardized data format for accountable interoperability;
- R2.2) Gateway-based interoperability framework and architecture;
- R2.3) Gateway reliability, decentralization security, and privacy assessment;
- R2.4) Gateway interoperability and standardization;
- R2.5) Guarantee atomicity, consistency, isolation, durability, and accountability (ACIDC) properties for gateway-operated transactions.

Addresses Problem 2

Similarly to Internet routing gateways, which enabled interoperability around private networks, and fostered the rise of the Internet, the global network of DLTs will require blockchain gateways. Gateways permit digital currencies and virtual assets to be transferred seamlessly between these systems, offering interesting characteristics to the enterprise domain, namely an audit trail, privacy, and the capacity to rollback transactions. Transferring an asset among blockchains via gateways is equivalent to an atomic swap [141] that locks an asset in a blockchain and creates its representation on another. However, how can one guarantee a fair exchange of assets (either all parties receive the requested assets, or none do) across gateways?

To start tackling these questions, we need to satisfy two conditions. First, processes requiring organizational interoperability require a standardized data format [142] and eventually, compliance with different regulations and laws. In the European Union, one notable regulation is the General

Data Protection Regulation (GDPR) [143, 144]. Therefore, we need a standardized data format for interoperability that can provide a level of privacy adequate to satisfy common regulations such as the GDPR (R2.1). For the reasons mentioned above, the blockchain gateway paradigm seems to be the most appropriate for interoperation capabilities in permissioned environments (R2.2). To ensure the properties that enable a fair exchange of assets in these environments, blockchain gateways must operate reliably and withstand various attacks (R2.3). Thus, a crash-recovery strategy must be a core design factor of blockchain gateways, where specific recovery protocols can be designed as part of the digital asset transaction protocol between gateways. A recovery protocol, allied to a crash recovery strategy, guarantees that the source and target DLTs are modified consistently, i.e., that assets taken from the source DLT are persisted into the recipient DLT, and no double spend can occur. The recovery process requires synchronization across gateways (R2.4). These requirements should translate to transactions achieving ACID + accountability (atomicity, consistency, isolation, durability) properties (R2.5).

RQ3: How to add robustness to blockchain interoperability mechanisms from a security perspective?
Associated requirements:

- R3.1) Reduction of the attack surface for IMs and cross-chain applications;
- R3.2) Secure decentralization of the IM;
- R3.3) Strategy for monitoring attacks on cross-chain applications;

Addresses Problem 3

For DLT technology to be adopted, it has to mature. Interoperability security is a critical requirement for the adoption of interoperability solutions and, consequently, DLT technology, but it seems its security is still not addressed adequately [121]. We pose several directions to increase the security (robustness) of interoperability solutions by minimizing some attack vectors (R3.1). First, we remove centralizing vectors in interoperability solutions, which have shown to contribute to many realized hacks (namely poorly decentralized multisignatures as the trust anchor for interoperability purposes) (R3.2). Secondly, robustness can be improved by promoting proactive and reactive incident response by modeling the interoperability solution, and the interoperable application, performing continuous monitoring, and real-time incident response (R3.3).

1.4.1 Methodology

We use the Design Science Research Methodology (DSRM) to guide our thesis dissertation writing. DSRM aims to provide a method to create innovative, purposeful solutions that solve a given problem [145], typically referred to as following the design science paradigm [146]. This methodology consists of several steps: 1) identification of the problem, 2) definition of the objectives for a solution, 3) development of the solution, 4) implementation and evaluation of the solution, 5) and finally, the communication of the results (scientific dissemination). For step 1, we identified the most prominent problems by reviewing the literature (i.e., the papers in the study [65]) and conducting our own

literature review [100].

This process outlines a series of problems in the space. In step 2, we define the solution space (next section), taking into account the observed problems. We translate the existing problems into research questions and their requirements. After that, We identify the specific contributions addressing the posed research questions. Steps 3 and 4 correspond to the research corpus, where we develop and evaluate each solution that is part of this thesis (Chapters 2-6). Note that in each chapter, we partially address the identification of the problem, as each contains a standalone publication. Likewise, each chapter implicitly or explicitly addresses the definition of objectives (namely by defining research questions and contributions). Step 5 corresponds to the communication of the results, which includes the elaboration and presentation of this dissertation, as well as the scientific publications that it is based on. Figure 1.6 illustrates the methodology.

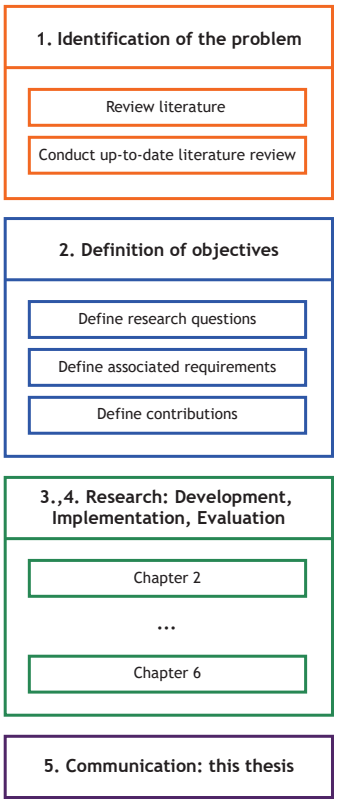


Figure 1.6: Application of DSRM to this thesis.

Such an approach allows extracting design principles from the design choices. This is a suitable methodology for the overall thesis due to the fast pace of the blockchain industry, where the industry consistently develops innovative products unbacked by academic research. Nonetheless, each of our research papers, realizing the scientific contributions may follow different methodologies. For each paper, we use different evaluation methodologies, including formal or semi-formal proofs [147], empirical evaluation [148], and qualitative evaluation [149].

1.5 Solution Space and Outline

Multiple contributions resulted from addressing the introduced research questions and requirements. Contributions take two major forms: 1) theoretical contributions, i.e., principles, theories, formalizations, methodologies, frameworks, and 2) applied computational methods, i.e., algorithms, protocols, and assessments that rely on one or more theoretical contributions. Some contributions are not only relevant to tackle the target problem, but can also be applied to answer other problems across the literature. As we address and answer each research question, we expect that the proposed research produces the following scientific contributions:

Contributions addressing RQ1

C1: A unified conceptual model and classification framework for blockchain interoperability solutions;

C2: A framework to assess the interoperability capabilities of a system utilizing multiple DLTs (in terms of potentiality, compatibility, and performance), based on the defined conceptual model;

C3: A pair of decision models that allow one to choose a blockchain interoperability solution, considering a set of criteria defined by our blockchain interoperability model;

Contributions addressing RQ2

C4: A common data format for accountable and privacy-preserving interoperability - blockchain views.

C5: Blockchain gateway paradigm as the enterprise-grade IM for auditable, private, and reliable interoperability

C6: Technical architecture, protocols, and algorithms to guarantee ACIDC properties for transactions amongst gateway-based IMs.

RQ1 - How to assess the interoperability capabilities of an interoperability mechanism?
R1.1 - Systematic classification of IMs
C1 - A unified conceptual model and classification framework for blockchain interoperability solutions;
R1.2 - Robust assessment of IMs
C2 - A framework to assess the interoperability capabilities of a system utilizing multiple DLTs [...]
R1.3 - Framework to choose an IM based on use case requirements
C3 - A pair of decision models that allow one to choose a blockchain interoperability solution [...]
RQ2 - How can IMs provide technical and semantic interoperability (and provide the basis for organizational interoperability)?
R2.1 - Privacy-preserving standardized data format for accountable interoperability;
C4 - A common data format for accountable and privacy-preserving interoperability - blockchain views.
R2.2 - Gateway-based interoperability framework and architecture;
R2.3 - Gateway reliability, decentralization security and privacy assessment
C5 - Blockchain gateway paradigm as the enterprise-grade IM for auditable, private, and reliable interoperability
R2.4 - Gateway interoperability and standardization
R2.5 - GuaranteeACIDC properties for gateway-operated transactions.
C6 - Technical architecture, protocols, and algorithms to guarantee ACIDC properties for transactions
RQ3 - How to add robustness to blockchain interoperability mechanisms from a security perspective?
R3.1 - Reduction of the attack surface for IMs and cross-chain applications
R3.2 - Secure decentralization of the IM
C7 - Protocols, algorithms [...] for decentralized, secure blockchain interoperability based on [...] SNARKs
R3.3 - Strategy for monitoring attacks to cross-chain application
C8 - Protocols, algorithms [...] for continuous monitoring and incident response of blockchain bridges.

Table 1.1: Contributions of this thesis per research question, requirements, and publication. Stacked requirements are addressed by the first following contribution (e.g., requirements R2.2 and R2.3 are both addressed by contribution C5).

Contributions addressing RQ3

C7: Protocols, algorithms, technical architecture, and frameworks for decentralized, secure blockchain interoperability based on the cryptography behind zero-knowledge proofs (namely SNARKs).

C8: Protocols, algorithms, technical architecture, and frameworks for continuous monitoring and incident response of blockchain bridges.

Table 1.1, maps the research questions, contributions, and respective publications, serving as a reference point for reasoning about the thesis hypothesis.

1.5.1 Scientific Dissemination

According to the introduced research questions, we list below the current status of the dissemination of the contributions from our thesis to the scientific community (contributing to the communication step of our methodology). This list contains only peer-reviewed publications, excluding other forms of dissemination, such as invited speeches, workshops, technical articles, contributions to open-source projects and standardization organizations, tutoring and teaching activities, collaborations in national and international projects, and scientific meetings and symposiums. Table 1.2 shows the current state of the publications that compose this dissertation.

Accepted and under revision publications per research question (RQ)	State (January 2024)	Dissertation Chapter
RQ1		
P1: Do You Need a Distributed Ledger Technology Interoperability Solution?	✓	Chapter 2
RQ2		
P2: BUNGEE: Dependable Blockchain Views for Interoperability	✓	Chapter 3
P3: HERMES: Fault-Tolerant Middleware for Blockchain Interoperability	✓	Chapter 4
RQ3		
P4: Harmonia: Securing Cross-Chain Applications Using Zero-Knowledge Proofs	⊙	Chapter 5
P5: Hephaestus: Modelling, Analysis, and Performance Evaluation of Cross-Chain Transactions	✓	Chapter 6

Table 1.2: State of the publications made in the context of this dissertation on January 2024, per research question. The state of publications is either accepted or published (represented by ✓) or work submitted for review, (represented by ⊙).

1.5.2 Outline

Figure 1.7 represents the thesis storyline: the chapters and their dependencies. The foundation chapters support theoretical research that was used to create *BUNGEE*, *Hermes*, `Hephaestus`, and `Harmonia`. `Hephaestus` adds monitoring and auditing to `Hermes` and `Harmonia`, based on the common data format introduced by BUNGEE.

After presenting the problem space, in this chapter, we proceed to propose a solution space. Chapter 2 presents the contributions relative to RQ1. It presents a study on the research area of DLT interoperability by dissecting and analyzing previous work (i.e., analyses the state of the art and conducts a literature review). We study the logical separation of interoperability layers, how a DLT can connect to others (connection mode), the object of interoperation (interoperation mode), and propose a new categorization for IMs, the first interoperability assessment for DLTs that systematically evaluates the interoperability degree of an IM.

Chapters 3 and 4 present the contributions relative to RQ2. Chapter 3 presents the concept of

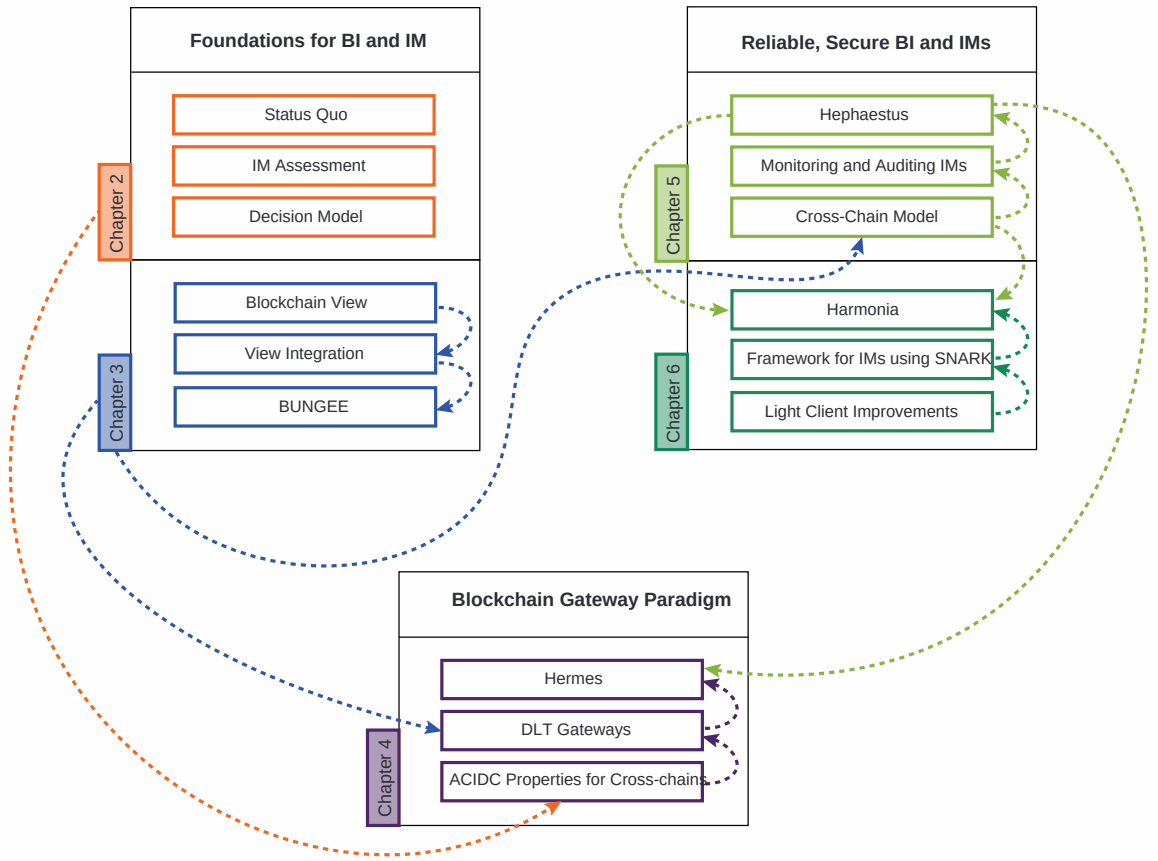


Figure 1.7: Thesis storyline. This figure represents the various building blocks of this thesis and their interdependencies.

blockchain view (view) – an abstraction of the state a participant can access at a certain point, as a common data format for gateways. Views allow us to systematically reason about either state partitions within the same DLT or an integrated view spanning across several DLTs. We introduce BUNGEE (Blockchain UNifier view GEnErator), the first DLT view generator, to allow capturing snapshots, constructing views from these snapshots, and merging views according to a set of rules specified by the view stakeholders. The blockchain research view is inspired by classical database view research [150]. We have done work on the necessary background to understand views [74].

In Chapter 4, we propose `Hermes`, a fault-tolerant middleware that connects blockchain networks and is based on SATP [91] (previously called the Open Digital Asset Protocol, ODAP). `Hermes` is crash fault-tolerant by allying a new protocol, ODAP-2PC (currently referred to as *SATP Crash Recovery Mechanism*), with a log storage API that can leverage blockchain to secure logs, providing transparency, auditability, availability, and non-repudiation, by verifiably communicating state changes from a source blockchain. We briefly explore a use case for cross-jurisdiction asset transfers. `Hermes` assumes that the way to communicate state changes follows a proof format like the one proposed in BUNGEE [4]. SATP is currently implemented within Hyperledger Cacti.

Chapters 5 and 6 present the contributions relative to RQ3. Chapter 5 presents `Harmonia`, an extensible, decentralized, secure, and efficient framework for building cross-chain applications. At its core, we leverage DendrETH, a smart contract implementation of Ethereum's light client sync protocol,

allowing blockchains applications to read Ethereum's state in a trust-minimized way. `Harmonia` solves the security problem by lowering the attack surface and relying on the properties of zero-knowledge proofs.

Chapter 6 presents `Hephaestus`, the first cross-chain model generator that captures the operational complexity of cross-chain applications. `Hephaestus` can generate cross-chain models from local transactions in different ledgers, realizing arbitrary cross-chain use cases and allowing operators to monitor their applications. Monitoring helps identify outliers and malicious behavior, which can enable programmatically stopping attacks ("a circuit breaker"), including bridge hacks.

Finally, Chapter 7 concludes this thesis, by inspecting its validation elements (cf. Figure 1.5), discussing the contributions, and proposing future work directions.

2 Do You Need a Distributed Ledger Technology Interoperability Solution?

The following chapter corresponds to the following publication [100]:

Status: Published ✓

Publication: Distributed Ledger Technologies: Research and Practice (ACM DLT)

Submitted: 23 January 2022

Accepted: 16 September 2022

Citation: Rafael Belchior, Luke Riley, Thomas Hardjono, André Vasconcelos, and Miguel Correia. 2023. Do You Need a Distributed Ledger Technology Interoperability Solution? Distrib. Ledger Technol. 2, 1, Article 1 (March 2023), 37 pages. https://doi.org/10.1145/3564532

Methodology: Design Science Research Methodology [145]

Evaluation Methodology: Qualitative Evaluation [149]

Journal Description: Distributed Ledger Technologies: Research and Practice (DLT) is a peer-reviewed journal that seeks to publish high-quality, interdisciplinary research on the research and development, real-world deployment, and evaluation of distributed ledger technologies, such as blockchain, cryptocurrency, and smart contract. DLT will offer a blend of original research work and innovative practice-driven advancements by internationally distinguished DLT experts and researchers from academia, and public and private sector organizations.

Do You Need a Distributed Ledger Technology Interoperability Solution?

RAFAEL BELCHIOR, INESC-ID, Instituto Superior Técnico, Universidade de Lisboa, Portugal and Quant Network, United Kingdom
LUKE RILEY, Quant Network, United Kingdom
THOMAS HARDJONO, Massachusetts Institute of Technology, USA
ANDRÉ VASCONCELOS and MIGUEL CORREIA, INESC-ID, Instituto Superior Técnico, Universidade de Lisboa, Portugal

Entrepreneurs, enterprises, and governments are using distributed ledger technology (DLT) as a component of complex information systems, and therefore interoperability capabilities are required. Interoperating DLTs enable network effects and synergies, and similarly to the rise of the Internet, it unlocks the full potential of the technology. However, due to the novelty of the area, interoperability mechanisms (IMs) are still not well understood, as interoperability is studied in silos. Consequently, choosing the proper IM for a use case is challenging.

Our article has three contributions: first, we systematically study the research area of DLT interoperability by dissecting and analyzing previous work. We study the logical separation of interoperability layers, how a DLT can connect to others (connection mode), the object of interoperation (interoperation mode), and propose a new categorization for IMs. Second, we propose the first interoperability assessment for DLTs that systematically evaluates the interoperability degree of an IM. This framework allows comparing the potentiality, compatibility, and performance among solutions.

Finally, we propose two decision models to assist in choosing an IM, considering different requirements. The first decision model assists in choosing the infrastructure of an IM, while the second decision model assists in choosing its functionality.

CCS Concepts: • **Computer systems organization → Dependable and fault-tolerant systems and networks;**

Additional Key Words and Phrases: Survey, blockchain interoperability, standards, interconnected DLT networks, cross-chain transactions, cross-blockchain communication, DLT infrastructure, interoperability assessment framework

ACM Reference format:
Rafael Belchior, Luke Riley, Thomas Hardjono, André Vasconcelos, and Miguel Correia. 2023. Do You Need a Distributed Ledger Technology Interoperability Solution?. *Distrib. Ledger Technol.* 2, 1, Article 1 (March 2023), 37 pages.
https://doi.org/10.1145/3564532

Work done while the Rafael Belchior was at Quant Network.
This work was partially supported by national funds through Fundação para a Ciência e a Tecnologia (FCT) with reference UIDB/50021/2020 (INESC-ID), and 2020.06837. B.D.R. was supported by Quant.
Authors' addresses: R. Belchior, INESC-ID, Instituto Superior Técnico, Universidade de Lisboa, Rua Alves Redol, 9, Lisboa 1000-029, Portugal and Quant Network, 20-22 Wenlock Road, London N1-7GU, United Kingdom; email: rafael.belchior@tecnico.ulisboa.pt; L. Riley, Quant Network, 20-22 Wenlock Road, London N1-7GU, United Kingdom; email: luke.riley@quant.network; T. Hardjono, Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, MA 02139; email: hardjono@mit.edu; A. Vasconcelos and M. Correia, INESC-ID, Instituto Superior Técnico, Universidade de Lisboa, Rua Alves Redol, 9, Lisboa 1000-029, Portugal; emails: {andre.vasconcelos, miguel.p.correia}@tecnico.ulisboa.pt.

2769-6472/2023/03-ART1 $15.00
https://doi.org/10.1145/3564532

1 INTRODUCTION

Before a technology unlocks its full range of applications, it first undergoes underestimation. **Distributed Ledger Technology (DLT)**, including blockchain, is no exception and is here to stay. A DLT (distributed ledger) or blockchain[1] implements a ledger that is shared across a network of nodes. Each node can create, broadcast, and validate *transactions*, which modify the distributed ledgers' state. DLTs typically provide support to run *smart contracts*, computer programs whose output is recorded on the ledger state. Smart contracts are triggered by transactions, which are recorded on the ledger. Nodes agree on the validity and ordering of transactions via a *consensus mechanism*. Typically, this environment provides transparency, tamper-resistance, and auditability of ledger information, providing desirable features that can alleviate some of the problems of the "centralized world."

Dozens of distributed ledger technology and blockchain systems [50] give rise to hundreds of blockchains [31] that, in its turn, support thousands of cryptocurrencies [31]. In the second quarter of 2021, decentralized exchanges (also called *automated market makers*) alone recorded a volume of $343 billion [34]. Along with Coinbase's total trading volume of $335 billion, the trends toward using blockchain for finance are increasing.

Payment networks, **central bank digital currencies (CBDCs)** and **decentralized finance (DeFi)** applications are already being leveraged by multiple players, such as (centralized and decentralized) hedge funds [130]. El Salvador adopted Bitcoin as a legal tender in June 2021. Several dozen projects on central bank digital currencies, including the Digital Pound consortium and the European Central Bank's Digital Euro [91] are displaying the increasing need for digitizing money [59]. Adoption seems inevitable as the world's financial ecosystems evolve [84]. Research suggests that the market for applications using DLTs will grow, with many organizations stating that blockchain is a critical priority [62, 102, 132], due to, for example, cost reduction. A recent report from Gartner predicts that "by 2023, 35% of enterprise blockchain applications will integrate with decentralized applications and services" [85]. Many blockchain ecosystems invest and promote projects that advance knowledge in cryptocurrencies [21, 57] and blockchain open source research [68], bringing more adoption to the space.

Thus, blockchain is slowly but steadily becoming an infrastructure for global value exchange and distributed computation [77]. However, blockchains have been created as standalone networks, as autonomy from most external systems was sufficient for the first applications.

Moreover, the need to securely and seamlessly connect DLTs (integration) is still an open problem [12, 131, 133]. Connecting those blockchains and making them cooperate (i.e., achieving interoperability [30]) have a practical utility and importance [12, 133, 134]. It allows communication between systems to exchange data and assets (fungible and non-fungible), leading to a higher heterogeneity of solutions in the market, synergies between projects, and higher liquidity to end-users. This way, no blockchain should become a single point of failure. Digital identity, supply chain, healthcare, voting [39], and CBDCs [12, 130] are just a few use cases benefiting from a multiple blockchain approach. We believe that blockchain will be adopted *en masse* when blockchains can use the capabilities of other systems in a unified approach [30].

Key Takeway 1. *Integrating blockchains*

Several blockchain projects already have embedded integration capabilities with other projects (e.g., Hyperledger Fabric can execute Solidity smart contracts). However, these existing integrations do not imply interoperability, as reutilizing functionality of a system does not imply cooperation across systems.

To connect DLTs and centralized systems, one needs blockchain interoperability techniques. A *centralized system* can still be distributed, typically for scaling purposes, but its components are trusted and operate under the

[1] We use the two terms interchangeably to mean a system with the characteristics explained in the rest of the paragraph. As a data structure, a blockchain is a distributed ledger but the opposite is not true. The reader is assumed to understand blockchain basics. For some references, please refer to [35, 84].

umbrella of one authority. More concretely, it is a system where the state consensus is decided by a single party or multiple parties under the same authority. A *decentralized system* is a distributed system where various parties control different components of the distributed system, and no party is fully trusted by all. In our context, we can consider a decentralized system to be a system where the state consensus is decided by conflicting or competing multiple parties, where accountability (from an external viewer's point of view) of individual decisions is *assured*. Each party composing the system can vote autonomously and has different incentives from other parties.

Interoperability allows a set of systems to cooperate, to achieve a common goal [66]—it is "the ability of two or more systems to cooperate despite differences in language, interface, and execution platforms" [128]. Studied since the 1980s [61, 66, 86, 123, 128], interoperability plays a major role connecting information systems. In this article, we propose a framework to assess the maturity of a DLT-based application to adapt to other systems (potentiality), its interoperation capabilities (compatibility), and its performance.

More recently, over a dozen academic papers surveyed the state of blockchain interoperability, identifying a few dozen solutions (for an updated list, see page 10 of [12]). In those surveys, examples of interoperation between networks of the same and different technologies are studied. Findings show that integrating multiple blockchains allows enterprise systems to be connected to DLTs and enables the creation of multi-ledger decentralized applications. Those applications can ideally run arbitrary cross-chain logic across DLTs. Cross-chain logic (or cross-chain rules) can be executed against a pair of homogeneous DLTs (a pair of DLTs running the same DLT protocol) or heterogeneous DLTs (a pair of DLTs running different DLT protocols). Interoperating heterogeneous blockchains is complex, as there may be differences in the underlying cryptographic primitives, data models, consensus models, privacy assumptions, integration capabilities, and others.

Key Takeway 2. *Emergent Solutions*

General-purpose blockchain interoperability solutions are still relatively unexplored, where complex logic can be programmed across chains.

Despite recent evolutions connecting homogeneous blockchains, many unsolved challenges in blockchain interoperability theory and practice are exacerbated by the lack of standardization among APIs, data models, and processes. Thus, integrating with different DLTs is an error-prone and tedious task [46]. This is one of the reasons why it is still difficult for centralized systems to exchange assets with blockchains, despite advances in developing higher-level APIs that simplify this process [11, 118, 124]. Exchanging assets between blockchains comes with critical challenges, where we highlight security: cross-chain protocol security flaws have already resulted in the loss of hundreds of millions of dollars [54] in 2021 alone. Given a set of security, scalability, and decentralization requirements, choosing the right blockchain interoperability solution can help prevent attacks, diminish costs, and bring products to the market faster. This work proposes to support the choice of an ***interoperability mechanism*** (**IM**), also known as interoperability solution. Specifically, we support the choice of the infrastructure and functionality of the IM, adjusted to specific project needs.

Key Takeway 3. *Interoperability is not binary*

To provide a better support for enterprise collaboration, synergies, and a richer ecosystem, integration processes should be verified and improved [36, 70]. Thus, integration is not a final step, but rather a continuous process that is also subject to change. Interoperability assessment tools exist for one to position a system in terms of how interoperable it is.

Research Questions and Contributions

Next, we present four fundamental interoperability questions that guide our research in this article. They are as follows.

Research Question 1. *What is the status quo of DLT interoperability?*

Recent years have seen extensive work on IMs. Several surveys condensed that knowledge, focusing on public connectors (connecting public blockchains) [16, 24, 76, 115, 119, 137], architecture for blockchains [126], and others [13, 72, 73, 111, 120]. Some surveys provide a systematic overview of the area, showcasing more modern interoperability solutions [12, 87].

However, the classification of solutions is typically inconsistent across surveys, making it challenging for researchers to consistently evaluate available options. What is a consistent classification framework that can improve past work and improve understanding when classifying solutions? Furthermore, we aim to clarify theoretical contributions to the DLT interoperability research area, including the current capabilities, components, connection modes, interoperation modes, practical applications, limitations, and strong points. This guide can prove helpful to researchers and practitioners by providing a mental model of existing interoperability solutions.

Contribution: a unified conceptual model and classification framework for blockchain interoperability solutions.

Research Question 2. *How to assess the interoperability capabilities of an IM?*

Not all IMs provide the same interoperation capabilities. To measure it, one needs to consider several key questions. Measuring the maturity of a system to adapt to others requires asking *can the system interoperate with other systems as is?* and *is the system able to be changed to adapt to other systems?* Assessing interoperability between systems can be done by asking *how well can a pair of systems interoperate?* and *what are the current problems or barriers that prevent the systems from interoperating better?* Finally, measuring performance requires studying cross-chain latency, cross-chain throughput, and cross-chain costs associated with an IM.

Contribution: a framework to assess the interoperability capabilities of a system utilizing multiple DLTs (in terms of potentiality, compatibility, and performance), based on conceptual models [44, 70].

Research Question 3. *How to choose an IM?*

This research question concerns a problem posed by academics and practitioners alike: does my project need an IM solution? What is the most suited IM given specific requirements? We build on top of the proposed classification and interoperability assessment to answer these questions, presenting a framework for choosing an appropriate blockchain interoperability solution, both from the infrastructure and the functionality perspectives.

Contribution: a framework that allows one to choose a blockchain interoperability solution, considering a set of criteria defined by our blockchain interoperability model.

Structure of the Article

In Section 2, we introduce the necessary background to read and understand this article, including background on DLT interoperability, and examples that motivate DLT interoperability research. After that, we present the current state of DLT interoperability, including its several layers and components, and our model in Section 3. In the same section, we present the interoperation modes, the connection modes, and the (IM) solution categories. Section 4 presents our framework for assessing the interoperability of a DLT-based solution. After that, we present a decision model for choosing an IM for a DLT project based on the infrastructure and functionality of the IM, and two concrete examples. Section 5 presents the related work and future research challenges. Finally, Section 6 concludes the article.

2 PRELIMINARIES

In this section, we present the background and motivating examples.

Table 1. Examples of DLT Networks, and Its Respective DLT Protocol and Subnetworks

DLT Protocol	DLT Network	DLT subnetwork
Hyperledger Fabric 2.3	Carbon Emission Network (Hyperledger Fabric)	Carbon emission channel
Hyperledger Fabric 2.3	Carbon Emission Network (Hyperledger Fabric)	Travel channel
Ethereum 1.0 (geth version Faryar v1.10.15)	Ethereum Mainnet	Ethereum Mainnet
Ethereum 1.0 (Besu 21.10.6)	Ethereum Mainnet	Ethereum Mainnet
Ethereum 1.0 (geth version Faryar v1.10.15)	Ethereum Testnet Ropsten	Ethereum Testnet Ropsten
Ethereum 1.0 Private Network (Besu 21.10.6)	Private Ethereum Network	Privacy group 1
Ethereum 1.0 Private Network (Besu 21.10.6)	Private Ethereum Network	Privacy group 2
Polkadot 0.9.14	Polkadot	Relay chain
Polkadot 0.9.14	Polkadot	Parachain 1

2.1 Blockchain and Interoperability

Distributed Ledger Technologies, such as Hyperledger Fabric, Corda, and Ethereum implement DLT protocols. Each DLT protocol is defined by its protocol version, e.g., Hyperledger Fabric v2.3 [5], Corda v4 [20], or Ethereum London Hard Fork [23]. These technologies and version combinations describe how each DLT state can be updated via transactions and the specific protocol that the nodes follow to come to agreement on the DLT state, as Figure 1(a) illustrates.

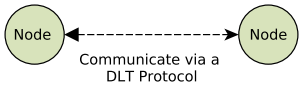
DLT Networks and Subnetworks. *DLT protocols* can be instantiated in *DLT networks* (Table 1). DLT networks are groups of DLT nodes that make up a DLT system [71]. For instance, Hyperledger Fabric might be instantiated in a DLT network composed by an enterprise consortium. The Ethereum mainnet, Bitcoin mainnet, and Substrate-based networks, such as Polkadot, Kusama, and Rococo [105] are other examples. DLT networks are typically called Layer-1 DLTs.

Each DLT network can be partitioned into *subnetworks*. Nodes of a subnetwork contain a logically separated state compared to another subnetwork [71].

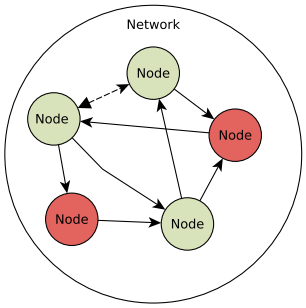
Each subnetwork may offer different functionalities (e.g., data isolation, processing capabilities, governance) and security properties (e.g., partial consistency vs. consistency, better confidentiality, and so on). At least one node of each subnetwork must connect to another node of another subnetwork for these two subnetworks to be contained within the same DLT network.

In Hyperledger Fabric, a subnetwork corresponds to a *channel*. Channels isolate execution environments and data from other channels belonging to the same Fabric network. Polkadot's Parachains could be considered subnetworks of the Polkadot network.

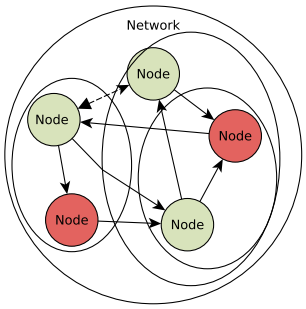
A *DLT network can therefore have multiple subnetworks*. If the DLT network state cannot be divided into multiple subnetworks, for the sake of simplicity of our evaluation, we say this DLT network has one subnetwork. Any node in a DLT subnetwork is also a node of the DLT network, implying that DLT network nodes and DLT subnetwork nodes must be running the same DLT protocol. In *permissionless* DLT networks, every compatible DLT node can join the network. In contrast, in *permissioned* DLT networks, only compatible DLT nodes with permissions can join the network, where each DLT node may be assigned a particular role restricting the functions it can perform. There are two subcategories of permissioned DLT networks: *private permissioned* DLT networks, where DLT nodes do not provide public access to the data contained in the distributed ledger; and



(a) DLT nodes run a DLT protocol. Different node implementations can communicate if they run the same DLT protocol and have the same DLT network configuration.



(b) DLT nodes can have different implementations of the same protocol (represented by different colors). A DLT network is a set of nodes connected in a mesh network manner.



(c) Networks can be organized into subnetworks. Nodes of each subnetwork share a partial view of the distributed ledger. Nodes can be in one or more subnetworks.

Fig. 1. Comparison between DLT nodes, DLT networks, and DLT subnetworks.

public permissioned DLT networks, where DLT nodes do provide public access to the data, such as via block explorers.

In permissionless DLT, anyone can run a node that interacts with the network; if it is permissioned, only nodes with permissions can access the network. Within a network boundary, a DLT stores the ledger state (which could be organized as a key-value store, such as the world state in Fabric [5], via the account model such as in Ethereum [23] or via a UTXO model such as in Bitcoin [96]), and have an identity management mechanism. Identities are then mapped to permissions that encode what each node can do on the network (in terms of reads and writes). Each identity typically has two main keys (public and private keys), an address, and a low-level storage. Nodes can perform updates to the ledger via *transactions*.

Transactions are "the smallest unit of a work process related to interactions with distributed ledgers" [71], that, parametrized and signed by its creator, can be issued against a smart contract, via a DLT node. Generally speaking (as different DLT technologies have different transaction lifecycles), transactions are sent to other DLT nodes via a network propagation protocol, such as a Gossip [35].

Internal Mechanisms. DLT nodes in the same DLT network will "gossip" to each other various messages, such as transactions to update the distributed ledger. DLT nodes agree on the order of transactions (and its content) to update the distributed ledger, by following a set of rules and procedures defined in a *consensus mechanism*. Typically, DLT networks have an *anti-sybil* component so that individual DLT nodes cannot replicate themselves to unfairly increase their influence on how the entire network reaches consensus. Some DLTs may allow for selectable consensus mechanisms (usually selectable only upon the genesis of the DLT network), such as with Ethereum or DLTs created with the Substrate framework. In contrast, other DLTs like Bitcoin have a hardcoded consensus mechanism.

How transactions affecting the distributed ledger are ordered gives different DLT types. A *blockchain* requires transactions to be grouped together in blocks, each block to be cryptographically linked to one previous valid block (a block that includes transactions that have modified the distributed ledger), and each DLT node must process each block in sequential order. In contrast, a ***directed acyclic graph (DAG)*** does not group transactions into blocks. Instead, each transaction references other valid transactions (that have modified the distributed ledger), and each DLT node can process each transaction in different orders. Finally, a *block DAG* groups transactions into blocks, each block is cryptographically linked to other previous valid blocks, and each DLT node can process blocks in different orders.

As regards collections of DLT networks, we say we have heterogeneous DLT networks when the technologies and their respective networks are different; we say we have homogeneous DLT networks when only the networks are different.

Interoperability among DLT Networks. Different DLT networks can connect to other DLT networks. An IM, often called a bridge, can connect networks to other networks, subnetworks, or centralized systems. Figure 2 depicts the mental model on DLT networks, subnetworks, and interoperability mechanisms. DLT protocols instantiate DLT networks that, in its turn, can be connected to DLT subnetworks. Subnetworks are more concerned with a specific scope (specialization of the network), i.e., they can focus on scalability, achieved, for example, via a different state model; or features. This figure considers the Carbon Emission Network (implemented with Hyperledger Fabric v2 and the Ethereum main net). The Hyperledger Fabric network contains two channels (subnetworks) that can communicate with each other, but not natively. On the other hand, we consider the Substrate framework as the DLT protocol (or, more concretely, the SDK to generate blockchains), instantiated as the Polkadot network. Polkadot can connect to its subnetworks via the relay chain. A relay is a smart contract in a target blockchain that functions as a light client of a source blockchain. Light clients are network nodes that are solely part of the blockchain history, i.e., relevant transactions (vs. full clients that store the whole blockchain history). They can verify that a transaction was included in the blockchain, typically using block headers (e.g., via Merkle proofs [43]), but not validate it. Relay smart contract receives block headers from relayers, nodes that fetch blocks from the source blockchain, and gives them to the target blockchain. Relays behave like oracles (except that they have to process it instead of receiving the processed information). Blocks given to the relay smart contract can be contested by other relayers by presenting a Merkle tree proof. The relay chain acts as a relay, realizing the bridge between parachains. Each parachain can connect to the relay chain via a module called Cumulus, and send messages to other parachains by using a message format XCMP [107].

Vertical interoperability (from networks to subnetworks and vice versa) and horizontal interoperability (between subnetworks and between networks of different systems) compose the spectrum of interoperability covered by this article.

Horizontal interoperability can be implemented in a multitude of ways. Layer-2 solutions are independent DLT networks connected to other networks via interoperability mechanisms. They aim at solving scalability problems with the DLT networks they interoperate with. Scalability is enhanced by allowing the network to offload transaction processing and enabling new features from the original DLT network. Those subnetworks are pegged to the networks via cryptographic mechanisms [12, 115].

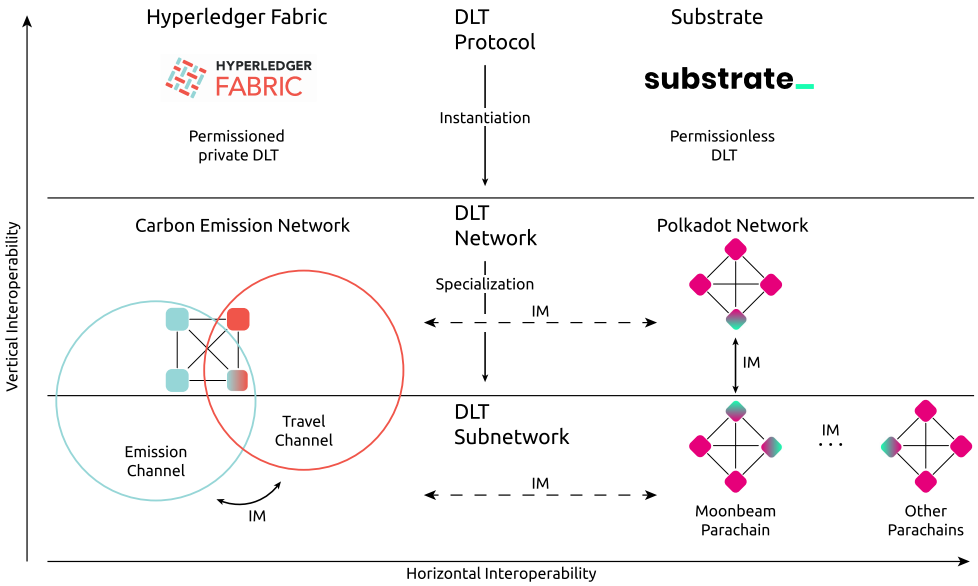


Fig. 2. DLT protocols, networks, and subnetworks.

Blockchain interoperability is challenging because it implies going beyond two different trust boundaries and establishing a new boundary. Network boundaries also influence state ownership: in a centralized system, the state is owned by a single party, and hence any party interoperating with such a system needs to trust it. A decentralized system, in its turn, defers the state ownership to the collective, where a protocol is used to update that state (and achieve *consensus*).

A new (trust) boundary is formed when two systems are interoperating. The trust assumptions of the new boundary can be lowered by systems to provide proof of state for a blockchain view[2] [3, 9]. The new boundary needs to assume that each ledger is secure. Security depends on the threat model and network assumptions. Generally, security properties such as safety assume an honest majority of participants and arbitrary Byzantine behavior. Furthermore, given those assumptions, each ledger can provide liveness and persistence [48]. Liveness says that all transactions originated by honest parties will eventually be included in the blockchain; persistence states that once a transaction is included in the blockchain of an honest party, it will be included in all honest parties.

Key Takeway 4. *Interoperability requires a trusted third party*

Cross-chain communication requires a trusted third party [137]. However, a trusted third party can be centralized or decentralized, being an example of the latter, a blockchain (whereby its consensus is used as an abstraction for a trusted third party) [12, 137]. A prerequisite for the system to be a trusted third party is its safety, i.e., common prefix and chain quality [48]. A decentralized interoperability solution implies the usage of a blockchain consensus as a trust anchor for the solution.

Different trust assumptions exist for each ledger, e.g., at least one honest node in Hyperledger Fabric, or the majority of the computational power, in the case of proof of work blockchains such as Bitcoin. Thus, when

[2]Stemming from the business process view integration research area, studying the creation, merging, and processing of views [9].

choosing an interoperability solution, the users who participate in the origin DLT must trust the involved DLTs and the IM. Ideally, both should be decentralized [115].

DLT as a System Component. Several studies have presented blockchain as an infrastructure for data storage and computation [88, 136]. Blockchain can be viewed as a system component that eases trust assumptions between mutually untrusting participants. DLTs can be accessed by centralized systems—and thus need software components that provide the necessary infrastructure for connecting with it (key management, secure connection, state storage). Interoperability across DLTs implies the existence of another middleware layer (another system component) that can bridge nodes.

2.2 Multiple DLT Decentralized Applications

DLT use cases are already in production, creating value [12]. As enterprises integrate blockchain in their business processes, the requirements will bring the need to use several types of DLTs. **Decentralized applications** (**dApps**) will then need to utilize multiple DLTs as their infrastructure, because one DLT cannot cover all use cases (i.e., offer the same functionality although there are different tradeoffs in security, scalability, and decentralization). We highlight two use cases that demonstrate the importance of this field, implemented by **multiple DLT decentralized applications** (**mDApps**).

Carbon Emissions. The first example is Hyperledger's Cactus implementation of the Carbon Emission App from the Hyperledger Carbon Accounting and Neutrality Working Group [28]. A detailed explanation of this use case can be found in Hyperledger [27]. The purpose of this use case is to reward carbon emission reduction by orchestrating heterogeneous blockchains: one focused on data collecting, and another on the reward incentives.

A Hyperledger Fabric network collects emission records (activity data), e.g., energy consumption, travel mileage, and widgets produced. The emissions records are not continuous because both the emissions factors and the data for calculating emissions are based on long time windows (e.g., utility bills are produced each month). Periodically, the activity is aggregated to be later converted to an emission token (ERC-721). Emission tokens are created on Ethereum's public network from the collected data on Fabric to be traded against allowances that reward emission reduction. Figure 3 depicts this network.

This example contemplates a private, permissioned ledger used for performance and privacy reasons, but where the final output (carbon emission tokens) are stored in public blockchains as a reward. In particular, the performance of Hyperledger Fabric in terms of throughput and end-to-end latency is superior to most public blockchains due to its consensus and low number of peers. Privacy can be assured because only the peers involved can read the global state or if needed, only a subset of peers could read part of the global state (i.e., by utilizing channels or private data).

LACChain. LACChain [79] is a Global Alliance for the Development of the Blockchain Ecosystem in Latin America and in the Caribbean, led by the Innovation Laboratory of the **Inter-American Development Bank Group** (**IDB LAB**) in cooperation with partners and strategic allies.

LACChain aims to provide infrastructure and technical tools, on top of the three layers the LACChain network comprises (DLT, self-sovereign identity, and tokenized money) that are useful for developing applications with social impact that contribute to the development of the countries of the region.

LACChain aims to provide a community and a general-purpose infrastructure for the realization of several use cases, such as supply chain, cross-border payments, and financial inclusion. This project currently utilizes two blockchain technologies, Ethereum and EOS [114].

Quant [113] and LACChain are piloting a project to provide private currency payments between retail customers from different financial institutions. This prototype involves retail customers creating transactions on a public permissioned Ethereum network. These payment amounts between the customers are tallied, even though the identities involved in the transactions are hidden via zero-knowledge proof technology. As all payments are

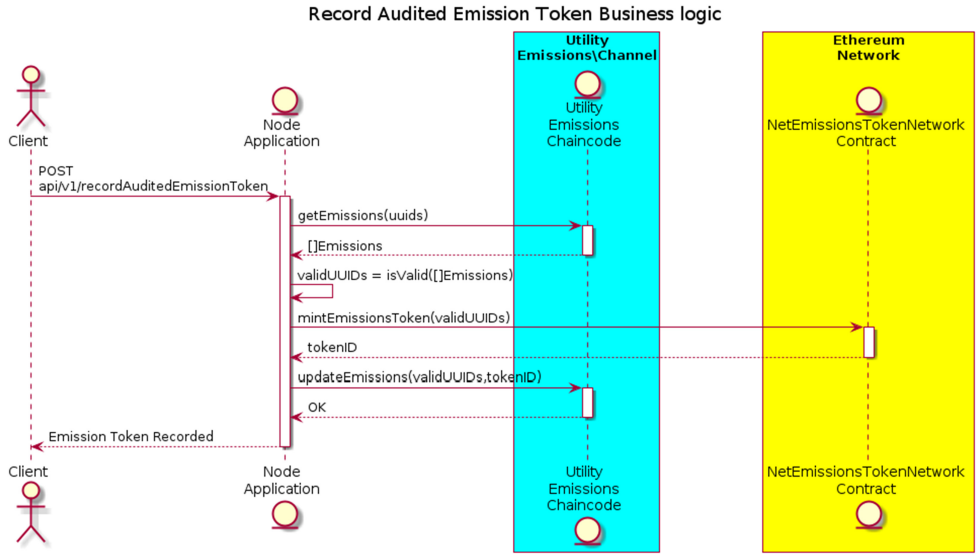


Fig. 3. Sequence diagram of the Carbon Emission use case.

recorded, currency exchanges between financial institutions can be netted and settled efficiently on a separate private permissioned Ethereum network. The blockchain interoperability solution infrastructure to allow this netting and settlement to occur is Quant's Overledger, where the related interoperability applications are built on top of it utilizing Overledger's cross-DLT standardized data model.

3 STATUS QUO OF BLOCKCHAIN INTEROPERABILITY

Throughout this article, we summarized and built on top of existing knowledge of blockchain interoperability, including existing solutions, challenges, and opportunities.

3.1 Interoperability Layers

This section studies the main interoperability layers based on existing interoperability platforms: we pave the way for systematically analyzing IMs.

Interoperability among computer systems is typically defined in terms of several layers [66]. Although it is possible to come with a detailed architecture for interoperability applications, interoperability has different meanings (and thus uses different techniques) depending on its domain (e.g., for European Union states [44], for language resources [66], supply chain [29], governments [56], and others). Hence, we adopt the European Interoperability Framework model [44] from the European Commission. This model is based on four layers, as depicted in Figure 4.

— Technical interoperability: links systems and services by adopting compatible data formats, communication protocols, interface specifications, and integration services [44]. Information exchange is achieved with technical interoperability, but there are no guarantees on how the received information is interpreted.
— Semantic interoperability: exists when systems can interpret information following a defined ontology (i.e., following a well-known model for information). As a consequence, information from one system can be interpreted in another. Some prerequisites of this type of interoperability are agreements (or conventions) on data formats. Protocol messages (and the protocols themselves) and the representation of assets are

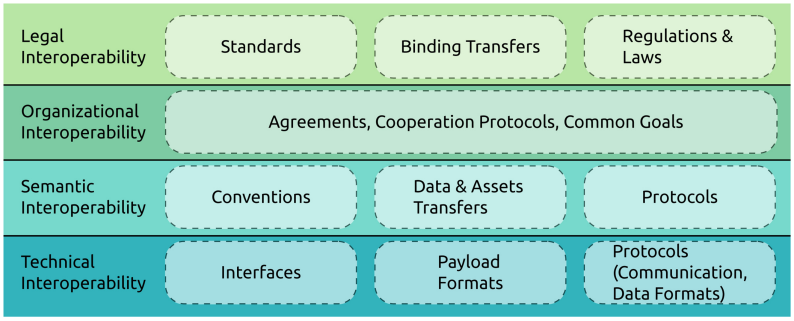


Fig. 4. Interoperability layers (see [44]).

part of this layer. Thus, semantic interoperability subsumes information syntax (what is the information format?) and information semantics (what does the information mean?).
— Organizational interoperability: concerns aligning the requirements and interests of the user community by leveraging cooperation and integration of business processes between organizations via arrangements and protocols, typically under a formal or semi-formal deal.
— Legal interoperability: ensures organizations can cooperate under "different legal frameworks, policies and strategies" [44]. This includes a certain degree of coherence between legislations so that the assets managed under the semantic interoperability layer can be managed consistently.

The orchestration of the four layers (arguably, at least the first two) could lead to a seamless integration of DLTs, leading to value exchange. For instance, if there is no regulatory framework that ensures the validity and legality of a cross-jurisdiction asset transfer (at the legal interoperability layer), organizations may not cooperate seamlessly (organizational interoperability layer). These incompatibilities affect different viewpoints, according to stakeholders' *concerns* [36]. Four concerns are proposed on the **Framework for Enterprise Interoperability (FEI)** [70]. The *business concern* regards barriers in organizations to cooperate despite differences in the decision-making process. The *process concern* regards how various artifacts that support the business (processes) work together. The *services concern* identifies the applications and their interfaces that support processes. Finally, the *data concern* regards data management from different supports. Each concern is related to all interoperability layers, and each layer is related to one another (typically following a bottom-top approach).

It is worth noting that other frameworks are equally valid, such as the Cloud Interoperability Standard ISO/IEC 19941:2017, being currently studied by ISO's WG7 (interoperability). In this framework, three layers exist: technical, business, and governance. For the sake of granularity, we choose the European Interoperability Framework model.

As an example, let us consider a cross-jurisdiction DLT-backed asset transfer [10]. Technical interoperability allows exchanging bytes across systems; semantic interoperability allows exchanging the asset—running a protocol creating entries representing ownership on both ledgers. Organizational interoperability concerns the deal between institutions that want to arrange digital asset transfers. Legal interoperability assures the validity/legal character of the asset. The last layer requires coordination between legal frameworks and, possibly, between the interoperability solution and the current regulatory framework.

Key Takeaway 5. *Legal and organizational efforts are lagging behind*

The lack of legal interoperability, in the form of standards and IMs hinders the development of dApps. Although governments and enterprises are interested in the technology alike, there is a gap between its potential adoption [12].

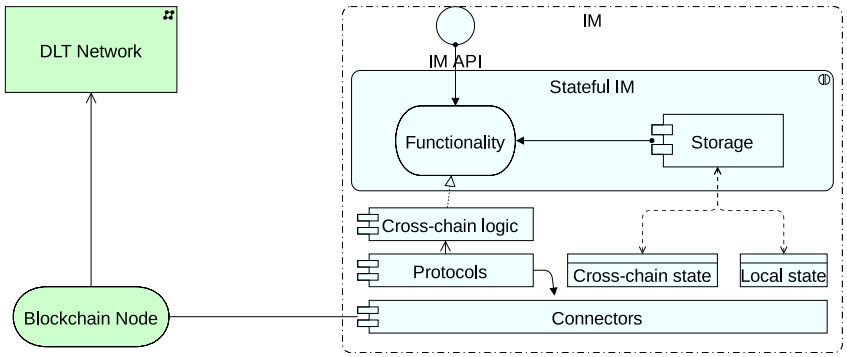


Fig. 5. Representation of a general-purpose blockchain interoperability solution in the Archimate modeling language [122].

3.2 A Model for Interoperability Solutions

In this section, we present a generic interoperability framework, as the blueprint to design an IM.

When connecting an application to one or more DLT networks, a developer (or software architect) has a few choices. Firstly the application could connect directly to a DLT node. Secondly, the application could connect to multiple DLT nodes, but then the complexity of managing the routing between the DLT nodes needs to be contained within the application. Instead and thirdly, the application could connect to a DLT node proxy which handles the routing and load balancing issues creating logical separation. The final option for an application is to connect to a DLT network via a DLT gateway, or a combination of gateways. DLT Gateways can implement data and asset transfers, as well as asset exchanges and come in many forms.

An IM is an application (classified as an oracle or cross-authentication, can be executed off-chain, on-chain, or both) that includes a connection mode (e.g., a DLT Gateway) and performs an interoperation mode. IMs provide access to its functionality via an API (we defer a formal definition of an IM for future work). The functionality an IM provides is to execute cross-chain logic via a set of protocols. The processing occurring in an IM can be persisted in a storage, composed of a local state and a cross-chain state. The local state stores all relevant data for local computation (e.g., processed output from business logic plugins, logs) and cross-chain state (joint state representing relevant computations performed over multiple systems).

Figure 5 illustrates a model for interoperability solutions. The IM exposes its functionality via a set of APIs. The APIs redirect the requests to the responsible module handling specific functionality upon being invoked. Those modules are called cross-chain rules, cross-chain logic, or **business logic plugins** (**BLPs**). Cross-chain logic modules process the request, translating it into transactions or requests to external systems, including DLTs. This processing can be persisted in storage. Protocols support the execution of cross-chain logic by acting as a middleware layer between high-level logic and specific interactions with other machines or DLT transactions. Eventually, the interaction with target nodes (either DLT nodes or other IMs) receives a response back, which is processed and optionally persisted. The processed responses can be redirected to an external system. A cross-chain state can be built from executing cross-chain logic to implementing cross-chain protocols. That state can be shared with multiple instances of the same IM, or another one.

This conceptual architecture effectively implements the technical and semantic layers of an IM. The organizational layer comprises how organizations cooperate across trust boundaries to achieve common goals in agreements valid on that trust boundary. In the newly formed trusted boundary (trust boundaries 1 and 2), the IM can interoperate with other systems (e.g., centralized systems) following specific protocols that realize cross-boundary cooperations. The legal layer applies to all trust boundaries. In particular, the applicable law

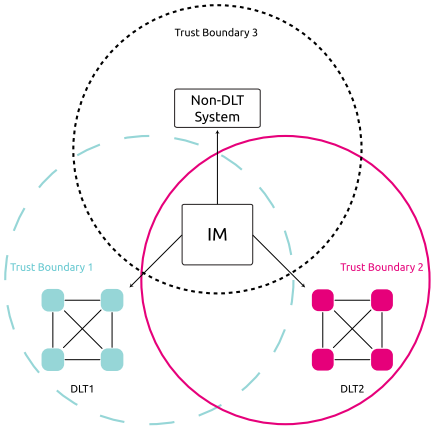


Fig. 6. Trust boundaries between a non-DLT system, two different DLT networks, and an IM.

varies according to the specific jurisdiction and norms. Figure 6 represents an IM connecting three different trust boundaries.

We call an IM *trustless* if two conditions hold:

- *verifiable correctness*: there is a method to check that the IM runs a well-defined *functionality* (e.g., protocol, arbitrary business logic) among chains. Misbehavior in the executing environment can be detected and thus held accountable. This implies that actions performed by the IM need to be stored, preferably in a public forum.
- *eventual data consistency*: all verified instances of the IM will eventually return the same result for any specific function call with the same input. However, a single IM may return to the user the result of a cross-chain transaction only when it is finalized, i.e., all sub-transactions have been committed. This stronger consistency guarantee can be provided at the expense of latency.

3.3 Blockchain Interoperability Solutions

Choosing a blockchain interoperability solution requires asking at least two questions: "what do you want to connect?" and "how does the interoperability solution connect the systems?"

3.3.1 Interoperation Mode. The "what" question concerns the artifact managed by the blockchain interoperability solution, i.e., the interoperation mode. The artifact exchanged can be data or assets. Data are arbitrary byte strings representing a piece of information on the blockchain (technical layer). It could be a key-value pair, metadata about the blockchain. Data can be copied from blockchain to blockchain.

Assets can be represented in the technical layer (by a string, for example). However, in the semantic layer, they "take form" by representing a fungible or non-fungible value which is or is not linked to a physical identity (in case it is, it is called a *digital twin* [110]). Therefore, they should not be copied among DLT networks but rather be *transferred under specific conditions*. More specifically, an asset transfer should abide by the rules of each DLT (e.g., no double spend), i.e., preserve at all moments the invariants of all the DLTs it affects. In DLTs, double spend can occur when an attacker sends tokens for a pending payment in a transaction to a victim in return for a product. The victim releases the product. Then, the attacker cancels the pending payment transaction, such that the original token transfer does not become recorded on the blockchain ledger (and thus their tokens are preserved). In the context of interoperability, double spend can happen in cross-chain asset transfers, when a lock/burn on the source DLT (the DLT in which the transaction is initiated to be executed on a recipient

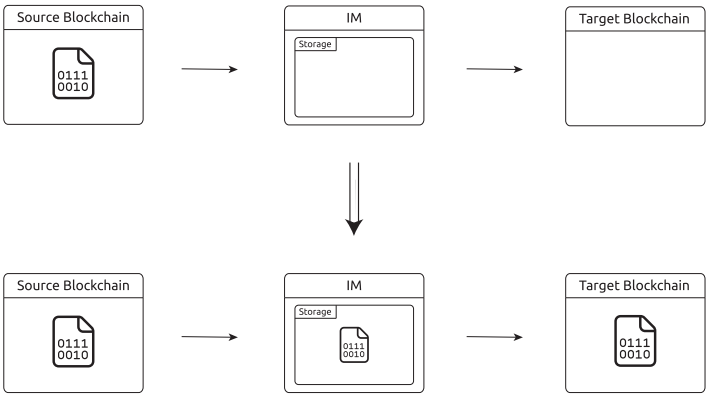


Fig. 7. Data transfer between two DLTs.

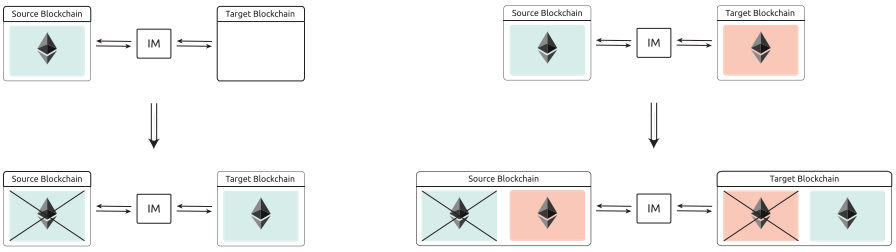
DLT [12, 58]) was not performed. The same representation of an asset could be valid in several DLTs, leading to a *new type of double spend*. Solving this problem implies synchronizing DLTs at the semantic layer, i.e., the involved blockchain interoperability solutions need to run the same protocol that prevents invariants from being violated, with on-chain notarization (e.g., via smart contracts).

The interoperation modes are as follows:

— *Data Transfer*: data is copied from one DLT to another, with an optional intermediate processing step. For example, copying price information from one DLT into another [3].
— *Asset Transfer*: unilateral or bilateral asset transfers. Assets are transferred from one DLT network to another (implies burning or locking the asset on the source DLT network). Tokens that are minted are typically called wrapped tokens [52], because its value is anchored on another asset. For example, locking Bitcoin to a multi-signature address on the bitcoin blockchain and minting a representative asset on the Ethereum blockchain (such as wBTC [97]).
— Asset Exchange: atomic asset transfers. Assets are exchanged in their respective DLT network, i.e., no transfers across DLT networks occur. Participants need to be present in both chains for this exchange to happen. For example, swapping promissory notes across two different distributed ledger systems between two users where both assets just change the address on their native DLT network [10].

Figure 7 represents a data transfer from the source blockchain to the target blockchain. A blockchain interoperability solution requests data from the source blockchain and writes it on the target blockchain. Data can be copied. As blockchains increasingly comprise more value, represented by assets, value transfer among blockchains needs to be handled carefully, as it exposes a new class of attacks: cross-chain attacks. In cross-chain attacks, attackers attempt to double spend an asset by manipulating cross-chain protocols. Assets are then more sensitive to manage in terms of interoperability.

Figure 8(a) represents an asset transfer from the source blockchain to the target blockchain. A blockchain interoperability solution requests data from the source blockchain and writes it on the target blockchain as an asset (semantic layer protocols are required). Thus, the IM needs to make sure the representation of the asset on the target blockchain is changed to used (or burned). This implies the IM needs to check for conformance on both DLTs. Figure 8(b) represents an asset exchange (bidirectional asset transfer, or two asset transfers) between both blockchains. Again, the IM needs to check that both blockchains are in a consistent state. Note that these figures are high-level and hide details. More detailed procedures show the rules for asset transfers in the next section.



(a) Unilateral asset transfer between two DLTs. (b) Bilateral asset transfer between two DLTs.

Fig. 8. Asset transfers.

Cross-chain asset exchanges can also be classified into two types, permanent and temporary:

— *Permanent asset exchange*: assets are exchanged between parties with no obligation to reverse the exchange later (e.g., hash time lock contract swaps).[3]
— *Temporary asset exchange*: assets are exchanged between parties where the conditions to reverse the swap are in place. Examples include a cross-ledger loan where a user places an asset into a smart contract on one chain for another asset to be borrowed on another chain [19].

In asset transfers, it is the responsibility of the interoperability solution to establish a new boundary of trust for both previously established boundaries. Since one asset transfer will typically involve several transactions (from the source and target blockchains), it is desirable to do so via atomic cross-chain transactions. Atomicity is desirable because, in the case not all transactions are completed, the union of systems might be left in an inconsistent state (although several solutions exist, such as rollback [11]).

3.3.2 Connection Modes. There are three methods for a dApp or mdApp to connect to a DLT. These mechanisms are called the *connection modes*. They are as follows:

— **DLT Nodes:** DLT nodes are the software systems that run a DLT protocol. The application could connect directly to a DLT node. While anyone can run their single DLT node, this is not crash resilience and not scalable from a load balancing perspective.[4] Example: an Ethereum node being run locally (Geth client).
— **DLT Proxy:** a DLT node proxy manages the routing and load balancing issues between an application and one or more DLT nodes, creating logical separation. To an application, interacting with a DLT node proxy is nearly identical to interacting with a DLT node as the message requests and responses will be virtually the same. The only possible difference may be identifying metadata in the messages to track the DLT node proxy users (e.g., for rate limiting reasons). Examples: a group of permissionless network nodes runs on Kubernetes (self-hosted or run by a third party). Some enterprises provide a DLT proxy, such as Infura's Ethereum DLT node as a service [67], or Blockdaemon [15].
— **DLT Gateways:** Like a DLT node proxy, a DLT gateway also manages the routing and load balancing issues between an application and one or more DLT nodes, creating logical separation. Example: Polkadot's block explorer; Self-hosted ODAP gateways; Quant Network's Overledger.

[3] A hash lock is an artifact that requires a preimage of a hash to trigger behavior. More concretely, a hash lock protocol relies on the preimage resistance property of a hash function H, such that hash = H(secret). A timelock is an artifact that triggers the ending of a protocol when a certain time has passed (e.g., in terms of the number of blocks). A hash lock time contract combines these concepts to realize a timed, programmable escrow supported by a DLT.
[4] Some enterprises provide a DLT Proxy service (node as a service) to solve this problem.

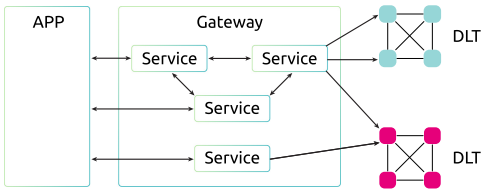


Fig. 9. Gateway architecture. A gateway provides a set of services to a client application, while connecting to different DLT networks.

DLT gateways are not DLT nodes. Instead, they are a collection of services built around DLT nodes, as Figure 9 shows. The services it provides (i.e., the protocols it runs) and identity management, access control, security, and liveness properties are left to be instantiated by the DLT gateway administrator. An example of a gateway is the ODAP Gateway [11, 60]. DLT proxies can also promote geographical diversification, alleviating some known blockchain cyberattacks (e.g., eclipse attack). However, it could be desirable to run non-native business logic from DLT nodes.

A DLT gateway provides additional non-DLT functionalities. Such additional functionality can include data analytics or complicated cross DLT network processes.

It could be desirable to use a DLT gateway instead of a DLT node or DLT node proxy if the gateway's additional services are crucial to your application or to smooth your application build process. For instance, a DLT gateway could utilize a standardized data model, meaning that, unlike DLT node proxies, a DLT gateway can be used to connect to many DLT networks of multiple DLT types. On the other hand, a DLT node proxy could be more desirable if the application developer is experienced and very familiar with the underlying APIs and data models of the DLT nodes.

Both DLT gateways and DLT node proxies can promote geographical node diversification. Additionally, DLT gateways and DLT node proxies can promote technical node diversification (e.g., running multiple different DLT node implementations for a particular DLT, such as geth and nethermind Ethereum nodes). This minimizes the risk of application failure if there is a bug in a particular node implementation.

However, it could be desirable to connect an application directly to a DLT node, for permissioned DLT networks where the attack vector of DLT nodes is significantly less and therefore running a DLT node in a container orchestration system (e.g., Kubernetes) would suffice.

3.3.3 Solution Categories. Categorization of DLT interoperability solutions attempt at answering *How is interoperation achieved?*, and what are the trust boundaries each solution creates. In the previous section, we presented the connection modes. In this section, we present a unified IM categorization that considers the connection mode, interoperation mode, and trust assumptions required by the solution category. Contrary to common knowledge, there are only two non-intersecting IM categories. All interoperability mechanisms are created from the following categories of solutions.

In particular, transferring data requires oracles, and transferring or exchanging assets can be done in two ways: (1) locking an asset in a source DLT, and creating its representation on a target blockchain, implying transactions on both DLTs—using oracles—or (2) via native transfer transactions (Alice transfers to Bob asset A in DLT X for asset B in DLT Y)—cross-authentication.

Solution Type 1 (Oracle). Oracle interoperability solutions allow a DLT system to make use of external data from another system [25, 41, 94], increasing the connectivity of DLT-based applications. There is a lot of ongoing research on the security and fairness of interoperability via oracles. In particular, oracles could be selecting certain transactions to be included in the target blockchain for its own benefit, similarly to the miner extractable

value problem [37]. We do not aim to cover such topics in this work, nor to present oracles in great detail, as this is well covered in the literature.

Code deployed into a distributed ledger cannot access external resources or data without the help of an intermediary. These intermediaries (known as oracles) gather external data, placing it into transactions that are subsequently added to the distributed ledger, therefore allowing this retrieved data to be read by deployed smart contracts. Note that oracles can fetch data from a non-DLT system or another DLT-system. Oracles are classified into two types [94]: pull-based or push-based.

The parties involved are the user and its smart contract (deployed on a DLT), the oracle and its smart contract, and external systems (decentralized, centralized).

— Pull-based oracle data transfers: upon request, those fetch data from off-chain systems and send the data to a DLT (via a transaction). These oracles operate differently depending on the transaction execution model of the DLT:

(1) order-validate-execute model (e.g., Ethereum): pull-based oracles on these DLT systems require multiple transactions to complete the data request process. An example would be the following: a smart contract issuing two transactions: (1) a transaction to the client smart contract which triggers a call to the oracle smart contract. This call details the data it wishes to obtain from an external system; (2) the oracle observes this request and creates a transaction with the necessary data, which is sent to the oracle smart contract called by the client. This way, the client smart contract now has the necessary information accessible via the oracle smart contract. This model could require one transaction if an oracle smart contract already holds the necessary data (requiring the oracle client to be pushing data periodically, i.e., the oracle is a push-based oracle).

    Figure 10 represents a pull-based oracle operating with an order-validate-execute DLT. The client smart contract calls the oracle smart contract with a request for information (step 1) the latter does not have (e.g., the request from the client smart contract includes a GET HTTP request). The transaction is recorded, and the oracle listens to transactions that call its smart contract, via an off-chain client (step 2). Upon recognizing them, it performs the requests by collecting information (steps 3–6). Upon eventual processing (step 7), the information is pushed to the oracle smart contract (step 8). The oracle smart contract now has the necessary information in its storage (or memory) (step 9). Finally, the oracle smart contract returns the information to the client smart contract (step 10).

(2) execute-order-validate model (e.g., Hyperledger Fabric): pull-based oracles on these DLT systems only require one transaction to complete the data request process as nodes involved in the execution and verification process perform the oracle functionality by fetching the data themselves. Before the transaction is confirmed, meaning that the external data can be immediately used by the requesting transaction (e.g., in the case of Hyperledger Fabric chaincode HTTP requests or Corda flows that include an Oracle node).

— Push-based oracle data transfers: obtain external data without an explicit request from a DLT transaction. These oracles usually add the data into smart contracts to be easily consumed by other smart contracts (via smart contract to smart contract calls).

Oracles inject information into a trusted network. This information can be used for decision-making by nodes, meaning oracles greatly influence the subsequent state of the network as a whole. It means that oracles are trusted third parties. The trust model of oracles might vary, from voting-based oracles to reputation-based oracles to a single trusted oracle [100]. The choice of an oracle then boils down to the choice of functionality offered, weighted with the acceptable level of decentralization of that oracle network.

Note that single DLT network oracles also exist (sometimes referred to as on-chain oracles, meaning located, performed, or run inside a blockchain system [71]). These oracles (when triggered by a transaction) collect data

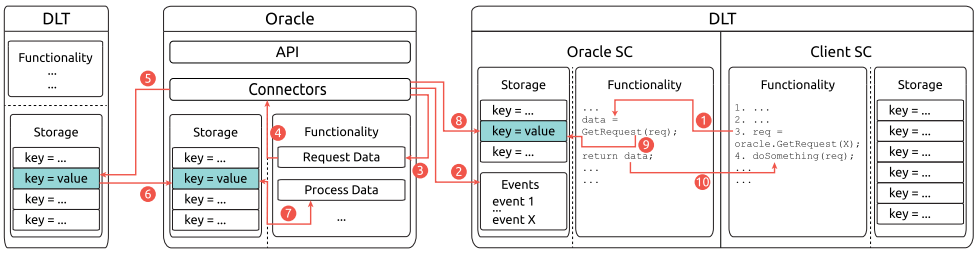


Fig. 10. Pull-based oracle architecture on execute-order-validate (e.g., Ethereum).

from different on-chain sources and present it in an easy to consume format [45]. But these oracles do not aim at providing cross-DLT network interoperability [45].

Oracles also support asset transfers across chains (sometimes known as bridges), typically unidirectionally, with mediated off-chain communication. These transfers are considered by commit-and-execute protocols [89, 138]. In this scheme, an asset residing in the source DLT network is burned (i.e., deleted) or locked, and a representation of that asset is minted (i.e., created) in the recipient DLT network. Both stages are implemented with smart contracts that implement specific rules for these operations.

Asset transfers via Oracles can occur using both push- and pull-based oracles.

— Pull-based oracle asset transfers: these are started by a transaction to a certain address or smart contract on a source DLT. This transaction contains the instruction to lock or burn the asset on the source DLT. Now an off-chain party, for example, the beneficiary of the transaction on the destination DLT (or an off-chain trusted third party) watches for the confirmation of the transaction on the source DLT. It sends a proof of that transaction to the destination DLT (e.g., a Merkle proof). More specifically, the proof is sent to the smart contract on the target destination DLT, along with information such as the beneficiary's address. This mechanism triggers the smart contract to mint and sends tokens to that address. At the end of the procedure, the number of assets is preserved.
— Push-based oracle asset transfers: these are started by a transaction on the destination DLT. This transaction contains the instruction to lock or burn the asset on the source DLT, create (or redeem) a representative asset on the destination DLT, and this transaction provides the related proof of ownership of that asset on the source DLT (such as a signature). In some interoperability solutions, this single transaction may be enough to complete the asset transfer (e.g., when a payment channel or Zero Knowledge Proof Rollup is the source DLT). Whereas in other interoperability solutions, other transactions, on the source DLT and possibly again on the destination DLT may be required to complete the asset transfer.

Optionally, cross-ledger audit trail information can be included in these transactions. Note that it is technically possible for token owners to operate their bridge, but this generates a problem around the general acceptance (e.g., by exchanges) of the token minted on the recipient DLT network. Some protocols that perform asset transfers include services to monitor the process for their users.

Interoperability Mode:

— Data Transfer (typically called oracles)
— Asset Transfer (typically called bridges)

Trust Assumptions:

— Centralized (single oracle)
— Decentralized (oracle consortium)

Advantages:

— Anyone can run their own oracle provided access to the recipient DLT.
— Oracles can allow the information collection phase to be separated from the processing phase, allowing arbitrary processing.
— Push-based oracles can send already processed data (i.e., ready to consume) or raw data. Processed data allows the amount of on-chain processing to be minimized.
— Pull-based oracles can allow a more transparent audit trail regarding who requested the data and who collected the data, as these actions are recorded inside distributed ledger transactions.
— Some DLTs allow smart contracts that can call external systems (e.g., Hyperledger's Fabric chaincode), i.e., the nodes running the protocol have the ability to perform as an oracle. This mechanism allows decentralized oracles, as oracle calls are made on-chain and are under consensus scrutiny.
— Oracles can be designed to declare who has permission to operate as an oracle. For instance, an IM can be designed to allow anytone to operate as an oracle (permissionless oracle system). In this case, there should be a mechanism that allows suspected invalid data to be challenged, and its authors possibly penalized. Alternatively, only certain oracles could be allowed, in which case a permissioned oracle system is used.

Disadvantages:

— Oracles support asset transfers, but trust needs to be put in the oracle group and the semantic layer supporting such transfer.
— Pull-based oracles can require multiple transactions for more generic calls (e.g., for generic HTTP requests vs. smart contract calls), raising the latency of the solution.
— Availability of oracles is a deciding factor, as smart contracts may rely on them to provide accurate information in real time. Failures in oracles (either crash faults or Byzantine faults) can occur and originate great losses (e.g., attacks on oracles DeFi [45]).
— Data is fed via DLT transactions that implies a minimum delay in the order of a few seconds to a few hours (if one considers finality).
— A smart contract depending on an oracle implies the trust of a (probably) smaller set of parties, compared to the number of nodes of the DLT network enforcing the correct execution of the smart contract. This weaker trust assumption is an attractive target for attackers.
— Can require synchronizing with off-chain parties (e.g., other exchange parties or the bridge operators).

Examples:

— Chainlink [18] (pull-based and push-based): provide external information to a set of smart contracts that can expose that information to other smart contracts for a fee.

  ChainLink selects qualified data feeders to provide data expected to represent the ground truth. Data feeders aggregate data via decentralized selection through staking their reputation (represented by LINK tokens). Data aggregation is done via statistical measures.
— BTC Relay [43] (push-based): provides information from the Bitcoin network to the Ethereum network.

  BTC Relay is a smart contract on Ethereum that stores block headers from the Bitcoin network. Nodes called relayers obtain the headers and send them to the BTC Relay smart contract. Smart contracts on the Ethereum network can then utilize information from the Bitcoin network by providing a Merkle proof referring to a certain block header.
— Polkadot (interoperability modules): the Polkadot ecosystem has several bridge projects allowing one to connect ecosystems [106]. These mechanisms mostly allow unidirectional asset transfers (albeit two unidirectional transfers can be done, realizing a bidirectional transfer), effectively connecting assets from different chains. For example, Snowfork is a general-purpose bridge between Ethereum and Polkadot. This will enable not only ETH to be transferred from Ethereum to Polkadot, but also ERC20 assets.

Solution Type 2 (Cross-Authentication). Cross-authentication interoperability solutions allow parties to exchange assets across DLTs, where each party sends a transaction on each DLT. To this end, parties need to authenticate on both chains to perform the transfers. This process typically happens without a trusted third party, as what is needed is some off-chain synchronization to set up the transactions to happen in both chains.

Trustless Asset Exchanges correspond to one asset exchange with non-mediated communication. The de-facto method for implementing this scheme is **Hash Lock Time Contracts** (**HTLCs**), where both parties deploy a smart contract in each chain that transfers the right amount of coins to the other party.

HTLCs consist in facilitating an asset exchange between two parties (typical case, although multi-party HLTCs exist [12, 63]) on a different blockchain (both parties can access it). Party from DLT 1 (P1) creates a secret s such that the hash of the secret, h(s), is put in a smart contract on DLT 1, transferring an asset to P2. The contract is hash locked with s, and a timelock t. Thus, P2 can redeem the assets from DLT 1 with secret s until time t. Upon confirming that the contract is correctly instantiated, P2 can create a smart contract on DLT 2 with the same hashlock h(s) but a timelock $t' < t$. The smart contract sends assets to P2 from DLT 2. This ensures that P1 can redeem assets before P2, with a slack t-t′. When P1 asserts that the contract from P2 is published, that party can send secret s, redeeming its assets. P2 now holds secret s, and can use it to redeem its assets on DLT 1.

HTLCs imply handling the technical layer (the hashing functions that are used to construct the secret needs to be supported by the involved DLTs) and the semantic layer (the information exchanged has meaning—assets—and needs to preserve a set of rules on both chains—avoiding double spending, for instance).

On the other hand, centralized asset exchanges are cross-authentication solutions that allow asset exchanges. These exchanges (also called notary schemes [12]) are a legal escrow that exchanges assets from one DLT for assets from another DLT. Decentralized exchanges are typically facilitators of such transactions by offering a bookkeeping system that matches buyers and sellers. Although decentralized exchanges running in heterogeneous DLTs are appearing [95], the mechanisms used for interoperation are classified as oracles. An overview of how centralized and decentralized exchanges work is present here [12].

New types of HTLCs are showing up. In Hyperledger Cactus [92], the *cactus-plugin-htlc-eth-besu-erc20* allows one to automatically deploy HTLCs on Ethereum via Hyperledger Besu. A similar package could automatically deploy two contracts: one in a permissioned DLT, and another in a permissionless DLT. As long as the parties exchanging assets are present in both networks, this scheme would work.

Interoperability Mode:

— Asset Exchange

Trust Assumptions:

— Centralized (centralized exchanges, notary schemes)
— Decentralized (HTLCs)

Advantages:

— Decentralized exchanges allow trustless asset exchanges between parties, by anchoring the correct operation of the process on the blockchain consensus).
— Platforms to create HTLCs running on heterogeneous DLTs, such as [92], streamline the process of setting up an exchange, diminishing the need for decentralized exchanges (and thus avoiding fees).

Disadvantages:

— Asset exchanges require multiple DLT transactions, which implies a minimum delay in the order of a few seconds to a few hours (if one considers finality).
— Network delays might render the execution transactions useless, wasting time and possibly transaction fees. However, some modern solutions eliminate this need.

Table 2. Summary Comparing Oracles and Cross-Authentication BISs

	Oracle	**Cross-Authentication**
Interoperability Mode	$\mathcal{D}$, $\mathcal{A}_t$	$\mathcal{A}_e$
Common Connection Mode	DLT Gateway	DLT Node, DLT Proxy
Can be used to build general-purpose use cases (vs. only transferring assets)	✓	✗
Native DLT security assumptions are enough to enable interoperation	✗	✓
Easily decentralizable	✗	✓
Easily implementable	✗	✓
Can parties be offline for interoperation to happen?	✓	✗ (for HTLCs)

$\mathcal{D}$ stands for data transfer, $\mathcal{A}_t$ for asset transfer, and $\mathcal{A}_e$ for asset exchange.

— Trustless approaches require some off-chain coordination between users wanting to exchange assets. Decentralized exchanges simplify this process at the expense of some decentralization.

Examples:

— Hyperledger Cactus cactus-plugin-htlc-eth-besu HTLC: Cactus provides a package that can deploy hash time lock contracts on Ethereum via Hyperledger Besu. The package provides functionality to deploy initialization, refund, and monitoring endpoints. Additional functionality (such as mediating off-chain agreements between the users of the HTLC) can be built on top of this package (i.e., a business logic plugin).
— Exchanges: with a centralized exchange, users deposit fiat or cryptocurrencies in a platform that is used to swap for other assets. It does not require all parties to authorize transactions on both chains, but only requires the sending user and the exchange to authorize the swap transaction.

Summary. Contrary to common knowledge, there are only two non-intersecting categories. We emphasize that this is a general overview. In-depth descriptions of the protocols and their implementations can be found in [12] (both), [25, 41, 94] (oracles), and [137] (cross-authorization). Table 2 summarizes the studied categories.

Oracles can perform data and asset transfers, typically using DLT gateways. This is due to the ability of gateways to process data to the format the oracle smart contracts accept. Oracles can enable general-purpose interoperability, thus they have implementation overhead, as well as decentralization overhead. On the other hand, cross-authentication solutions are used for asset exchanges only—a DLT node or DLT proxy suffice.

The immense variability of IM solutions stems from the fact that many design patterns are built on top of those two categories. A detailed study on the available design patterns for IM is left for future work.

Key Takeway 6. *There is no technical distinction between "Layer 1" and "Layer 2" solutions*

The industry typically classifies DLTs into layer 1 infrastructure or layer 2. While layer 1's are standalone DLTs, layer 2's are DLTs extending the capabilities of layer 1's, attempting to solve, for instance, the scalability problem [12]. Both layer 1 and layer 2 solutions are groups of nodes (i.e., chains) running a protocol. Some of the node groups might anchor their security on another one (typically layer 2 chains share security [49] or re-utilize work [82] from layer 1's). This implies that layer 2's are better viewed as separate networks (technically similar to layer 1's) connected by an IM (typically called bridges).

4 WHICH BLOCKCHAIN INTEROPERABILITY SOLUTION DO YOU NEED?

Few studies provide guidelines to improve interoperability in blockchain solutions [8]. Specifically, there are no frameworks to evaluate cross-chain solutions in terms of interoperation capabilities systematically, performance, security, cost, and user friendliness. In this section, we put forward a first effort, based on our recent work [90]. Since standardized approaches to evaluate interoperability are needed [93], we propose our interoperability assessment framework for DLTs. We start this section by presenting our interoperability framework, allowing the end-user to assess the current state, in terms of interoperability, of their DLT-based solution. After that, given that an initial assessment has been conducted, we present our framework to choose the infrastructure and functionality of an IM. Users can then improve the interoperability of their solutions by picking a suitable IM, based on the proposed decision models. Finally, one can re-evaluate the interoperability of their DLT-based solution by running the interoperability assessment again.

4.1 Interoperability Assessment

The goal of the interoperability assessment is to provide concrete, systematic guidelines for solutions to be compared in terms of interoperation capabilities—a concept called **Interoperability Assessment** (**INAS**) [36]. This work focuses on evaluating interoperation capabilities (and performance, to a lower extent). To this end, we propose three assessments. Each assessment defines a set of criteria, where each item yields a score. The score of all criteria outputs the score of that assessment. Summing the score of the three types of assessments yields the final score for the interoperability assessment. The higher the score, the better the interoperability capabilities of such a solution. Table 3 shows the score for each criteria. There are three assessments a system can take to measure the ease of interoperability regarding external systems [36]:

— *Potentiality assessment*: this assessment evaluates the maturity of a system to adapt to other systems. It answers the question *can the system interoperate with other systems as is*?
  This assessment provides an understanding of which infrastructures a DLT-based solution can connect to. The score for this assessment is divided into four categories, for a maximum of 4 points.
— *Compatibility assessment*: this assessment evaluates the interoperability between two known systems before or after changes to interoperation capabilities of both. It answers the question *how well can a pair of systems interoperate*? And *what are the current problems or barriers that prevent the systems from interoperating better?*
  This assessment provides an understanding of the "capabilities" the interoperability mechanisms offer (can it make two DLTs understand each other? can it comply with rules and laws?). The score for this assessment is divided into three categories, for a maximum of 3 points.
— *Performance assessment*: this assessment evaluates the interoperation processes during runtime concerning cross-chain transactions key metrics. It answers *what are the values for the interoperation metrics cross-chain latency, cross-chain throughput, and cross-chain costs?*
  The score for this assessment is divided into two categories, for a maximum of 3 points.

4.1.1 Potentiality Assessment. The potentiality assessment evaluates technical interoperability (see Section 3). It takes a system based on a DLT protocol and evaluates its maturity toward interacting with other systems (requesting/providing data). Four levels of interoperability exist (cf. Figure 11), in increasing order of complexity:

— *Level P1*: interoperation across different functionalities (e.g., smart contracts) on the same subnetwork can happen. An example is smart contracts calling other smart contracts (on the Ethereum network, on the same Hyperledger Fabric channel). Even though level 1 interoperability may seem standard for some DLTs, e.g., smart contracts on Ethereum and Fabric, this is not the case for all DLTs. For instance, on Corda, Cordapps are deployed onto certain nodes. Each Cordapp includes a set of smart contracts used for

Table 3. DLT Interoperability Solution Assessment

Potentiality Assessment (PA)	**Score (0–4)**
P1: Interoperation within the same DLT network, same subnetworks	☐
P2: Interoperation within the same DLT network, different subnetworks	☐
P3: Interoperation within different DLT networks	☐
P4: Interoperation within different DLT protocols	☐
Compatibility Assessment (CA)	**Score (0–3)**
C1: Provides semantic-level interoperability (shared protocols)	☐
C2: Provides organization-level interoperability (shared agreements)	☐
C3: Provides legal-level interoperability (follow regulations)	☐
Performance Assessment (PeA)	**Score (0–3)**
PE1: Provides acceptable cross-chain transaction end-to-end latency/throughput	☐
PE2: Provides acceptable cross-chain transaction end-to-end cost	☐
PE3: Complies with desirable energetic consumption goals	☐
PA + CA + PeA	**Total (0–10):**

Interoperability assessment is divided into PE, CA, and PeA assessments. A higher score corresponds to a more interoperable solution.

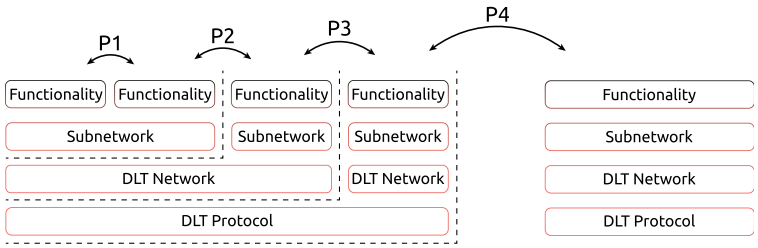


Fig. 11. Potentiality assessment of an IM.

transactions relating to this Cordapp. Allowing smart contracts created in one Cordapp to be utilized by transactions created in another Cordapp is a non-trivial task due to the UTXO architecture of Corda.

— *Level P2*: interoperation within the same DLT network, different subnetworks (smart contracts can call other smart contracts from other subnetworks, e.g., across Hyperledger Fabric channels).

— *Level P3*: interoperation across DLT networks of the same DLT protocol, i.e., homogeneous blockchains (e.g., Ethereum Ropsten to Ethereum mainnet, Hyperledger Fabric network A to Hyperledger Fabric network B, as seen in [51]).

— *Level P4* interoperation across different networks of different DLT protocols, i.e., heterogeneous DLTs, as seen in [51, 92, 113].

We could also consider *Level P5*, providing interoperation with non-DLT systems (e.g., enterprise systems, payment systems). However, all IM solutions and DLT nodes provide capabilities for accessing the ledger. In the compatibility assessment, we consider interactions (e.g., digital asset exchanges across DLTs) to have legal binding.

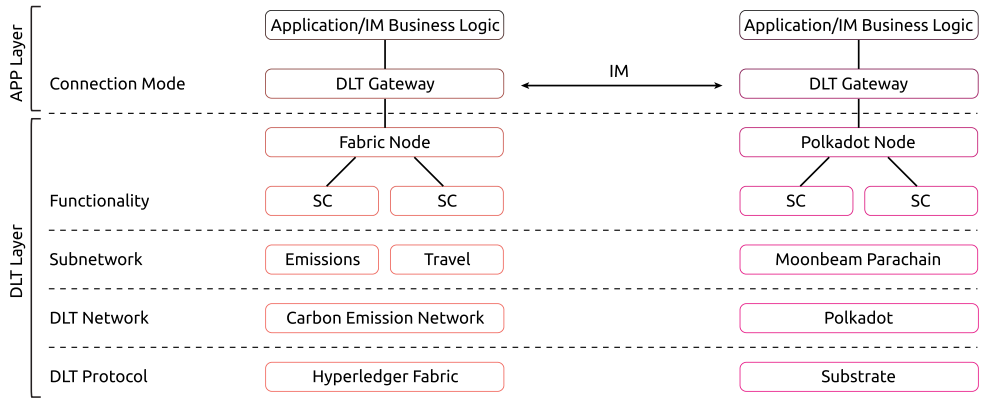


Fig. 12. Example of vertical interoperation in a Hyperledger Fabric network and the Polkadot network. Horizontal interoperability can be achieved via an IM using, for example, a DLT gateway.

For instance, Hyperledger Fabric–based networks can provide and receive information from and to the exterior, respectively, via smart contracts. Figure 12 depicts a practical example of a potentiality assessment. A high score for this assessment shows that *the system can interoperate with systems significantly different from it as is.* One could consider an IM as the cross-chain logic plus a connection mode (in this case a DLT gateway). The IM can then connect to multiple DLT networks, depending on how many of the latter the DLT gateway supports. Thus, business logic from the IM can spawn across several DLT networks. If this is the case, cross-chain logic can be implemented. Another interesting possibility can be implemented: connecting the IMs via the DLT gateways, allowing for second-order interoperability. Provided accountability guarantees, such as smart contracts as trust anchors, or a decentralized log storage for IMs, this enables cross-chain use cases operated by mutually untrusted IMs

4.1.2 Compatibility Assessment. The compatibility assessment evaluates compatibility aspects regarding semantic, organizational, and legal interoperability. *Given a pair of systems, do they run protocols that both understand? Do they share similar organizational goals? Do they follow the same jurisdiction and regulations?*
Figure 13 depicts this assessment. Three levels of compatibility maturity exist:

— *Level C1*: semantic interoperability is achieved by a pair of systems (see Section 2).
— *Level C2:* semantic and organizational interoperability are achieved.
— *Level C3:* semantic, organizational, and legal interoperability is achieved.

The score for this assessment is obtained by summing the weights of each level cumulatively, since the last layer typically depends on the previous (1, 2, and 3, respectively). In this article, we focus on the semantic aspect, leaving pointers for future work on the organizational and legal aspects. The granularity regarding the three layers can be defined by the users of the framework.
Figure 14 depicts an architecture of an interoperability solution (e.g., it could be connected by a network of gateways) that has a compatibility assessment conducted. A network of gateways is a set of gateways that run cross-chain logic shared by gateways, following a protocol.

4.1.3 Performance Assessment. The performance assessment studies how efficiently an IM executes its processes. The efficiency can be measured in metrics related to the **Cross-chain transaction** (**CC-Tx**) concept. A Cross-chain transaction (CC-Tx) is composed of transactions directed to the target systems (called *subtransactions of a CC-Tx*) plus the internal transactions of the IM. We call the logic that an IM executes *cross-chain logic*

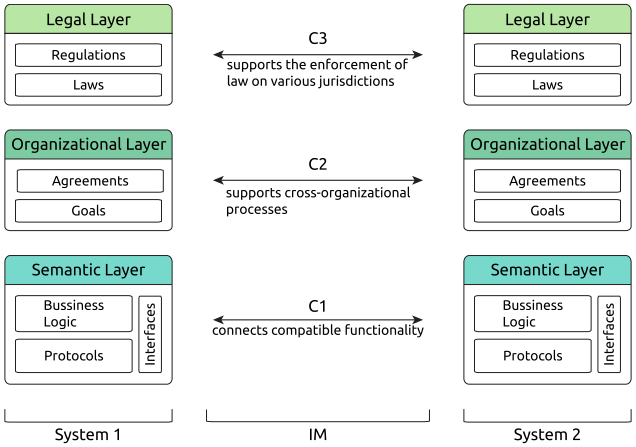


Fig. 13. Compatibility assessment between two systems.

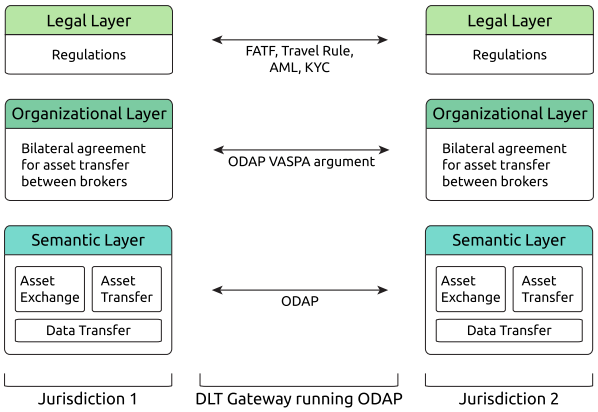


Fig. 14. Example of interoperation between two DLTs connected by a DLT gateway network running a digital asset transfer protocol such as ODAP [60]. A compatibility assessment can be performed regarding the participating DLTs.

or *cross-chain rules*. The cross-chain logic accounts for business logic plugins' execution, which can modify the final transactions issued against DLTs.

Following the blockchain integration framework, there are three main metrics to assess the performance of an IM [90]:

— *End-to-end latency*: calculated by summing the time to execute the IM cross-chain logic plus the latency of every sub-transaction (until it is committed and finalized, in its local DLT). For example, the Carbon Emission use case (see Figure 3) would have a latency calculated by the execution of the mdApp logic plus two transactions on the Emissions Channel plus one transaction on the Ethereum network. The end-to-end latency answers the question *how long does it take for the cross-chain transaction to be incorporated on the target systems*?

— *End-to-end throughput*: the throughput is the number of cross-chain transactions executed per second. The more complex the cross-chain logic, and the longer it takes for each sub-transaction to be committed on their target system, the lower the throughput. The throughput metric answers the question *how many cross-chain transactions are finalized on the ledgers per second?*
— *End-to-end cost*: cost can come in two forms: transaction cost and energetic consumption. Cost can be measured in transaction fees and/or direct transaction costs. Transaction fees are paid to support the network (happening in public, permissionless DLTs more often). Direct transaction cost (in a certain period) can be calculated by dividing the cost of being in the network by the number of transactions. This latter model is usual on permissioned networks with the subscription business model. The energetic consumption is calculated by dividing the energetic cost of a transaction per number of transactions. This metric answers the question *what is the cost in transaction fees and energetic consumption of the sum of the cross-chain transactions issued by the system?*

At the moment, it is not possible to establish specific guidelines for this type of assessment due to the lack of systematic evaluations of interoperability solutions. Although several solutions bring performance evaluations [12], they are not standardized according to any framework, making it difficult, if not impossible, to compare solutions systematically. Thus, we leave the judgment of a reasonable latency, throughput, and cost for interoperability solutions and a rigorous model to evaluate the performance for future work. We emphasize the challenges to measuring the energetic consumption of an IM, given that it interacts will multiple decentralized systems. Although some work has been done in evaluating the energetic consumption of Bitcoin [53, 55], the literature falls short in exploring other DLTs. Thus, this remains a problematic metric to assess.

4.2 Choosing the Right Interoperability Solution

In this section, we help the reader choose an IM, using two decision models. The proposed decision models are directed to researchers, developers, and software architects. The first decision model focuses on assisting the choice of the IM's infrastructure (connection mode), for a given use case. The second decision model assists in the choice of the IM's functionality (interoperation mode, potentiality, and compatibility). The output of functionality diagram suggests a group of IMs for the chosen functionality.

How to Use the Decision Models. The reader should start navigating from left to right, starting on the node with the *START* label. After that, a path should be followed by answering yes (✓) or (✗) to the proposed questions until an end node is reached (blue nodes), or a *proceed* flag is present in one of the arrows connected to the current node. More details on the proceed flag are available on the functionality decision model. The blue nodes output recommendations regarding the infrastructure or functionality of an IM.

Infrastructure. Choosing an infrastructure refers to choosing the hosting infrastructure for an IM: DLT node, DLT proxy, or DLT gateway. Hosting an IM implies several challenges that require specialized, well-trained experts: (1) node and hardware management, including installation, maintenance, load balancing, software version management, redundancy, and scaling; (2) security, including monitoring the node and responding to cyber-attacks; (3) and others, such as adhering to regulations (e.g., GDPR). Thus, different needs require a different infrastructure. Depending on the use case, it is acceptable to defer the management of the infrastructure to third parties (versus self-hosting the infrastructure).

The following decision model, in Figure 15, guides on choosing the infrastructure (i.e., connection mode) for an IM and if it should be self-hosted or not.

DLT nodes are native blockchain clients, e.g., Geth Ethereum Node [42], Hyperledger Besu node (Ethereum node) [5], Hyperledger Fabric peer node [5], Bitcoin Core [14], and Polkadot node [108]. DLT proxies include Infura [67], Blockdaemon [15], and nodes hosted and accessible via cloud providers. DLT gateways include Quant

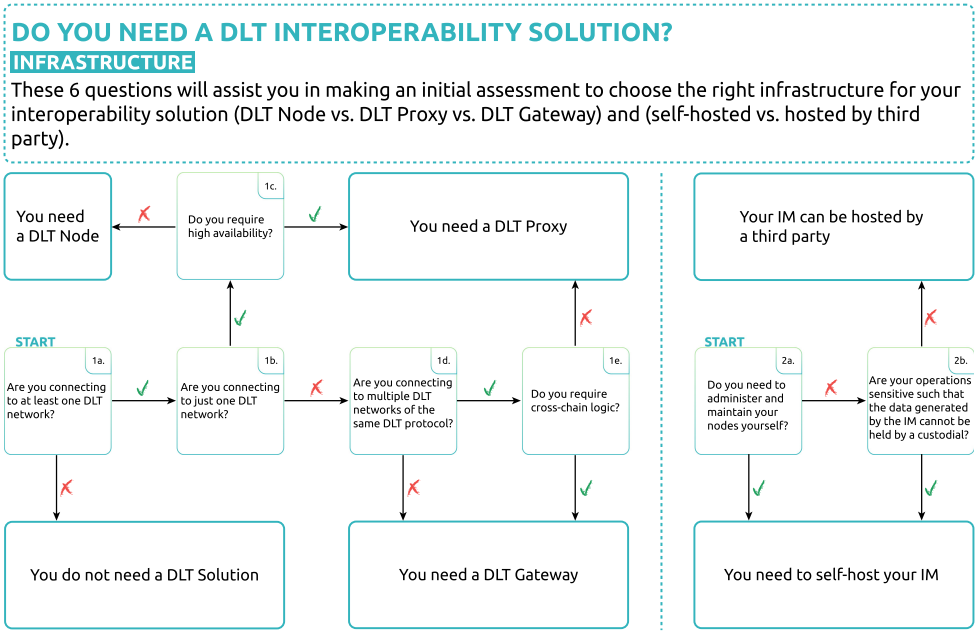


Fig. 15. Decision map guiding the choice of the infrastructure of a DLT interoperability solution. Start on the topmost left node (with the green *START* label) and answer the questions until you arrive to an end node (blue nodes). The output of this process is a group of IMs that respect the requirements stated on the process flow.

Overledger [113], Hyperledger Cactus nodes [92], Weaver gateways [127], among others. More examples of each connection mode can be found in [12].

Functionality. Choosing the functionality refers to choosing the IM functionality in terms of interoperation mode, P levels, and C levels.

Most IMs assure P $\leq$ 3, while a few provide all P levels (P1, P2, P3, and P4) [12]. On the other hand, IMs can be divided on C1, because most do provide $C < 2$. Most IMs can provide C1 (in fact, a system providing P4 implies that it provides C1), while a few attempt at implementing standards that could, in the future, support the legal layer. At the moment, we do not know of any IM providing level C3.

The following decision model, in Figure 16, guides on choosing the functionality. This decision model uses the *proceed* flag, meaning that when it is present in an arrow connected to the current node, the user should evaluate the condition, and then proceed (instead of stopping), *accumulating* the recommendations, until a node without a proceed node is found (and therefore the last decision is made on that node). Take, for example, the following flow: one starts in node *3a*. As the *proceed* flag is present on that node, the reader will answer to the question and then (independently of the answer) move to the next node. In case the answer was yes (✓), the recommendation is saved. At the end of Figure 16, a maximum of five recommendations may be collected.

4.2.1 Solution Groups. In this section, we define each solution group depicted in Figure 16. In particular, we systematically compare IMs according to their P level, C level, and interoperation mode. Table 4 shows examples of solutions belonging to each group proposed by Figure 16.

The first group comprises solutions providing levels P1–P3, and supporting data transfers, asset transfers, or asset exchanges. Most blockchains provide P1 interoperability by enabling functionality re-usage. For instance,

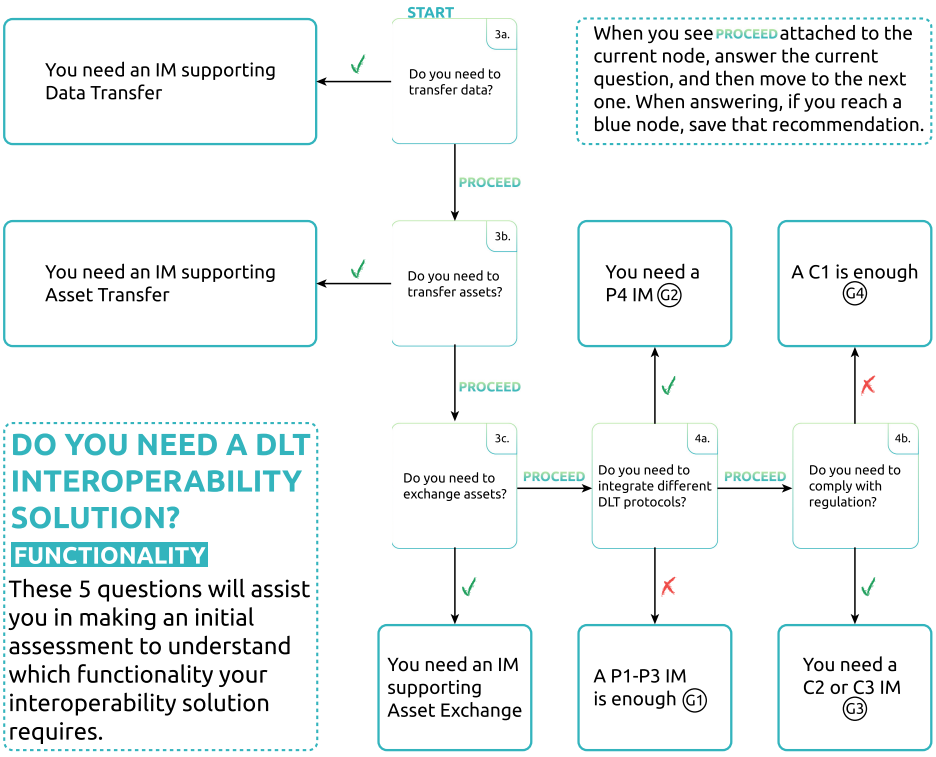


Fig. 16. Decision map guiding the choice of the functionality of a DLT interoperability solution. Start on the topmost left node (with the green label START) and answer the questions until you arrive to an end node (gray ones). The output of this process is a group of IMs that respect the requirements stated on the process flow.

smart contracts can call other smart contracts (even across subnetworks, providing level P2) in most blockchains. Level P2 requires some orchestration. For example, interoperability across subnetworks in Ethereum can be done via oracles or gateways running bespoke cross-chain logic. A similar level of orchestration happens when P3 is needed but could be more complex, as different networks may differ more than different subnetworks. Level P3 and P4 systems imply the usage of blockchain interoperability middleware (in some DLT protocols, level P3 can be achieved without specialized middleware) that can be centralized or decentralized.

Many of the examples present in group G2 supporting data transfers are classified as *trusted relays* or *blockchain agnostic protocols*, while G2 solutions supporting asset transfers and exchanges belong to *sidechains and relays*. Blockchain of blockchain platforms (Polkadot, Cosmos) provides level P3. The design and implementation of bridges to external DLTs would carry these systems to level P4, but they are still in development, and, furthermore, these bridges are not native to the DLT networks.

Example 1 (Oracle Solution). In this section, we present an example of the choice of the infrastructure and functionality of an oracle solution based on a simple use case. Let us consider an IM administrator who transfers data between the Ethereum blockchain and a Polkadot's parachain.

Infrastructure: the administrator does not want to host the infrastructure and would like to have high availability. There is no preference to whether the IM can access the DLT nodes directly or not. The administrator

Table 4. Categorization of IM According to P Levels, C Levels, and Interoperation Mode ($\mathcal{D}$ Stands for Data Transfer, $\mathcal{A}_t$ for Asset Transfer, and $\mathcal{A}_e$ for Asset Exchange)

Solution Group	Solution	P1	P2	P3	P4	C1	C2	C3	$\mathcal{D}$	$\mathcal{A}_t$	$\mathcal{A}_e$
G1	[2, 22, 38, 104, 117]	✓	✓	✓	✗	✓	✗	✗	✓	✗	✗
	[1, 4, 40, 98, 112, 116], HTLCs [12]	✓	✓	✓	✗	✓	✗	✗	✗	✓	✓
	Blockchain of blockchains (e.g., [78, 129])	✓	✓	✓	✗	✓	✗	✗	✓	✓	✓
G2	[46, 47, 51, 92, 99, 127]	✓	✓	✓	✓	✓	✗	✗	✓	✗	✗
	[11, 17, 32, 33, 64, 65, 69, 92, 138], HTLCs [12]	✓	✓	✓	✓	✓	✗	✗	✗	✓	✓
	[92, 109, 117], Bridges (e.g., [7, 81, 82])	✓	✓	✓	✓	✓	✗	✗	✓	✓	✓
G3	Solutions supporting P ≥ 3										
G4	[6, 113]	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓

accepts the risk of its data being read by third parties, for the comfort of an easier to deploy solution. Using the decision model from Figure 15, the administrator would answer:

(1) **What is your desired connection mode?**
   — 1a: Are you connecting to at least one DLT? ✓
   — 1b: Are you connecting to just one DLT? ✗
   — 1c: Do you need high availability? ✓
   — 1d: Do you need to run cross-chain logic? ✓

   Therefore, a DLT gateway is needed.
(2) **What is your desired hosting mode?**
   — 2a: Do you need to administer and maintain your nodes yourself? ✗
   — 2b: Are your operations sensitive such that the data generated by the IM cannot be held by a custodial? ✗

   Therefore, the IM can be hosted by a gateway hosted by a third party.

Functionality: the IM would only need to transfer data. It needs to connect different DLT protocols and does not need to comply with any regulations. Using the decision model from Figure 16, the administrator would answer:

(1) **What is the desired interoperation mode?**
   — 3a: Do you need to transfer data? ✓
   — 3b: Do you need to transfer assets? ✗
   — 3c: Do you need to exchange assets? ✗

   Therefore, IM supporting asset transfers is needed.
(2) **What is the desired supported functionality?**
   — 4a: Do you need to integrate different DLT protocols? ✓
   — 4b: Do you need to comply with regulation? ✗

   Therefore, solutions from G2 (supporting P1–P4, C1, and data transfers) are needed.

Therefore, the chosen IM is a P4 + C1 third-party hosted DLT gateway (see Table 4 to choose a solution).

Example 2 (Cross-Authentication Solution). In this section, we present an example of the choice of the infrastructure and functionality of a cross-authentication solution based on a simple use case. Let us consider an end-user who wants to perform an asset exchange. The user wants to exchange asset a in DLT a for asset b in DLT b.

Infrastructure: the user is looking to connect to two different DLTs, does not need to run cross-chain logic (as the only logic needed is a time-locked transfer), with no high availability requirements. The user wants to have direct access and control over the node. Using the decision model from Figure 15, the administrator would answer:

(1) **What is your desired connection mode?**
— 1a: Are you connecting to at least one DLT? ✓
— 1b: Are you connecting to just one DLT? ✗
— 1c: Do you need high availability? ✗
— 1d: Do you need to run cross-chain logic? ✗

Therefore, a DLT proxy (or DLT node) is needed.

(2) **What is your desired hosting mode?**
— 2a: Do you need to administer and maintain your nodes yourself ✓
— 2b: Are your operations sensitive such that the data generated by the IM cannot be held by a custodial? ✗

Therefore, the IM can be self-hosted.

Functionality: the IM needs to exchange assets (or perform two independent asset transfers). It needs to connect different DLT protocols and does not need to comply with any regulation, but it must comply with rules on exchanging assets (e.g., hashlock time contract). Using the decision model from Figure 16, the administrator would answer:

(1) **What is the desired interoperation mode?**
— 3a: Do you need to transfer data? ✗
— 3b: Do you need to transfer assets? ✗
— 3c: Do you need to exchange assets? ✓

Therefore, an IM supporting asset exchanges is needed.

(2) **What is the desired supported functionality?**
— 4a: Do you need to integrate different DLT protocols? ✓
— 4b: Do you need to comply with regulation? ✗

Therefore, we need solutions from G2 (supporting P1–P4, C1, and supporting asset exchanges).

Therefore, the chosen IM is a P4 + C1 self-hosted DLT proxy supporting asset exchanges.

5 RELATED WORK AND OPEN RESEARCH CHALLENGES

In this section, we compare the contributions of our article to the state-of-the-art.

Status quo of Blockchain Interoperability. Compared to the existing literature, our study focuses on allowing researchers and developers (or software architects) to choose a blockchain interoperability solution, which was only partially addressed before. In particular, a number of surveys studied blockchain interoperability solutions or architectures [12, 13, 16, 24, 72, 73, 76, 87, 103, 111, 119, 120, 126, 137], but did not provide a decision model to choose one. Each survey comes with tradeoffs (technical explanation depth vs. IM coverage) and often with

conflicting categories of solutions. This implies that the reader will not have a holistic overview of the area. Our survey performs an analysis of these surveys, attempting to unify and synthesize existing knowledge.

The latest systematic survey on IM is that of Belchior et al. [12]. In this survey, the authors categorize IM into public connectors, hybrid connectors, and blockchain of blockchains. Instead of answering *how* are IMs connecting DLTs, in the survey, we classify IMs according to *what they connect*.

Our categorization departs from older ones because we consider trusted relays, sidechains, notary schemes, relays, gateways, and other design patterns built on top of oracles. We clarify that only two types of interoperability exist and propose a general IM model.

Assessing the Degree of Interoperability of an IM. There is extensive work in the area of interoperability assessment. Leal et al. discuss and systematically compare 21 interoperability assessment approaches [36]. Most approaches aim to assess the interoperability of (centralized) systems in the technical, semantic, and organizational layers, while some generically evaluate interoperability.

In [125], the authors discuss different interoperability testing architectures for assessing interoperability between distributed systems. The authors provide guidelines to generate interoperability tests for the proposed testing architectures. Our work provides guidelines (or interoperability tests) for DLTs. Numerous other works evaluate interoperability for IoT [139], cloud providers [74, 83, 121], and more generic ones [75, 80].

The closest to our work is the *blockchain interoperability evaluation framework* of Mihaiu et al. [90]. In this study, the authors introduce the concepts of cross-chain rules and propose a list of metrics to compare blockchain interoperability solutions. In an older study, the authors compare two interoperability solutions based on 12 ad-hoc criteria [76].

To the best of our knowledge, our framework is the first to provide insights on the potentiality, compatibility, and performance of a blockchain interoperability solution while explicitly providing support to choose the infrastructure and functionality of an IM. Our framework helps the reader confirm the IM's adequacy for interoperating with other systems at technical, semantic, organizational, and legal levels. Our framework for choosing a blockchain interoperability solution may resemble studies aimed at facilitating the choice of an IM on functional and non-functional requirements, such as [135].

Open Research Challenges. Although recent years have assisted in the skyrocketing increase in cross-chain research, several research challenges are left unsolved: First and foremost, there is no formalization of a general model for IMs (using frameworks to define and prove security properties such as the universal composability framework [26]) that can answer the following questions: *What are the technical requirements that a DLT must provide, in order to be interoperable? What are the technical requirements an IM must provide to assure safety and liveness properties?* These research questions are important because often implementations of IM typically do not follow specific guidelines [25].

Secondly, as pointed out by [8, 12] there is still a lack of supporting tools for IMs, such as monitoring tools (e.g., visualization and analysis of cross-chain transactions and state), cross-chain digital identity, migration tools (change the DLT infrastructure on the go for a dApp or mDApp), security tools (automatic detection of frontrunning attacks, formal analysis of cross-chain protocols), and others.

Following the reasoning line of this work, there are still no methods to systematically evaluate and compare the performance of an IM. Although we propose an initial set of metrics to respond to this need, there are no baseline benchmarks.

Finally, regulation on cross-chain asset transfers, mainly among institutional players, will significantly impact how modern financial systems interact and evolve. Although there have been recent and numerous efforts on regulating certain aspects of interoperability (such as the ODAP protocol [60]), currently, there are no available standards.

Key Takeway 7. *DLT interoperability standards*

Although none are official standards, there is a wide range of standardization efforts on blockchain interoperability. The works in [12, 101, 131] survey the major standardization efforts in the field.

6 CONCLUSION

The future of DLT technology depends on its capability to interoperate across different dimensions. To address such a gap, interoperability solutions flourished in recent years. However, several challenges are posed to researchers and practitioners when trying to understand blockchain interoperability. For instance, there is still no systematic way for classifying, assessing, comparing, and choosing DLT interoperability solutions.

Our article closes this gap by exposing three main contributions. First, we propose a unified conceptual model and classification framework for blockchain interoperability solutions. Second, we propose a framework to assess the interoperability capabilities of a system utilizing one or more DLTs. Lastly, we propose two decision models that allow one to choose the infrastructure and functionality for an IM, to enable their off-chain systems to interoperate with multiple DLTs. We provide practical examples of this decision process. Finally, based on the most recent research, we provide a list of updated open research challenges in the blockchain interoperability research area.

ACKNOWLEDGMENTS

We warmly thank our colleagues in the IETF's forming working group ODAP for fruitful discussions. We thank the Hyperledger Cactus community and Iulia Mihaiu for insightful discussions on blockchain interoperability. We thank Si Chen for supporting us on specifying the Carbon Emission use case.

REFERENCES

[1] 0x Protocol Team. 2021. 0x: Powering the decentralized exchange of tokens on Ethereum. Retrieved January 10, 2022 from https://0x.org/.

[2] Ermyas Abebe, Dushyant Behl, Chander Govindarajan, Yining Hu, Dileban Karunamoorthy, Petr Novotny, Vinayaka Pandit, Venkatraman Ramakrishna, and Christian Vecchiola. 2019. Enabling enterprise blockchain interoperability with trusted data transfer. In *Proceedings of the 20th International Middleware Conference Industrial Track*. Association for Computing Machinery, 29–35.

[3] Ermyas Abebe, Dileban Karunamoorthy, Jiangshan Yu, Yining Hu, Vinayaka Pandit, Allison Irvin, and Venkatraman Ramakrishna. 2021. Verifiable observation of permissioned ledgers. arXiv 2012.07339v2. Retrieved from https://arxiv.org/abs/2012.07339.

[4] Hayden Adams, Noah Zinsmeister, and Dan Robinson. 2020. *Uniswap v2 Core*. Technical Report. Uniswap. https://docs.uniswap.org/, last accessed on 2022-01-10.

[5] Elli Androulaki, Artem Barger, Vita Bortnikov, Srinivasan Muralidharan, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Chet Murthy, Christopher Ferris, Gennady Laventman, Yacov Manevich, Binh Nguyen, Manish Sethi, Gari Singh, Keith Smith, Alessandro Sorniotti, Chrysoula Stathakopoulou, Marko Vukolić, Sharon Weed Cocco, and Jason Yellick. 2018. Hyperledger fabric: A distributed operating system for permissioned blockchains. In *Proceedings of the 13th EuroSys Conference (EuroSys'18)*, Vol. 2018-Janua. AACM, New York, New York, 1–15.

[6] ARK Team. 2019. ARK Whitepaper Version 2.1.0. (2019). Retrieved January 1, 2022 from https://whitepaper.ark.io/prologue.

[7] Tal Baneth. 2019. Waterloo—A Decentralized Practical Bridge between EOS and Ethereum. (2019). Retrieved February 19, 2021 from https://blog.kyber.network/waterloo-a-decentralized-practical-bridge-between-eos-and-ethereum-1c230ac65524.

[8] Rafael Belchior. 2021. PhD Thesis Proposal—Blockchain Interoperability. Technical Report. Instituto Superior Técnico. Retrieved August 10, 2021 from https://www.researchgate.net/publication/355370486_PhD_Thesis_Proposal.

[9] Rafael Belchior, Sérgio Guerreiro, André Vasconcelos, and Miguel Correia. 2020. A survey on business process view integration. (Nov. 2020). http://arxiv.org/abs/2011.14465. To appear: Business Process Management Journal.

[10] Rafael Belchior, André Vasconcelos, Miguel Correia, and Thomas Hardjono. 2021. Enabling cross-jurisdiction digital asset transfer. In *IEEE International Conference on Services Computing*. IEEE.

[11] Rafael Belchior, André Vasconcelos, Miguel Correia, and Thomas Hardjono. 2021. HERMES: Fault-tolerant middleware for blockchain interoperability. *Future Generation Computer Systems* (March 2021). https://doi.org/10.36227/TECHRXIV.14120291.V1

[12] Rafael Belchior, André Vasconcelos, Sérgio Guerreiro, and Miguel Correia. 2021. A survey on blockchain interoperability: Past, present, and future trends. *ACM Computing Surveys* (May 2021). http://arxiv.org/abs/2005.14282.
[13] Monika Bishnoi and Rajesh Bhatia. 2020. Interoperability solutions for blockchain. In *Proceedings of the International Conference on Smart Technologies in Computing, Electrical and Electronics (ICSTCEE'20)*. 381–385.
[14] Bitcoin. 2021. Bitcoin Core integration/staging tree. Retrieved October 14, 2021 from https://github.com/bitcoin/bitcoin.
[15] Blockdaemon. 2021. Platform—Blockdaemon. Retrieved January 1, 2022 from https://blockdaemon.com/platform/.
[16] Michael Borkowski, Philipp Frauenthaler, Marten Sigwart, Taneli Hukkinen, Oskar Hladky, and Stefan Schulte. 2019. Cross-Blockchain Technologies: Review, State of the Art, and Outlook. Retrieved August 10, 2021 from https://dsg.tuwien.ac.at/projects/tast/pub/tast-white-paper-4.pdf.
[17] Michael Borkowski, Marten Sigwart, Philipp Frauenthaler, Taneli Hukkinen, and Stefan Schulte. 2019. DeXTT: Deterministic cross-blockchain token transfers. *IEEE Access* 7 (Aug. 2019), 111030–111042.
[18] Lorenz Breidenbach, Christian Cachin, Benedict Chan, Alex Coventry, Steve Ellis, Ari Juels, Farinaz Koushanfar, Andrew Miller, Brendan Magauran, Daniel Moroz, Sergey Nazarov, Alexandru Topliceanu, Florian Tramèr, and Fan Zhang. 2021. *Chainlink 2.0: Next Steps in the Evolution of Decentralized Oracle Networks*. Technical Report. Chainlink. Available online https://research.chain.link/whitepaper-v2.pdf.
[19] Crypto Briefing. 2021. Aave is Exploring Solana, Avalanche, Layer 2 Expansion—Crypto Briefing. Retrieved January 10, 2022 from https://cryptobriefing.com/aave-is-exploring-solana-avalanche-layer-2-expansion/.
[20] Richard Brown. 2018. The Corda Platform: An Introduction White Paper. Retrieved August 10, 2021 from https://www.r3.com/reports/the-corda-platform-an-introduction-whitepaper/.
[21] Matthew BSC. 2021. Binance Launches $1B Binance Smart Chain Fund to Reach One Billion Crypto Users. Retrieved January 10, 2022 from https://www.binance.org/en/blog/binance-launches-one-billion-binance-smart-chain-fund-to-reach-one-billion-crypto-users/.
[22] Gewu Bu, Riane Haouara, Thanh Son Lam Nguyen, and Maria Potop-Butucaru. 2020. Cross hyperledger fabric transactions. In *CRYBLOCK 2020—Proceedings of the 3rd Workshop on Cryptocurrencies and Blockchains for Distributed Systems, Part of MobiCom 2020*. ACM, 35–40. https://doi.org/10.1145/3410699.3413796
[23] Vitalik Buterin. 2009. Ethereum White Paper. Retrieved January 10, 2022 from https://www.networkworld.com/article/2177684/lan-wan/the-growth-in-east-west-traffic.html.
[24] Vitalik Buterin. 2016. *R3 Report—Chain Interoperability*. Technical Report. R3 Corda. https://www.r3.com/wp-content/uploads/2017/06/chain_interoperability_r3.pdf.
[25] Giulio Caldarelli and Joshua Ellul. 2021. The blockchain oracle problem in decentralized finance–A multivocal approach. *Applied Sciences (Switzerland)* 11, 16 (2021). https://doi.org/10.3390/app11167572
[26] R. Canetti. 2001. Universally composable security: A new paradigm for cryptographic protocols. In *Proceedings of the Annual Symposium on Foundations of Computer Science*. 136–145. https://doi.org/10.1109/SFCS.2001.959888
[27] Carbon Accounting and Certification WG. 2021. Carbon Accounting and Certification WG—Climate Action SIG—Hyperledger Foundation. Retrieved January 10, 2022 from https://wiki.hyperledger.org/display/CASIG/Carbon+Accounting+and+Certification+WG.
[28] Carbon Emission Working Group. 2021. Hyperledger Working Groups—Blockchain Carbon Accounting. Retrieved January 10, 2022 from https://github.com/hyperledger-labs/blockchain-carbon-accounting.
[29] Christos E. Chalyvidis, Jeffrey A. Ogden, Alan W. Johnson, John M. Colombi, and Thomas C. Ford. 2016. A method for measuring supply chain interoperability. Supply Chain Forum 17, 4 (Jan. 2016), 246–258. https://doi.org/10.1080/16258312.2016.1247655 Available online https://www.tandfonline.com/doi/abs/10.1080/16258312.2016.1247655, last accessed on 2021-08-10.
[30] David Chen, Guy Doumeingts, and François Vernadat. 2008. Architectures for enterprise integration and interoperability: Past, present and future. *Computers in Industry* 59, 7 (Sept. 2008), 647–659. https://doi.org/10.1016/J.COMPIND.2007.12.016
[31] Coin Market Cap. 2021. All Coins. Retrieved August 10, 2021 from https://coinmarketcap.com/coins/views/all/.
[32] Composable Finance. 2021. Composable Finance Whitepaper. Retreived Dec. 21, 2021 from https://paper.composable.finance/.
[33] Connext. 2021. Welcome! | Connext Documentation. Retrieved August 10, 2021 from https://docs.connext.network/.
[34] Consensys. DeFi Report Q2 2021 | Consensys. Retrieved August 12, 2021 from https://consensys.net/reports/defi-report-q2-2021.
[35] Miguel Correia. 2019. From byzantine consensus to blockchain consensus. *Essentials of Blockchain Technology* (2019), 41.
[36] Gabriel da Silva Serapião Leal, Wided Guédria, and Hervé Panetto. 2019. Interoperability assessment: A systematic literature review. *Computers in Industry* 106 (Apr. 2019), 111–132. https://doi.org/10.1016/J.COMPIND.2019.01.002
[37] Philip Daian, Steven Goldfeder, Tyler Kell, Yunqi Li, Xueyuan Zhao, Iddo Bentov, Lorenz Breidenbach, and Ari Juels. 2020. Flash boys 2.0: Frontrunning in decentralized exchanges, miner extractable value, and consensus instability. In *Proceedings of the IEEE Symposium on Security and Privacy*. 1106–1120. https://doi.org/10.1109/SP40000.2020.00040
[38] DAML. 2021. Canton. Retrieved August 10, 2021 from https://www.canton.io/.
[39] Amrita Dhillon, Peter McBurney, Grammateia Kotsialou, and Luke Riley. 2020. Voting over a distributed ledger: An interdisciplinary perspective. *SocArXiv* (2020). https://doi.org/10.31235/osf.io/34df5
[40] DYdX. 2021. dYdX. Retrieved August 10, 2021 from https://dydx.exchange/.

[41] Shayan Eskandari, Mehdi Salehi, Wanyun Catherine Gu, and Jeremy Clark. 2021. SoK: Oracles from the ground truth to market manipulation. arXiv:2106.00667. https://doi.org/10.1145/3479722.3480994.
[42] Ethereum Foundation. 2021. Getting Started with Geth | Go Ethereum. Retrieved December 17, 2021 from https://geth.ethereum.org/docs/getting-started.
[43] Ethereum Foundation and Consensys. 2015. BTC-relay: Ethereum contract for Bitcoin SPV. Retrieved March 20, 2020 from https://github.com/ethereum/btcrelay.
[44] European Commission. 2021. NIFO—National Interoperability Framework Observatory, Interoperability Layers. Retrieved April 20, 2020 from https://joinup.ec.europa.eu/collection/nifo-national-interoperability-framework-observatory/3-interoperability-layers.
[45] Extropy.io. 2021. Price Oracle Manipulation | by Extropy.IO | Sep, 2021 | Medium. Retrieved September 17, 2021 from https://extropy-io.medium.com/price-oracle-manipulation-d46fd413cc17.
[46] Ghareeb Falazi, Uwe Breitenbücher, Florian Daniel, Andrea Lamparelli, Frank Leymann, and Vladimir Yussupov. 2020. Smart contract invocation protocol (SCIP): A protocol forthe uniform integration of heterogeneous blockchain smart contracts. *CAiSE 2020* 1 (2020), 134–149. https://doi.org/10.1007/978-3-030-49435-3
[47] Philipp Frauenthaler, Marten Sigwart, Christof Spanring, Michael Sober, and Stefan Schulte. 2020. ETH relay: A cost-efficient relay for ethereum-based blockchains. In *Proceedings of the 2020 IEEE International Conference on Blockchain (Blockchain 2020)*. 204–213. https://doi.org/10.1109/BLOCKCHAIN50366.2020.00032
[48] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. 2015. The bitcoin backbone protocol: Analysis and applications. In *Advances in Cryptology*, Vol. 9057. 281–310. https://doi.org/10.1007/978-3-662-46803-610
[49] Alberto Garoffolo, Dmytro Kaidalov, and Roman Oliynykov. 2020. Zendoo: A zk-SNARK verifiable cross-chain transfer protocol enabling decoupled and decentralized sidechains. In *Proceedings of the 2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS'20)*.
[50] Gartner. 2021. Blockchain Platforms Reviews 2021 | Gartner Peer Insights. (2021). Retrieved July 22, 2021 from https://www.gartner.com/reviews/market/blockchain-platforms.
[51] Sara Ghaemi, Sara Rouhani, Rafael Belchior, Rui S. Cruz, Hamzeh Khazaei, and Petr Musilek. 2021. A pub-sub architecture to promote blockchain interoperability. Submitted to *Computer Communications*. http://arxiv.org/abs/2101.12331. Retrieved August 10, 2021 from https://deepai.org/publication/a-pub-sub-architecture-to-promote-blockchain-interoperability.
[52] Caldarelli Giulio. 2021. *Wrapping Trust for Interoperability. A Study of Wrapped Tokens*. Technical Report. Department of Business Administration, University of Verona. Preprint available online https://arxiv.org/pdf/2109.06847v1.pdf.
[53] Pasquale Giungato, Roberto Rana, Angela Tarabella, and Caterina Tricase. 2017. Current trends in sustainability of bitcoins and related blockchain technology. *Sustainability* 9, 12 (11 2017), 2214. https://doi.org/10.3390/SU9122214
[54] Eliza Gkritsi and Muyao Shen. Cross-Chain DeFi Site Poly Network Hacked; Hundreds of Millions Potentially Lost | Nasdaq. Retrieved August 10, 2021 from https://www.nasdaq.com/articles/cross-chain-defi-site-poly-network-hacked-hundreds-of-millions-potentially-lost-2021-08-10.
[55] Jesus M. Gonzalez-Barahona. 2021. Factors Determining Maximum Energy Consumption of Bitcoin Miners. Retrieved August 10, 2021 from https://digiconomist.net/bitcoin-energy-consumption.
[56] Petter Gottschalk. 2009. Maturity levels for interoperability in digital government. *Government Information Quarterly* 26, 1 (1 2009), 75–81. https://doi.org/10.1016/J.GIQ.2008.03.003
[57] Blockchain Grants. Defi & NFT Blockchain Grants. Retrieved August 10, 2021 from https://www.blockchaingrants.org/home.
[58] T. Hardjono, M. Hargreaves, N. Smith, and V. Ramakrishna. 2021. *An Interoperability Architecture for Blockchain Gateways*. Internet-Draft draft-hardjono-blockchain-interop-arch-03. IETF. https://datatracker.ietf.org/doc/draft-hardjono-blockchain-interop-arch/.
[59] Martin Hargreaves and Thomas Hardjono. 2021. *Implementing a CBDC: The Challenges and A Solution*. Technical Report. Quant Network. https://www.quant.network/insights/implementing-a-cbdc-the-challenges-and-a-solution.
[60] Martin Hargreaves, Thomas Hardjono, and Rafael Belchior. 2021. *Open Digital Asset Protocol Draft 02*. Technical Report. draft-hargreaves-odap-02. IETF. https://datatracker.ietf.org/doc/html/draft-hargreaves-odap-02.
[61] Sandra Heiler. 1995. Semantic interoperability. *ACM Computing Surveys (CSUR)* 27 (1995), 271–273.
[62] Annegret Henninger and Atefeh Mashatan. 2021. Distributed interoperable records: The key to better supply chain management. *Computers* 10, 7 (7 2021), 89. https://doi.org/10.3390/COMPUTERS10070089
[63] Maurice Herlihy, Barbara Liskov, and Liuba Shrira. 2019. Cross-chain deals and adversarial commerce. *Very Large Databases* 13, 2 (2019), 100–113. https://doi.org/10.14778/3364324.3364326
[64] Hop Exchange. 2021. Hop Exchange. Retrieved January 10, 2022 from https://kovan.hop.exchange/send.
[65] Hyphen. 2021. Hyphen—Instant Cross-Chain Transfers—Biconomy. Retrieved January 10, 2022 from https://docs.biconomy.io/products/hyphen-instant-cross-chain-transfers?ref=hackernoon.com.
[66] Nancy Ide and James Pustejovsky. 2010. What does interoperability mean, anyway? Toward an operational definition of interoperability for language technology. *Conference on Global Interoperability* (2010). http://www.language-archives.org.
[67] Infura. 2021. Ethereum | Infura Documentation. Retrieved January 10, 2011 from https://infura.io/docs/ethereum.
[68] Interchain Foundation. Funding | Interchain Foundation. Retrieved January 1, 2022 from https://interchain.io/funding/.

[69] Interledger. 2020. Interledger Protocol V4 (ILPv4) | Interledger. Retrieved January 10, 2022 from https://interledger.org/rfcs/0027-interledger-protocol-4/.
[70] ISO. Requirements for establishing manufacturing enterprise process interoperability—Part 1: Framework for enterprise interoperability. Retrieved July 22, 2021 from https://www.iso.org/standard/50417.html.
[71] ISO - TC 307. 2020. *Blockchain and Distributed Ledger Technologies-Vocabulary (Draft ISO/TC 307/WG 1 N 783 ISO/TC)*. Technical Report. International Organization for Standardization. Available online https://www.iso.org/standard/73771.html.
[72] Sandra Johnson, Peter Robinson, and John Brainard. 2019. Sidechains and Interoperability. (March 2019). Retrieved January 10, 2022 from http://arxiv.org/abs/1903.04077.
[73] Niclas Kannengießer, Michelle Pfister, Malte Greulich, Sebastian Lins, and Ali Sunyaev. 2020. Bridges between Islands: Cross-chain technology for distributed ledger technology. In *Proceedings of the Hawaii International Conference on System Sciences*.
[74] Kiranbir Kaur, Sandeep Sharma, and Karanjeet Singh Kahlon. 2017. Interoperability and portability approaches in inter-connected clouds: A review. *ACM Computing Surveys* 50, 4 (2017). https://doi.org/10.1145/3092698
[75] Ralf Klischewski, Al Tagamoa, and Al Khames. 2004. Information integration or process integration? How to achieve interoperability in administration. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Vol. 3183. Springer. 57–65. https://doi.org/10.1007/978-3-540-30078-6_10
[76] T. Koens and E. Poll. 2019. Assessing interoperability solutions for distributed ledgers. *Pervasive and Mobile Computing* 59 (10 2019), 101079. https://doi.org/10.1016/J.PMCJ.2019.101079
[77] John Kolb, Moustafa Abdelbaky, Randy H. Katz, and David E. Culler. 2020. Core concepts, challenges, and future directions in blockchain: A centralized tutorial. *ACM Computing Surveys* 53, 1 (2 2020). https://doi.org/10.1145/3366370
[78] Jae Kwon and Ethan Buchman. 2016. *Cosmos Whitepaper*. Technical Report. Cosmos Foundation. Available online https://github.com/cosmos/cosmos/blob/master/WHITEPAPER.md.
[79] LACChain Consortium. 2021. LACChain. Retrieved August 10, 2021 from https://www.lacchain.net/home.
[80] Hasnae L'Amrani, Badr Eddine Berroukech, Younès El Bouzekri El Idrissi, and Rachida Ajhoun. 2017. Toward interoperability approach between federated systems. In *Proceedings of the 2nd International Conference on Big Data, Cloud and Applications*. https://doi.org/10.1145/3090354.3090391
[81] Rongjian Lan, Ganesha Upadhyaya, Stephen Tse, and Mahdi Zamani. 2021. Horizon: A gas-efficient, trustless bridge for cross-chain transactions. (Jan. 2021). Retrieved January 10, 2022 from http://arxiv.org/abs/2101.06000.
[82] Sergio Lerner. 2015. *RSK Whitepaper*. Technical Report. RSK. https://docs.rsk.co/RSK_White_Paper-Overview.pdf. Available online https://docs.rsk.co/RSK_White_Paper-Overview.pdf.
[83] Wenjuan Li and Lingdi Ping. 2009. Trust model to enhance security and interoperability of cloud environment. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Vol. 5931. Springer. 69–79. https://doi.org/10.1007/978-3-642-10665-1_7
[84] Alexander Lipton. 2021. Cryptocurrencies change everything. *Quantitative Finance* 21, 8 (2021), 1257–1262. https://doi.org/10.1080/14697688.2021.1944490
[85] Avivah Litan and Adrian Leow. Hype Cycle for Blockchain, 2021. Retrieved July 22, 2021 from https://www.gartner.com/en/documents/4003463/hype-cycle-for-blockchain-2021.
[86] Witold Litwin, Leo Mark, and Nick Roussopoulos. 1990. Interoperability of multiple autonomous databases. *ACM Computing Surveys (CSUR)* 22, 3 (Sept. 1990), 267–293. https://doi.org/10.1145/96602.96608
[87] Ankur Lohachab, Saurabh Garg, Byeong Kang, Muhammad Bilal, Junmin Lee, Shiping Chen, and Xiwei Xu. 2021. Towards interconnected blockchains: A comprehensive review of the role of interoperability among disparate blockchains. *ACM Computing Surveys (CSUR)* 54, 7 (July 2021), 1–39. https://doi.org/10.1145/3460287
[88] Francesco Longo, Letizia Nicoletti, Antonio Padovano, Gianfranco D'Atri, and Marco Forte. 2019. Blockchain-enabled supply chain: An experimental study. *Computers & Industrial Engineering* 136 (Oct. 2019), 57–69. https://doi.org/10.1016/J.CIE.2019.07.026
[89] Jack Lu, Boris Yang, Zane Liang, Ying Zhang, Shi Demmon, Eric Swartz, and Lizzie Lu. 2017. Wanchain: Building Super Financial Markets for the New Digital Economy. (2017), 34 pages. https://wanchain.org/files/Wanchain-Whitepaper-EN-version.pdf.
[90] Iulia Mihaiu, Rafael Belchior, Sabrina Scuri, and Nuno Nunes. 2021. A Framework to Evaluate Blockchain Interoperability Solutions. (2021). Retrieved January 10, 2022 from https://doi.org/10.36227/techrxiv.17093039.v1
[91] Igor Mikhalev, Kaj Burchardi, Igor Struchkov, Bihao Song, and Jonas Gross. 2021. Central Bank Digital Currency (CBDC) Tracker. (2021). Retrieved January 10, 2022 from https://cbdctracker.org/cbdc-tracker-whitepaper.pdf.
[92] Hart Montgomery, Hugo Borne-Pons, Jonathan Hamilton, Mic Bowman, Peter Somogyvari, Shingo Fujimoto, Takuma Takeuchi, Tracy Kuhrt, and Rafael Belchior. 2020. *Hyperledger Cactus Whitepaper*. Technical Report. Hyperledger Foundation. https://github.com/hyperledger/cactus/blob/master/docs/whitepaper/whitepaper.md. Available online https://github.com/hyperledger/cactus/blob/master/docs/whitepaper/whitepaper.md.
[93] Terry Moon, Suzanne Fewell, and Hayley Reynolds. 2008. The what, why, when and how of interoperability. *Defence & Security Analysis* 24, 1 (2008), 5–17. https://doi.org/10.1080/14751790801903178

[94] Roman Mühlberger, Stefan Bachhofner, Eduardo Castelló Ferrer, Claudio Di Ciccio, Ingo Weber, Maximilian Wöhrer, and Uwe Zdun. 2020. Foundational oracle patterns: Connecting blockchain to the off-chain world. In *Lecture Notes in Business Information Processing*, Vol. 393. Springer Science and Business Media Deutschland GmbH, 35–51. https://doi.org/10.1007/978-3-030-58779-6_3
[95] MYDA. 2021. MYDA—Whitepaper. Retrieved January 10, 2022 from https://www.mydacoin.com/whitepaper.
[96] Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. Retrieved on January 10, 2022 from https://bitcoin.org/en/bitcoin-paper.
[97] WBTC Network. 2021. Wrapped Bitcoin (WBTC) an ERC20 token backed 1:1 with Bitcoin. Retrieved January 10, 2022 from https://wbtc.network/.
[98] Pancake Swap. 2021. Home | PancakeSwap - $11.911. (2021). Retrieved January 10, 2022 from https://pancakeswap.finance/.
[99] Yan Pang. 2020. A new consensus protocol for blockchain interoperability architecture. *IEEE Access* 8 (2020), 153719–153730.
[100] Amirmohammad Pasdar, Zhongli Dong, and Young Choon Lee. 2021. Blockchain oracle design patterns. arXiv:2106.09349. Retrieved from https://scholar.google.com.au/.
[101] Linda Pawczuk, Margi Gogh, and Nadia Hewett. 2020. *Inclusive Deployment of Blockchain for Supply Chains: Part 6—A Framework for Blockchain Interoperability. Part 6—A Framework for Blockchain Interoperability in Collaboration with Deloitte.* Technical Report. World Economic Forum.
[102] Linda Pawczuk, Roy Massey, and Jonathan Holdowsky. Deloitte's 2019 Global Blockchain Survey. Retrieved on January 10, 2022 from https://www2.deloitte.com/content/dam/insights/us/articles/2019-global-blockchain-survey/DI_2019-global-blockchain-survey.pdf.
[103] Babu Pillai, Kamanashis Biswas, Zhé Hóu, and Vallipuram Muthukkumarasamy. 2022. Level of conceptual interoperability model for blockchain based systems. In *2022 IEEE Crosschain Workshop (ICBC-CROSS'22)*. IEEE, 1–7.
[104] Babu Pillai, Kamanashis Biswas, and Vallipuram Muthukkumarasamy. 2020. Cross-chain interoperability among blockchain-based systems using transactions. *Knowledge Engineering Review* 35 (2020), 1–18. https://doi.org/10.1017/S0269888920000314
[105] Polkadot. Glossary · Polkadot Wiki. Retrieved January 10, 2022 from https://wiki.polkadot.network/docs/glossary.
[106] Polkadot. Polkadot Bridges—Connecting the Polkadot Ecosystem with External Networks. Retrieved January 10, 2022 from https://polkadot.network/blog/polkadot-bridges-connecting-the-polkadot-ecosystem-with-external-networks/.
[107] Polkadot. 2019. Cross-chain Message Passing (XCMP) · Polkadot Wiki. Retrieved January 10, 2022 from https://wiki.polkadot.network/docs/en/learn-crosschain.
[108] Polkadot. 2021. paritytech/polkadot: Polkadot Node Implementation. Retrieved January 10, 2022 from https://github.com/paritytech/polkadot.
[109] Polygon. 2021. Polygon | Ethereum's Internet of Blockchains. Retrieved January 10, 2022 from https://polygon.technology/.
[110] Benedikt Putz, Marietheres Dietz, Philip Empl, and Günther Pernul. 2021. EtherTwin: Blockchain-based secure digital twin information management. *Information Processing & Management* 58, 1 (Jan. 2021), 102425. https://doi.org/10.1016/J.IPM.2020.102425
[111] Ilham A. Qasse, Manar Abu Talib, and Qassim Nasir. 2019. Inter blockchain communication: A survey. In *Proceedings of the ArabWIC 6th Annual International Conference Research Track*.
[112] Minfeng Qi, Ziyuan Wang, Donghai Liu, Yang Xiang, Butian Huang, and Feng Zhou. 2020. ACCTP: Cross chain transaction platform for high-value assets. In *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Vol. 12404. Springer Science and Business Media Deutschland GmbH, 154–168. https://doi.org/10.1007/978-3-030-59638-5_11
[113] Quant. 2021. Overledger Network for Enterprise. (2021). Retrieved January 10, 2022 from https://www.quant.network/overledger-network-for-enterprise.
[114] Luke Riley. 2021. Universal DLT interoperability is now a practical reality. Retrieved January 10, 2022 from https://www.hyperledger.org/blog/2021/05/10/universal-dlt-interoperability-is-now-a-practical-reality#.YKzdCEdGRmw.twitter.
[115] Peter Robinson. 2021. Survey of crosschain communications protocols. *Computer Networks* 200 (Dec. 2021), 108488. https://doi.org/10.1016/J.COMNET.2021.108488
[116] Robinson Peter, Ramesh Raghavendra, and Johnson Sandra. 2022. Atomic crosschain transactions for ethereum private sidechains. *Blockchain: Research and Applications* 3, 1 (2022), 100030.
[117] Peter Robinson and Raghavendra Ramesh. 2020. General purpose atomic crosschain transactions. http://arxiv.org/abs/2011.12783.
[118] Eder J. Scheid, Timo Hegnauer, Bruno Rodrigues, and Burkhard Stiller. 2019. Bifröst: A modular blockchain interoperability API. In *IEEE 44th Conference on Local Computer Networks*. IEEE, 332–339.
[119] Amritraj Singh, Kelly Click, Reza M. Parizi, Qi Zhang, Ali Dehghantanha, and Kim Kwang Raymond Choo. 2020. Sidechain technologies in blockchain networks: An examination and state-of-the-art review. *Journal of Network and Computer Applications* 149 (2020).
[120] Vasilios A. Siris, Pekka Nikander, Spyros Voulgaris, Nikos Fotiou, Dmitrij Lagutin, and George C. Polyzos. 2019. Interledger approaches. *IEEE Access* 7 (2019), 89948–89966.
[121] Meriem Thabet, Mahmoud Boufaida, and Fabrice Kordon. 2014. An approach for developing an interoperability mechanism between cloud providers. *International Journal of Space-Based and Situated Computing* 4, 2 (2014), 88. https://doi.org/10.1504/IJSSC.2014.062469
[122] TO Group. 2016. *ArchiMate® 3.0 Specification.* Van Haren Publishing.
[123] Andreas Tolk and James A. Muguira. 2003. The levels of conceptual interoperability model. In *Simulation Interoperability Workshop*.

[124] Gilbert Verdian, Paolo Tasca, Colin Paterson, and Gaetano Mondelli. 2018. *Quant Overledger Whitepaper v0.1.* Technical Report. Quant. 1–48 pages. http://objects-us-west-1.dream.io/files.quant.network/Quant_Overledger_Whitepaper_v0.1.pdf.

[125] César Viho, Sébastien Barbin, and Lénaïck Tanguy. 2001. Towards a formal framework for interoperability testing. *Formal Techniques for Networked and Distributed Systems* (Feb. 2001), 53–68. https://doi.org/10.1007/0-306-47003-9_4

[126] Hoang Tam Vo, Ashish Kundu, and Mukesh Mohania. 2018. Research directions in blockchain data management and analytics. *Advances in Database Technology—EDBT* 2018-March, 445–448. https://doi.org/10.5441/002/edbt.2018.43

[127] Weaver. Weaver Interoperability RFCs. Retrieved January 10, 2022 from https://github.com/hyperledger-labs/weaver-dlt-interoperability/tree/main/rfcs.

[128] Peter Wegner. 1996. Interoperability. *ACM Computing Surveys* 28, 1 (1996).

[129] Gavin Wood. 2017. Polkadot: Vision for a heterogeneous multi-chain framework. *Whitepaper* (2017), 1–21. https://github.com/w3f/polkadot-white-paper/raw/master/PolkaDotPaper.pdf.

[130] World Bank Group. 2020. Blockchain Interoperability. Retrieved August 10, 2021 from https://www.ft.com/content/1cfb6d46-5d5a-11e9-939a-341f5ada9d40.

[131] World Economic Forum. Global Standards Mapping Initiative: An Overview of Blockchain Technical Standards | World Economic Forum. Retrieved January 10, 2022 from https://www.weforum.org/whitepapers/global-standards-mapping-initiative-an-overview-of-blockchain-technical-standards.

[132] World Economic Forum. 2015. Global Agenda Council on the Future of Software & Society Deep Shift Technology Tipping Points and Societal Impact. Retrieved January 10, 2022 from https://www3.weforum.org/docs/WEF_GAC15_Technological_Tipping_Points_report_2015.pdf.

[133] World Economic Forum. 2020. *Bridging the Governance Gap: Interoperability for Blockchain and Legacy Systems.* Technical Report. World Economic Forum. https://www.weforum.org/whitepapers/bridging-the-governance-gap-interoperability-for-blockchain-and-legacy-systems.

[134] World Economic Forum. 2021. *Defining Interoperability —Digital Currency Governance Consortium White Paper Series.* Technical Report. World Economic Forum. https://www.weforum.org/reports/digital-currency-governance-consortium-white-paper-series.

[135] Karl Wüst and Arthur Gervais. 2018. Do you need a blockchain?. In *Proceedings of the 2018 Crypto Valley Conference on Blockchain Technology (CVCBT'18).* 45–54. https://doi.org/10.1109/CVCBT.2018.00011

[136] Xiwei Xu, Cesare Pautasso, Liming Zhu, Vincent Gramoli, Alexander Ponomarev, An Binh Tran, and Shiping Chen. 2016. The blockchain as a software connector. In *Proceedings of the 2016 13th Working IEEE/IFIP Conference on Software Architecture (WICSA'16).* 182–191. https://doi.org/10.1109/WICSA.2016.21

[137] Alexei Zamyatin, Mustafa Al-Bassam, Dionysis Zindros, Eleftherios Kokoris-Kogias, Pedro Moreno-Sanchez, Aggelos Kiayias, and William J. Knottenbelt. 2019. SoK: Communication across distributed ledgers. International Conference on Financial Cryptography and Data Security, Springer, 3–36.

[138] Alexei Zamyatin, Dominik Harz, Joshua Lind, Panayiotis Panayiotou, Arthur Gervais, and William J. Knottenbelt. 2019. XCLAIM: A framework for blockchain interoperability. In *IEEE Symposium on Security & Privacy.*

[139] Ivana Podnar Zarko, Sergios Soursos, Ivan Gojmerac, Elena Garrido Ostermann, Gianluca Insolvibile, Marcin Plociennik, Peter Reichl, and Giuseppe Bianchi. 2017. Towards an IoT framework for semantic and organizational interoperability. In *Proceedings of the Global Internet of Things Summit*. https://doi.org/10.1109/GIOTS.2017.8016253

Received 23 January 2022; revised 21 July 2022; accepted 16 September 2022

3 BUNGEE: Dependable Blockchain Views for Interoperability

The following chapter corresponds to the following publication [4]:

Status: Published ✓

Publication: Distributed Ledger Technologies: Research and Practice (ACM DLT)

Submitted: 28 January 2023

Accepted: 22 January 2024

Citation: Rafael Belchior, Limaris Torres, Jonas Pfannschmidt, André Vasconcelos, and Miguel Correia. 2024. BUNGEE: Dependable Blockchain Views for Interoperability. Distrib. Ledger Technol. 3, 1, Article 7 (March 2024), 25 pages. https://doi.org/10.1145/3643689

Research Methodology: Design Science Research Methodology [145]

Evaluation Methodology: Formal Proofs [147], Qualitative Evaluation [149]

Journal Description: Distributed Ledger Technologies: Research and Practice (DLT) is a peer-reviewed journal that seeks to publish high-quality, interdisciplinary research on the research and development, real-world deployment, and evaluation of distributed ledger technologies, such as blockchain, cryptocurrency, and smart contract. DLT will offer a blend of original research work and innovative practice-driven advancements by internationally distinguished DLT experts and researchers from academia, and public and private sector organizations.

BUNGEE: Dependable Blockchain Views for Interoperability

RAFAEL BELCHIOR, INESC-ID, Instituto Superior Técnico, Universidade de Lisboa, Portugal and Blockdaemon, Portugal
LIMARIS TORRES, Independent Researcher, United States
JONAS PFANNSCHMIDT, Blockdaemon, Ireland
ANDRÉ VASCONCELOS, INESC-ID, Instituto Superior Técnico, Universidade de Lisboa, Portugal
MIGUEL CORREIA, INESC-ID, Instituto Superior Técnico, Universidade de Lisboa, Portugal

With the evolution of distributed ledger technology (DLT), several blockchains that provide enhanced privacy guarantees and features, including Corda, Hyperledger Fabric, and Canton, are being increasingly adopted. These distributed ledgers only provide partial consistency, meaning that participants can observe the same ledger differently, i.e., observe some transactions but not others, providing higher levels of privacy to the end-user.

Choosing privacy instead of transparency leads to delicate trade-offs that are difficult to manage during runtime, hampering the development of applications that depend on reasoning about shared state, e.g., asset transfers across blockchains. We propose using the concept of *blockchain view* (view) – an abstraction of the state a participant can access at a certain point to address this problem. Views allow us to systematically reason about either state partitions within the same DLT or an integrated view spanning across several DLTs. We introduce BUNGEE (Blockchain UNifier view GEnErator), the first DLT view generator, to allow capturing snapshots, constructing views from these snapshots, and merging views according to a set of rules specified by the view stakeholders. Creating views and operating views allows new applications built on top of dependable blockchain interoperability, such as stakeholder-centric snapshots for audits, cross-chain analysis, blockchain migration, and combined on-chain-off-chain analytics.

ACM Reference Format:
Rafael Belchior, Limaris Torres, Jonas Pfannschmidt, André Vasconcelos, and Miguel Correia. 2018. BUNGEE: Dependable Blockchain Views for Interoperability. *J. ACM* 37, 4, Article 111 (August 2018), 27 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Blockchains[1] provide trustworthy and transparent services, leveraging a network of mutually untrusting participants. A highly desirable property of DLTs is *consistency* [1]: the guarantee that all honest parties share a common prefix of the blockchain, i.e., they see the same *transactions* registered in the ledger. Based on this property, each ledger holds a single source of truth for

[1]We use the terms DLT and blockchain interchangeably. A DLT subsumes a blockchain, i.e., a blockchain is a DLT.

Authors' addresses: Rafael Belchior, INESC-ID, Instituto Superior Técnico, Universidade de Lisboa, Rua Alves Redol, 9, Lisboa, Portugal, 1000-029 and Blockdaemon, Lisboa, Lisboa, Portugal, 0000-0000, rafael.belchior@tecnico.ulisboa.pt; Limaris Torres, Independent Researcher, , Miami, United States, 0000-0000, limarist@gmail.com; Jonas Pfannschmidt, Blockdaemon, Dublin, Dublin, Ireland, 0000-0000, jonas@blockdaemon.com; André Vasconcelos, INESC-ID, Instituto Superior Técnico, Universidade de Lisboa, Rua Alves Redol, 9, Lisboa, Portugal, 1000-029, andre.vasconcelos@tecnico.ulisboa.pt; Miguel Correia, INESC-ID, Instituto Superior Técnico, Universidade de Lisboa, Rua Alves Redol, 9, Lisboa, Portugal, 1000-029, miguel.p.correia@tecnico.ulisboa.pt.

0004-5411/2018/8-ART111 $15.00
https://doi.org/XXXXXXX.XXXXXXX

all its *participants*: consistency is the foundation of the decentralized trust that DLTs offer[2]. The view concept has its roots in database schema integration and, more recently, in business process view integration [2]. To account for the multitude of business process views, *business view process integration* (BVPI) studies the consolidation of different views regarding a business process [2, 3]. Business view process integration (BPVI) addresses the challenges of processes that involve several participants with different incentives, alleviating them by merging models that represent a different view of the same model.

Despite the importance of consistency, some permissioned DLTs offer only *partial consistency*, providing a trade-off between transparency and privacy. Partial consistency is a weaker notion of consistency that implies that honest parties can read only subsets of the same global transaction graph, i.e., of the ledger. For every transaction ID a set of parties share, they also agree on the contents and dependencies of such transaction [4]. Partial consistent blockchains are very useful in enterprise-grade applications, where privacy and accountability are paramount and necessary to enforce [5]. Organizations working with DLTs that provide partial consistency have been putting resources in place to enable *interoperability* with other blockchains, following the growing trend in the space to accommodate DLTs offering different properties and features [6, 7].

A problem naturally emerges from a multichain ecosystem. Since participants might have different views of a chain, see different data partitions[3].

A solution to address interoperability across blockchains is to trust a centralized third party [9], such as a cryptocurrency exchange. However, we have seen that such platforms have been consistently attacked and even defrauded its users (see the FTX crash [10], and many more [11, 12]). Trusting a decentralized protocol could be the way forward. However, many decentralized protocols useful for interoperability have few privacy-protecting mechanisms [7, 13] and are still immature and insecure [14]. However, there are alternatives to conventional decentralized bridge designs. A promising one is systems based on zero-knowledge proofs [15**?** , 16]. Despite having potential to be reasonable long-term solutions, these are two shortcomings: 1) the technology is very recent, and thus it is immature and not hard-tested (e.g., zk-rollup technology [17]) and 2) it requires significant engineering effort, to the point of even creating a new blockchain from scratch (see ZCash, Monero [18]).

We then search for a balanced approach that includes support for both centralized and decentralized protocols. A view offers a stakeholder-centric, generalizable, self-describing commitment to the state of a blockchain, allowing for representing states from different blockchains in a standardized way. Blockchain views are created, processed, and shared by consortium participants that are legally identified and have contractual obligations to follow a protocol, namely *blockchain gateways* [19, 20]. Gateways are trusted systems that run an interoperability protocol [21] and can use blockchain views as proof of state for asset and data transfers. Despite being trusted and therefore suitable for enterprises, several decentralization features in gateways promote accountability among consortium members. In this context, building and analyzing views is important to understand each stakeholder's view of each DLT accurately as a tool for dependable *interoperability* across heterogeneous systems.

[2]A similar assumption could be extended to business processes, where a single model can capture simple processes. However, different representations of the same process are possible as soon as its complexity increases.

[3]Note that is different from the traditional database field, where different views exist and are processed according to a different number of methods [8]. In particular, we are interested in the problem of managing views of a decentralized system, where access control and data management (who can see what) are decentralized. This has implications for governance and privacy, as no single authority manages the view creation, processing, and sharing processes. Furthermore, operations in the decentralized database must generate and share proof that such an operation is valid.

Problem Definition and Research Questions

Let us consider a blockchain consortium c made up of n nodes and v different views. The research problem we propose to solve is to find a view v' that is agreed by a subset of n, includes a subset of v, and respects a privacy policy defined in c. To answer this problem, we divide it into several research questions (RQs):

- *RQ 1. How to provide a data format for views that balance the characteristics of centralized and decentralized systems?*

  Multiple DLT data formats result from their architecture, consensus, and identity models. Formalizing the blockchain view and related concepts is necessary to clarify data representation across chains. *Motivation*: the absence of standards in the interoperability area is a well-known concern [22–24]. Providing a standardized data format would enable the industry and academia to converge to more efficient design patterns, enhancing organizational and legal interoperability.

- *RQ 2. How to guarantee privacy-preserving properties in blockchain interoperability?*

  This contribution addresses that need by creating views that allow one to see a stakeholder's perspective over the entire ledger. However, how do we obtain a holistic view of a DLT providing partial consistency, i.e., combined perspectives of all participants, according to privacy restrictions? We ensure that the view's creation, merging, and processing come with privacy, integrity, and accountability guarantees. *Motivation*: Recent research highlights the need and inexistence of privacy-preserving mechanisms in the cross-chain setting as a priority for next-generation interoperable systems [25–27]. Providing privacy-preserving capabilities for cross-chain solutions would enable a new range of users, including enterprises, governments, and organizations that transact sensitive data.

By answering these research questions, we expect to analyze, model, design, and provide implementation guidelines for systems generating views, making it easier to reason about systems interacting with several blockchains, hence delivering the following contributions:

Contributions

- We define the concept of *blockchain view*. We present a formalization of concepts surrounding the view, rooted in the state abstraction and causality relationships between transactions, states, and views. This formalization is the foundation to specify systems handling views, and also for practical implementations.
- We specify the first view integrator, *Blockchain UNifier view GEnErator* (BUNGEE). BUNGEE is a flexible, modular middleware that sits between the data and the semantic layers of a blockchain, allowing data to be abstracted into different data models and formats. To the best of our knowledge, this is the first time views are used to take stakeholder-specific snapshots of the ledger, allowing for several applications.
- We specify BUNGEE's algorithms to create, merge, and process a view, providing a comprehensive discussion of decentralization, efficiency, and privacy trade-offs.

Paper Outline

This document is organized as follows: Section 2 introduces the background necessary to comprehend this paper. In Section 3 we formalize the blockchain view and related concepts. Next, Section

4 presents BUNGEE. After that, we present a discussion in Section 5. Next, we present the related work, in Section 6. Finally, we conclude the paper, in Section 7.

2 PRELIMINARIES

This section presents the background necessary to understand the paper, along with motivational use cases.

Blockchains Providing Partial Consistency. Blockchains providing partial consistency create partitions over the global state according to some criteria. Private blockchains require their participants to be authenticated and only expose their content to trusted parties (although those parties do not necessarily trust each other). In private blockchains, different views are common and desirable for privacy reasons [5]. Parties may want to share information with a selected group.

For example, Hyperledger Fabric (Fabric), a private blockchain framework, provides a feature called private data collections. Private data allows sets of participants to hide part of the state they hold, only sharing a hash of that private data as proof of existence [28, 29]. This feature effectively implements partial consistency in Fabric, allowing for the existence of different views. In Corda [30], transactions are ordered as a set of (potentially) disconnected directed acyclic graphs – parties can access certain subgraphs, i.e., Corda provides partial consistency. Other examples exist, such as Quorum [31], IOTA [32] and Digital Asset's Canton [33]. Figure 1 shows a visualization of the concept.

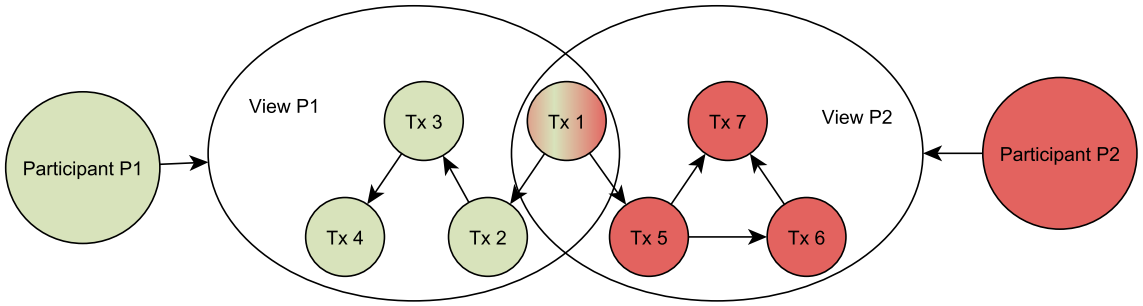


Fig. 1. Two different participant views over the same DLT. Tx stands for a transaction. A green or red labeled transaction is available (read access) to Participant P1 or Participant P2, respectively. Transaction Tx1 is available for both participants.

Blockchain Interoperability. The emergence of many blockchains raised the debate about the need for interoperability [7, 13]. Interoperability can be defined as the ability of multiple parties to work together by sharing/exchanging information [34].

The first use case for interoperability is cross-chain state creation, management, and visualization. While some preparatory work has been done [14], it is hard to visualize and reason about private data partitions (different views), not only in the cross-chain setting but also in a single blockchain setting. Blockchain platforms could leverage views to improve view analysis for auditors, cybersecurity experts, and developers. Auditors and cybersecurity professionals can facilitate audits [35] because different data partitions can be analyzed from a specific angle. Developers can gain insight into their applications and processes. Representing on-chain data through a DLT view in multiple chains allows for a visualization of the cross-chain state, making it easier to manage and reason. A specific application could be having one view across multiple Cosmos zones, Polkadot parachains, or Layer 2 solutions (Polygon, Arbitrum, and others, for instance) [36].

The second use case is decentralized application migration. Migration of blockchain-based applications is necessary and increasingly common [13, 37, 38]. Migration allows enterprises to experiment with other DLT infrastructures without the risk of vendor lock-in. The key idea behind application migration is to capture the DLT state relevant to that application (data and functionality) and move it to a different DLT infrastructure. With several views on participants' concerns operating on the source blockchain, one might need to consolidate their diverse views into an integrated view that serves as the foundation of the migration. The integrated view comprises a holistic view of the application's state at the source DLT. This view can then be transferred to the target infrastructure and its functionality (i.e., smart contract migration). We leave the treatment of this interesting problem and its details (for example, how to manage user keys) for future work.

Finally, the third use case is to allow asset transfers between chains [20]. Transferring assets between public DLTs and centralized systems (or private DLTs) is hard because it relies on strong trust assumptions or transparency. An asset transfer is typically implemented by locking an asset in the source chain and unlocking it in the target chain. However, if one of the chains is private (or centralized), such a state is not visible (by design) [39]. Therefore, decentralized transfers across these types of system rely on proofs (or, rather, a notarization) of the current state of each chain concerning the representation of an asset [40]. Blockchain gateways that perform asset transfers need to translate the state of assets "to some DLT-neutral standard that other gateways can interpret and then hand over to the networks they are acting on behalf of" [41] - the DLT-neutral standard is the view.

Thus, blockchain views can be the bridge that allows decentralized blockchain interoperability across heterogeneous systems by representing such notarization on a public forum [42, 43], with a standardized data format, independent of any specific blockchain implementation. A related use case to this would be to build a cross-chain wallet that, giving a private key, outputs all tokens in all blockchains associated with that wallet. This could be particularly useful for people with LUNA tokens spread across several blockchains, especially after the value of the coin plummeted [44].

3 BLOCKCHAIN VIEWS

This section introduces a running example that applies view integration to the supply chain industry. After that, we formalize concepts related to the view, such as the access point, blockchain view, and view generator.

3.1 Running Example

We present a typical use case on private blockchains, supply chain [45], that benefits from representing the various internal views to an external observer.

A supply chain transfers value between parties, from the raw product (physical or intellectual) to its finalized version. Managing a supply chain is complex because it includes many nontrusting participants (e.g., enterprises and regulators), and implies keeping an audit trail of all operations. As many markets are open and fluid, companies do not take the time to build trust and instead rely on a paper trail that logs the state of an object in the supply chain. This paper trail is necessary for auditability and can typically be tampered with, which leads to the suitability of blockchain to address these problems by monitoring the execution of the collaborative process. Blockchain smart contracts can ensure that the execution of the process complies with defined business rules [46, 47].

Audits inspect the trail of transactions referring to a product's lifecycle. Therefore, different perspectives might need to be analyzed. A challenge naturally emerges: balancing the necessary transparency for audits while maintaining privacy about the transactions across other business partner groups is not trivial. By selectively sharing a common domain, parties can have more

efficient processes while performing data-sensitive operations within the same supply chain. A domain is a state shared by parties enrolled in a private relationship.

Let us consider a group of five organizations on a Hyperledger Fabric blockchain that produces, transport, and trade:

- A Supplier, producing goods.
- A Shipper, moving goods between parties.
- A Distributor, moving goods abroad. Buys goods from Suppliers and sells them to Wholesalers.
- A Wholesaler, acquiring goods from the Distributor.
- A Retailer, acquiring goods from shippers and wholesalers.

The Distributor may prefer to make private transactions with the Supplier and the Shipper to keep confidentiality towards the Wholesaler and Retailer (hiding their profit margins). On the contrary, the Distributor may want a different relationship with the Wholesaler. It charges them a lower price than it does with the Retailer (as it sells assets in bulk). The Wholesaler may want to share the same data with the Retailer and the Shipper (because the Wholesaler may charge the Retailers a higher price than the Shipper). We call these private relationships *domains*. The combination of all domains represents the whole ledger.

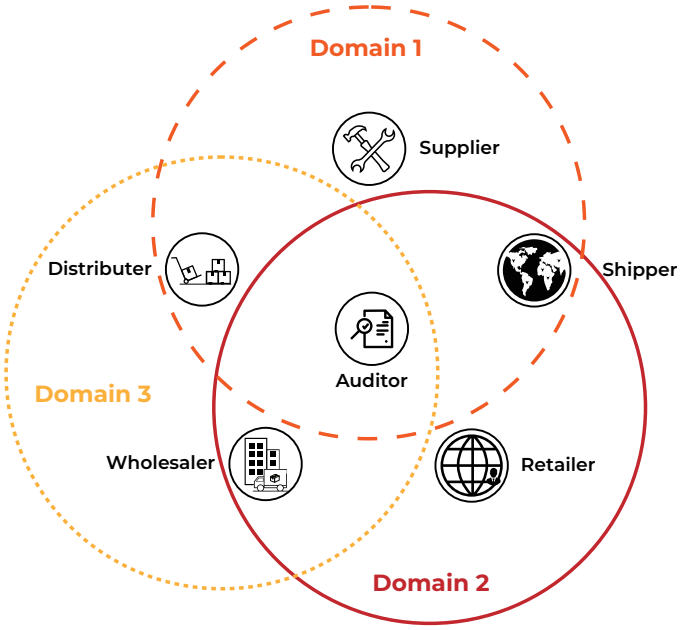


Fig. 2. Different domains on the blockchain supporting the supply chain scenario. For instance, the Supplier has access to domain 1, while the Shipper accesses domains 1 and 2. Access to different domains leads to the creation of different views.

Domains hold a subset of the ledger that is only accessible by authorized parties. By sectioning the shared ledger, different views on the same blockchain are possible, depending on a stakeholder's participation in a given domain, as shown in Table 1. In this table, the asset ID is one across all domains. However, its price differs across domains, translating into different views. For instance, the Supplier has access to the asset's price on d_1, but only access to its price's hash on d_2 and d_3 (i.e., does not have access to the price on d_2 and d_3. This three-dimensional tuple access-deny-deny corresponds to v_1. Retailers' view, v_5 can see the asset's price only in d_2. The three existing domains (see Figure 2, translate into three different price values for the same item – five participants originate five different views.

Now, assume that we identify each asset tracked in a supply chain by an ID and a price. For the same asset (and thus the same state on the blockchain, as it is uniquely identifiable by its ID), the Distributor-Supplier-Shipper (Domain 1, d_1) has the same view of the price, but the Wholesaler-Retailer-Shipper (Domain 2, d_2) and Distributor-Wholesaler (Domain 3, d_3) and have different views. As every stakeholder has a different combination of the domains that are accessible, the DLT infrastructure yields five different views.

Participant\Domain	d_1	d_2	d_3	
Supplier	ID: 1 Price: 1	ID: 1 Price: hidden	ID: 1 Price: hidden	v_1
Shipper	ID: 1 Price: 1	ID: 1 Price: 2	ID: 1 Price: hidden	v_2
Distributor	ID: 1 Price: 1	ID: 1 Price: hidden	ID: 1 Price: 3	v_3
Wholesaler	ID: 1 Price: hidden	ID: 1 Price: 2	ID: 1 Price: 3	v_4
Retailer	ID: 1 Price: hidden	ID: 1 Price: 2	ID: 1 Price: hidden	v_5

Table 1. Participant views on the supply-chain blockchain regarding an asset with ID = 1.

Let us now imagine that an auditor wants to inspect the Distributor's operations regarding an asset. The auditor would retrieve blockchain snapshots in light of each participant's view. After that, the auditor can analyze each view from the perspective of each participant. If a general picture is needed, all views can be merged into an integrated one and jointly analyzed. Since there are different viewpoints, there are different prices for the same object, and different merge procedures are possible. This is not a trivial aspect to deal with, since views can originate from *multiple* decentralized systems, and therefore, aspects such as governance, access control, and privacy-preserving mechanisms are not easily managed by a centralized party.

In particular, the different views are translated into an integrated view that refers only to a consolidated price as a summary of the prices of the different views. The processing and the merging of the views are the consortium's responsibility for managing the blockchain, and thus, several options are possible. We will address this point later in this paper.

3.2 Formalizing Views

This section formally defines the terms necessary for blockchain view integration. We provide the conceptual framework to build programs that can merge blockchain views. The first concept of our framework is the ledger. A ledger is a simple key-value database with two functionalities: `read` and `store`. It supports a state machine that implements a DLT. We define the ledger as follows:

Definition 1. *Ledger.* A ledger $\mathcal{L}$ is a tuple $(\mathcal{D}, \mathcal{A})$ such that:

- $\mathcal{D}$ is a database, specifically a key-value store. Each entry in the database is a key-value tuple, i.e., $d \in \mathcal{D} : (k, v)$, where k stands for key and v for value.
  $\mathcal{D}$ has two functions: `read` and writes (storing). `read` returns the value associated with a key, the empty set $\emptyset$ (if there is no value for that key) or an error $\perp$ (if the user does not have access permissions), i.e., `read` $\rightarrow \{v, \emptyset, \perp\}$. The `store` primitive saves the (k, v) pair in the database, indexed by k, returning 1 if the operation was successful and 0 otherwise, i.e., `store` $: k \times v \rightarrow \{0, 1\}$.

The `read` and `store` primitives support the representation of simple UTXO blockchains (e.g., Bitcoin) [48] or more complex ones, an account model (e.g., Ethereum) [49], or others by combining the operations mentioned above (e.g., Hyperledger Fabric [50]).

- $\mathcal{A}$ is an access control list that specifies access rights to read entries from the database. Each entry in the list has the form (p, k). Each entry indicates that the participant p can read the state with key k. A participant p can access a key k when the primitive `access`$(\mathcal{A}, p, k)$ returns 1, or 0 otherwise.

The simple functionality of the notion of ledger given in Definition 1 allows us to represent Bitcoin [51] as follows: the database (collection of all states) is a list of UTXO entries (states). A UTXO has a unique identifier, the transaction hash, and the state key (we present a simplified version of UTXO). Its value is in the form *(input, output, metadata)*. The input corresponds to a reference to the previous transaction and a key to unlock the previous output to the current input. The output consists of a cryptographic lock and time. Metadata is any other relevant information for a transaction using that UTXO (for example, the timestamp and the fees). Hyperledger Fabric's state is more straightforward to map since it is a key-value store.

We define the entities that can read or write in the ledger by *participants*:

Definition 2. *Participant.* A participant $p \in \Upsilon$ is an entity (K_k^{id}, K_P^{id}), capable of reading and writing to a ledger L, where:

- K_k^{id} is a private key. The private key is used as the signing key.
- K_P^{id} is a public key. The public key is used as the verification key.

Participants interact with the ledger via nodes. Nodes are software systems that participate in ledger consensus by aggregating and executing transactions and sending them to other nodes. We introduce the concept of *Access Point* to formalize the relationship between participants and nodes as follows:

Definition 3. *Access Point (AP).* An AP ω maps a set of nodes n connected to ledger $\mathcal{L}$ to a set of participants p_n, i.e., $\omega(n) \longrightarrow p_n \subseteq \Upsilon_{\mathcal{L}}$. Conversely, ω^{-1} returns the node set n_p that a participant can access, i.e., $\omega^{-1}(v) \rightarrow n_p$.

An access point tells us which participants can access the ledger through a specific node. Nodes can access a DLT through the primitive `obtainDLT`. The result of `obtainDLT`$(n_p) = \mathcal{L}_v$, where $\mathcal{L}_v$ is a *virtual ledger*. A virtual ledger only allows the participants to read and write in the ledger according to their permissions (namely, the defined access control list). In more detail, a virtual ledger:

Definition 4. *Virtual ledgers.* A virtual ledger $\mathcal{L}_v$ is a projection of a ledger $\mathcal{L}(\mathcal{D}, \mathcal{A})$ in the form $(\mathcal{L}, \mathcal{F}_\pi)$ such that:

- $\mathcal{L}$ is the ledger that provides the database where projections are made.
- $\mathcal{F}_\Pi$, a set of projection functions $\{\mathcal{F}_{\pi_1}, \mathcal{F}_{\pi_2}, ..., \mathcal{F}_{\pi_n}\}$ that returns a subset d_π of the database $\mathcal{D}$ from $\mathcal{L}$, i.e., $\mathcal{F}_\pi \in \mathcal{F}_\Pi$: $\mathcal{L}_\mathcal{D} \times \mathcal{L}_\mathcal{A} \times p \rightarrow \{\emptyset, d_\pi\}$, according to the entries in the access control list of the participant defined by $\mathcal{A}$ (or $\emptyset$, if the participant is not authorized to access the ledger). This corresponds to "what the participant can see".

Recall that the database or a subset is a collection of keys and their values. We can simplify its representation by referring to the projection of the ledger l against the participant p (this is, the projection function $\mathcal{F}_\pi$ that is chosen projects the states of the virtual ledger that are accessible by p). The projection in the ledger $\mathcal{L}$ using the projection function $\mathcal{F}_p$ outputs a set of states $\{s_1, ..., s_n\}$ that the participant p can access, i.e., $d_{\mathcal{L},\mathcal{F}_p} = \{s_1, ..., s_n\}$. We use the notation $\bar{s}$ to represent the absence of a state in a projection. Consider the following projections, illustrated by Table 2:

$d_{\pi_{l,pn}}$	d_1	d_2	d_3
p_1 = Supplier	s_1	$\overline{s_2}$	$\overline{s_3}$
p_2 = Shipper	s_1	s_2	$\overline{s_3}$
p_3 = Distributor	s_1	$\overline{s_2}$	s_3
p_4 = Wholesaler	$\overline{s_1}$	s_2	s_3
p_5 = Retailer	$\overline{s_1}$	s_2	$\overline{s_3}$
p_6 = Retailer	$\overline{s_1}$	s_2	$\overline{s_3}$

Table 2. Ledger l projections onto all the participants from the use case depicted in Section 3.1. The projection function is a simple read of the database. A state s_i in the green background is a state that a participant can access, whereas a state $\overline{s_i}$ is a state that is not accessible to a given participant.

- $d_{\mathcal{L},\mathcal{F}_{p_1}} = \{ s_1, \overline{s_2}, \overline{s_3} \} = s_1$
- $d_{\mathcal{L},\mathcal{F}_{p_2}} = \{ s_1, s_2, \overline{s_3} \} = s_1, s_2$
- $d_{\mathcal{L},\mathcal{F}_{p_3}} = \{ s_1, \overline{s_2}, s_3 \} = s_1, s_3$
- $d_{\mathcal{L},\mathcal{F}_{p_4}} = \{ \overline{s_1}, s_2, s_3 \} = s_2, s_3$
- $d_{\mathcal{L},\mathcal{F}_{p_5}} = \{ \overline{s_1}, s_2, \overline{s_3} \} = s_2$
- $d_{\mathcal{L},\mathcal{F}_{p_6}} = \{ \overline{s_1}, s_2, \overline{s_3} \} = s_2$

We can retrieve a projection $d_{\mathcal{L},\mathcal{F}_p}$ (or empty set) with the primitive `obtainVirtualLedger`. This primitive receives as input a ledger $\mathcal{L}$ and a projection function $\mathcal{F}_p$. Some projections are not unique (e.g., $d_{\mathcal{L},\mathcal{F}_{p_5}}$ and $d_{\mathcal{L},\mathcal{F}_{p_6}}$). The concept of projection is the basis for the DLT view, which we will define later in this section. Both ledgers and virtual ledgers are abstractions to access a state, represented by a key-value store. Participants issue transactions to change the state. A transaction is defined as follows:

Definition 5. *Transaction.* A transaction t is a tuple $(t_{id}, t, payload, \sigma_{K_s^p}(message), S_{tid}, target)$, where:

- t_{id} is a unique increasing sequence identifier for a transaction. This identifier allows one to construct a transaction ID, a unique identifier for a transaction. Transaction t_i precedes t_j ,i.e., $t_i \preccurlyeq t_j$ if and only if $j > i$.
- t is the transaction timestamp
- $payload$ is the transaction payload. The payload can carry arbitrary information (smart contract parameters, UTXO input value).
- a signature on the transaction $\sigma_{K_s^p}(message)$, where $message \doteq (t_{id}, t, target, payload)$.
- S_{tid}, a set of input states given as input to the transaction with transaction ID sid.
- $target$ is the state ID to which the transaction refers.

A transaction takes as input a S_{tid} and outputs S'_{tid}. We define a primitive `VerifyTx(.)` that takes a set of initial states, a transaction, and a set of output states and outputs one if and only if the state transition is valid according to some algorithm ρ, i.e., `VerifyTx`$(S_{tid}, t_{id}, S'_{tid}, \rho = 1)$. Checking the validity of a transaction w.r.t. the states it changes implies re-running the transaction on its run environment. Transactions produce state changes. We define the state as:

Definition 6. *State.* A state s is a tuple $(s_k, s_{k,v}, \mathcal{T}, \pi_k)$, where

- s_k is a unique identifier (the state's key).
- a transaction list $\mathcal{T}$, referring to that state, i.e., $\forall t \in \mathcal{T} : t.target = s_k$

- the value it holds $s_{k,v}$. The value of a state can be calculated using the set of transactions $\forall i, TS = \{t_i \subset \mathcal{T}, s \in \mathcal{S} : t_i.target = s_k\}$, i.e., the transactions referring to that state, in the following manner:

$$s_{k,v} = \begin{cases} \emptyset & i = 0 \\ apply(t_i, s_{k,i-1}) & i \leq |TS| \end{cases}$$

  where *apply* is a function that executes the payload of the transaction t_i on the state s_k. The data is the most recent state value, the result of the successive transformations (over the previous versions of the same state). The function *apply* is blockchain-dependent.
- a proof of state validity $\pi_k = \sigma_{K_s^P}(s_k, s_{k,v}, v)$, where $s_{k,v}$ is the value of state s_k at version v, and σ_m is the set of signatures of participants $P \subset \Upsilon$ that creates the proof, over a payload m.

A state has a unique reference (or key) s_k and a version v such that when v is updated, it yields $v' > v$. We denote the value pointed by that reference by $s_{k,v}$. If we omit the version, we refer to s_k as the latest value in a certain state. Thus, for all $k \neq k'$, s_k and s'_k represent the latest value of different states. In practice, the value of a state is the result of successively executing transactions over the same object. The value for $s_{k,v}$ or (s_n) can be calculated as follows, where transaction set $\{t_1, ...t_{k-1}\} \in TS$ are the transactions referring specifically to s_k:

$$s_{k,0} \xrightarrow{t_1} s_{k,1} \xrightarrow{t_2} \dots \xrightarrow{t_{k-1}} s_{k,v}$$

Each ledger database stores states in its key-value store. The state identifier, s_k, is the key, while the tuple $(s_{k,v}, t, \pi_k)$ is the value. Each proof $\pi \in \Pi$ is a string accounting for the validity of the item it describes (e.g., signature over transaction).[4].

We define the *cardinality* of a state s_n as $|s_n|$ as the number of transactions that compose it. If s_n has a set of transactions $\mathcal{T} = \{t_1, ..., t_i\}$, then $|s_n| = i$. The state of a particular object can be reconstructed from the execution of all the transactions that refer to it. The global state is then the set of all states, the set $\mathcal{S}$. The set of states visible by a certain participant (states that authorize the participant to read/write/update) is a view of the blockchain.

Having introduced all the basilar concepts, we can define a DLT view:

Definition 7. *View.* A view $\mathcal{V}$ is a projection of a virtual ledger $\mathcal{L}$, in the form of $(v_k, t_i, t_f, p, d_{\pi_{l,p}}, S, \Pi)$, where

- its key v_k, is a unique ID
- an initial time t_i and a final time t_f that restrict the states belonging to that view. A view may have no restriction on the temporal interval, i.e., all states that a participant p accesses through $d_{\pi_{l,p}}$ are included in the view.
- $p \subseteq \Upsilon$ is the set of participants associated with the view. A participant *upsilon* can be associated with one or more nodes, accessible by a blockchain access point ω.
- a projection function $d_{\pi_{\mathcal{L},p}}$ used to build the view.
- S corresponds to the set of versioned states that the participant in the view has access to (via the projection function).
- Π is a set of proofs accounting for the validity of a view (e.g., accumulator value for over states ordered by the last update).

The consolidated view, or the global view $\mathcal{V}$, is the set of all participant views, i.e., $\mathcal{V} = \cup_{i=0}^{i} p_i$, that captures the entire ledger $\mathcal{L}$.

[4]In Bitcoin, for instance, the proof of validity for a transaction is the issuer's signature, along with a nonce whereby its hash begins with a certain number of zeroes and is smaller than a certain threshold (valid transaction within a valid block). In Hyperledger Fabric, the proof is a collection of signatures from the endorsing peer nodes that achieved consensus on the transaction's validity.

Definition 8. *View Cardinality.* Let there be a DLT view $v_{l,p}$ belonging to a participant p. A view $v_{l,p}$ has cardinality i when the number of states composing that view is i, i.e., $d_{\pi_{l,p}} = \{s_1, ..., s_i\}$. In other words, $|v_{l,p}| = i$.

Definition 9. *DLT Domain t* is a tuple $(d, s_k, s_{k,v}, \mathcal{F}_d, P, value)$, where:

- d is the identifier of the domain
- s_k is the state key to which that domain refers.
- $s_{k,v}$ is the state value corresponding to s_k.
- projection function that generates the state value $s_{k,v}$ of domain d indexed by s_k.
- P is the set of participants that can read the value v of state s_k on a domain d.

Domains represent private relationships; they capture how many participants can share the same state. Domains then capture if a state is accessible (value readable) or not by a set of participants. The same state's key can have different values on different domains (depending on the projection function generating the domain). Two participants sharing the same domain does not mean having the same view.

Definition 10. *View equivalence.* Let there be a ledger $\mathcal{L}$ composed of a set of views $\{v_1, v_2, ..., v_n\} \in \mathcal{V}$, holding respectively the sets of states $\{S_1, S_2, ..., S_n\} \in \mathcal{S}$, i.e., there are the pairs $\{(v_1, S_1), (v_2, S_2), ...(v_n, S_n)\}$. There is view equivalence, denoted by $\texttt{equivalent}(v_i, v_j)$ if, for any pair of views $v_i, v_j \in \mathbb{V}$ there is a bijection $\varphi : s_i \rightarrow s_j$ such that if both sets of states from the views are the same, their views are equivalent, i.e., $\varphi(s_i) = s_j \implies \forall s \in s_i, s$, could replace all $s' \in s_j \implies v_i \equiv v_j$.

Following the example of Table 2, v_1 and v_6 are equivalent views because the states that are accessible to those views are the same. However, those views are unequal, as the other parameters might change.

Definition 11. *View Transparency.* Let there be a ledger $\mathcal{L}$ with a set of participants Υ and a set of DLT views $\{v_1, ..., v_n\} = \mathcal{V}$. We define the transparency grade κ of a DLT view v_v, denoted as $\kappa(v_v)$, as the ratio of participants who can access the states encoded in that view. More formally,

$$\kappa(v_v) = \frac{\sum_{i \neq v}^{n} \forall v_i [\texttt{equivalent}(v_n, v_j) + 1]}{|\mathcal{V}|}$$

This concept is useful to understand how many participants can access a certain set of states. Taking the example from Table 2, $\kappa(v_{l,\mathcal{F}_{p_6}}) = \frac{2}{6}$ because the view created by projecting the ledger l with $d_{\mathcal{F}_{p_6}}$ is equivalent to v_5 (summing with the view being compared). Therefore, two out of six participants can access the same set of states (i.e., each participant has a different view, apart from p_5 and p_6).

4 BUNGEE, A MULTI-PURPOSE VIEW GENERATOR

In this section, we present BUNGEE. First, we present the system, key management processes, and adversary models. After that, we present the snapshot process. Next, we present how views are built and then merged. The section ends with discussion on the processes of creating snapshots and views, as well as merging views.

4.1 System Model

We consider an asynchronous distributed system, the DLT, that hosts a ledger $\mathcal{L}$. Three types of participants interact with the ledger: i) participants Υ: entities that transact on the network (can use `read` and `write` operations) via the nodes that their AP exposes; ii) nodes $\mathcal{N}$, who hold the full state of the DLT, and contribute to the consensus of the later; and iii) view generators $\mathbb{G}$, programs

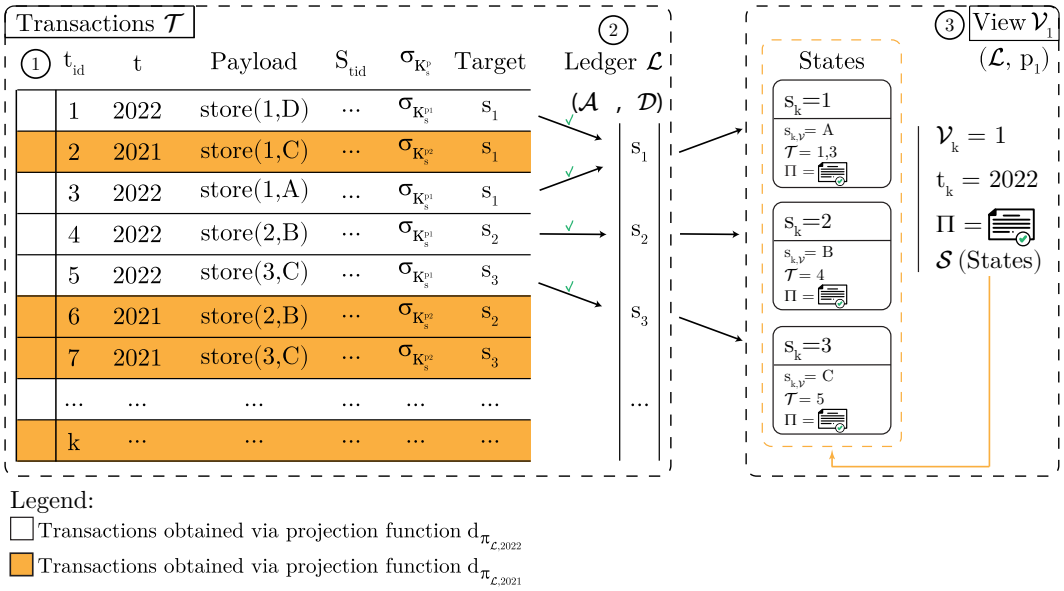


Fig. 3. View generation process for participant p_1, ledger $\mathcal{L}$, using projection functions $d_{\pi_{\mathcal{L},2021}}$ (yellow rows) and $d_{\pi_{\mathcal{L},2022}}$ (white rows)

that build views, via node that has access to the target participant of the view. This implies that a view generator trusts the node that is the access point to the DLT. Each DLT is assumed to be able to preserve its safety and liveness abilities despite the possible existence of malicious nodes. This implies that building and operating views based on networks that cannot guarantee safety properties (e.g., DLT forks due to attack) are invalid.

Key management. Each participant $p \in \Upsilon$, node $n \in \mathcal{N}$, and view generator $\mathbb{G}$ is identified by a pair of keys (K_p^p, K_k^p), (K_p^n, K_k^n), and $(K_p^{\mathbb{G}}, K_k^{\mathbb{G}})$ respectively. The private key is the signing key, while the public key is the verification key. The generated keys are independent of all other keys, implying that no adversary with limited computational resources can distinguish a key from one selected randomly. We assume that keys are generated and distributed in an authenticated channel, preserving integrity; digital signatures cannot be forged. We say that an entity x signs a message m with its private key with the following notation: $\texttt{sign}_x(m)$. Verifying a message m with the public key from x can be done with a `verify` primitive, which outputs one if the message was correctly signed by m, i.e., $\texttt{verify}(m, x, K_x^p) = 1$, and 0 otherwise.

4.2 Adversary Model

The DLT where view generators operate is trusted, meaning that most internal nodes are honest, and thus the network is trusted. Given this assumption, there can be different adversary models for nodes, generators, and view generators. Nodes can be honest by following the DLT protocol, establishing consensus with other honest nodes, and reporting the actual status of the DLT to participants who request it. Nodes can, instead, be malicious, i.e., Byzantine, being able to deviate from the protocol and falsely report the DLT status to participants (endangering the creation of truthful views). Nodes can be malicious but cautious, meaning that they are only malicious if there are no accountability checks that can penalize them (i.e., if they know that they cannot get caught).

View generators constitute a trusted group with the participant that is the target of the view because the generator needs the participant's credentials to access a (private) subset of the ledger.

We then assume that each participant runs its view generator. View generators can only build views for participants whose keys they do not control if the ledger has no partition (i.e., it is public). Since participants might access DLT partitions from different nodes, the trust group (participant, view generator) does not include a node or set of nodes, i.e., view generators and DLT participants are independent of the nodes that sustain the DLT.

4.3 View Generation Process Overview

BUNGEE constructs views from a set of states from an underlying DLT called a snapshot. This is done by obtaining a virtual ledger on behalf of a certain participant with a projection function. Then, each state accessible by the participant is collected in the snapshotting phase. The states are processed, and a representation of the ledger is built to which the participant has access. We can think of the snapshot as the capture of available transactions from the perspective of the participant in a table (*c.f* Figure 3, step (1)) upon having the necessary permissions from the ledger ((2)). Right after that, in the view building phase, a view is built from the virtual ledger that the view generator can access by temporarily limiting the states that one can see ((3)). Views can be stored in a local database, providing relational semantics and rich queries. Views are assured to provide provenance, i.e., BUNGEE can trace each component constituting a view down to the transaction.

After that, the view merging phase (optional) comprises merging views into an integrated one (see Section 4.7). For that, an extended state is created from the states present in each view that share the same key. Following that step, a merging algorithm is applied to the extended state. Finally, each view generator signs the integrated view, which can optionally be published in a public forum. The publication in a public forum can be decided by the participants that generate views (social consensus).

Let us focus on the high-level snapshot generation and view generation processes, as exemplified in Figure 3. In this example, we are building two views $\mathcal{V}_1$ and $\mathcal{V}_2$ (from participants p_1 and p_2, which capture states whose projection functions are $d_{\pi_{\mathcal{L},2022}}$ and $d_{\pi_{\mathcal{L},2021}}$, respectively. The semantics for the projection functions are simple: $d_{\pi_{\mathcal{L},2022}}$ refers to transactions timestamped as of 2022, and $d_{\pi_{\mathcal{L},2021}}$ refers to transactions timestamped as of 2021. In the figure, we focus on building $\mathcal{V}_1$ - the white rows. Thus, the transactions timestamped 2021 (in the yellow rows) are not captured on view $\mathcal{V}_1$. Transactions t_1, t_2 and t_3 alter state s_1, but t_2 belongs to view $\mathcal{V}_2$, so its not included. Transaction t_4 changes stores B as the value of s_3, while transaction t_5 sets s_3 as C.

Once the three steps are completed, BUNGEE returns the views (the generated and the integrated views) to the client application (for example, a blockchain migration application). For example, the client application might use BUNGEE to retrieve snapshots that refer to a period relevant to an audit. Due to BUNGEE's modularity, adding support for different applications is facilitated. Next, we present each phase depicted in this overview in finer detail.

4.4 Snapshot

A snapshot is a set of states that a certain stakeholder can access, plus proof of the validity of that state. We view each state as a versioned (key, value) store. A `snapshot` has a snapshot identifier id, a version v, a participant p, a set of states bins, sb, an initial time t_i that refers to the timestamp of the first transaction of any of the states belonging to sb, a final time t_f that refers to the timestamp of the last transaction of any of the states belonging to sb, i.e., $\texttt{snapshot} \doteq \{id, v, sb, t_i, t_f\}$. Each state bin is indexed by a state id s_k, the latest value to that key, $s_{k,\overrightarrow{v}}$, a version v that refers to the number of transactions applied on the state key s_k to produce the latest value $s_{k,\overrightarrow{v}}$ and a list of transactions T referring to that state (as in Definition 7). Versioning snapshots allows one to

efficiently build snapshots from older snapshots (i.e., building snapshots from incremental changes from older snapshots).

Algorithm 1: Snapshotting of ledger $\mathcal{L}$ through node n and participant p, via projection function $\mathcal{F}_p$

Input: Access point AP, participant p, projection function $\mathcal{F}_p$, snapshot identifier $\texttt{snapshot}_{id}$
Output: Snapshot from participant p through node n, `snapshot`

1 $\texttt{snapshot.id} \leftarrow \texttt{snapshot}_{id}$
2 $\texttt{snapshot.v} \leftarrow 1$
3 $\texttt{snapshot.sb} \leftarrow \emptyset$
4 $\texttt{snapshot}.t_i \leftarrow \perp$
5 $\texttt{snapshot}.t_f \leftarrow \perp$
6 $t_{it} \leftarrow \infty$ ▷ temporary variable to hold minimum state timestamp to date
7 $t_{ft} \leftarrow 0$ ▷ temporary variable to hold maximum state timestamp to date
8 $n = \omega^{-1}(p)$ ▷ choose any available node
9 $\mathcal{L} = \texttt{obtainDLT}(n)$ ▷ depends on the DLT client implementation
10 $d_{\mathcal{L},\mathcal{F}_p} = \texttt{obtainVirtualLedger}(\mathcal{L}, \mathcal{F}_p)$ ▷ obtain projection of $\mathcal{L}$ according to p
11 **foreach** $s_k \in d_{\mathcal{L},\mathcal{F}_p}$ **do**
12 $s_{k,it} \leftarrow \emptyset$ ▷ the timestamp of the first transaction applied to state s_k
13 $s_{k,lt} \leftarrow \emptyset$ ▷ the timestamp of the last transaction applied to state s_k
14 $\texttt{snapshot}.sb[s_k].s_k = s_k$
15 $\texttt{snapshot}.sb[s_k].version = d_{\mathcal{L},\mathcal{F}_p}[s_k].T.length$
16 $\texttt{snapshot}.sb[s_k].\texttt{latestValue} = d_{\mathcal{L},\mathcal{F}_p}[s_k].s_{k,v}$
17 $\texttt{snapshot}.sb[s_k].T = d_{\mathcal{L},\mathcal{F}_p}[s_k].T$ ▷ save list of transactions referring to each state key
18 $s_{k,it} = d_{\mathcal{L},\mathcal{F}_p}[s_k].T[0]$ ▷ transaction list is ordered chronologically
19 $s_{k,lt} = d_{\mathcal{L},\mathcal{F}_p}[s_k].T.length$
20 **if** $s_{k,it} < t_{it}$ **then**
21 $t_{it} = s_{k,it}$ ▷ update the auxiliary first timestamp
22 **end if**
23 **if** $s_{k,lt} > t_{ft}$ **then**
24 $t_{ft} = s_{k,lt}$ ▷ update the auxiliary last timestamp
25 **end if**
26 **end foreach**
27 $\texttt{snapshot}.t_i = t_{it}$
28 $\texttt{snapshot}.t_f = t_{ft}$
29 **return** `snapshot`

Algorithm 1 depicts the snapshotting process. The snapshot phase occurs when the BUNGEE client requests the beginning of the view integration process to a node n on behalf of the participant p (line 8). After that, the node connects to the DLT. Upon a successful connection, n retrieves the ledger (line 9). Obtaining a list of states from a ledger requires checking all transactions that performed state updates. For each transaction, a BUNGEE has to check its target. BUNGEE creates a new state if there is no state key with a target equal to the current transaction. The version of

the new state is one. Then, BUNGEE runs the transaction's payload against the current state value (empty at initialization). Otherwise, if the transaction target refers to an existing state key, run the transaction payload against the state's current value, yielding the new value and incrementing the version by one. This process outputs a list of states. According to the participant's perspective, the process is abstracted by the ledger's projection (according to the participant's perspective) that the algorithm uses (line 11). The snapshot maps each state to a state bin. For each state, we collect its key (line 15), version (line 16), latest value (line 17), the auxiliary first timestamp (line 18), and auxiliary latest timestamp (line 19). After that, the first and last timestamps are updated (lines 28 and 29), and, at last, the algorithm returns a snapshot.

4.5 View Building

This section explains how views are built. A view generator can generate a set of views depending on the input p. The following steps occur for each view to be built: first, the view generator generates a snapshot. After that, the snapshot is limited to a time interval and signed by the view generator.

Algorithm 2 shows the process of building a view from a snapshot. First, the view generator temporarily limits each included state, proceeding to abort if no states are within its boundaries (line 8). If there are, each state in the snapshot is included if it belongs to the temporal limit (line 18) and removed otherwise (line 15). Finally, the view generator signs the view (line 20) and returns it to the client application (line 21).

4.6 Merging views

In this section, we describe how to merge views in a privacy-preserving way. The merging of views creates an integrated view $\mathbb{I}$ from a set $\mathcal{V}$ of input views. The idea is to compare the state keys indexed by every view and their value according to a merging algorithm $\mathcal{M}$ that is given as input. This merging algorithm controls how the merge is performed, and therefore, a user can set up policies that comply with the privacy needs (of all participants).

Algorithm 3 shows the procedure for merging views. The algorithm receives the views to be merged and returns an integrated (or consolidated) view as input. We initialize an auxiliary list $\mathcal{S}_{\mathcal{V}_1,...,\mathcal{V}_n}$ (on line 1) that holds all the values (coming from different views) for each state key. We propose a construct called an extended state. An extended state is a state where each state key maps to a set of values. Additionally, an extended state has a *metadata* field holding a list of operations applied to that extended state.

Definition 12. An *Extended State* $\overrightarrow{s}$ is a tuple $\overrightarrow{s_k}, \overrightarrow{s_{k,v}}, t, \pi_k, metadata, version)$, where

- $\overrightarrow{s_k}$ is a unique identifier (the state's key);
- $\overrightarrow{s_{k,v}}$ is a list of values;
- a transaction list $\mathcal{T}$;
- a proof of state validity π_k;
- metadata, which holds a list of operations that have been applied to the extended state;
- version, a monotonically increasing integer. The counter increases when an update is done to the extended state (the number of elements in the metadata field is the same as the version).

Thus, each index of the set of extended states $\mathcal{S}$ will index all different values for each key for all the views to be merged, i.e.,

$$\mathcal{S}_{\mathcal{V}_1,...,\mathcal{V}_n} = \{\forall s_i \in \mathcal{S} : \exists k_i \in s_i : k_i \implies (s_{\mathcal{V}_1(k_i,v)}, ..., s_{\mathcal{V}_n(k_i,v)})\}$$

Algorithm 2: Constructing a view $\mathcal{V}$ of ledger $\mathcal{L}$ with snapshot snapshot, from the perspective of participant p.

Input: Snapshot snapshot, view id id, initial time t_i, final time t_f
Output: View $\mathcal{V}$

1 $\mathcal{V}.k \leftarrow id$
2 $\mathcal{V}.t_i \leftarrow t_i$
3 $\mathcal{V}.t_f \leftarrow t_f$
4 $\mathcal{V}.d_{\pi_{l,p}} \leftarrow \text{snapshot}.\mathcal{F}_p$
5 $\mathcal{V}.p \leftarrow \text{snapshot}.p$
6 $\mathcal{V}.\Pi \leftarrow \perp$
7 $\mathcal{V}.S_{k,v} \leftarrow \perp$
8 **if** $t_i < \text{snapshot}.t_f$ ***OR*** $t_f > \text{snapshot}.t_i$ **then**
9 return; ▷ there are no intersecting states that we want to capture, on the snapshot
10 **end if**
11 ▷ each $sb = \{s_k, s_{k,\overrightarrow{v}}, \text{v}\}$
12 **foreach** $s_k \in \text{snapshot}.sb$ **do**
13 **foreach** $t \in s_k$ **do**
14 **if** $t.timestamp < t_i$ ***OR*** $t.timestamp > t.f$ **then**
15 $\text{snapshot}.sb[s_k] \leftarrow \text{snapshot}.sb[s_k].\mathcal{T} \setminus t$ ▷ removes transaction that is not within the specified time frame
16 **end if**
17 **end foreach**
18 $\mathcal{V}.S_{k,v} \leftarrow \text{snapshot}.sb[s_k]$
19 **end foreach**
20 $\mathcal{V}.\Pi \leftarrow sign_{\mathbb{G}}(\mathcal{V})$
21 **return** $\mathcal{V}$

After we initialize the list of extended states, in Algorithm 3, we initialize the integrated view properties: its initial timestamp (line 2), final timestamp (line 3), projection functions (taken as the union of the projection functions of all the views, on line 4), participants (the participants from each view, on line 5), a set of proofs (line 6) and a set of states (line 7). The set of states to be assigned as the set of states of the integrated view is a function of the processed auxiliary set of states $\mathcal{S}$. After all, we check each state key to merge each view. If the tested state is already on the auxiliary state set (line 10), then we add its value $\overrightarrow{s_{k,v}}$ as a value for the current extended state key (line 11). This outputs a list of values (between one and the number of views to be merged) for each extended state key. Otherwise, we set a new extended state, adding the current state value (as the first value for that key, on line 15).

On line 21, we apply an optional view processing phase by giving our list of states $\mathcal{S}$ to an arbitrary algorithm that needs to respect a simple interface and functionality (later defined). After that, we add algorithm $\mathcal{M}$ as a projection function for $\mathcal{I}$ for future traceability and auditing. Next, we adjust the initial and final timestamps (lines 23 and 24) because the merging algorithm might have changed the time boundaries of the included states (for example, the state corresponding to the lowest timestamp might have been removed). All view generators must sign $\mathcal{I}$ (line 25) to

Algorithm 3: Merging a set of views $\mathcal{V} = \mathcal{V}_1, \mathcal{V}_2, ..., \mathcal{V}_n$, where each view was built referring to participant $p_1, p_2, ...p_n$ respectively by a set of view generators $\mathbb{G} = \mathbb{G}_1, \mathbb{G}_2, ..., \mathbb{G}_n$

Input: Views to be merged $\mathcal{V} = \mathcal{V}_1, \mathcal{V}_2, ..., \mathcal{V}_n$, merging algorithm $\mathcal{M}$
Output: Integrated view $\mathcal{I}$

$\mathcal{S} \leftarrow []$ ▷ state list $\mathcal{S}_{\mathcal{V}_1,...,\mathcal{V}_n}$ ($\mathcal{S}$ for simplicity) where each index (representing a state key) maps to tuple of values from referring to that key, from each view to be merged
$\mathcal{I}.t_i \leftarrow \emptyset$
$\mathcal{I}.t_f \leftarrow \emptyset$
$\mathcal{I}.d_{\pi_{l,p}} \leftarrow \bigcup_{i=0}^{n} \mathcal{V}_n.d_{\pi_{l,p}}$
$\mathcal{I}.p \leftarrow \bigcup_{i=0}^{n} \mathcal{V}_n.p$
$\mathcal{I}.\Pi \leftarrow \perp$
$\mathcal{I}.S_{k,v} \leftarrow \perp$
foreach $v \in \mathcal{V}$ **do**
  foreach $s \in v.S_{k,v}$ **do**
    if $s \in \mathcal{S}$ **then**
      $\mathcal{S}[\overrightarrow{s.k}] = \mathcal{S}[\overrightarrow{s.k}] \cup \overrightarrow{s_{k,v}}$ ▷ if state exists, add value referring to that state, from current view
      $\mathcal{S}[\overrightarrow{s.k}].version \leftarrow \mathcal{S}[\overrightarrow{s.k}].version + 1$
    end if
    else
      $\mathcal{S}[\overrightarrow{s.k}] = \overrightarrow{s_{k,v}}$ ▷ otherwise, initialize state key list
      $\mathcal{S}[\overrightarrow{s.k}].version \leftarrow 0$
      $\mathcal{S}[\overrightarrow{s.k}].metadata \leftarrow \{MERGE - INIT\}$
    end if
  end foreach
end foreach
$\mathcal{I}.S_{k,v} = \texttt{call}_{\texttt{algorithm}\mathcal{M}}(\mathcal{S})$ ▷ OPTIONAL. Computes the state list of the integrated view according to $\mathcal{M}$ (see for example algorithm 4)
$\mathcal{I}.d_{\pi_{l,p}} \leftarrow \mathcal{I}.d_{\pi_{l,p}} \cup \{\mathcal{M}\}$ ▷ add reference to the merging algorithm
$\mathcal{I}.t_i = \texttt{min}\{\mathcal{I}.S_{k,v}.t_i\}$ ▷ initial timestamp correspond to the initial timestamp of the processed states
$\mathcal{I}.t_f = \texttt{min}\{\mathcal{I}.S_{k,v}.t_f\}$
$\mathcal{I}.\Pi \leftarrow sign_{\mathbb{G}}(\mathcal{I})$ ▷ signed collectively by $\mathbb{G}$
return $\mathcal{I}$

promote accountability. Signing the integrated view can be distributed using a multi-signature algorithm (for example, BLS Multi-Signatures [52]).

Each merging phase has an optional application of a merging algorithm $\mathcal{M}$, which dictates how the merge is carried out (otherwise, all states are included without further processing). We define a simple interface for merging algorithms: a merging algorithm receives a set of extended states as input and outputs a set of extended states. The functionality of the merging functions should be: 1) apply arbitrary operations on the set of extended states, 2) add a reference to the current merging algorithm to the *metadata* field of each extended state key that is altered, 3) increase the version of

each extended state key that is altered. Each merging algorithm should be public and well-known to the parties involved.

Examples of merging algorithms are:

- *Pruning:* removes the values coming from a particular view.
  Algorithm 4 prunes the values belonging to a particular view from a set of extended states. Note that times do *not* need to be updated because they are recalculated in steps 23 and 24 of Algorithm 3. Applications include removing sensitive information in the context of existing regulations and laws.

Algorithm 4: Merging algorithm example – PRUNE (by $\mathcal{V}_1$)

Input: The set of states to be processed $\mathcal{S}$
Output: A processed set of states $\mathcal{S}'$

1 $\mathcal{S}' \leftarrow \emptyset$ ▷
2 **foreach** $s \in \mathcal{S}$ **do**
3 **if** $s_k[0]$ **then**
4 $\mathcal{S}'[s_k] = \mathcal{S}'[s_k] \setminus s[0]$ ▷ if there exists a value for view $\mathcal{V}_1$, then remove that value from the state list
5 $\mathcal{S}'[s_k].metadata \leftarrow$ `PRUNE-VIEW-1`
6 $\mathcal{S}'[s_k].version \leftarrow \mathcal{S}'[s_k].version + 1$
7 **end if**
8 **end foreach**
9 **return** $\mathcal{S}'$

4.7 Example: merging two views

In this section, we graphically show an example of a merge view, by applying Algorithm 3 (merge view) and `MERGE-ALL`. Informally, `MERGE-ALL` works by keeping the values from both views included in the final view (a rather simple merge algorithm). Let us consider two views $\mathcal{V}_1$ and $\mathcal{V}_2$ (create from the table of Figure 3), and its merging into a consolidated view $\mathcal{V}_{\mathcal{I}}$, c.f. Figure 4. View $\mathcal{V}_1$ and $\mathcal{V}_2$ differ on the value for s_1, A and C, respectively. The integrated view will hold an extended state with 1) a timestamp including both views, 2) references to the participants generating each view, 3) the joint projection function, 4) a set of proofs, and 5) a set of extended states. For s_1, we have included the different values from the different views.

5 DISCUSSION

In this section, we discuss BUNGEE. The proliferation of blockchain interoperability solutions is increasing interest in exploring cross-chain logic and the need to model and analyze it [53]. Our proposal constitutes the foundation to make sense of that diversity by allowing us to systematically create and integrate views from different blockchains. In this section, we discuss the studied research questions, with considerations on the integrity, accountability, and privacy of views.

5.1 RQ 1: providing a data format for views

Views can be generated from different sources, as long as they are accompanied by a valid proof. The existence of proofs on states is a proof of creation by the entities that created or executed the transactions referring to that state. For example, a signed transaction hash qualifies as proof of a transaction that makes part of the state proof (as many proofs as signed transactions referring

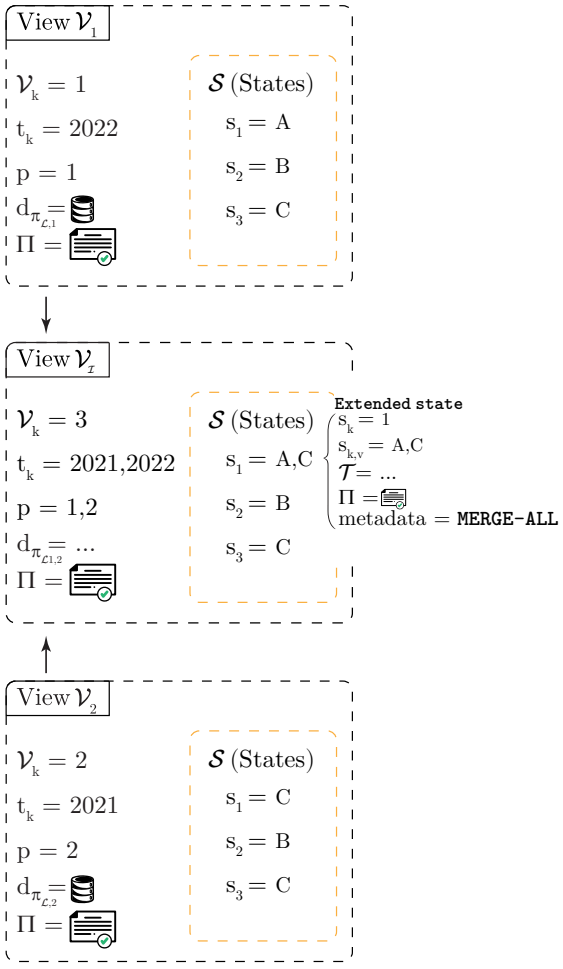


Fig. 4. Merging of views $\mathcal{V}_1$ and $\mathcal{V}_2$ into a consolidated view $\mathcal{V}_I$ according to merging algorithm MERGE-ALL.

to a certain state). On the other hand, views are also signed by the view generator that either generates, merges views, applies a merging algorithm or notarizes the view as true. This set of proofs allows independent parties to validate the truthfulness of the view (by verifying each state) and hold view generators accountable. In a permissioned environment, an auditor can confirm that the views are valid and complete. The metadata fields on each extended state and the views allow one to understand who, when, and how a view generator changes a certain view. However, more accountability measures can be implemented. In particular, if a view is only shared across the view generators that endorsed it, there might be limited exposure and, therefore, limited transparency. To enhance transparency, our key insight is to store a view in a public forum such as the InterPlanetary File System [54] (a distributed peer-to-peer file system maintained by a network of public nodes) or a public blockchain, similar to some related work [42, 43]. If a view is deemed false, automatic view conflict detection and resolution can occur.

For the network to enforce the integrity of views, we need two conditions to hold. First, each group of nodes that accesses a subset of a ledger (and thus creates a view) must have at least two elements. Second, at least there is one honest element for every group. Thus, an honest view generator connected to an honest node holds the knowledge of the view v, and publishes it. If a malicious node broadcasts a false view v', an honest node can dispute it. Disputes can be calculated

by calculating the difference between views and checking the proofs constituting each view. In particular, if an instance of BUNGEE, on behalf of participant A, holds the knowledge of a pair of different views v, v' referring to the same participant at the same time frame, then one of the views is false. Thus, the creator of one of the views is malicious. An honest view generator can reconstruct the disputed view and compare it to the view publicized by the malicious participant.

It is unlikely that all participants are colluding to change the perception of the inner state because, in principle, participants have different interests; however, there might be several situations in which the whole network gains if it colludes (i.e., blockchain with financial information). The ledger is unreliable if all internal nodes collude because the safety properties cannot be guaranteed. We hypothesize that using a view similarity metric could be a good tool to assess the quality of the view merging process. In other words, one could systematically compare how the final integrated view is different from each view that composes it.

5.2 RQ 2: privacy-preserving integrated views

In this paper, we have introduced how to create, merge, and process views. However, a challenge remains unsolved: how to share views in a decentralized way? How does one manage the lifecycle of a view, including its creation, endorsement, and dispute? Although the work of Abebe et al. [43] sheds some light on this, how can one verify that a view is false? The solution offered by Abebe et al. includes parties voting on an invalid view, but this does not solve the problem per se because if the source blockchain is private, there is no canonical answer. Suppose that at least one view from the integrated view comes from a private blockchain; the signatures of the view guarantee that a certain participant has voted on the validity of that view. This could introduce problems if all participants collude to show a false view. However, assuming that at least one view generator is honest, the view generator could initiate a dispute with the suspect of a false view.

A view generator could use fraud proofs [55] to create disputes about the validity of views, allowing an efficient and decentralized view management protocol. Application clients can then use the proof field from views, states, and transactions to validate a certain fact on a ledger. However, when BUNGEE merges views, completeness may not be guaranteed because the merged view depends on each input view, and processing might be applied (including pruning), possibly leading to information being excluded. A case to apply pruning might be when sensitive data is recorded in a ledger and later removed from the processing stage or even to remove "obsolete" data from the blockchain and therefore contribute to efficient bootstrapping of light clients [56]. An interesting detail is that each view only includes the state and respective proofs in timeframe t_k. However, to ensure that it is possible to validate the view, a pointer to the validity of the latest state before t_k should be available.

Our integration process follows a semantic approach to information based on a conceptual standard data model that we define as a view. Thus, for each practical implementation of BUNGEE, there needs to be a mapping between the data model of the underlying blockchain and the view concept. Being all views uniform, we can not only represent data in all blockchains, but we can merge views belonging to different blockchains. The applicability is to build a complete picture of the activity of a participant in each network, but it can also be used to disclose information according to an access control policy [57]. While selective access control to views has been explored, there is space to explore decentralized identity access control mechanisms to provide fine-grain access over views, leveraging the need to unify the different notions of identity that emerge from different blockchains.

The reader might inquire how BUNGEE would ensure the privacy that partial consistent blockchains attempt to enforce when views are unified and then shared. To address this problem, we envision two solutions: first, merging views requires *tacit* consent from all parties sharing the

input views. If there is sensitive data, the data is removed *apriori* or removed in the snapshotting phase. This is essentially encoded by the projection function $\mathcal{F}_p$ used to obtain the virtual ledger. The second solution is to encrypt the data (or hash the data) [57], so the resulting view contains obfuscated information or a notarization proof [58], respectively. However, the scientific community agrees that storing sensitive data on-chain, even if encrypted, is a bad security practice due to the threat of cryptographic algorithms being broken in the future [59–62]. Zero-knowledge proofs can also be explored as a vehicle to prove facts on a ledger by disclosing limited information about such facts [63]. We leave those interesting research paths for future work.

5.3 Considerations on Privacy

Privacy is of the utmost importance when dealing with views. There is intra-group privacy (within participants sharing the same domain, and view equivalence) and intra-blockchain privacy (where two disjoint groups of participants might have different view cardinality and view transparency). Likewise, we can consider the privacy of a merged view against the environment (participants from other systems that are interested in reading the created views).

Consider two groups of participants, α and β that share a disjoint set of domains d_α and d_β within a blockchain. These participant groups will at least not have access to the states a and b, for the first and second groups, respectively. The higher the view transparency within a domain, the more shared states exist, and therefore, the easier a merge operation becomes (namely because if everyone knows the states within a view, there is no need to run preprocessing over those states - only when sharing with the exterior). The states group α cannot access from group β are then given by $c = d_\alpha \setminus d_\beta$. Upon a merge, states $d_\alpha \cap d_\beta$ can be freely processed without intra-blockchain privacy concerns (i.e., both groups have access to those states) - so it is a matter of the appointed view generator to apply a commonly-agreed algorithm over those states, for public exposure.

Regarding sharing the unified view with the exterior, the consortium agrees on a common procedure to enhance privacy (e.g., pruning sensitive information that only should be shared within the consortium).

5.4 Future Work

There is ongoing work on the implementation of BUNGEE in a flagship interoperability project called Hyperledger Cacti, in the scope of standardizing asset transfers, within the IETF[5]. We would also like to empirically validate our work by providing an implementation of BUNGEE that can provide support for building blockchain migrator applications.

Another future work venue deals with the evolution of blockchains, and consequently the evolution of views. Our solution diverges from the classical definition of a view in databases [2] since the data shown is not up-to-date, nor easily updatable. We propose designing algorithms to update a view, perhaps similarly to how Git manages updates to versioned files [64]. An API that inspects and creates updates to views can be used by applications to efficiently use up-to-date data. Finally, we find potential in studying zero-knowledge proofs to enhance view privacy.

6 RELATED WORK

In this section, we present the related work.

Partial Consistency. Graf et al. propose the concept of partial consistency [4]. An implementation of this principle is given by blockchains with state partitions such as channels. Although blockchains that provide this property have existed for several years, e.g., Quorum [31], IOTA [32], Corda [30], Hyperledger Fabric, Hyperledger Besu [65], and Ripple [66], to the best of our knowledge, this is

[5]https://datatracker.ietf.org/group/satp/about/

the first formalization of the concept. Some solutions build partial consistency realizations on top of blockchains [67], such Canton [33], but are not formalized, and thus privacy properties of such channels are not clear. Our concept of view brings another way to reason about blockchain sharding [68], where each validator that is part of a shard runs a view generator and can communicate the state of the shard to different blockchains. Sharding is a technique to improve throughput, typically in public blockchains. A sharding scheme offloads the transaction processing to several groups of nodes called shards [68]. A shard is thus a de-facto logical view that guarantees the integrity and correctness of states regarding the participants that can access those states. Like a shard, a view is a logical separation of the ledger according to each participant.

Decentralized privacy-preserving computation. Some work surveys [69, 70] privacy-preserving techniques for blockchain interoperability, which we emphasize the latest survey [25], that supports the need for blockchain views. Several surveys claim that most implementations focus on privacy-preserving techniques for permissionless homogeneous blockchains, while we focus on both permissionless and permissioned heterogeneous blockchains.

Indirecly related work inclues the work from Kosba et al.[71], who propose a privacy-preserving decentralized smart contract system. The Enigma network [**?**] (uses multi-party computation technology to enforce smart contract privacy). Others use trusted hardware to enforce privacy [72], or zero-knowledge proofs [73, 74]. Contrarily to these works, BUNGEE focuses on delivering privacy for merged views (i.e., data from different sources, or from different domains), for interoperability purposes, while the related work focuses on ensuring privacy in a single domain.

Generating views. Katsis al. [75] has summarized view-based data integration techniques. Our approach follows a Global and Local as View [76] because views are created from a subset of the global state, but then can be merged and processed. We call the reader's attention to the survey on view integration techniques, mostly used in the database and business process management research areas [8]. Abebe et al. [43] have proposed the concept of external view, a construct to prove the internal state of permissioned blockchains. However, this concept only applies to private blockchains. We extend and generalize the concept of view so that it can be used for both public and private blockchains, and thus for heterogeneous interoperability purposes (by allowing the integration and merging of views).

Some proposals in industry and academia propose general data models for cross-chain interaction, namely the Rosetta API, Quant Overledger's gateways [36, 77], Blockdaemon's Ubiquity API [78], Polkadot's XCMP [79, 80], Cosmos's IBC [81]. The Rosetta API and Blockdaemon's Ubiquity API only support public blockchains. Quant Overledger supports public and private blockchains but does not allow them to realize complex operations such as merging views. Polkadot and Cosmos have the previous limitation and can only support blockchains created with Substrate and Tendermint, respectively. None of those are well-accepted standards. On the other hand, BUNGEE aims to create views independent of the underlying blockchains, aiming to follow the efforts of IETF's standardization group SATP [41, 42, 82].

View applications. In [57], views provided fine-grain dynamic access control over private data in Hyperledger Fabric. In addition to the applications referred to in Section 1, we identify some studies using the concept of view for different purposes. Some authors use views to perform audits of participants on different blockchains [83, 84]. In particular, a view is created and then merged with other views from the same participant on different blockchains to create a global view of the participant's activity. Applications are, for instance, cross-chain tax audit [85], or cross-chain portfolio tracking [86], and even cross-chain security, by representing and monitoring cross-chain

state [87], all applications that could benefit from a more formal treatment that BUNGEE can provide.

7 CONCLUSION

Views directly support blockchain interoperability since it is easier to share the perspectives of all participants across heterogeneous DLTs. This enables complex orchestration of cross-blockchain services and supports the new research areas of DLT interoperability, including blockchain gateway-based interoperability. In this paper, we introduce the concept of blockchain view, a foundational concept for handling cross-chain state. Views represent different perspectives of blockchain participants, allowing one to reason about their different incentives and goals. We present BUNGEE, a system that can create views from a set of states according to a projection function, yielding a collection of states accessible by a certain participant. BUNGEE can create a snapshot by retrieving the state of a blockchain, and based on participants' permissions, build a view of the global state. After that, BUNGEE creates extended states, the basis for merging blockchain views. Different views (possibly from different blockchains) can be merged into a consolidated view, enabling applications such as cross-chain audits and analytics. Finally, we discuss different aspects of BUNGEE, including decentralization, security, privacy, and its applications. An important area for future work is the use of zero-knowledge proofs to enhance view privacy.

ACKNOWLEDGMENTS

We warmly thank Luis Pedrosa, the colleagues of the academic paper workforce at Hyperledger, and the colleagues of IETF's SATP working group for many fruitful discussions. We thank Afonso Marques, Iulia Mihaiu, Benedikt Putz, and Diogo Vaz for reviewing and providing valuable feedback on an earlier version of this paper. We thank the anonymous reviewers for suggestions and feedback that helped improve the paper. The work of Limaris Torres was done while she was with Blockdaemon. This work was developed within the project scope nr. 51 "BLOCKCHAIN.PT - Agenda Descentralizar Portugal com Blockchain", financed by European Funds, namely "Recovery and Resilience Plan - Component 5: Agendas Mobilizadoras para a Inovação Empresarial", included in the NextGenerationEU funding program. This work was also supported by the European Commission through contract 952226 (BIG), and national funds through Fundação para a Ciência e a Tecnologia (FCT) with reference UIDB/50021/2020 (INESC-ID) and 2020.06837.BD.

REFERENCES

[1] J. Garay, A. Kiayias, N. Leonardos, The Bitcoin Backbone Protocol with Chains of Variable Difficulty, in: J. Katz, H. Shacham (Eds.), Advances in Cryptology – CRYPTO 2017, Springer International Publishing, Cham, 2017, pp. 291–323.

[2] R. Belchior, S. Guerreiro, A. Vasconcelos, M. Correia, A survey on business process view integration: past, present and future applications to blockchain, Business Process Management Journal 28 (3) (2022) 713–739, publisher: Emerald Publishing Limited. doi:10.1108/BPMJ-11-2020-0529.
URL https://doi.org/10.1108/BPMJ-11-2020-0529

[3] R. Dijkman, Diagnosing differences between business process models, in: International Conference on Business Process Management, 2008. doi:10.1007/978-3-540-85758-7_20.
URL https://link.springer.com/chapter/10.1007/978-3-540-85758-7_20

[4] M. Graf, D. Rausch, V. Ronge, C. Egger, R. Küsters, D. Schröder, A Security Framework for Distributed Ledgers, in: Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security, CCS '21, Association for Computing Machinery, 2021, pp. 1043–1064, event-place: New York, NY, USA. doi:10.1145/3460120.3485362.
URL https://doi.org/10.1145/3460120.3485362

[5] M. Graf, R. Küsters, D. Rausch, Accountability in a Permissioned Blockchain: Formal Analysis of Hyperledger Fabric (2020). doi:10.1109/EuroSP48549.2020.00023.

[6] R. Belchior, L. Riley, T. Hardjono, A. Vasconcelos, M. Correia, Do you need a distributed ledger technology interoperability solution?, Distributed Ledger Technologies: Research and Practice (ACM DLT) 2 (1) (2022) 1–37, citation Key: belchiorYouNeedDistributed2022a. doi:10.1145/3564532.
[7] R. Belchior, J. Süßenguth, Q. Feng, T. Hardjono, A. Vasconcelos, M. Correia, A Brief History of Blockchain Interoperability (9 2023). doi:10.36227/techrxiv.23418677.v3.
URL https://www.techrxiv.org/articles/preprint/A_Brief_History_of_Blockchain_Interoperability/23418677
[8] R. Belchior, S. Guerreiro, A. Vasconcelos, M. Correia, A survey on business process view integration: Past, present and future applications to blockchain ahead-of-print. doi:10.1108/BPMJ-11-2020-0529.
URL https://doi.org/10.1108/BPMJ-11-2020-0529
[9] A. Lohachab, S. Garg, B. Kang, M. B. Amin, J. Lee, S. Chen, X. Xu, Towards Interconnected Blockchains: A Comprehensive Review of the Role of Interoperability among Disparate Blockchains, ACM Comput. Surv. 54 (7), place: New York, NY, USA Publisher: Association for Computing Machinery (Jul. 2021). doi:10.1145/3460287.
URL https://doi.org/10.1145/3460287
[10] FTX: An Overview of the Exchange and Its Collapse.
URL https://www.investopedia.com/ftx-exchange-5200842
[11] Centralised Exchanges Are Terrible At Holding Your Money: A Timeline of Catastrophes - LocalCryptos Blog.
URL https://blog.localcryptos.com/centralised-exchanges-are-terrible-at-holding-your-money/
[12] Crypto Hack: The Mt. Gox Tragedy | CoinMarketCap.
URL https://coinmarketcap.com/alexandria/article/crypto-hack-the-mt-gox-tragedy
[13] R. Belchior, A. Vasconcelos, S. Guerreiro, M. Correia, A Survey on Blockchain Interoperability: Past, Present, and Future Trends 54 (8) 168:1–168:41. doi:10.1145/3471140.
URL https://doi.org/10.1145/3471140
[14] R. Belchior, P. Somogyvari, J. Pfannschmid, A. Vasconcelos, M. Correia, Hephaestus: Modelling, Analysis, and Performance Evaluation of Cross-Chain Transactions. doi:10.36227/techrxiv.20718058.v1.
URL https://www.techrxiv.org/articles/preprint/Hephaestus_Modelling_Analysis_and_Performance_Evaluation_of_Cross-Chain_Transactions/20718058/1
[15] T. Xie, J. Zhang, Z. Cheng, F. Zhang, Y. Zhang, Y. Jia, D. Boneh, D. Song, zkBridge: Trustless Cross-chain Bridges Made Practical. arXiv:2210.00264, doi:10.48550/arXiv.2210.00264.
URL http://arxiv.org/abs/2210.00264
[16] S. Armenchev, R. Belchior, D. Dimov, E. Ivanichkov, Z. Karadjov, K. Kirkov, P. Kirov, Y. Miladinov, Dendreth: A smart contract implementation of the ethereum light client sync protocol, accessed: 21-June-2023 (2023).
URL https://github.com/metacraft-labs/DendrETH
[17] T. Chen, H. Lu, T. Kunpittaya, A. Luo, A Review of zk-SNARKs. arXiv:2202.06877, doi:10.48550/arXiv.2202.06877.
URL http://arxiv.org/abs/2202.06877
[18] A. Biryukov, S. Tikhomirov, Security and privacy of mobile wallet users in Bitcoin, Dash, Monero, and Zcash 59 101030. doi:10.1016/j.pmcj.2019.101030.
URL https://www.sciencedirect.com/science/article/pii/S1574119218307181
[19] T. Hardjono, A. Lipton, A. Pentland, Toward an Interoperability Architecture for Blockchain Autonomous Systems 67 (4) 1298–1309. doi:10.1109/TEM.2019.2920154.
[20] R. Belchior, L. Riley, T. Hardjono, A. Vasconcelos, M. Correia, Do You Need a Distributed Ledger Technology Interoperability Solution?doi:10.1145/3564532.
URL https://doi.org/10.1145/3564532
[21] R. Belchior, A. Vasconcelos, M. Correia, T. Hardjono, HERMES: Fault-Tolerant Middleware for Blockchain Interoperabilitydoi:10.36227/TECHRXIV.14120291.V1.
[22] R. Belchior, L. Riley, T. Hardjono, A. Vasconcelos, M. Correia, Do you need a distributed ledger technology interoperability solution?, Distributed Ledger Technologies: Research and PracticeJust Accepted (Sep 2022). doi:10.1145/3564532.
URL https://doi.org/10.1145/3564532
[23] R. Belchior, J. Süßenguth, Q. Feng, T. Hardjono, A. Vasconcelos, M. Correia, A brief history of blockchain interoperabilityCitation Key: belchiorBriefHistoryBlockchain2023a (Sep. 2023). doi:10.36227/techrxiv.23418677.v3.
URL https://www.techrxiv.org/articles/preprint/A_Brief_History_of_Blockchain_Interoperability/23418677/3
[24] EU Blockchain Observatory Forum, The current state of interoperability between blockchain networks, EU Blockchain Observatory Forum Note, [Online]. Available: https://www.eublockchainforum.eu/sites/default/files/reports/EUBOF_Interoperability%20Report-30112023.pdf (Nov 2023).
URL https://www.eublockchainforum.eu/sites/default/files/reports/EUBOF_Interoperability%20Report-30112023.pdf
[25] A. Augusto, R. Belchior, M. Correia, A. Vasconcelos, L. Zhang, T. Hardjono, Sok: Security and privacy of blockchain interoperability, 2023, citation Key: SoKSecurityPrivacy.

URL http://tinyurl.com/sok-sp-interop
[26] T. Haugum, B. Hoff, M. Alsadi, J. Li, Security and privacy challenges in blockchain interoperability - a multivocal literature review, in: Proceedings of the International Conference on Evaluation and Assessment in Software Engineering 2022, EASE '22, Association for Computing Machinery, New York, NY, USA, 2022, p. 347–356.
[27] R. Yin, Z. Yan, X. Liang, H. Xie, Z. Wan, A survey on privacy preservation techniques for blockchain interoperability, Journal of Systems Architecture (2023) 102892doi:10.1016/j.sysarc.2023.102892.
[28] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, S. Muralidharan, C. Murthy, B. Nguyen, M. Sethi, G. Singh, K. Smith, A. Sorniotti, C. Stathakopoulou, M. Vukolić, S. W. Cocco, J. Yellick, Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains, in: Proceedings of the Thirteenth EuroSys Conference, EuroSys '18, Association for Computing Machinery, 2018, event-place: New York, NY, USA. doi:10.1145/3190508.3190538.
URL https://doi.org/10.1145/3190508.3190538
[29] Hyperledger, Private data — hyperledger-fabricdocs master documentation (2022).
URL https://hyperledger-fabric.readthedocs.io/en/release-2.2/private-data/private-data.html
[30] R3 Foundation, R3's Corda Documentation.
URL https://docs.r3.com/
[31] JP Morgan, Quorum White Paper (2017).
URL https://github.com/jpmorganchase/quorum/blob/master/docs/QuorumWhitepaperv0.2.pdf
[32] W. F. Silvano, R. Marcelino, Iota Tangle: A cryptocurrency to communicate Internet-of-Things data, Future Generation Computer Systems 112 (2020) 307–319. doi:10.1016/j.future.2020.05.047.
URL https://www.sciencedirect.com/science/article/pii/S0167739X19329048
[33] C. team, Introduction to Canton — Daml SDK 2.1.1 documentation (2021).
URL https://docs.daml.com/canton/about.html
[34] P. Wegner, Interoperability, ACM Computing Surveys (CSUR) 28 (1) (1996) 285–287, publisher: ACM New York, NY, USA.
[35] EY, EY announces general availability of EY Blockchain Analyzer: Reconciler (2022).
URL shorturl.at/hpMQ8
[36] R. Belchior, A. Vasconcelos, S. Guerreiro, M. Correia, A Survey on Blockchain Interoperability: Past, Present, and Future Trends, ACM Computing Surveys 54 (8) (2021) 1–41.
URL http://arxiv.org/abs/2005.14282
[37] H. Bandara, X. Xu, I. Weber, Patterns for blockchain migration, arXiv preprint arXiv:1906.00239Publisher: Jun (2019).
[38] M. Westerkamp, A. Küpper, SmartSync: Cross-Blockchain Smart Contract Interaction and Synchronization, in: 2022 IEEE International Conference on Blockchain and Cryptocurrency (ICBC), pp. 1–9. doi:10.1109/ICBC54727.2022.9805524.
[39] C. Pedreira, R. Belchior, M. Matos, A. Vasconcelos, Securing Cross-Chain Asset Transfers on Permissioned BlockchainsPublisher: TechRxiv (Jun. 2022). doi:10.36227/techrxiv.19651248.v3.
URL https://www.techrxiv.org/articles/preprint/Trustable_Blockchain_Interoperability_Securing_Asset_Transfers_on_Permissioned_Blockchains/19651248/3
[40] A. Augusto, R. Belchior, A. Vasconcelos, T. Hardjono, Resilient Gateway-Based N-N Cross-Chain Asset TransfersPublisher: TechRxiv (Jun. 2022). doi:10.36227/techrxiv.20016815.v1.
URL https://www.techrxiv.org/articles/preprint/Resilient_Gateway-Based_N-N_Cross-Chain_Asset_Transfers/20016815/1
[41] SAT working group, Re: [Sat] SAT entity identifiers and DIDs (2022).
URL https://mailarchive.ietf.org/arch/msg/sat/4WOAW1JEFRBU6TQHU1PNgdob6cs/
[42] R. Belchior, A. Vasconcelos, M. Correia, T. Hardjono, HERMES: Fault-Tolerant Middleware for Blockchain Interoperability, Future Generation Computer Systems (Mar. 2021). doi:10.36227/TECHRXIV.14120291.V1.
[43] E. Abebe, Y. Hu, A. Irvin, D. Karunamoorthy, V. Pandit, V. Ramakrishna, J. Yu, Verifiable Observation of Permissioned Ledgers, in: 2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC), IEEE, 2021, pp. 1–9.
[44] Inside Bitcoins, What is Terra LUNA - Explaining the LUNA Crash (May 2022).
URL https://insidebitcoins.com/news/explaining-the-luna-crash
[45] Hyperledger Foundation, Hyperledger Fabric Private Data (2020).
URL https://hyperledger-fabric.readthedocs.io/en/release-1.4/private-data/private-data.html
[46] N. Hewett, W. Lehmacher, Y. Wang, Inclusive deployment of blockchain for supply chains, World Economic Forum, 2019.
[47] S. Saberi, M. Kouhizadeh, J. Sarkis, L. Shen, Blockchain technology and its relationships to sustainable supply chain management, International Journal of Production Research 57 (7) (2019) 2117–2135. doi:10.1080/00207543.2018.1533261.

URL https://www.tandfonline.com/doi/full/10.1080/00207543.2018.1533261
[48] M. Bartoletti, S. Lande, L. Pompianu, A. Bracciali, A general framework for blockchain analytics, in: Proceedings of the 1st Workshop on Scalable and Resilient Infrastructures for Distributed Ledgers, 2017, pp. 1–6.
[49] L. Brünjes, M. J. Gabbay, Utxo-vs account-based smart contract blockchain programming paradigms, in: International Symposium on Leveraging Applications of Formal Methods, Springer, 2020, pp. 73–88.
[50] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, et al., Hyperledger fabric: a distributed operating system for permissioned blockchains, in: Proceedings of the thirteenth EuroSys conference, 2018, pp. 1–15.
[51] S. Nakamoto, Bitcoin: A peer-to-peer electronic cash system (2008).
URL http://bitcoin.org/bitcoin.pdf
[52] D. Boneh, M. Drijvers, G. Neven, Compact multi-signatures for smaller blockchains, in: International Conference on the Theory and Application of Cryptology and Information Security, Springer, 2018, pp. 435–464.
[53] R. Belchior, P. Somogyvari, J. Pfannschmid, A. Vasconcelos, M. Correia, Hephaestus: Modelling, analysis, and performance evaluation of cross-chain transactions (Sep 2022). doi:10.36227/techrxiv.20718058.v1.
URL https://www.techrxiv.org/articles/preprint/Hephaestus_Modelling_Analysis_and_Performance_Evaluation_of_Cross-Chain_Transactions/20718058/1
[54] J. Benet, Ipfs-content addressed, versioned, p2p file system, arXiv preprint arXiv:1407.3561 (2014).
[55] M. Al-Bassam, A. Sonnino, V. Buterin, Fraud proofs: Maximising light client security and scaling blockchains with dishonest majorities, arXiv preprint arXiv:1809.09044 160 (2018).
[56] P. Chatzigiannis, F. Baldimtsi, K. Chalkias, SoK: Blockchain Light Clients, Tech. Rep. 1657 (2021).
URL http://eprint.iacr.org/2021/1657
[57] P. Ruan, Y. Kanza, B. C. Ooi, D. Srivastava, Ledgerview: Access-control views on hyperledger fabric, in: Proceedings of the 2022 International Conference on Management of Data, SIGMOD '22, Association for Computing Machinery, New York, NY, USA, 2022, p. 2218–2231. doi:10.1145/3514221.3526046.
URL https://doi.org/10.1145/3514221.3526046
[58] Y. Zhang, S. Wu, B. Jin, J. Du, A blockchain-based process provenance for cloud forensics, in: 2017 3rd IEEE international conference on computer and communications (ICCC), IEEE, 2017, pp. 2470–2473.
[59] A. K. Fedorov, E. O. Kiktenko, A. I. Lvovsky, Quantum computers put blockchain security at risk (2018).
[60] W. Buchanan, A. Woodward, Will quantum computers be the end of public key encryption?, Journal of Cyber Security Technology 1 (1) (2017) 1–22.
[61] T. M. Fernandez-Carames, P. Fraga-Lamas, Towards post-quantum blockchain: A review on blockchain cryptography resistant to quantum computing attacks, IEEE access 8 (2020) 21091–21116.
[62] R. A. Grimes, Cryptography apocalypse: preparing for the day when quantum computing breaks today's crypto, John Wiley & Sons, 2019.
[63] X. Yang, W. Li, A zero-knowledge-proof-based digital identity management scheme in blockchain, Computers & Security 99 (2020) 102050. doi:10.1016/j.cose.2020.102050.
URL https://www.sciencedirect.com/science/article/pii/S0167404820303230
[64] J. D. Blischak, E. R. Davenport, G. Wilson, A quick introduction to version control with git and github, PLoS computational biology 12 (1) (2016) e1004668.
[65] Hyperledger, Hyperledger Besu Ethereum client - Hyperledger Besu (2019).
URL https://besu.hyperledger.org/en/stable/
[66] T. Qiu, R. Zhang, Y. Gao, Ripple vs. swift: transforming cross border remittance using blockchain technology, Procedia computer science 147 (2019) 428–434.
[67] R. Henry, A. Herzberg, A. Kate, Blockchain access privacy: Challenges and directions, IEEE Security & Privacy 16 (4) (2018) 38–45.
[68] G. Yu, X. Wang, K. Yu, W. Ni, J. A. Zhang, R. P. Liu, Survey: Sharding in Blockchains, IEEE Access 8 (2020) 14155–14181. doi:10.1109/ACCESS.2020.2965147.
[69] R. Yin, Z. Yan, X. Liang, H. Xie, Z. Wan, A survey on privacy preservation techniques for blockchain interoperability, Journal of Systems Architecture 140 (2023) 102892. doi:https://doi.org/10.1016/j.sysarc.2023.102892.
URL https://www.sciencedirect.com/science/article/pii/S1383762123000711
[70] M. Li, J. Weng, Y. Li, Y. Wu, J. Weng, D. Li, G. Xu, R. Deng, Ivycross: a privacy-preserving and concurrency control framework for blockchain interoperability, Cryptology ePrint Archive (2021).
[71] A. Kosba, A. Miller, E. Shi, Z. Wen, C. Papamanthou, Hawk: The blockchain model of cryptography and privacy-preserving smart contracts, in: 2016 IEEE symposium on security and privacy (SP), IEEE, 2016, pp. 839–858.
[72] M. Li, Y. Chen, L. Zhu, Z. Zhang, J. Ni, C. Lal, M. Conti, Astraea: Anonymous and secure auditing based on private smart contracts for donation systems, IEEE Transactions on Dependable and Secure Computing (2022).

[73] Z. Guan, Z. Wan, Y. Yang, Y. Zhou, B. Huang, Blockmaze: An efficient privacy-preserving account-model blockchain based on zk-snarks, IEEE Transactions on Dependable and Secure Computing 19 (3) (2020) 1446–1463.
[74] R. Belchior, D. Dimov, Z. Karadjov, J. Pfannschmidt, A. Vasconcelos, M. Correia, Harmonia: Securing Cross-Chain Applications Using Zero-Knowledge Proofs, 2023, citation Key: belchiorHarmoniaSecuringCrossChain2023. doi:10.36227/techrxiv.170327806.66007684/v1.
URL https://www.techrxiv.org/users/679023/articles/695183-harmonia-securing-cross-chain-applications-using-zero-knowledge-proofs?commit=8565e0802b73dd884f9a20de9edef48f4b1aa0fb
[75] Y. Katsis, Y. Papakonstantinou, View-based Data Integration, in: Encyclopedia of Database Systems, Springer US, 2009, pp. 3332–3339. doi:10.1007/978-0-387-39940-9_1072.
URL https://link.springer.com/referenceworkentry/10.1007/978-0-387-39940-9_1072
[76] A. Y. Levy, Logic-Based Techniques in Data Integration, in: Logic-Based Artificial Intelligence, Springer US, 2000, pp. 575–595. doi:10.1007/978-1-4615-1567-8_24.
URL https://link.springer.com/chapter/10.1007/978-1-4615-1567-8_24
[77] G. Verdian, P. Tasca, C. Paterson, G. Mondelli, Quant overledger whitepaper, Release V0. 1 (alpha) (2018).
[78] Blockdaemon, Ubiquity.
URL https://blockdaemon.com/platform/ubiquity/
[79] G. Wood, Polkadot: Vision for a heterogeneous multi-chain framework, White Paper 21 (2016) 2327–4662.
[80] Polkadot, Cross-Consensus Message Format (XCM) · Polkadot Wiki (2021).
URL https://wiki.polkadot.network/docs/learn-crosschain
[81] J. Kwon, E. Buchman, Cosmos whitepaper, A Netw. Distrib. Ledgers (2019).
[82] M. Hargreaves, T. Hardjono, R. Belchior, Secure Asset Transfer Protocol (SATP), no. draft-ietf-satp-core-02, 2023, citation Key: hargreavesSecureAssetTransfer2023a.
URL https://datatracker.ietf.org/doc/draft-ietf-satp-core
[83] A. M. Rozario, C. Thomas, Reengineering the audit with blockchain and smart contracts, Journal of emerging technologies in accounting 16 (1) (2019) 21–35, publisher: American Accounting Association.
[84] Y. Jo, J. Ma, C. Park, Toward Trustworthy Blockchain-as-a-Service with Auditing, in: ICDCS, 2020. doi:10.1109/ICDCS47774.2020.00068.
[85] A. Li, G. Tian, M. Miao, J. Gong, Blockchain-based cross-user data shared auditing, Connection Science (2021) 1–21Publisher: Taylor & Francis.
[86] Blockdaemon, Introducing Blockdaemon's New Staking Dashboard (Sep. 2021).
URL https://blockdaemon.com/blog/introducing-blockdaemons-new-staking-dashboard/
[87] I. Mihaiu, R. Belchior, S. Scuri, N. Nunes, A Framework to Evaluate Blockchain Interoperability Solutions, Tech. rep., TechRxiv (Dec. 2021). doi:10.36227/TECHRXIV.17093039.V2.
URL https://www.techrxiv.org/articles/preprint/A_Framework_to_Evaluate_Blockchain_Interoperability_Solutions/17093039

4 Hermes: Fault-Tolerant Middleware for Blockchain Interoperability

The following chapter corresponds to the following publication [83]:

Status: Published ✓

Publication: Future Generation Computer Systems

Submitted: 30 August 2021

Accepted: 8 November 2021

Citation: R. Belchior, A. Vasconcelos, M. Correia, and T. Hardjono, "HERMES: Fault-Tolerant Middleware for Blockchain Interoperability," Future Generation Computer Systems, Volume 129, April. 2022.

Research Methodology: Design Science Research Methodology [145]

Evaluation Methodology: Formal Proofs [147], Qualitative Evaluation [149]

Journal Description: Computing infrastructures and systems are rapidly developing and so are novel ways to map, control and execute scientific applications which become more and more complex and collaborative. Computational and storage capabilities, databases, sensors, and people need true collaborative tools. Over the last years there has been a real explosion of new theory and technological progress supporting a better understanding of these wide-area, fully distributed sensing and computing systems. Big Data in all its guises require novel methods and infrastructures to register, analyze and distill meaning. FGCS aims to lead the way in advances in distributed systems, collaborative environments, high performance and high performance computing, Big Data on such infrastructures as grids, clouds and the Internet of Things (IoT).

H-Index: 151 **Coverage**: 1984-present **Quartile**: Q1 **Scimago Journal Rank 2022**: 2.04

Future Generation Computer Systems 129 (2022) 236–251

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

HERMES: Fault-tolerant middleware for blockchain interoperability

Rafael Belchior [c,b,*], André Vasconcelos [c,b], Miguel Correia [c,b], Thomas Hardjono [a]

[a] MIT Connection Science & Engineering, Massachusetts Institute of Technology, Cambridge, USA
[b] Department of Computer Science and Engineering, Instituto Superior Técnico, Universidade de Lisboa, Lisbon, Portugal
[c] INESC-ID, Lisbon, Portugal

ARTICLE INFO

Article history:
Received 30 August 2021
Received in revised form 3 November 2021
Accepted 8 November 2021
Available online 8 December 2021

Keywords:
Distributed ledger technology
Blockchain
Interoperability
Digital asset

ABSTRACT

Blockchain interoperability reduces the risk of investing in blockchain systems by avoiding vendor lock-in, enabling a new digital economy, and providing migration capabilities. However, seamless interoperability among enterprises requires service providers to comply with different regulations, e.g., data privacy regulations and others that apply to financial services. For supporting interoperability, organizations can connect to each blockchain via a *gateway*. However, these gateways should be resilient to crashes to maintain a consistent state across ledgers. To realize this vision, we propose HERMES, a fault-tolerant middleware that connects blockchain networks and is based on the Open Digital Asset Protocol (ODAP). HERMES is crash fault-tolerant by allying a new protocol, ODAP-2PC, with a log storage API that can leverage blockchain to secure logs, providing transparency, auditability, availability, and non-repudiation. We briefly explore a use case for cross-jurisdiction asset transfers, illustrating how one can leverage HERMES to support cross-chain transactions compliant with legal and regulatory frameworks.

1. Introduction

There is an increasing interest in digital currencies and virtual assets as the foundation of the next generation digital economy. Blockchain technology has been shown to be a dependable enabler due to its properties, such as immutability, transparency, and auditability [1–3]. Both private organizations and governments are actively investigating and investing in blockchain-based digital assets by, for example, promoting new platforms for digital transactions [4]. A key challenge to enabling this digital economy is to safely connect different networks, enabling network effects among them [5–7].

Interoperability of *blockchains* is, therefore, key to the area [2, 8–10]. Hash lock time contracts, sidechains, oracles and relays, and exchanges are already allowing users to take advantage of this new digital economy in a permissionless environment (see [11] for a detailed survey on the topic).

Although significant progress on interoperability has been made, public blockchains, private blockchains, and legacy systems cannot communicate seamlessly yet [11]. Moreover, current solutions are not standardized and do not offer the possibility to seamlessly transfer data and value across legal jurisdictions, hampering enterprise adoption of blockchain. There is a need for building solutions capable of complying with legal frameworks and regulations.

We believe that similar to Internet routing gateways, which enabled interoperability around private networks, and fostered the rise of the Internet, the global network of decentralized ledgers (DLTs) will require blockchain gateways [2,6]. Gateways permit digital currencies and virtual assets to be transferred seamlessly between these systems. Within the Internet Engineering Task Force (IETF), there is currently ongoing work on an asset transfer protocol that operates between two gateway devices, the *Open Digital Asset Protocol* (ODAP) [2]. ODAP is a cross-chain communication protocol handling multiple digital asset cross-border transactions by leveraging blockchain gateways. Gateways agree on the assets to be exchanged via an asset profile, i.e., a structured, regulation-compliant data model for representing assets (e.g., digital, physical). Transferring an asset among blockchains via gateways is equivalent to an atomic swap that locks an asset in a blockchain and creates its representation on another. However, how can one guarantee a fair exchange of assets (either all parties receive the assets they requested, or none do) across gateways?

To assure the properties that enable a fair exchange of assets, blockchain gateways must operate reliably and be able to withstand a variety of attacks. Thus, a crash-recovery strategy must be a core design factor of blockchain gateways, where specific recovery protocols can be designed as part of the digital asset transaction protocol between gateways. A recovery protocol, allied to a crash recovery strategy, guarantees that the source and

* Corresponding author at: Department of Computer Science and Engineering, Instituto Superior Técnico, Universidade de Lisboa, Lisbon, Portugal.
E-mail address: rafael.belchior@tecnico.ulisboa.pt (R. Belchior).

https://doi.org/10.1016/j.future.2021.11.004
0167-739X/

target DLTs are modified consistently, i.e., that assets taken from the source DLT are persisted into the recipient DLT, and no double spend can occur.

We present Hermes,[1] a middleware for blockchain interoperability that focuses on providing crash recovery capabilities to gateways. The main component of Hermes is an extension of the ODAP protocol that we also introduce in this paper. We denominate this protocol *ODAP-2PC* as it is inspired in the two-phase commit protocol (2PC) [13,14]. Hermes also leverages a log storage API that persists evidence on asset transfer processes across gateways for posterior accountability and dispute-resolution. Hermes's architecture is layered, allowing for cross-chain logic (implemented by business logic plugins, or *BLPs* [15]) to be executed among gateways.

By modeling and developing this system, we expect to address four *research questions*:

- *RQ1* What is the reliability of a cross-chain transaction issued by a gateway, i.e., how can one be sure that a gateway can effectively deliver transactions?
- *RQ2* What is the trade-off between resiliency and performance of gateways?
- *RQ3* How decentralized is Hermes, and how can it be accountable for the transactions it manages?
- *RQ4* What to expect in terms of security and privacy of gateway-based interoperability solutions?

The contributions of this paper are three-fold: first, the blockchain (or DLT) *gateway concept* is explained from a theoretical and practical perspective. Second, we present the Hermes fault-tolerant middleware and its main component, *ODAP-2PC*, a new protocol that provides ACID properties for cross-blockchain transactions. ODAP is also presented. A preliminary implementation of ODAP[2] is available at Hyperledger Cactus, an Hyperledger Foundation project dedicated to DLT interoperability.[3] We provide a comprehensive discussion on Hermes as a solution for blockchain interoperability, focusing on the four research questions we address. Third, and lastly, we present a use case on the exchange of promissory notes across jurisdictions. This use case illustrates how one can leverage Hermes to achieve blockchain interoperability compliant with legal and regulatory frameworks. To be clear, Hermes and ODAP-2PC are contributions of this paper, whereas ODAP is a protocol being designed in the context of the IETF, although by some of the authors of the present paper.

The rest of this paper is structured as follows: Section 2 presents the background. We introduce the gateway concept and Hermes, in Section 3. After, in Section 4, we present ODAP, including the message and logging procedure, the log storage API, and the distributed recovery protocol (Sections 4.2, 4.3, and 4.4, respectively). Section 5, presents a use case that benefits from Hermes. Section 6 presents our discussion on gateways, ODAP, and ODAP-2PC in the light of the presented research questions. The related work follows, in Section 7. Finally, we conclude the paper in Section 8.

2. Preliminaries

This section presents the background on fault tolerance, atomic commit, and nomenclature both on logging and blockchain interoperability.

[1] A preliminary, short, version of this paper appears in IEEE SCC 2021 [12].
[2] We plan to study the performance of our implementation as future work.
[3] https://github.com/hyperledger/cactus.

Fault tolerance

A *fault* is an event that alters the expected behavior of a system. Faults can imply the transition from a correct state of the system to an incorrect state, called *errors*. Errors can provoke *failures* if the system deviates from its specification, possibly causing loss of information or compromising business logic. Nodes can experience failures where for various reasons (e.g., power outage, network partitions, faulty components). We consider four failure types: message loss, communication link failure, site failure, and network partition. Albeit common, failures can be detected by different mechanisms, such as timeouts, defined as the upper bound δ_t that a message is expected [13].

Fault recovery

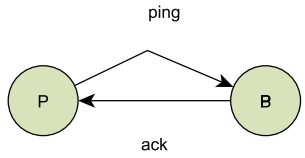
Typically, crash fault-tolerant (CFT) services can tolerate $\frac{n}{2}$ nodes crashing, with n being the number of nodes. As long as there is a majority of nodes with the latest state, failures can be tolerated. The *primary-backup model* defines a set of n hosts (or nodes) that, as a group, assures service resiliency, thus improving availability. In this model, an application client sends messages to a primary node $\mathcal{P}$. The primary nodes redirects the message updates to a set of replicas (backups) $\mathcal{B} = \{\mathcal{B}_1, \ldots, \mathcal{B}_n\}$, when it receives a message. The backup server k propagates the new incoming message to the backup server *k+1*, $k \leq n, k \in \mathbb{R}$. Node $\mathcal{P}$ is then notified of an update when n-host resiliency is met, i.e., the message was at least replicated in n nodes. Should such acknowledgment fail to be retrieved by $\mathcal{P}$, a message update request is re-sent. If $\mathcal{P}$ crashes, then a new leader $\mathcal{P}_{new} \in \mathcal{B}$ is elected. If a backup node receives a request from the application client, it redirects it to $\mathcal{P}$, only accepting it when the latter sends the update request. When an update is received, $\mathcal{P}$ sends the message update to its right-hand neighbor, sending back an acknowledgment.

Another recovery mechanism is *self-healing* [13]. In self-healing, when nodes crash, they are assumed to recover eventually. While this mode is cheaper than primary backup, fewer nodes, fewer exchanged messages, and lower storage requirements, it comes at the expense of availability. In particular, the protocol may block until nodes recover. Fig. 1 depicts a simplified self healing protocol for two nodes. Node P sends a ping to node B, which responds with an ACK. In case of a crash, node B awaits a ping.

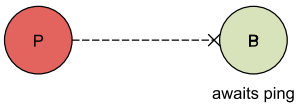
Atomic commit protocols

An atomic commit protocol (ACP) is a protocol that guarantees a set of operations being applied as a single operation. An atomic transaction is indivisible and irreducible: either all operations occur, or none does. ACPs consider two roles: a *Coordinator* that manages the execution of the protocol, and *Participants* that manage the resources that must be kept consistent. ACPs assume stable storage with a write-ahead log (a history of operations is persisted before executed). Examples of ACPs are the two-phase commit protocol, 2PC, the three-phase commit protocol, 3PC, and non-blocking atomic commit protocols [13].

2PC achieves atomicity even in case of temporary system failure, accounting for a wide adoption in academia and in the industry. It has two phases: the voting phase and the commit phase. In the voting phase, the Coordinator prepares all participants to take place in a distributed transaction by inspecting each participant's local status. Each participant executes eventual local transactions required to complete the distributed transaction. If those are successful, participants send a *YES* response to the Coordinator, and the protocol continues. Else, if the *NO* response is sent, it means that the participant chose to abort; this happens when there are problems at the local partition. Next, in the commit phase, when the Coordinator obtains *YES* from all



(a) Both nodes operational. P sends ping to B and gets ACK response.



(b) Node P crashes, leaving B awaiting for a ping.

(c) Node P recovers, re-establishing the connection to B.

Fig. 1. Self-healing mode with two nodes.

participants, a *COMMIT* message is sent to the participants that voted *YES*. This message triggers the execution of local transactions that implement the distributed transaction. Otherwise, the Coordinator sends an *ABORT* message, triggering a rollback on each local partition.

Logging

A log $\mathcal{L}$ is a list of log entries $\{l_1, l_2, \ldots, l_n\}$ such that entries have a total order, given by the time of its creation. A log is considered *shared* when a set of nodes can read and write from the log. On the other hand, a log is *private* (or *local*) when only one node can read and write it. Logs are associated to a process p running *operations* on a certain node. We denote the nth step of process p as (n, p). We denote the ith log entry, as l_i, and the log entry referring to process p and step k as $l^{p,k}$. Both i and k are monotonically increasing positive integers. To manipulate the log, we define a set of *log primitives*, that translate *log entry requests* from a process p into log entries. The log primitives are *writeLogEntry* (writes a log entry), *getLogLength* (obtains the number of log entries), and *getLogEntry(i)* (retrieves a log entry l_i). A log entry request typically comes from a single event in a given protocol.

A *log storage API* provides access to the primitives. Log entry requests have the format <`phase, step, operation, nodes`>, where the field *operation* corresponds to an arbitrary command and the field *nodes* to the parties involved in the process p. We define five operations types to provide context to the protocol being executed:

- Operation type *init*- states the intention of a node to execute a particular operation.
- Operation type *exec*- expresses that the node is executing the operation.
- Operation type *done*- states when a node successfully executed a step of the protocol.
- Operation type *ack*- refers to when a node acknowledges a message received from another.
- Operation type *fail*- indicates to when an agent fails to execute a specific step.

The field *nodes* contains a tuple with a node A issuing a command, or a node A commanding a node B the execution of a command c, if the form is A or $A \rightarrow B$ (c may be omitted), respectively.

Fig. 2 illustrates the logging procedure of some process (or protocol) A, executed by two nodes: Node and Node 2. Process A has three steps. While typically each gateway has its log (and log storage API), we only represent one for simplicity. Note that nodes can also have a common log. Log entry $l_1 = l^{init,1}$ corresponds to the node's first message to the log storage API, which on its turn persists it on a log, using the `writeLogEntry` primitive. The log storage API writes the message that is received. For instance, in step 2, the log storage API executes writeLogEntry<Process A, 1, init-node, Node>. Log entry l_1 is created in step (2), coming from the command issued at step 1. Conversely, writing $l_2 = l^{init,2}$ (steps 4 and)) corresponds to the command that the node issues towards node 2 (step 6), *initAllNodes*, which causes node 2 to issue an init operation. Log entry $l^{init,3}$ corresponds to the execution of init by Node 2 (step 9). At step 12, `getLogLength` returns 3.

Blockchain interoperability

A recent survey classifies blockchain interoperability studies in three categories: Cryptocurrency-directed interoperability approaches, Blockchain Engines, and Blockchain Connectors [11]. Cryptocurrency-directed approaches enable the transfer of digital assets (e.g., cryptocurrencies) across homogeneous and heterogeneous blockchains. The cryptocurrency-directed approaches typically rely on protocols leveraging public blockchains, as they assume that gateways are not trusted. As a result, these approaches are challenging to integrate with permissioned blockchains that support arbitrary assets and smart contracts.

The second category is the blockchain engines, enabling an application-specific blockchain that can communicate with its other instances. These solutions can benefit from implementing gateways, providing each application-specific blockchain (e.g., applications running on a parachain) self-sovereignty regarding communications with other blockchains.

The third category, blockchain connectors, includes trusted relays, blockchain agnostic protocols, blockchain of blockchains solutions, and blockchain migrators. Trusted relays are software components, typically centralized, where escrows route cross-blockchain transactions.

3. Hermes

In this section, we introduce out interoperability middleware, Hermes.

3.1. The concept of gateway

A *gateway* is a hardware device running software capable of interacting with blockchains (e.g., issuing transactions, reading state), and performing computation based on such interactions. Depending on the distributed ledger gateways are connected, they might need to be full nodes, i.e., they may need to implement the whole functionality of a node of that blockchain (e.g., Ethereum). By being present in different DLTs, gateways can perform *cross-chain transactions* (CC-Tx), i.e., transactions including both blockchains, including asset transfers [6]. A *primary gateway* is the DLT system node acting as a gateway in a CC-Tx. Primary gateways may be supported by *backup gateways* for fault tolerance. Primary gateways can be a *source gateway* $\mathcal{G}_S$ or a *recipient gateway* $\mathcal{G}_R$, depending on the role they play in a CC-Tx. Source gateways initiate the gateway-to-gateway protocol, e.g., an asset transfer, data pushing/pulling. Gateways use

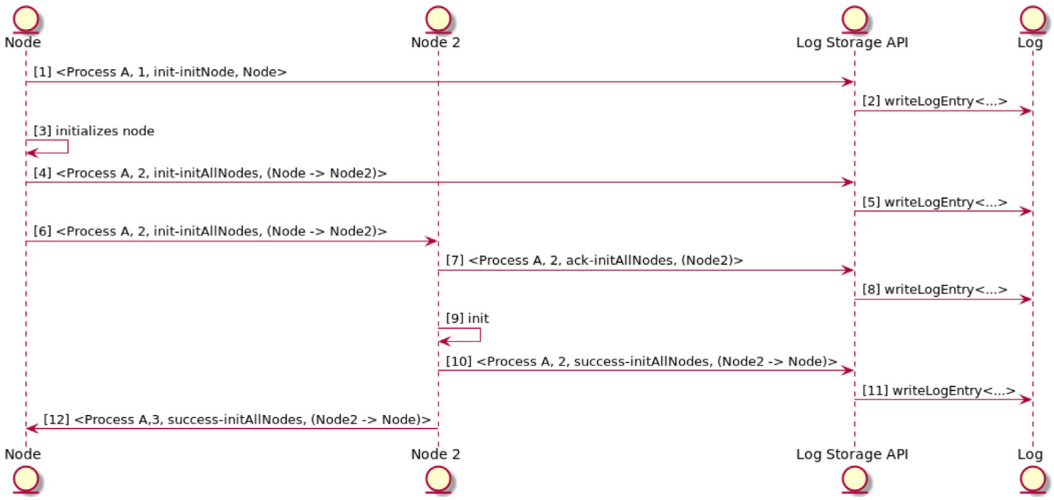


Fig. 2. Example of logging in the context of a process/protocol *A* executed by two nodes. The log storage API writes the incoming message. A single log represents the logs of nodes Node and Node2, but the intent is clear.

machine-resolvable addresses (e.g., URIs/URLs) to communicate with other gateways, obtaining information such as public-key certificates and protocol-specific messages.

For gateways to be crash fault-tolerant, they keep track of each operation they do in a log (of operations). The log is a sequence of log entries, each entry representing a step of the gateway-to-gateway protocol. Each message has a schema, defining the parameters and the payload employed in each message flow. The log data comprises the log information retained by a gateway within a protocol using gateways. A gateway-to-gateway protocol specifies the set of messages and procedures between two gateways for their correct functioning. The gateway-to-gateway protocol considered in this paper is ODAP [2].

3.2. The architecture of hermes

Hermes is a middleware that enables DLT interoperability by implementing part of the software component of gateways. ODAP defines the set of messages (the protocol) for asset transfers at the base layer, realizing technical interoperability. On top of it, ODAP-2PC, a fault-tolerant gateway-to-gateway protocol, provides reliability in the presence of crashes. In case a cross-chain transaction is aborted, ODAP-2PC attempts to issue a rollback on the affected DLTs. ODAP-2PC also provides support for disputes and accountability by providing logging capabilities via the log storage API. Finally, business logic plugins can be implemented (i.e., asset transfer), providing the core rules for a gateway to operate (when to initiate or refuse a transfer). This layer implements semantic interoperability. Clients can use Hermes to support standards that a specific gateway implementation needs to comply with (e.g., Travel Rule [16]). Fig. 3 represents the layers of Hermes.

Our architecture is flexible and modular, as its components are pluggable. By decoupling the protocol from the crash recovery component, and the latter from the business logic plugins, our system can be adapted to specific needs. For instance, in a trustless environment, where gateways do not fully trust each other, a gateway might have a more robust fault recovery mechanism; conversely, if gateways operate in a permissioned environment and completely trust each other, logging capabilities might be reduced to a local log. In this paper, we instantiate Hermes with ODAP and its crash fault-tolerant distributed recovery protocol, ODAP-2PC. The chosen business logic plugin allows promissory note exchanges, presented in detail in Section 5. We presented

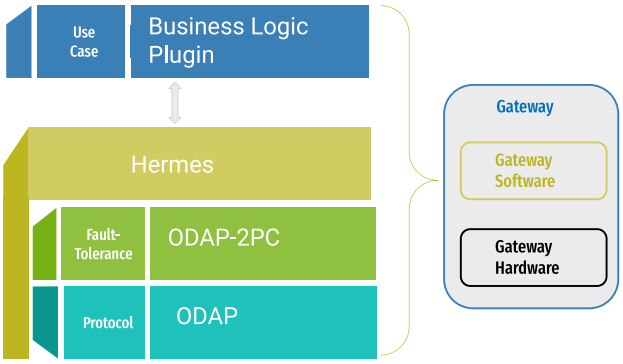


Fig. 3. Hermes's layers.

the architecture of a single Hermes-enabled gateway. In Fig. 4, we present a network compressed of two organizations (A and B), each one with its gateway (Gateway A and Gateway B, respectively). Gateway A is connected to DLT 1, while Gateway B is connected to DLT 2, and a centralized system (e.g., invoice system). This network connects data and assets from DLT 1 to DLT 2 (via Gateway A), DLT 2 to DLT 1 (via Gateway B).

Gateway A establishes a connection to Gateway B via ODAP, exchanging protocol messages. Each gateway has an instance of ODAP-2PC, that guarantees the current state to be preserved. State is written to and read from the distributed log storage, i.e., DLT-based or cloud-based. This storage is accessible by both gateways. Each gateway has its local log, where private information on the gateways operations might be saved (e.g., for data analytics). Hermes redirects ODAP messages to business logic plugins that, in its turn, issue transactions against distributed ledgers (or centralized systems).

3.3. System model

We consider a partially synchronous distributed system (there are unknown bounds on transmission delay and processing time) composed of two types of participants: clients and gateways. Participants have access to a globally synchronized clock (although minor deviations are tolerated).

Clients are in charge of starting transactions and are connected to gateways that are connected to blockchains. More specifically,

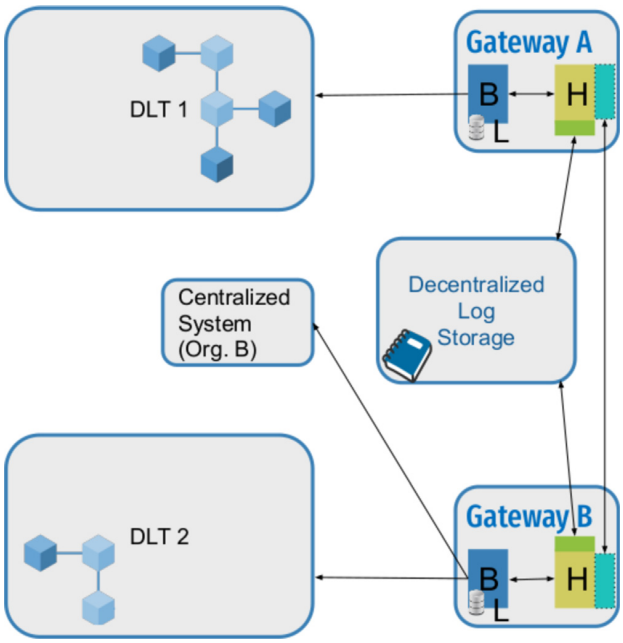


Fig. 4. A Hermes-powered gateway network compressed of two gateways. Within each gateway, the dark blue box represents a business logic plugin (B); the local log is represented by L; the yellow box represents Hermes (H); the light blue box represents ODAP (dotted border); the green box represents ODAP-2PC.

every gateway is a node from a DLT system authorized to act on it (manage assets, identify participants) via, for example, smart contracts. Gateways can communicate with other gateways and can crash (i.e., becoming unresponsive). We assume that there are no Byzantine or arbitrary faults. We assume blockchains are *secure*, so they fail only by crashing. We consider a blockchain secure if the data stored is immutable, transparent to all their participants, traceable, and, generally, the consensus mechanism cannot be subverted by malicious parties.

Gateways store log data about the step of their protocol. This information allows a gateway to construct a state, and recover, in case of a crashe. Gateways are *honest-but-curious*, i.e., follow the protocol, but will attempt to learn all possible information from legitimately received messages. Hermes provides the following properties:

- *P1 Atomicity:* Transactions either commit on all underlying ledgers or entirely fail.
- *P2 Consistency:* All gateways that decide on a CC-Tx reach the same, either commit or abort. The state of the underlying ledgers reflects that decision.
- *P3 Durability:* Once a transaction has been committed, it must remain so regardless of any component crashes.
- *P4 Isolation:* When a transaction is issued, all the underlying assets are locked.
- *P5 Auditability:* Any CC-Tx executed can be inspected by the involved parties.
- *P6 Termination:* If a gateway proposes a transaction, it is eventually committed or aborted.

To satisfy these properties, Hermes leverages ODAP and ODAP-2PC.

3.4. Threat model

ODAP assumes a trusted, secure communication channel between gateways (i.e., messages cannot be spoofed or altered by an adversary) using TLS 1.3 or higher, i.e., the receiver of the communication will ascertain the authenticity validity of the communication. Each gateway has a public and private key pair. New TLS sessions [17] are created when a gateway crashes and then recovers. Clients connect to gateways using a credential scheme such as OAuth2.0 [18].

The distributed recovery protocol has assumptions regarding log management. Log entries need integrity, durability, availability, and confidentiality guarantees, as they are an attractive attack point [19]. Every log entry contains a hash of its payload for guaranteeing integrity. If extra guarantees are needed (e.g., non-repudiation), a log entry might be signed by the gateway creating it (e.g., with ECDSA [20]). Availability is guaranteed using the log storage API, which connects a gateway to dependable storage (local, external, or DLT-based). Each underlying storage provides different guarantees. Access control can be enforced via the *access control profile* that each log can have associated with, i.e., the profile can be resolved, indicating which client can access the log in which condition. Access control profiles can be implemented with access control lists for simple authorization. The authentication of the entities accessing the logs is done at the log storage API level (e.g., username and password authentication in local storage vs. blockchain-based access control in a DLT). We assume the log is not tampered with or lost.

While we consider both gateways to be trusted, we consider a probabilistic, polynomial-time adversary who can corrupt any gateway to prevent the protocol from achieving liveness. The adversary can do this by causing a gateway crash, interrupting an asset transfer. However, we assume that gateways do not deviate from the protocol. We assume the underlying ledgers where gateways operate are safe (i.e., consensus cannot be subverted by an adversary).

4. ODAP-2PC

In this section, we present Hermes' main building block: ODAP-2PC. We start by presenting ODAP, in which ODAP-2PC is based.

4.1. ODAP

The ODAP protocol is a gateway-to-gateway unidirectional asset transfer protocol that uses gateways as the systems conducting the transfer [2]. An asset transfer is represented in the form $T : G_1 \overset{a,x}{\rightarrow} G_2$, where a source gateway G_1 transfers x asset units from type a from a source ledger $\mathcal{B}_S$ to a recipient ledger $\mathcal{B}_R$, via a gateway G_2.

The source gateway issues a transfer such that x asset units will be unavailable at the source DLT and become available at the target DLT. A recipient gateway is the target of an asset transfer, i.e., follows instructions from the source gateway. Hermes leveraged ODAP to provide as strong durability guarantees as to the underlying durability guarantees of the chosen data store. If the datastore is a blockchain, Hermes can achieve transaction durability if transactions are immutable and permanently stored in a secure decentralized ledger.

Durability

Hermes provides the durability guarantees that the infrastructure gateways are connected to.

The transfer process is started by a client (application) that interacts with the source gateway. The source gateway then deals with the complexity of translating an asset transfer request to transactions targeting both the source and the target DLT systems.

The gateway also knows other gateways, either directly or via a decentralized gateway registry. ODAP has several operating modes, but here we solely consider the relay mode. The relay mode realizes client-initiated gateway to gateway asset transfers.

In ODAP, a client application interacts with its local gateway (source gateway GS) over a Type-1 API. The existence of this API allows the client to provide instructions to GS (corresponding to the source gateway) concerning the assets stored in the source DLT and the target DLT (via the recipient gateway, GR). The client may have a complex business logic code that triggers behavior on the gateways. Hence, ODAP allows three flows: the *transfer initiation flow*, where the process is bootstrap, and several identification procedures take place; the *lock-evidence flow*, where gateways exchange proofs regarding the status of the asset to be transferred; and the *commitment establishment flow*, where the gateways commit on the asset transfer. The schema of the messages exchanged by the ODAP protocol is depicted in the "Simplified ODAP Message Format" figure.

Fig. 5 represents ODAP. When an end-user wants to perform an asset transfer, gateways conduct such a process. In the transfer initiation flow (Phase 1), both gateways resolve identities, asset information (via the asset profiles) and establish a secure channel. This verification includes verifying the asset profile validity, the travel rule status, and the pair originator-beneficiary of the transaction [2]. In the lock-evidence verification flow (Phase 2), claims on the status of assets are exchanged, and their correspondent proofs are persisted. The persistence of asset status proof allows for non-repudiation and accountability, proving useful proofs in resolving a dispute.

Theorem 1 (*Isolation*)**.** *Let there be an instance of ODAP, with a source gateway $\mathcal{G}_S$ and a recipient gateway $\mathcal{G}_R$, operating on an asynchronous environment. Given a lock primitive LOCK that prevents assets from being used, if there is a timeout δ_t (applied to steps 2.3 and 3.3 of ODAP), then ODAP provides transaction isolation.*

Proof (*informal*)**.** In this context, transaction isolation implies that a certain asset is locked. At various points of the protocol, both $\mathcal{G}_S$ and $\mathcal{G}_R$ are waiting for messages before proceeding. In particular, in steps 2.3 and 2.4, the logging procedure depends on the success of the asset lock. A trigger δ_t, defining an interval before an asset is used to assure that an asset is securely locked, even in probabilistic-based consensus blockchains. After δ_t counterparty $\mathcal{G}_R$ can produce a log entry with the asset locking proof. When LOCK is called, assets are locked, rendering any attempt of writing fruitless. A similar process occurs in step 3.3. Thus, as assets cannot be changed up to step 3.7, ODAP guarantees transaction isolation. □

Isolation

ODAP-2PC provides transaction isolation by pre-locking assets before the commitment of an asset transfer.

Finally, at the Commitment Establishment Flow (Phase 3), assets are escrowed. In practice, assets are locked on the source ledger and represent those created on the target ledger. The lock of assets prevents double-spend attacks. ODAP aims at providing termination, a non-trivial problem when considering distributed transactions [21]. Thus, we consider three processes on ODAP, p_1 = *transfer initiation flow*, p_2 =*lock-evidence flow*, and p_3 = *commitment establishment flow*. Process p_1 has 2 steps, p_2 has 6 steps, and $p3$ has 6 steps. Thus, a normal end-to-end ODAP flow would have 14 steps.

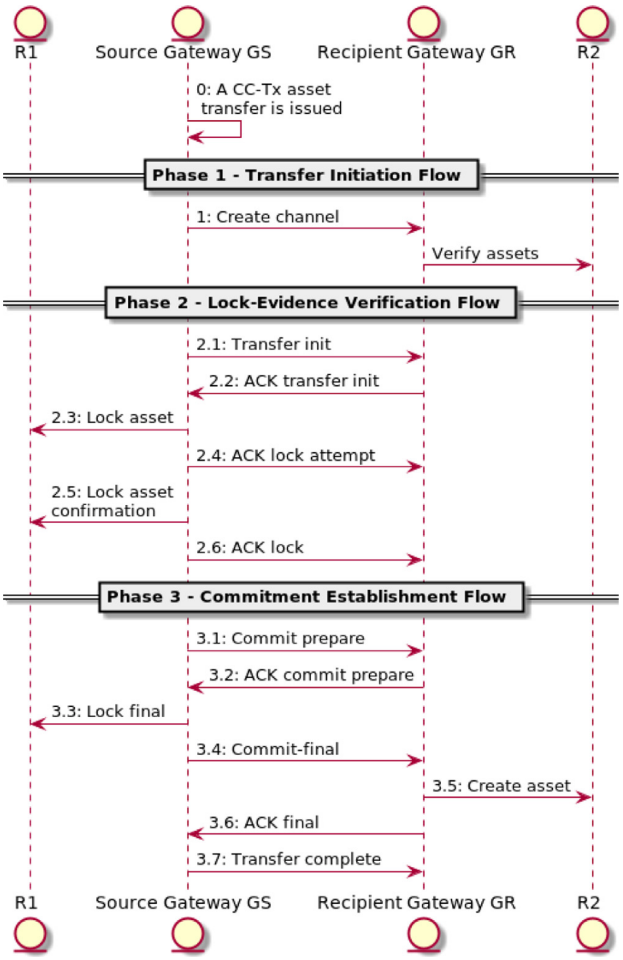


Fig. 5. Simplified sequence diagram depicting ODAP. A transfer is issued by an end-user to the gateway (G1), which then manages on-chain resources (L1), and communicates with a counterparty gateway (G2). The asset transfer corresponds to the creation of L2.

4.2. ODAP-2PC: Message and logging flow

ODAP-2PC aims to solve an important practical limitation of ODAP. ODAP does not handle gateway crashes. If they crash, the protocol may leave the DLTs in some inconsistent state. ODAP-2PC allows the ODAP to continue operating when the faulty gateway recovers, e.g., when the server where it runs reboots.

ODAP messages are exchanged between client applications and gateway servers (DLT nodes). They consist of functional messages allowing protocol negotiation [2]. Messages are encoded in JSON format, allowing for serialization, with protocol-specific mandatory fields. Support for authentication and authorization is provided, allowing for plaintext or encrypted payloads. This servers enterprise needs. ODAP-2PC stores these messages in logs to allow recovery.

We consider the set of logging nodes $\mathcal{N} = \{\mathcal{G}_S, \mathcal{G}_R\}$, with log entry requests with the format <phase, step, type-operation operation, nodes>. Within processes, two types of operations are considered: *private operations* and *public operations*. Private operations involve only one gateway, requiring two log entries, the intention of executing a command, and the execution's confirmation. This serves to handle crashes in systems with only one node. Public operations are operations in which a state is known by more than one node. Intuitively, a private

Table 1
Logging flow regarding the validation operation, of ODAP's phase 1.

Event	From	To	Log ID	Log Content	Operation	Type
$\mathcal{G}_S$ triggers the validation operation	$\mathcal{G}_S$	$\mathcal{G}_R$	$l_4 = l^{p_1,1}$	<p1, 1, init-validate, (GS->GR)>	validate	init
$\mathcal{G}_R$ executes the validation operation	✗	$\mathcal{G}_R$	$l_5 = l^{p_1,2}$	<p1, 2, exec-validate, (GR)>	validate	exec
$\mathcal{G}_R$ completes the validation operation	✗	$\mathcal{G}_R$	$l_6 = l^{p_1,3}$	<p1, 3, done-validate, (GR)>	validate	done
$\mathcal{G}_R$ informs $\mathcal{G}_S$	$\mathcal{G}_R$	$\mathcal{G}_S$	$l_7 = l^{p_1,4}$	<p1, 4, ack-validate, (GR->GS)>	validate	ack

operation is only known by the node executing it, whereas public operations involve several nodes and are thus perceived by more nodes than those executing it.

Simplified ODAP Message Format

1. **Version**: ODAP protocol Version (major, minor)
2. **Resource URL**: Location of Resource to be accessed.
3. **Developer URN**: Assertion of developer/application identity.
4. **Action/Response**: GET/POST and arguments (or Response Code)
5. **Credential Profile**: Specify type of auth (e.g. SAML, OAuth, X.509)
6. **Credential Block**: Credential token, certificate, string
7. **Payload Profile**: Asset Profile provenance and capabilities
8. **Application Profile**: Vendor or Application specific profile
9. **Payload**: Payload for POST, responses, and native DLT txns
10. **Sequence Number**: Sequence Number.

The message flow generates a variable number of log entries, depending on the situation: i) a private operation completes successfully, generating three log entries (init-X, exec-X, done-X); ii) a private operation fails, generating three log entries (init-X, exec-X, fail-X); iii) a public operation completes successfully, generating at least four log entries (init-X, exec-X, done-X, ack-X), and (iv) a public operation fails, generating four log entries (init-X, exec-X, fail-X, ack-X). Given that a normal ODAP flow has 14 steps, one would expect at least 42 log entries.

Let us consider an example where there is an asset transfer $\mathcal{G}_S \overset{a,1}{\rightarrow} \mathcal{G}_R$. We depict a message exchange with content m from $\mathcal{G}_S$ to $\mathcal{G}_R$ by $\mathcal{G}_S \overset{m}{\rightarrow} \mathcal{G}_R$. The reply from $\mathcal{G}_R$ to $\mathcal{G}_S$ is represented by $\mathcal{G}_R \overset{\alpha(m)}{\rightarrow} \mathcal{G}_S$, where α is a function that given an operation, step, and input from a counterparty gateway, returns the response to it. Fig. 6 illustrates part of the message flow involving the public operation $p1$, the ODAP's first phase. Note that one operation has been performed before, corresponding to three log entries (init, exec, done), and to a $\mathcal{G}_S$ client issuing an asset transfer. Thus, the first log entry from p_1 has index 4. In the transfer initiation flow, where $\mathcal{G}_S$ initiates a transfer of one asset a to $\mathcal{G}_R$, the first step is to resolve identities.

To fulfill step 1, $\mathcal{G}_S$ takes two actions: 1) it expresses that $\mathcal{G}_R$ will be informed to initiate an asset transfer; and 2) it sends that message to $\mathcal{G}_R$. These messages are sent to the Log Storage API, that generates the appropriate log entries $l_4 = l^{p_1,1} =< p_1$, 1, init-validate, $(\mathcal{G}_S \rightarrow \mathcal{G}_R) >$, $l_5 = l^{p_1,2} =< p_1$,2, init, $(\mathcal{G}_R) >$, $l_6 = l^{p_1,3} =< p_1$, 3, done-init, $(\mathcal{G}_R) >$, and $l_7 = l^{p,4} =< p_1$, 4, ack-validate, $\mathcal{G}_R >$. Table 1 summarizes the exchanged messages and the log entries they generate. Note that these log entries are simplified, for illustration purposes. ODAP logs have a well-defined schema, and extra parameters, illustrated later in .

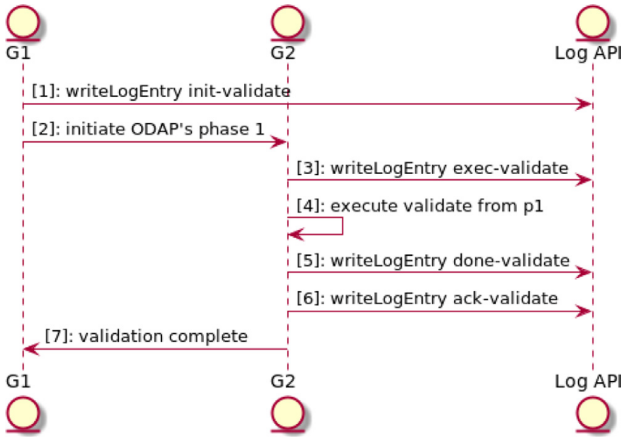


Fig. 6. Message flow regarding the validation operation, of ODAP's phase 1.

We consider a log storage API that allows developers to be abstracted from the storage details (e.g., relational vs. non-relational, local vs. cloud vs. DLT-based) and handles access control if needed. In the next section, we detail the functioning of the log storage API.

4.3. ODAP-2PC: Log storage API

The log storage API allows developers to abstract operations on the log, focusing on the development of gateway-to-gateway protocols. Our API uses the following primitives:

- `initializeLog`(γ): returns a reference to an empty log $\mathcal{L}$, stored on the support γ. The support can be local γ_{local}, cloud γ_{cloud}, or a blockchain γ_{bc}.
- `getLogSupport()`: returns the support γ.
- `writeLogEntry`(l, $\mathcal{L}$): writes a log entry l in the log $\mathcal{L}$, stored on the support γ.
- `getLogEntry`(i): returns the log entry l_i.
- `getLogLength`: returns the length of the log, i.e., $|\mathcal{L}|$.
- `getLatestLogEntry`: returns the log entry l_j such that $\nexists l_i : i > j$
- `getLog`: returns $\mathcal{L}$.

This API can be exposed as a REST API, allowing the log storage API to be hosted in an execution environment different from the one running the gateway implementation. We consider the log file to be a stack of log entries. Each time a log entry is added, it goes to the top of the stack (has the highest index). Logs can be saved either locally (e.g., γ_{local} = computer's disk) and may also be saved in an external service (e.g., γ_{cloud} = cloud storage service) or even in a DLT (e.g., γ_{bc} = Ethereum).

Depending on the support, logs will have different privacy levels. On support γ_{local}, logs are isolated, each gateway keeping its entries private. In case of a crash, the crashed gateway will retrieve the most updated version of the log: if it is local, it needs to require it from other gateways (thus being susceptible

to misbehavior from other gateways). This mode thus requires substantial trust in other gateways. The DLT-based repository, γ_{bc}, offers strong reliability concerning log-saving due to its immutability, transparency, and traceability [19,22]. In particular, this method offers accountability because persisted log entries are non-repudiable, traceable, and cannot be changed; it offers high availability because they are replicated across all nodes participating in the network. The cloud support γ_{cloud} offers a tradeoff between γ_{local} and γ_{bc}, both in terms of cost and integrity guarantees. As a cloud provider mediates this support, trust is put on the provider instead of uniquely on the counterparty gateway. However, it is likely to be more costly than the local support.

Format of log entries

The log entries' format should account for three phases, in case the gateway-to-gateway protocol is ODAP. In Section 4.2 we introduced a simplified version of a log entry for illustration purposes. The mandatory fields for a log entry for ODAP-2PC are:

ODAP-2PC Log Schema – Mandatory Fields

1. **Session ID**: unique identifier (UUIDv2) representing an ODAP interaction (corresponding to a particular flow)
2. **Sequence Number**: represents the ordering of steps recorded on the log for a particular session
3. **ODAP Phase ID**: flow to which the logging refers to. Can be Transfer Initiation flow, Lock-Evidence flow, and Commitment Establishment flow.
4. **Source Gateway ID**: the public key of the gateway initiating a transfer Source DLT ID: the ID of the gateway initiating a transfer
5. **Recipient Gateway ID**: the public key of the gateway involved in a transfer Recipient DLT ID: the ID of the gateway involved in a transfer
6. **Timestamp**: timestamp referring to when the log entry was generated (UNIX format)
7. **Payload**: Message payload: contains subfields *Votes* (optional), *Msg*, *Message type*. The field Votes refers to the votes parties need to commit in the 2PC. Msg is the content of the log entry. Message type refers to the different logging actions (e.g., command, backup).
8. **Payload Hash**: hash of the current message payload

Apart from mandatory log fields, the log schema for ODAP-2PC contains optional fields. The *logging profile* field contains the profile regarding the logging procedure. If not present, $\gamma = \gamma_{local}$ is assumed. The *Source Gateway UID* is the unique identifier (UID) of the gateway initiating a transfer. The *Recipient Gateway UID* is the UID of the gateway involved in a transfer. The *Message Digest* is a gateway signature over the log entry. The *Last Log Entry* is the hash of the previous log entry. Finally, the *Access Control Profile* is the field specifying a profile regarding the confidentiality of the log entries being stored; in particular, this field can be used to parse access control policies to the supports managing logs. Next, we introduce the ODAP-2PC, a distributed recovery mechanism for gateways.

4.4. ODAP-2PC: distributed recovery procedure

One of the key deployment requirements of gateways for asset transfers is a high degree of gateways availability. A distributed recovery procedure then increases the resiliency of a Hermes gateway by tolerating faults. Next, we present an overview of ODAP-2PC.

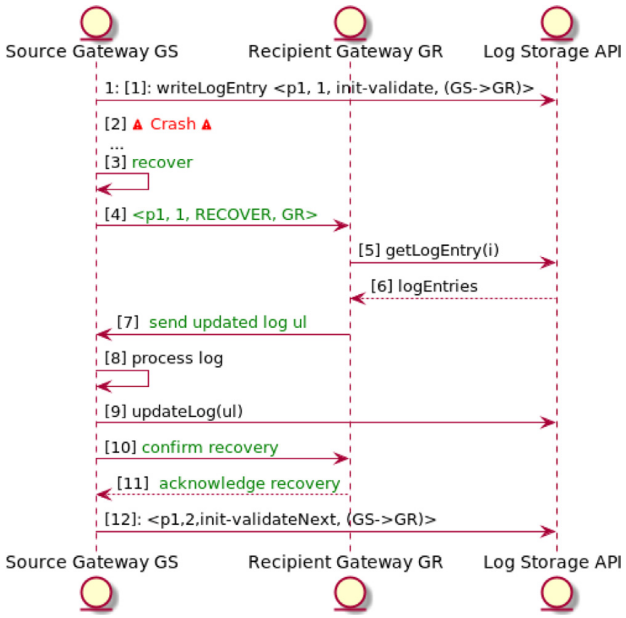


Fig. 7. $\mathcal{G}_S$ crashing before issuing init-validation to $\mathcal{G}_R$.

Overview

The protocol is crash fault-tolerant, so it does not tolerate Byzantine faults (i.e., gateways that behave arbitrarily). Gateways are trusted to operate the ODAP protocol as specified unless they crash.

ODAP-2P support two alternative fault tolerance strategies:

1. *self-healing mode:* after a crash, a gateway eventually recovers, informs other parties of its recovery, and continues executing the protocol;
2. *primary-backup mode:* after a crash, a gateway may never recover, but that timeout can detect this failure [21]; if a node is crashed indefinitely, a backup is spun off, using the log storage API to retrieve the log's most recent version.

In self-healing mode – the mode we detail in this paper – when a gateway restarts after a crash, it reads the state from the operation log and executes the protocol from that point on. We assume that the gateway does not lose its long-term keys (public–private key pair) and can reestablish all TLS connections. In Primary-backup mode, we assume that after a period δ_t of the failure of the primary gateway, a backup gateway detects that failure unequivocally and takes the role of the primary. The failure is detected using heartbeat messages and a conservative value for δ_t. For that purpose, the backup gateway does essentially the same as the gateway in self-healing mode: reads the log and continues the process. In this mode, the log must be shared between the primary and the backup gateways. If there is more than one backup, a leader-election protocol must be executed to decide which backup will take the primary role.

In both modes, logs are written before operations (write-ahead) to provide atomicity and consistency to the protocol used for asset exchange. The log data is considered as resources that may be internal to the DLT system, accessible to the backup gateway and possible other gateway nodes.

There are several situations when a crash may occur. Fig. 7 represents the crash of $\mathcal{G}_S$ before it issues a validation operation to $\mathcal{G}_R$ (steps 1 and 2). Both gateways keep their log storage APIs, with γ_{local}. For simplicity, we only represent one log storage API. In the self-healing mode, the gateway eventually recovers (step

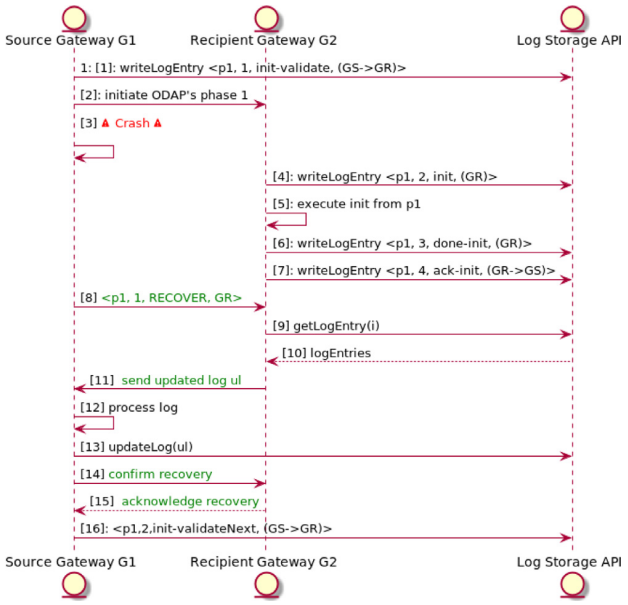


Fig. 8. $\mathcal{G}_S$ crashing after issuing the init command to $\mathcal{G}_R$.

3), building a recovered message in the form `<phase, step, RECOVER, nodes>` (step 4). The non-crashed gateway queries the log entries that the crashed gateway needs (steps 5, 6). In particular, $\mathcal{G}_S$ obtains the necessary log entries at step 7 and compares them to its current log. After that, $\mathcal{G}_S$ attempts to reconcile the changes with its current state (step 8). Upon processing, if both log versions match, the log is updated, and the process can continue. If the logs differ, then $\mathcal{G}_S$ calls the primitive `updateLog`, updating its log (step 9) and thus allowing the crashed gateway to reconstruct the current state. In this particular example, step 9 would not occur because operations exec-validate, done-validate, and ack-validate were not executed by $\mathcal{G}_R$. If the log storage API is on the shared mode, no extra steps for synchronizations are needed. After that, it confirms a successful recovery (steps 10, 11). Finally, the protocol proceeds (step 12).

Fig. 8 represents a recovery scenario requiring further synchronization. At the retrieval of the latest log entry, $\mathcal{G}_S$ notices its log is outdated. It updates it upon necessary validation and then communicates its recovery to $\mathcal{G}_R$. The process then continues as normal. (for instance, corresponding to exec-validate, done-validate, and ack-validate)

4.4.1. The ODAP-2PC protocol

In this section, we present the ODAP-2PC protocol itself. In particular, this protocol is used at ODAP's Phase 3, crucial for the atomicity and the consistency of asset transfers. We consider two parties: the coordinator $\mathcal{G}_S$, and the participant $\mathcal{G}_R$. The coordinator manages the protocol execution while the participant follows the coordinator's instructions.

ODAP-2PC is a 2PC protocol able to detect and recover from crashes, delivering the effort to execute an asset transfer starting at ODAP's phase 3: the commitment establishment flow. Crashes at other phases of the ODAP are handled by the self-healing mechanism, supported by the messaging and logging mechanism, as depicted by Figs. 7 and 8. In phase 3, sensitive messages that include the lock and unlocking of assets may not arrive due to failures (e.g., communication failures, gateway crash due to power outage). To detect crashes, we use a timeout δ_C. However, processes may wait for the crashed gateway to recover for an unbounded timespan, wasting resources (e.g., locked assets). To avoid this, we introduce an additional timeout $\delta_{rollback}$. When a gateway does not recover before this timeout, a *timeout action* is triggered, corresponding to the *rollback protocol*. A possible rollback protocol cancels the current transactions by issuing transactions with the contrary effect, guaranteeing the consistency of the DLT whose gateway is not crashed. Upon recovery, the crashed gateway is informed of the rollback, performing a rollback too. This process guarantees the consistency of both underlying DLTs.

Algorithm 1: ODAP-2PC Protocol

Input: Coordinator $\mathcal{G}_S$, Participant $\mathcal{G}_R$, Asset a, Gateway primitives PRE_LOCK, LOCK, COMMIT,CREATE_ASSET, COMPLETE, ROLLBACK
Result: Asset a transferred from $\mathcal{G}_S$ to $\mathcal{G}_R$

1 $PO_{\mathcal{G}_S} = \perp$ ▷ rollback list for $\mathcal{G}_S$
2 $PO_{\mathcal{G}_R} = \perp$ ▷ rollback list for $\mathcal{G}_R$
3 ▷ Pre-Voting Phase
4 preLock $= \mathcal{G}_S$.PRE_LOCK(a) ▷ step 2.3
5 $PO_{\mathcal{G}_S}$.append(preLock)
6 ▷ Voting Phase
7 $\mathcal{G}_S \xrightarrow{vote-req} \mathcal{G}_R$ ▷ step 3.1
8 **wait until** $\mathcal{G}_R \xrightarrow{\alpha(vote-req)} \mathcal{G}_S$ ▷ step 3.2
9 ▷ Decision Phase
10 **if** $\mathcal{G}_R \xrightarrow{\alpha(vote-req)} \mathcal{G}_S = NO$ **then**
11 $\quad \mathcal{G}_S \xrightarrow{abort()} \mathcal{G}_R$ ▷ otherwise, $\mathcal{G}_R \xrightarrow{\alpha(vote-req)} \mathcal{G}_S = YES$
12 $\quad \mathcal{G}_S$.ROLLBACK($PO_{\mathcal{G}_S}$) ▷ undo $\mathcal{G}_S$.*preLock*(a)
13 **end if**
14 *lock* $= \mathcal{G}_S$.LOCK(a) ▷ step 3.3
15 $PO_{\mathcal{G}_S}$.append(lock)
16 *commit* $= \mathcal{G}_S$.COMMIT() ▷ step 3.4
17 **if** *commit* $= \perp$ **then**
18 $\quad \mathcal{G}_S \xrightarrow{abort()} \mathcal{G}_R$
19 $\quad \mathcal{G}_S$.*rollback*($PO_{\mathcal{G}_S}$) ▷ undo $\mathcal{G}_S$.LOCK(a)
20 **end if**
21 $\mathcal{G}_S \xrightarrow{commit} \mathcal{G}_R$
22 $a' = \mathcal{G}_R$.CREATE_ASSET() ▷ step 3.5
23 $PO_{\mathcal{G}_R}$.append(a')
24 **wait until** $\mathcal{G}_R \xrightarrow{\alpha(commit)} \mathcal{G}_S$ ▷ step 3.6
25 **if** $\mathcal{G}_R \xrightarrow{\alpha(commit)} \mathcal{G}_S = COMMIT$ **then**
26 $\quad \mathcal{G}_S$.COMPLETE() ▷ step 3.8
27 **end if**
28 **else**
29 $\quad \mathcal{G}_S \xrightarrow{abort()} \mathcal{G}_R$ ▷ otherwise, $\mathcal{G}_R$ failed the commit
30 $\quad \mathcal{G}_S$.ROLLBACK($PO_{\mathcal{G}_S}$) ▷ undo $\mathcal{G}_S$ locks
31 $\quad \mathcal{G}_R$.ROLLBACK($PO_{\mathcal{G}_R}$) ▷ undo $\mathcal{G}_R$.CREATE_ASSET()
32 **end if**
33 return ▷ asset transferred

Algorithm 1 depicts the ODAP-2PC. A coordinator $\mathcal{G}_S$ and a participant $\mathcal{G}_R$ perform a CC-Tx T, that typically is an asset transfer of x number of a assets, i.e., $T : \mathcal{G}_S \xrightarrow{a,x} \mathcal{G}_R$. Any time a party ABORTS, the protocol stops, and that transaction is considered invalid (and thus the run of the protocol fails). We define a set of gateway primitives Σ = {PRE_LOCK, UNLOCK, LOCK, COMMIT, CREATE_ASSET, COMPLETE, ROLLBACK}, such that they realize pre-locking an asset, locking an asset, unlocking an asset, committing to a CC-Tx, creating an asset, asserting for the end of the protocol, and performing a rollback, respectively. The gateway primitives are divided into two types: off-chain primitives and on-chain primitives, represented by $\sigma^{offchain}$ and $\sigma^{onchain}$, respectively. Some off-chain primitives call their respective on-chain primitive. The protocol receives a set of gateway primitives that realize the commit, locking, rollback, and other operations. Lists $PO_{\mathcal{G}_S}$ and $PO_{\mathcal{G}_R}$ track the operations to be rolled back in case of failure for $\mathcal{G}_S$or $\mathcal{G}_R$, respectively.

First, in the session opening, the asset to be transferred is agreed on. At the pre-voting phase, the source gateway initiates the process, pre-locking an asset (executing the transaction right to the point before its commitment, at step 2.3, line 4). The

recipient gateway confirms this pre-locking, issuing a VOTE-REQ to its counterparty (line 7). The recipient gateway replies either YES or ABORT (line 8), starting the decision phase. Note that the eventual ABORT, at line 8, does not require a rollback because, so far, no on-chain operations took place. At the beginning of the decision phase, if $\mathcal{G}_R$ replies NO, then the pre-lock is rolled back, and the transaction aborted (lines 11 and 12). Otherwise, $\mathcal{G}_S$ tries to lock the asset to be transferred (line 14) and commit that action (line 16). The recipient gateway completes the pending transactions (line 22) and sends an acknowledgment message back to the source gateway (line 24). Upon the second commit, the source gateway completes the process, closing the session (line 26). However, if $\mathcal{G}_S$ cannot commit (line 25 is not COMMIT), the transaction is aborted, and the respective rollbacks are triggered.

If the participant $\mathcal{G}_R$ does not reply on the blocking operations (within $t < \delta_R$, $\mathcal{G}_S$ considers $\mathcal{G}_R$ crashed, and starts the *recovery protocol*). The recovery protocol may be trivial: in ODAP-2PC, firstly, the gateway awaits the counterparty gateway to recover (by assumption, it does). Upon recovery, the process depicted by steps 4–11 from Fig. 7 takes place. Conversely, if $\mathcal{G}_S$ does not respond within $t < \delta_S$, the same process occurs. It is worth noting that the coordinator may issue the rollback at any point $t > \delta_{rollback}$, where $\delta_{rollback} > \delta_R$, i.e., it does not need to wait indefinitely for the participant to recover. For both cases, if the recovering awaiting period is greater than the rollback timeout protocol, i.e., $t > \delta_{rollback}$, the *rollback protocol* is triggered.

Consistency

ODAP-2PC provides transaction consistency by employing a self-healing strategy based on a write-ahead log: all parties either COMMIT or ABORT the CC-Tx (i.e., the asset transfer). The rollback protocol assures the consistency of the underlying DLTs.

Theorem 2 (*Termination*)**.** *Let there be an instantiation of ODAP-2PC in the self-healing mode, with a coordinator $\mathcal{G}_S$ and a participant $\mathcal{G}_R$, operating on an asynchronous environment. Given a coordinator timeout δ_S and a participant timeout δ_R, ODAP-2PC assures that ODAP terminates.*

Proof (informal): At various points of the protocol, both $\mathcal{G}_S$ and $\mathcal{G}_R$ are waiting for messages before proceeding, in particular at lines 3, 4, and 17. In line 3, $\mathcal{G}_R$ waits for a VOTE-REQ message. Since this gateway can decide to abort before it votes YES, it can abort and stop the process if it has the timeout action triggered. In line 4, $\mathcal{G}_S$ is waiting for a YES or NO message. At this stage, there is still no decision on how to proceed ($\mathcal{G}_R$ still did not decide to COMMIT). Thus, the coordinator can decide to abort in case of timeout by sending ABORT to the other gateway and stopping the process. In line 17, in case it voted YES, gateway $\mathcal{G}_R$ is waiting for a COMMIT or ABORT message. In case of a crash, $\mathcal{G}_R$ gateway remains blocked until $\mathcal{G}_S$ recovers. By assumption, there is an upper bound in which gateways recover from crashes. Thus, gateways will be able to communicate and thereby reach a decision. □

Termination

ODAP-2PC does not block indefinitely, providing liveness regarding its termination.

4.4.2. Rollback protocol

The process of rolling back blockchain-based transactions is not trivial. As most blockchains are immutable, rolling back means issuing a transaction with the opposite effect of the first. We call this a *canceling* a transaction. For example, canceling a PRE_LOCK(a) and LOCK would imply issuing a transaction unlocking a, whereas CREATE_ASSET would imply the destruction of a created asset. The rollback protocol includes two parties: the canceling gateway and the counterparty gateway. The canceling gateway realizes the need to cancel one or more transactions, initiate the rollback protocol, and propagate eventual corrective measure commands to the counterparty gateway. There is a need to involve a counterparty gateway to ensure the consistency of the assets handled by the protocol.

The rollback process occurs as follows: 1) the canceling gateway undoes the transactions to be rolled back by issuing transactions with the contrary effect; 2) the same gateway sends an acknowledgment back to the counterparty gateway, and 3) counterparty gateway undoes all its pending transactions, which can lead back to step one, where the counterparty gateway serves as the canceling gateway. This recursive protocol may generate a cascade effect where several transactions from both blockchains need to be canceled. Our rollback protocol is triggered at step 3.3 or 3.5. At step 2.3, if a lock is unsuccessful, there is still no transaction to undo (an ABORT is sent). Steps 2.4 and 3.7 are assumed to be successful, i.e., issuing a transaction that creates a log entry succeeds. In particular, if the log entry cannot be persisted in the blockchain support, alternative support is used by the Log Storage API, and the respective party is warned.

Atomicity

The ODAP-2PC protocol provides transaction atomicity,

It is worth noting that the ODAP-2PC and its rollback protocol depend on the implementation of a set of gateway primitives, as well as a specific asset schema. In the next section, we briefly present a use case leveraging gateway primitives.

5. Use case: Gateway-supported cross-jurisdiction promissory notes

This section presents a use case implementing digital asset transfers, benefiting from the gateway paradigm. The digital assets to be exchanged are defined are standardized in as an *asset profile*, which is ongoing work at the IETF [23]. An asset profile is "the prospectus of a regulated asset that includes information and resources describing the virtual asset". A virtual asset, on its turn, is "a digital representation of value that can be digitally traded" [23]. Asset profiles can be emitted by authorized parties, having the capability to represent real-world assets (e.g., real estate) legally.

5.1. Asset profile

The *Asset Profile Definitions for DLT Interoperability* draft presents an unambiguous manner of representing a digital asset, independently of its concrete implementation [23]. The representation of an asset via an asset profile allows for representing physical assets (called tokenization) that then can be exchanged across DLTs with Hermes. Hermes validates a given asset profile definition, allowing gateways to agree on the asset to be exchanged, in the *Transfer Initiation Flow*. An asset profile contains the following fields (from [23]):

Asset Profile Schema

1. **Issuer**: The registered name or legal identifier of the entity issuing this asset profile document.
2. **Asset Code**: The unique asset code under an authoritative namespace assigned to the virtual asset.
3. **Asset Code Type**: The code type to which the asset code belongs under an authoritative namespace.
4. **Issuance date**: The issuance date of the Asset Profile JSON document.
5. **Expiration date**: The expiration of the Asset Profile JSON document in terms of months or years.
6. **Verification Endpoint**: The URL endpoint where anyone can check the current validity status of the Asset Profile JSON file.
7. **Digital signature**: The signature of the Issuer of the Asset Profile.
8. **Prospectus Link**: The link to any officially published prospectus, or non-applicable.
9. **Key Information Link**: The link to any Key Information Document (KID), or non-applicable.
10. **Keywords**: The list of keywords to make the Asset Profiles easily searchable. It can be blank or non-applicable.
11. **Transfer Restriction**: Information about transfer restrictions (e.g., prohibited jurisdictions), or non-applicable.
12. **Ledger Requirements**: Information about the specific ledger mechanical requirement, or non-applicable.

We refer to this asset profile as $\mathcal{A}_p$. For generic protocols manipulating assets (e.g., transfer, creating), this asset profile can provide the necessary attributes for trust establishment. For instance, gateways should verify their counterparty identity in case of an asset transfer. Moreover, the asset profile and asset code should be identifiable and retrievable, allowing different attributes to be parsed as inputs to the asset gateway primitives.

5.2. Asset gateway primitives

Based on the proposed digital asset schema, we present pseudo-code for the gateway primitives used in ODAP-2PC. We recall the gateway primitives: off-chain primitives (COMMIT, ROLLBACK, and COMPLETE) and on-chain primitives (PRE-LOCK, LOCK, UNLOCK, and CREATE_ASSET). The sequencing of off-chain operations, performed by gateways and on-chain operations, allows the asset transfer. For instance, based on a specific asset profile $\mathcal{A}_p$, gateways validate eventual restrictions (e.g., jurisdiction restrictions) on a certain asset, at the validation phase, before PRE_LOCK an asset (in case the protocol comprises transferring an asset).

To implement the primitives, we define an additional field on $\mathcal{A}_p$ to represent a digital asset: *state*. Four possible states exist: an asset is unlocked (can be used without constraints on that ledger), pre-locked (the asset will be transferred, and thus cannot be used), locked (asset was transferred and cannot be used), and burnt (asset was destroyed or permanently locked).

Algorithm 2 depicts the procedure to implement a PRE-LOCK, LOCK, and UNLOCK, if the level is pre-lock, lock, or unlock, respectively. If the level is burnt, then an additional DLT-specific operation needs to eliminate (burn) the asset. The PRE_LOCK primitive issues a LOCK, temporarily locking an asset on $\mathcal{G}_S$,

Algorithm 2: On-chain set state

```
Input: Asset a, Ledger connector c, lock level l
Result: Asset a locked at l
1 assetRepresentation = c.getStateById(a.assetCode)      ▷ DLT-specific
2 assetRepresentation.state = l      ▷ pre-lock, lock, unlocked, burnt
3 c.setState(assetRepresentation)      ▷ DLT-specific
```

setting the state of the asset to *pre-locked*. After that, the gateway waits for confirmation from the counterparty gateway of such an operation. In case the protocol fails before COMMIT, a ROLLBACK is issued by $\mathcal{G}_S$, triggering an UNLOCK transaction. The UNLOCK sets the state of the pre-locked asset to *unlocked*, reverting the effect of the PRE-LOCK.

If a COMMIT is successful, then two operations happen: 1) in $\mathcal{G}_S$ a LOCK is issued, setting the state of the asset to *locked*, meaning it cannot be used; 2) $\mathcal{G}_R$ issues a CREATE_ASSET, creating a representation of the original asset on the recipient ledger. If the whole process is successful, according to ODAP-2PC, $\mathcal{G}_S$ issues a COMPLETE. All operations are logged via the log storage API; an additional on-chain primitive LOG is considered if the logging takes place on-chain.

5.3. Using hermes to exchange promissory notes

Promissory notes are freely transferable financial instruments where issuers denote a promise to pay another party (payee) [24]. Notes are globally standardized by several legal frameworks, providing a low-risk instrument to reclaim liquidity from debt. Notes contain information regarding the debt, such as the amount, interest rate, maturity date, and issuance place. Notes are useful because they allow parties to liquidate debts and conduct financial transactions faster, overcoming market inefficiencies. In practice, promissory notes can be both payment and credit instruments. A promissory note typically contains all the terms about the indebtedness, such as the principal amount, credit rating, interest rate, expiry date, date of issuance, and issuer's signature. Despite their benefits, paper promissory notes are hard to track, require hand signatures and not-forgery proofs, accounting for cumbersome management. To address these challenges, recent advances in promissory notes' digitalization include FQX's eNote [25]. Blockchain-supported digital promissory notes (eNotes) worth about half a million dollars were used by a "Swiss commodity trader to finance a transatlantic metal shipment" [26]. eNotes are stored in a trusted ledger covered by the legal framework, belonging to a specific jurisdiction. Consider the following supply chain scenario: a producer (P) produces a certain amount of goods that sells to a wholesaler (W). W accepted the goods, and now P issues an invoice of value V. The wholesaler could pay in, for example, 90 days. Because P does not want to wait up to 90 days for its payment, it requests a promissory note from W, stating that V will be paid in 90 days. This way, P can sell that same promissory note to a third party. The promissory note is abstract from any physical good being exchanged. Depending on the issuer, collateral might not be needed, as the accountability for liquidating the debt is tracked by the blockchain where it is stored.

Blockchain-based promissory notes belonging to a particular jurisdiction are stored in a certified blockchain that exposes a gateway. When a promissory note needs to change jurisdictions (e.g., a promissory note issued in the USA that needs to be redeemed in Europe), the gateways belonging to the source and target blockchains perform an asset transfer the asset is a digital promissory note. Alternatively, the gateway extends to several jurisdictions. Below is an example of an asset profile of a digital promissory note. Such digital promissory notes can be trivially

exchanged between blockchains using Hermes and the ODAP-2PC protocol, where gateways belonging to different jurisdictions (e.g., representing different blockchains regulated by different entities) perform asset transfers.

Promissory Note Example

1. **Issuer**: FQX AG
2. **Asset Code**: CH0008742519
3. **Asset Code Type**: ISIN
4. **Keywords**: Electronic Promissory Note; eNote; Debt
5. **Prospectus Link**: N/A
6. **Key Information Link**: N/A
7. **Transfer Restriction**: shall not be transferred to the U.S., Canada, Japan, United Kingdom, South Africa. Shall not be transferred to non-qualified investors anywhere.
8. **Ledger Requirements**: Hyperledger Fabric v2.x.
9. **Original Asset Location**: N/A
10. **Previous Asset Location**: N/A
11. **Issuance date**: 04.09.2020
12. **Verification Endpoint**: https://fqx.ch/profile-validate
13. **Signature Value**: (signature blob)

6. Discussion

Hermes can include different gateway implementations, different gateway-to-gateway protocols, and different distributed recovery mechanisms. Modularity and pluggability allow Hermes to be flexible regarding different legal frameworks, supporting different privacy and performance requirements. In particular, Hermes can be instantiated in blockchains supporting smart contracts that implement functionality for locking and unlocking assets. The gateway paradigm allows integrating DLT-based systems to centralized legacy systems by leveraging existing legal frameworks. For extra robustness, data integrity and counterparty performance can be attested using trusted hardware [27,28]. Remote attestations are particularly important since provably exposing the internal state to external parties is a crucial requirement for CC-Txs [29].

Gateways can also be leveraged for tasks other than asset transfers; they can perform the function of oracles, either centralized or decentralized [11], allowing to integrate blockchains with external systems and data providers. An oracle's general goal is to retrieve data, validate and deliver it to a blockchain, or pull information from a blockchain [30]. An oracle may provide extra functions, such as showing proof of original data, incentivizing oracle services (e.g., rewarding nodes providing information to the oracle), and even privacy (encrypting data). As a gateway, Hermes can implement asset transfers through the ODAP protocol or serve as an oracle.

6.1. RQ1: Reliability

Our solutions implement a strong consistency model among gateways, where there are no dirty writes or repeated reads during an atomic transaction (e.g., atomic asset exchange). This is achieved by sacrificing liveness and using one of two mechanisms: on the primary backup mode, n-host resiliency is provided by sequencing backups and using acknowledgment messages. These messages assure that the update has progressed at least to the following backup beyond itself. However, primary backup introduces a latency overhead, as the client application only retrieves the output from the message update request after n replicas have been updated. On the other hand, the self-healing mechanism, allied to a resilient log storage API, provides means for developers to save the ODAP state, even in the presence of crashes. We should point out that this strong consistency model solely applies to shared state among gateways and not shared state among blockchains. Hermes delivers eventual consistency among blockchains (either both operations eventually happen, or none does), but no stronger guarantees (e.g., strict consistency, strong consistency).

ODAP and ODAP-2PC assume a trade-off between reliability and efficiency, according to the end-to-end principle [31]. The more reliable a gateway is (in terms of accountability, termination, and ACID properties), the higher the overhead is in terms of performance. The storage capability of gateways, abstracted by the log storage API, determines gateways' robustness, as logs are used to dispute resolution and accountability. Shared, non-repudiable, and immutable log entries provide better guarantees than locally stored logs [19,22]. Thus, the log storage API serves two purposes: 1) it provides a reliable means to store logs created by all gateways involved in an asset transfer, and thus ensures consistency, atomicity, and isolation; and 2) promotes accountability across parties, reducing the risk of counterparty fraud.

6.2. RQ2: Performance

As mentioned, a trade-off between reliability and performance exists. Storing logs in local storage typically has lower latency but delivers weaker integrity and availability guarantees than store them on the cloud or in a ledger. Generally, the more resilient the support γ is, the higher the latency ($\gamma_{bc} > \gamma_{cloud} > \gamma_{local}$). For critical scenarios where strong accountability and traceability are needed (e.g., financial institution gateways), blockchain-based logging storage may be appropriate. Conversely, for gateways that implement interoperability between blockchains belonging to the same organization (i.e., a legal framework protects the legal entities involved), local storage might suffice.

ODAP-2PC exchanges messages to assure atomicity, leading to blocking operations, where operations depend on the state of the other gateway. In particular, γ_{bc} implies issuing a blockchain transaction, several orders of magnitude slower than writing on disk or even writing on a cloud-based storage [32], especially if one waits for confirmation, depending on the blockchain, it may require up to dozens of minutes. The self-healing mode is compatible with the three types of logs, but the primary backup mode could require the log storage API on support external to the gateway.

6.3. RQ3: Decentralization

Gateway-to-gateway business transactions depend on the social and technological trust that stakeholders build. In particular, as every operation is saved on a log, this log can be used for disputes in case of misbehavior by any stakeholder. In particular, in case of dispute, the involved parties can inspect the logs and recur to the legal frameworks [22] from the jurisdiction in which the asset transfer occurs. Thus, for the legislated spaces and proper log storage support, Hermes might be sufficiently decentralized. While this is acceptable for enterprise scenarios, as accountability is guaranteed, there may be cases in which gateways are not trusted. Considering non-trusting gateways, Hermes might not be sufficiently decentralized. Besides picking the appropriate log storage support, one could choose several techniques to decentralize gateways or enhance the accountability level.

A first option is to implement a gateway as a smart contract: this does not allow a gateway to deviate from its configured behavior but has shortcomings, such as inflexibility, lack of scalability, and operation costs. In particular, smart contracts often lack the possibility of being integrated with external resources and systems; oracles may provide some extra flexibility [11]. Smart contract-based gateways could also need to pay transaction fees in public blockchains, such as gas on Ethereum [33], raising additional costs. Additional costs imply that adding gateways on the same blockchain is not scalable.

Second, to decentralize Hermes, one could implement a Byzantine fault-tolerant version of a gateway, similarly to what is planned on Cactus [15]. In this case, it is not a single gateway conducting the message delivery process but a quorum of gateways that belong to different stakeholders. In a permissioned scenario, stakeholders could represent different departments, with the caveat that they should periodically publish proofs of state in an external repository [19]. If gateways are sufficiently decentralized, gateways do not need to be implemented as smart contracts. This allows better scalability than the smart contract and flexibility in integrating legacy systems and infrastructure with the gateways.

A third option is to secure computation leveraging trusted hardware to enable remote attestation [27,28]. Remote attestation is a method allowing a device to authenticate its hardware and software to a centralized service, proving its integrity, and thus its trustworthiness. Working as an additional security layer, device-level attestations would enable gateways to provide truthful evidence of their internal state. Evidence would then promote trust across gateways, diminishing the risk of collusion and misbehavior. This solution would be essential for financial institution gateways involving digital asset transfers with monetary value.

6.4. RQ4: Security and privacy

Gateways should assure the integrity and non-repudiation of log entries and ensure that the protocol terminates. If an adversary performs a denial-of-service on either gateway, the asset transfer is denied, but ODAP-2PC assures eventual consistency of the underlying DLTs. Accountability promoted by robust storage can diminish the impact of these attacks. The connection between gateways should always provide an authentication and authorization scheme, e.g., based on OAuth and OIDC [34], and use secure channels based on TLS/HTTPS [17].

Gateways should be flexible enough to accommodate not only different legal frameworks but also different notions of privacy. Reasoning about different privacy levels, one key question is: what should be the privacy granularity level regarding an issuer and beneficiary transaction of a digital asset? Some regulations imply that both parties are identified, and such records are maintained for several years. However, for cryptocurrency exchanges across public blockchains, privacy might be of more significant concern. A second question follows: what are the privacy guarantees of the gateway performing such transfers, mainly if logging functions are jointly performed, on blockchain-based support? This question can be answered with privacy policies and cherry-picking the information written in publicly available logs. Future research on the security and privacy of gateways is needed before they are ready for production use.

Another privacy-related aspect is the encapsulation of internal asset representation. Although gateways are working with a specific asset schema, each gateway needs to be aware of the asset represented by the underlying DLT (or at least DLT client), i.e.; it needs to convert ODAP messages to blockchain-specific transactions. Thus, the gateway has the responsibility of converting a standard representation on a DLT-specific one. If desirable, gateways can hide representation details, providing privacy regarding asset management.

7. Related work

Building dependable and trusted middleware for blockchain interoperability requires the orchestration of several disciplines. This section introduces related work on blockchain interoperability solutions, crash recovery, and other contributions.

Interoperability solutions

Blockchain interoperability solutions are diverse and numerous. However, few solutions can accommodate a seamless integration among public blockchains and non-public blockchains (private blockchains, legacy systems). Thus, we focus on the comparison of Hermes with this class of solutions. Table 2 presents existing related work with regard to several criteria (and *sub-criteria)*:

- Digital Asset Support: whether the solution can transfer (1) *Utility Tokens* (non-fungible tokens, such as ERC-20 tokens [35]), or (2) *Payment Tokens* (fungible tokens, such as Bitcoin, or Ether).
- Regulation: if the solution is implemented such that it is compliant with (1) a *Legal* framework or (2) an existing or ongoing *Standard*.
- Crash Recovery: the solution supports *Crash Faults* and/or *Byzantine Faults*.

Hardjono et al. proposed a gateway-based architecture inspired by the architecture of the Internet [6], further expanded by recent work [51]. The gateway-based paradigm bootstrap the emergence of standardization efforts such as ODAP [2] at the IETF [50]. Such protocols, in which Hermes is based, aim to comply with the Travel Rule and FAFT regulations [49].

Ghaemi et al. [45] proposed a publisher-subscriber architecture for blockchain interoperability, based on connector applications that publish on the connector smart contract held by the broker blockchain. However, the connectors do not have crash recovery mechanisms and thus are not suitable for a production environment.

Hyperledger Cactus [15] is a trusted relay connecting DLTs, whereby a consortium of Cactus Nodes endorses transactions. Cactus aims to be a general-purpose interoperability solution that uses two families of software components that, in its sum, constitute a gateway: validators and connectors. Validators are components that retrieve state from blockchains, while connectors are active components that issue transactions. The consortium can run arbitrary business logic, including logic for asset transfers, making Cactus a suitable infrastructure to implement gateways. However, Cactus Nodes have no crash recovery mechanism implemented and thus are not suitable for a production environment. Weaver also aims to provide general-purpose interoperability, using proofs of state of private blockchains [29], called a blockchain view [52]. On top of that, they propose interoperability RFCs [48].

Quant Overledger is a blockchain interoperability enterprise solution [36] compliant with the ongoing standardization effort from ISO [47]. Overledger exposes functionalities from different ledgers and legacy systems via a REST API. Overledger has crash recovery mechanisms and can be deployed on different clouds. However, this solution is not open source, and it is not clear if a rollback protocol is provided.

Other solutions are equally promising, but lack crash recovery capabilities, unlike Overledger, ODAP, and Hermes [38–44]. We refer readers interested in interoperability to the survey in [11], where each solution is analyzed in greater detail.

A short paper on some of the ideas presented in this paper appeared before [12]. The present paper is three times larger and presents in detail what the other barely sketches: Hermes, ODAP-2PC, etc.

Table 2
Classification of blockchain interoperability solutions connecting public and non-public blockchains. The green checkmark (✓) indicates a subcriteria is fulfilled. A red cross (✗) indicates otherwise. The gray question mark (?) indicates that the criteria may or not be fulfilled (we lack information to decide).

		Digital asset support		Regulation		Crash recovery	
Paper	Year	Payment tokens	Utility tokens	Legal	Standardized	Crash faults	Byzantine faults
Quant Overledger [36]	2018	✓	✓	✓[a]	✓[b]	✓	✗
Bifrost [37]	2019	✓	✓	✗	✗	✗	✗
Abebe et al. [38]	2019	✗	✓	✗	✗	✗	✗
Wang et al. [39]	2020	?	?	✗	✗	✗	✗
Zhao et al. [40]	2020	?	?	✗	✗	✗	✗
Cactus [15]	2020	✓	✓	✗	✗	✗	✗
Weaver [29]	2020	✓	✓	✗	✓[c]	✗	✗
Gewu et al. [41]	2020	✗	✓	✗	✗	✗	✗
SCIP [42]	2020	✗	✓	✗	✓[d]	✗	✗
Nissl et al. [43]	2020	✓	✓	✗	✗	✗	✗
Fynn et al. [44]	2020	✓	✓	✗	✗	✗	✗
ODAP [2]	2021	✓	✓	✓[e]	✓[f]	✓	✗
Ghaemi et al. [45]	2021	✗	✓	✗	✗	✗	✗
This paper	2021	✓	✓	✓[e]	✓[f]	✓	✗

[a]International standards [46].
[b]ISO/TC 307/SG 7 [47].
[c]Weaver Interop. RFCs [48].
[d]SCIP Protocol [42].
[e]Travel Rule, FAFT [49].
[f]IETF [50].

Crash recovery on cross-chain transactions

Two-phase commit was initially developed as an atomic commitment protocol that coordinates distributed transactions. Thus, it could be seen as a consensus mechanism over the global state encompassing each distributed database. Generally, 2PC is not used for blockchain consensus [53], but rather for assuring the reliability of atomic cross-chain transactions. Fynn et al. presented a Move operation that can migrate accounts and arbitrary computation across Ethereum virtual machine-based chains [44]. An atomic Move operation can be implemented with 2PC. Wang et al. [39] presented a 2PC protocol for conducting CB-Tx. A blockchain is elected as the coordinator in this scheme, managing the process between an arbitrary number of blockchains. This protocol includes a heartbeat monitoring mechanism to guarantee liveness.

However, it is unclear how ACID properties are assured, e.g., atomicity, as the authors do not provide a rollback protocol. Our work provides ACID properties via ODAP-2PC and the rollback protocol. Herlihy et al. discuss the need for developing models for cross-chain transactions that evolve from traditional ACID properties when non-trusted actors are involved [54]. Gateways in Hermes are assumed to be trusted, and thus ACID properties seem reasonable to model transactions across these systems. However, and for future work, a decentralized ODAP system, where a set of mutually non-trusting gateways represents each jurisdiction, can benefit from the modeling mentioned above. A decentralized ODAP and its recovery mechanism, ODAP-3PC, could pave the way for resilience towards Byzantine faults and malicious actors.

Standardization efforts

Standardization processes typically take years from inception until publishing. While several standardization efforts are focusing on blockchain interoperability, none have been published and widely adopted. While it is likely that there will be several competing standards, the most active ones seem to be ODAP (IETF [50]), ISO 307/SG 7 [47], and the IEEE Blockchain Initiative [55].

Hermes is one of the few solutions (along with Quant Overledger) that is being built considering the existing standardization efforts.

Other contributions

Orthogonally, several contributions support the gateway paradigm: Vo et al. propose decentralized blockchain registries that can identify and address blockchain oracles [7]. Chen and Hardjono proposed an IETF draft proposing a method for identification of computer systems that act as gateways and the correct validation of the ownership of the gateway [56]. This identification occurs via DNS, where a gateway owner registers for an "Autonomous System number from ARIN, or other region networking authorities (such as RIPE NCC for Europe and APNIC for East and South Asia)" [56]. Self-sovereign based identity promotes identity portability, by deferring the authentication and authorization processes to the end-user [11,57,58]. Gateways could then be identified by a decentralized identifier and issued verifiable credentials by certified authorities that manage virtual asset providers.

8. Conclusion

In this paper, we presented Hermes, a blockchain interoperability middleware that enables gateway-to-gateway asset transfers via the Open Asset Digital Protocol. Hermes can support asset transfer across jurisdictions, contributing towards regulation-compliant, standardized, blockchain interoperability middleware. We have shown that our solution is resilient to crashes by leveraging ODAP-2PC, a distributed recovery mechanism. This implies that asset transfers are atomic, fair, and no double spending can occur. A use case on the exchange of digital promissory notes is presented, showing that Hermes is an appropriate trust anchor for enterprise use cases requiring cross-blockchain asset transfers. Future work will enable several gateways to be involved in an atomic asset transfer (using ODAP-3PC), paving the way for efficient multiparty atomic swaps.

CRediT authorship contribution statement

Rafael Belchior: Conceptualization, Formal analysis, Investigation, Validation, Writing – original draft. **André Vasconcelos:** Conceptualization, Supervision, Writing – review & editing. **Miguel Correia:** Conceptualization, Investigation, Supervision, Writing – review & editing. **Thomas Hardjono:** Conceptualization, Investigation, Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

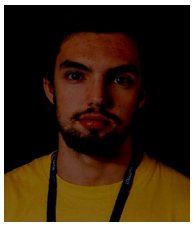
Acknowledgments

We warmly thank Benedikt Schuppli for discussions on digital promissory notes and our colleagues in the IETF's forming working group on ODAP for fruitful discussions. This work was partially supported by national funds through Fundação para a Ciência e a Tecnologia (FCT) with reference UIDB/50021/2020 (INESC-ID) and 2020.06837.BD, and by the European Commission program H2020 under grant agreement 822404 (QualiChain).

References

[1] C. Catalini, J.S. Gans, Some Simple Economics of the Blockchain, Working Paper Series 22952, National Bureau of Economic Research, 2016, https://doi.org/10.3386/w22952, URL http://www.nber.org/papers/w22952.
[2] M. Hargreaves, T. Hardjono, R. Belchior, Open digital asset protocol draft 02, draft-hargreaves-odap-02, 2021, Internet Engineering Task Force, URL https://datatracker.ietf.org/doc/html/draft-hargreaves-odap-02.
[3] W. Viriyasitavat, L. Da Xu, Z. Bi, V. Pungpapong, Blockchain and internet of things for modern business process in digital economy—the state of the art, IEEE Trans. Comput. Soc. Syst. 6 (6) (2019) 1420–1432.
[4] A. Pentland, A. Lipton, T. Hardjono, Time for a new, digital bretton woods, 2021, Barron's, URL https://rb.gy/yj31vq.
[5] L. Pawczuk, M. Gogh, N. Hewett, Inclusive Deployment of Blockchain for Supply Chains: A Framework for Blockchain Interoperability, Tech. Rep., World Economic Forum, 2020, URL www.weforum.org.
[6] T. Hardjono, A. Lipton, A. Pentland, Towards an interoperability architecture blockchain autonomous systems, IEEE Trans. Eng. Manag. 67 (4) (2019) 1298–1309, URL doi:10.1109/TEM.2019.2920154.
[7] H. Tam Vo, Z. Wang, D. Karunamoorthy, J. Wagner, E. Abebe, M. Mohania, Internet of Blockchains: Techniques and Challenges Ahead, in: 2018 IEEE International Conference on Internet of Things (IThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), 2018, pp. 1574–1581.
[8] B. Pillai, K. Biswas, Blockchain interoperable digital objects innovative applications of blockchain technology view project blockchain interoperable asset classes view project, 2019, http://dx.doi.org/10.1007/978-3-030-23404-1_6, URL https://doi.org/10.1007/978-3-030-23404-1_6.
[9] M. Borkowski, M. Sigwart, P. Frauenthaler, T. Hukkinen, S. Schulte, DeXTT: Deterministic cross-blockchain token transfers, IEEE Access 7 (2019) 111030–111042, arXiv:..
[10] S. Schulte, M. Sigwart, P. Frauenthaler, M. Borkowski, Towards blockchain interoperability, in: C. Di Ciccio, R. Gabryelczyk, L. García-Bañuelos, T. Hernaus, R. Hull, M. Indihar Štemberger, A. Kő, M. Staples (Eds.), Business Process Management: Blockchain and Central and Eastern Europe Forum, Springer International Publishing, Cham, 2019, pp. 3–10.
[11] R. Belchior, A. Vasconcelos, S. Guerreiro, M. Correia, A survey on blockchain interoperability: Past, present, and future trends, ACM Comput. Surv. 58 (8) (2022) 1–41, http://dx.doi.org/10.1145/3471140, arXiv:2005.14282.
[12] R. Belchior, A. Vasconcelos, M. Correia, T. Hardjono, Enabling cross-jurisdiction digital asset transfer, in: IEEE International Conference on Services Computing, IEEE, 2021.
[13] P.A. Bernstein, V. Hadzilacos, N. Goodman, Concurrency Control and Recovery in Database Systems, Addison-Wesley, 1987.
[14] R.J. Patton, Fault-tolerant control: The 1997 situation, IFAC Proc. Vol. 30 (18) (1997) 1029–1051, http://dx.doi.org/10.1016/s1474-6670(17)42536-5.
[15] H. Montgomery, H. Borne-Pons, J. Hamilton, M. Bowman, P. Somogyvari, S. Fujimoto, T. Takeuchi, T. Kuhrt, R. Belchior, Hyperledger cactus whitepaper, 2020, URL https://github.com/hyperledger/cactus/blob/master/docs/whitepaper/whitepaper.md.
[16] T. Hardjono, A. Lipton, A. Pentland, Towards a public key management framework for virtual assets and virtual asset service providers, J. FinTech 1 (1) (2020) http://dx.doi.org/10.1142/S2705109920500017.
[17] E. Rescorla, RFC 8446 - The transport layer security (TLS) protocol version 1.3, 2014, URL https://tools.ietf.org/html/rfc8446.
[18] E. D. Hardt, RFC 6749 - The oauth 2.0 authorization framework, 2012, URL https://tools.ietf.org/html/rfc6749.
[19] B. Putz, F. Menges, G. Pernul, A secure and auditable logging infrastructure based on a permissioned blockchain, Comput. Secur. 87 (2019) 101602.
[20] D. Johnson, A. Menezes, S. Vanstone, The elliptic curve digital signature algorithm (ECDSA), Int. J. Inf. Secur. 1 (1) (2001) 36–63, http://dx.doi.org/10.1007/s102070100002.
[21] P. ALsberg, J. Day, A principle for resilient sharing of distributed resources, J. Chem. Inform. Model. (1976) arXiv:arXiv:1011.1669v3.
[22] R. Belchior, A. Vasconcelos, M. Correia, Towards Secure, Decentralized, and Automatic Audits with Blockchain, in: European Conference on Information Systems, 2020.
[23] A. Sardon, T. Hardjono, Benedikt Schuppli, Asset Profile Definitions for DLT Interoperability (draft-sardon-blockchain-interop-asset-profile-00), Tech. Rep., 2021, URL https://datatracker.ietf.org/doc/draft-sardon-blockchain-interop-asset-profile/.
[24] J.S. Waterman, The promissory note as a substitute for money, Minn. Law Rev. 14 (1929) 313.
[25] FQX, eNI™ Infrastructure - fqx.ch - Electronic Negotiable Instruments - FQX, 2020, URL https://fqx.ch/.
[26] Transatlantic Shipment of Metals Financed via FQX eNote | Treasury Management International, URL https://treasury-management.com/news/transatlantic-shipment-of-metals-financed-via-fqx-enote/.
[27] T. Hardjono, N. Smith, Towards an attestation architecture for blockchain networks, World Wide Web J. 24 (9) (2021) 1587–1615, http://dx.doi.org/10.1007/s11280-021-00869-4.
[28] G. Coker, J. Guttman, P. Loscocco, A. Herzog, J. Millen, B. O'Hanlon, J. Ramsdell, A. Segall, J. Sheehy, B. Sniffen, Principles of remote attestation, Int. J. Inf. Secur. 10 (2) (2011) 63–81, http://dx.doi.org/10.1007/s10207-011-0124-7, URL https://link.springer.com/article/10.1007/s10207-011-0124-7.
[29] E. Abebe, D. Karunamoorthy, J. Yu, Y. Hu, V. Pandit, A. Irvin, V. Ramakrishna, Verifiable observation of permissioned ledgers, 2021, arXiv, arXiv:2012.07339v2.
[30] R. Mühlberger, S. Bachhofner, E. Castelló Ferrer, C. Di Ciccio, I. Weber, M. Wöhrer, U. Zdun, Foundational oracle patterns: Connecting blockchain to the off-chain world, in: Lecture Notes in Business Information Processing, Vol. 393 LNBIP, Springer Science and Business Media Deutschland GmbH, 2020, pp. 35–51, http://dx.doi.org/10.1007/978-3-030-58779-6_3, arXiv:2007.14946.
[31] J.H. Saltzer, D.P. Reed, D.D. Clark, End-to-end arguments in system design, ACM Trans. Comput. Syst. (TOCS) 2 (4) (1984) 277–288, http://dx.doi.org/10.1145/357401.357402, URL https://dl.acm.org/doi/10.1145/357401.357402.
[32] A. Bessani, M. Correia, B. Quaresma, F. Andre, P. Sousa, DepSky: Dependable and secure storage in a cloud-of-clouds, ACM Trans. Storage 9 (4) (2013) 1–33.
[33] G. Wood, Ethereum: A Secure Decentralised Generalised Transaction Ledger. Byzantium version 7e819ec, Tech. Rep., 2019, URL https://ethereum.github.io/yellowpaper/paper.pdf.
[34] Final: OpenID Connect Core 1.0 incorporating errata set 1, URL https://openid.net/specs/openid-connect-core-1_0.html.
[35] F. Vogelsteller, V. Buterin, EIP 20: ERC-20 Token standard, 2015, URL https://eips.ethereum.org/EIPS/eip-20.
[36] G. Verdian, P. Tasca, C. Paterson, G. Mondelli, Quant Overledger Whitepaper Vo.1, Tech. Rep., Quant, 2018, pp. 1–48, URL http://objects-us-west-1.dream.io/files.quant.network/Quant_Overledger_Whitepaper_v0.1.pdf.
[37] E.J. Scheid, T. Hegnauer, B. Rodrigues, B. Stiller, Bifröst: a modular blockchain interoperability API, in: IEEE 44th Conference on Local Computer Networks, Institute of Electrical and Electronics Engineers (IEEE), 2019, pp. 332–339.
[38] E. Abebe, D. Behl, C. Govindarajan, Y. Hu, D. Karunamoorthy, P. Novotny, V. Pandit, V. Ramakrishna, C. Vecchiola, Enabling enterprise blockchain interoperability with trusted data transfer, in: Proceedings of the 20th International Middleware Conference Industrial Track, Association for Computing Machinery, 2019, pp. 29–35.
[39] X. Wang, O.T. Tawose, F. Yan, D. Zhao, Distributed nonblocking commit protocols for many-party cross-blockchain transactions, 2020, arXiv, arXiv:2001.01174.
[40] D. Zhao, T. Li, Distributed cross-blockchain transactions, 2020, arXiv, arXiv:2002.11771v1.
[41] G. Bu, R. Haouara, T.S.L. Nguyen, M. Potop-Butucaru, Cross hyperledger fabric transactions, in: CRYBLOCK 2020 - Proceedings of the 3rd Workshop on Cryptocurrencies and Blockchains for Distributed Systems, Part of MobiCom 2020, Association for Computing Machinery, 2020, pp. 35–40, http://dx.doi.org/10.1145/3410699.3413796.
[42] G. Falazi, U. Breitenbücher, F. Daniel, A. Lamparelli, F. Leymann, V. Yussupov, Smart contract invocation protocol (SCIP): A protocol for the uniform integration of heterogeneous blockchain smart contracts, in: International Conference on Advanced Information Systems Engineering, Vol. 12127 LNCS, 2020, pp. 134–149.
[43] M. Nissl, E. Sallinger, S. Schulte, M. Borkowski, Towards cross-blockchain smart contracts, 2020, arXiv:2010.07352.
[44] E. Fynn, F. Pedone, B. Alysson, Smart contracts on the move, in: Dependable Systems and Networks, 2020.
[45] S. Ghaemi, S. Rouhani, R. Belchior, R.S. Cruz, H. Khazaei, P. Musilek, A pub-sub architecture to promote blockchain interoperability, 2021, arXiv:2101.12331.
[46] Quant, Overledger Enterprise, URL https://www.quant.network/overledger-enterprise.
[47] ISO, ISO - ISO/TC 307 - Blockchain and distributed ledger technologies, URL https://www.iso.org/committee/6266604.html.

[48] Weaver, Weaver Interoperability RFCs, URL https://github.com/hyperledger-labs/weaver-dlt-interoperability/tree/main/rfcs.
[49] T. Hardjono, A. Lipton, A. Pentland, Towards a Public Key Management Framework for Virtual Assets and Virtual Asset Service Providers, Tech. Rep., 2019, arXiv:1909.08607v1.
[50] IETF, IETF | Internet Engineering Task Force, URL https://www.ietf.org/.
[51] T. Hardjono, Blockchain gateways, bridges and delegated hash-locks, 2021, arXiv:2102.03933.
[52] R. Belchior, S. Guerreiro, A. Vasconcelos, M. Correia, A survey on business process view integration, 2020, arXiv:2011.14465.
[53] J. Nijsse, A. Litchfield, A taxonomy of blockchain consensus methods, Cryptography 4 (4) (2020) 32.
[54] M. Herlihy, B. Liskov, L. Shrira, Cross-chain deals and adversarial commerce, PVLDB, in: Cross-Chain Deals and Adversarial Commerce, vol. 13, (2) 2019, pp. 100–113, http://dx.doi.org/10.14778/3364324.3364326, URL https://doi.org/10.14778/3364324.3364326.
[55] IEEE, P3205 - Standard for Blockchain Interoperability - Data Authentication and Communication Protocol, URL https://standards.ieee.org/project/3205.html.
[56] S. Chen, T. Hardjono, Gateway identification and discovery for decentralized ledger networks, 2021, Internet-Draft draft-chen-dlt-gateway-identification-00, Internet Engineering Task Force Work in Progress, URL https://datatracker.ietf.org/doc/html/draft-chen-dlt-gateway-identification-00.
[57] R. Belchior, B. Putz, G. Pernul, M. Correia, A. Vasconcelos, S. Guerreiro, SSI-BAC : SElf-sovereign identity based access control, in: The 3rd International Workshop on Blockchain Systems and Applications, IEEE, 2020.
[58] M.S. Ferdous, F. Chowdhury, M.O. Alassafi, In search of self-sovereign identity leveraging blockchain technology, IEEE Access 7 (2019) 103059–103079, http://dx.doi.org/10.1109/access.2019.2931173.



Eng. Rafael Belchior is a researcher at INESC-ID in the Distributed Systems Group and Ph.D. student at Técnico Lisboa, where he started lecturing in 2018.

Studying the intersection of blockchain interoperability and security, he contributes for Hyperledger Cactus, ODAP's forming group at IETF, and European Commission projects Qualichain and DE4A. He worked as a blockchain research engineer at the Portuguese government, Linux Foundation, and Quant. Rafael also volunteers as a speaker and mentor (invited classes, Hyperledger Summer Internship program).

André is Assistant Professor (Professor Auxiliar) in the Department of Computer Science and Engineering, Instituto Superior Técnico, Lisbon University, and researcher in Information and Decision Support Systems Lab at INESC-ID, in Enterprise Architecture domains namely representation, and modeling of Architectures of Information Systems, and Evaluation of Information Systems Architectures.

He has over 15 years of experience in advising organizations in information systems and enterprise architectures in eGovernment and private sector projects. Head of teams accountable for delivering state of art Information and Communication Technology (ICT) innovation aligned with business needs, in distinctive projects.

Miguel Correia is a Full Professor at Instituto Superior Técnico (IST), Universidade de Lisboa, in Lisboa, Portugal. He is vice-president for faculty at DEI. He is coordinator of the Doctoral Program in Information Security at IST. He is a senior researcher at INESC-ID, and member of the Distributed Systems Group (GSD). He is co-chair of the European Blockchain Partnership that is designing the European Blockchain Services Infrastructure (EBSI). He is a member of the Board of Técnico+ and a non-executive member of the Board of Associação .PT. He is Associate Editor for IEEE Transactions on Computers. He has a Ph.D. in Computer Science from the Universidade de Lisboa Faculdade de Ciências. He has been involved in several international and national research projects related to cybersecurity, including the DE4A, BIG, QualiChain, SPARTA, SafeCloud, PCAS, TCLOUDS, ReSIST, CRUTIAL, and MAFTIA European projects. He has more than 200 publications and is Senior Member of the IEEE.

Dr. Thomas Hardjono is currently the CTO of Connection Science and Technical Director of the MIT Trust-Data Consortium, located at MIT in Cambridge, MA. For several years prior to this he was the Executive Director of the MIT Kerberos Consortium, helping make the Kerberos protocol to become the most ubiquitously deployed authentication protocol in world today. Over the past two decades Thomas he has held various industry technical leadership roles, including Distinguished Engineer at Bay Networks, Principal Scientist at VeriSign PKI, and CTO roles at several start-ups. He has been at the forefront of several industry initiatives around identity, data privacy, trust, applied cryptography, and cybersecurity.

5 Harmonia: Securing Cross-Chain Applications Using Zero-Knowledge Proofs

The following chapter corresponds to the following publication [151][1]:

Status: Submitted ⊙

Publication: IEEE Transactions on Dependable and Secure Computing

Submitted: 15 January 2024

Accepted: N/A

Citation: N/A

Research Methodology: Design Science Research Methodology [145]

Evaluation Methodology: Formal Proofs [147], Empirical Evaluation [148], Qualitative Evaluation [149]

Journal Description: IEEE Transactions on Dependable and Secure Computing (TDSC) publishes archival research results focusing on research into foundations, methodologies, and mechanisms that support the achievement-through design, modeling, and evaluation-of systems and networks that are dependable and secure to the desired degree without compromising performance. The focus also includes measurement, modeling, and simulation techniques, and foundations for jointly evaluating, verifying, and designing for performance, security, and dependability constraints.

H-Index: 92 **Coverage**: 2004-present **Quartile**: Q1 **Scimago Journal Rank 2022**: 1.83

---

[1]The full version of this paper can be accessed here `https://www.techrxiv.org/users/679023/articles/695183-harmonia-securing-cross-chain-applications-using-zero-knowledge-proofs`.

Harmonia: Securing Cross-Chain Applications Using Zero-Knowledge Proofs

Rafael Belchior*†, Dimo Dimov‡, Zahary Karadjov‡†, Jonas Pfannschmidt†, André Vasconcelos*, Miguel Correia*, *INESC-ID, Instituto Superior Técnico, Universidade de Lisboa, Portugal
†Blockdaemon Ltd., Dublin, Ireland
‡Metacraft Labs

***Abstract*—The field of blockchain interoperability plays a pivotal role in blockchain adoption. Despite these advances, a notorious problem persists: the high number and success rate of attacks on blockchain bridges. We propose Harmonia, a framework for building robust, secure, efficient, and decentralized cross-chain applications. A main component of Harmonia is DendrETH, a decentralized and efficient zero-knowledge proof-based light client. This light client protocol is implemented as a smart contract, allowing blockchains to read the state of the source blockchain in a trust-minimized way. We show that DendrETH solves critical security flaws in Ethereum's light client protocol.**

We implemented Harmonia in 9K lines of code. Our implementation is compatible with the Ethereum Virtual Machine (EVM) based chains and some non-EVM chains. Our experimental evaluation shows that Harmonia can generate light client updates with reasonable latency and costs (a dozen to a few thousand US dollars per year). We provide an open-source implementation and reproducible environment for researchers and practitioners to replicate our results.

I. Introduction

With the development of increasingly more complex cross-chain logic, the trend is for the major *decentralized applications* (dApps) to go either *cross-chain* or *multi-chain* [1], where various chains coexist, sharing data and digital assets. This allows developers and users to choose the best infrastructure based on trade-offs (e.g., cost, performance, and convenience [2]). This enables workflows supported by different infrastructure components that process *data transfers*, *asset transfers*, and *asset exchanges*, the so-called *interoperability modes* [2]. In practice, realizing these interoperability modes is orchestrating a set of coordinated reads and writes of transactions settled in different blockchains. Asset transfers are particularly vulnerable. In fact, current bridge implementations are insecure to the point of having caused around $3B in losses [3], [4]. The causes include large attack surface, lack of transparency, poor monitoring techniques, lack of incident response plans, reliance on a single point of failure (either individual nodes or committees) [5], cybersecurity attacks that steal private keys [6], bad operational practices [7], attacks on economic incentives [8], and others [3], [4], [9].

A. Problem and Solution Overviews

Academia and industry agree that *interoperability mechanisms* (IMs) relying native verification of transactions across the *source* and *destination (or target) blockchains* are the safest [1], [10], [11]. Bridges allow the transfer of funds by providing facts on the source chain and relaying those to the destination chain. The destination chain independently verifies that the received state is valid and final according to the state transition and consensus rules of the source network. For example, a user can *lock* (or *burn*) some tokens in a source chain, and the bridge can *mint* the corresponding amount of that asset on the destination chain provably (asset transfer). To prove a transaction is valid on an external blockchain, cross-chain applications use light clients [4].

A *light client protocol* (or simply light client) is a protocol that allows proving the inclusion of a transaction in a blockchain without downloading the full blockchain (typically only the block headers [12]. This requires, e.g., in the case of Ethereum, that 1) there is a valid block header that includes the transaction to be proved and 2) a valid *Merkle proof* against the block header root is provided. The sequence of transactions happening in a blockchain creates a "history", that is appended to the blocks that form the blockchain. However, light clients allow multiple valid histories to be validated as long as they respect consensus rules. This creates an attack vector that, combined with the lack of incentives for different parties to behave correctly, can lead to exploiting the light-client system. We call this exploitation *Ghost Checkpoint Attestation Attack*. We explain this attack and a possible solution later in the paper.

In this work, we propose *Harmonia*[1], a framework to build reliable cross-chain applications, realizing data or asset transfers. At its core, Harmonia leverages a light client protocol, assuring the safety of interoperability. Our implementation allows us to prove facts on the direction *Ethereum* $\rightarrow$ *other blockchains*. An additional contribution is *DendrETH*, an improved version of Ethereum's light client protocol [14] that prevents the *Ghost Checkpoint Attestation Attack*. Introduced in the Altair hard fork, the Altair Light Client protocol suffered from critical security vulnerabilities. DendrETH takes as input a *zero-knowledge proof*, specifically, a succinct non-interactive argument of knowledge, or SNARK [15]–[17], which allows verifying on-chain the light client protocol rules in a cost-efficient way.

The key intuition of the paper is that SNARKs can be used to prove that DendrETH rules are correctly executed,

[1]The full version of this paper is available as a preprint at [13]. We will refer the reader to this version when appropriate.

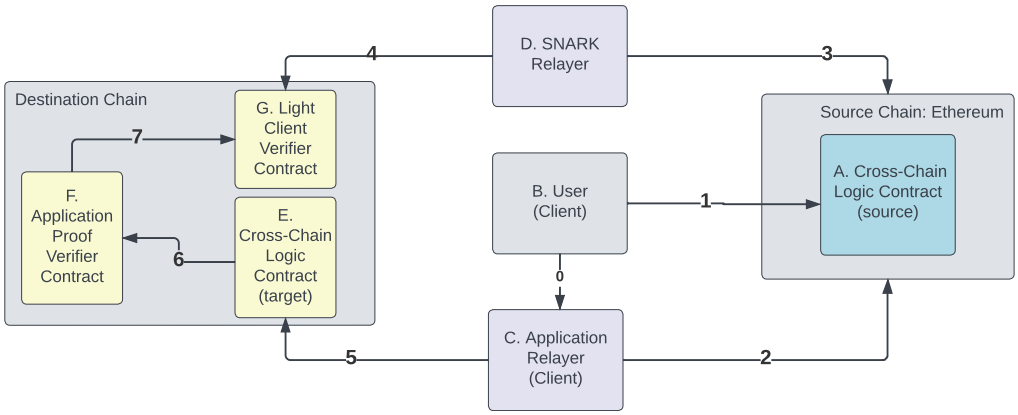


Fig. 1: Harmonia's high-level architecture. The cross-chain contract is represented by . Relayer components . Destination chain components .

which is useful because SNARKs can be verified on-chain. If the verification is successful, the light client is updated. The output of the light client update is a validated *block header*. In this way, cross-chain applications can verify the state of other chains by using different cross-chain proof mechanisms [18] such as Merkle proofs [19] against the updated block header. This allows cross-chain applications to have guarantees on the cross-chain state, e.g., transaction inclusion, and to perform arbitrary *cross-chain logic* (rules that orchestrate cross-chain transactions) [3].

Figure 1 shows the high-level architecture of our framework. The starting point is two cross-chain smart contracts, A and E. Contract E will have `read` and/or `write` dependencies on A, depending on the defined cross-chain logic. In steps 0 and 1, user B interacts with the source chain smart contract A directly or through the *Application Relayer* C (respectively), issuing transactions that change the local state of A; therefore, we deem the relayers blockchain clients. The Application Relayer (C) will see changes to the contract in step 2 and create a Merkle proof that attests to the state changes that step 1 triggered. The Merkle proof and use-case-specific data are sent to E according to the cross-chain logic.

In parallel, the *SNARK Relayer* D gathers the state and data of the light client (step 3) to create a SNARK proof that proves a light client update. The SNARK, along with the necessary input data are sent to the light client verifier contract G, which is a SNARK verifier contract (step 4). If the verification succeeds, the necessary data to validate Merkle proofs will be available on the smart contract to be consumed by applications. The Application Relayer can transact with E following a determined cross-chain logic (step 5). The cross-chain logic contract on the target chain will only execute the logic if the Merkle proof verification on the Application Proof Verifier Contract (contract F) succeeds (step 6). Contract F calls the SNARK verifier to obtain validated state roots to run the Merkle proofs against.

B. Technical Challenges

We aim to provide a light client protocol that addresses the following key challenges ($\mathcal{C}$), some on-chain and the last off-chain:

- On-chain **Safety Failure** ($\mathcal{C}_1$): A light client that has been compromised presents risks to cross-chain use cases. A compromised light client can lead to the validation of invalid block headers, which can then be submitted and executed on a remote deployment.
- On-chain **Safety Failure Propagation** ($\mathcal{C}_2$): If the light client is compromised, an invalid message could be created and executed on a remote deployment with immediate effect, giving no time for stakeholders to react.
- On-chain **Accountability** ($\mathcal{C}_3$): The source chain light client should only be able to sign one light client update (one valid version of the history) at any time. Relayers should detect on-chain misbehavior in the form of multiple signatures, so that it is punished by slashing an amount of collateral.
- Off-chain **Liveness Failure** ($\mathcal{C}_4$): Failure of the source chain for long periods could significantly delay or altogether prevent updates on the target chain. These liveness failures come in the form of occasional intermittent interruptions that lead to delayed or overlooked messages.

C. Contributions

This paper provides a safer way to implement cross-chain applications by minimizing the attack surface, learning from the past while focusing on high performance and cost reduction. In particular:

- We propose Harmonia, a framework to build robust, secure, efficient, and decentralized cross-chain applications based on SNARK-based light clients.
- We present the Altair Light Client (ALC), the canonical light client protocol for the Ethereum 2.0 network. We present an improvement that strengthens the cryptoeconomic security of the ALC, which we call DendrETH. To our knowledge, we are the first to propose improvements that fix critical issues in ALC security.
- We implement and experimentally evaluate Harmonia instantiated with DendrETH in terms of latency and costs. We implement a data transfer use case using Harmonia

and DendrETH. After that, we provide a qualitative evaluation that addresses the problems and technical challenges of this new technology.

D. Outline

This paper is organized as follows: in Section II we introduce the background. Section III presents Harmonia and its light client protocol, DendrETH. Next, Section IV showcases the implementation details of the relayers and verifier smart contracts. In Section V, we present the empirical evaluation. Section VI presents the qualitative evaluation. Section VII presents the related work. Section VIII concludes the paper.

II. Preliminaries

A. Blockchain

We consider a ledger (or blockchain) $\mathcal{L}$ a versioned key-value store. The block number (also called the *height* of the blockchain) allow us to version the key-value store.

Definition 1. *Secure ledger. A ledger is deemed secure if it satisfies three properties [20]:*

- *Consistency*: honest nodes possess a large common prefix, i.e., if n honest parties prune x blocks from their local chains, the probability that the resulting pruned chains will not be mutual prefixes of each other drops exponentially with the number of blocks belonging to the common prefix.
- *Chain quality*: there is an upper bound on the ratio of blocks proposed to the chain of any honest party n contributed by malicious parties.
- *Liveness*: if an honest node receives a valid transaction, it is eventually included in the blockchain by all honest nodes.

For these properties to hold, it is required that the number of malicious nodes f is bounded by the number of honest nodes n - the typical byzantine fault tolerance threshold is $n > 3f+1$.

B. Cryptographic Building Blocks

1) Cryptographic Keys: Accounts of a blockchain are tuples (K_k^{id}, K_P^{id}, id), capable of reading and writing to a ledger via a blockchain client (or node), where [21]:

- K_k^{id} is a private key, used as the signing key.
- K_P^{id} is a public key, used as the verifying key.
- id is the unique identifier of the participant. It is the output of a function over the participant's public key.

2) Hash Functions: A cryptographic hash function is a function that takes an input and returns a fixed-size string of bytes, typically called a hash value or digest. The output is unique, with overwhelming probability to each unique input (one-way function). Given a hash h, it is computationally infeasible to find an input value x such that $hash(x) = h$ (pre-image resistance property) or to find two different inputs that hash to the same output (collision resistance).

3) Signatures: A signature is a mathematical scheme for verifying the authenticity of messages. A node N can create a signature σ via algorithm `SIGN` over a message m using its private key, i.e., $\texttt{SIGN}_{K_k^n}(m) \rightarrow \sigma$. To verify a signature σ, a verifier can run `VERIFY`, taking as input a signature, a message, and the public key of the signer, i.e., $\texttt{VERIFY}_{K_p^n}(m, \sigma) \rightarrow \{0, 1\}$.

Aggregate signatures [22] are digital signatures where a set of nodes uses a mathematical function to combine their signatures $\{\sigma_1, \sigma_2, \ldots, \sigma_N\}$ into a single signature σ_{1-N}. A widely used aggregate signature scheme is BLS (Boneh-Lynn-Shacham) [23], which is also used in Ethereum [24]. Verification of a BLS signature is computationally expensive compared with verification of an ECDSA signature. The advantage of aggregate signatures in the context of this paper is that aggregate signatures provide a succinct way to represent a set of signatures in constant size. Signatures can also be aggregated with SNARKs (which we will discuss later).

C. Merkle Trees and Merkle Proofs

Merkle trees are accumulators implemented as binary trees. The leaves of the tree are data items, and the parent nodes are hashes of that data. The Merkle tree is calculated recursively: each parent element is the hash of its two children until the root is obtained. The tree's root is a succinct vector commitment to a state at a certain time. Merkle trees allow proving data inclusion. In particular, to construct a proof, we include all the nodes along the path needed to allow a recursive hashing up to the tree root (i.e., hashes of sibling nodes that are not in the direct path).

Definition 2. *Merkle proof. A Merkle proof (proof of inclusion) is a path between the tree's root and a leaf node. Given a vector v of i elements, we have three algorithms [25]:*

- $root \leftarrow tree.\texttt{commit}(v)$.
- $(v[i], \pi_i) \leftarrow tree.\texttt{proof}(v, i)$.
- $\{0, 1\} \leftarrow tree.\texttt{verify}(\pi_i, root, v[i])$.

The `commit` algorithm adds an element to the Merkle tree. The `proof` algorithm creates a Merkle proof (a path) for the i^{th} element of v. The `verify` algorithm verifies the proof.

D. Light Client Protocol

Ethereum offers a way to verify whether a transaction is included in a block using only the block header, avoiding downloading the full block.

Definition 3. *Light client. A light client $\mathcal{L}$ is an algorithm that has the following primitives [26]:*

- $\texttt{INIT}(\texttt{BlockHeader}_i) \rightarrow (\mathcal{L}_{\mathcal{S}_i}, \pi)$: The light client takes as input a bootstrap block (either genesis block or a pre-agreed block), and initializes the state with an interactive protocol with a full node and receives a proof π of correct initialization. We assume the bootstrap block is trusted (either by social consensus or cryptographically).
- $\texttt{QUERY}(\mathcal{L}_{\mathcal{S}_i}, data) \rightarrow (resp, \pi)$: a client can read the global state of the light client (and, indirectly, the global state of the underlying blockchain). The light client

returns a response $resp$ and a proof π that authenticates the response or an error $\perp$.
- $\texttt{LCU}(\mathcal{L}_{\mathcal{S}_r}, data) \rightarrow (\mathcal{L}_{\mathcal{S}_{r+1}}, \perp)$: a light client update LCU takes as input the current light client state $\mathcal{S}$ and auxiliary data, and outputs the next light client state or an error.

Note that $resp$ and $data$ vary according to the specific light client protocol.

Definition 4. *Secure and Efficient Light Client. A secure and efficient light client is a client that respects the following properties:*

- Soundness: After INIT, a malicious adversary should not be able to convince a light client $\mathcal{L}$ to accept a forged transaction. On the other hand, the adversary should not be able to convince $\mathcal{L}$ not to accept a valid transaction.
- Liveness: Valid transactions received by an honest full node are eventually included in the chain. A light client protocol eventually includes such transactions in a block header. This means that QUERY returns up-to-date requests up to a liveness parameter λ_l.
- Succinctness: For each state update, the light client protocol takes linear time to synchronize the state. INIT and LCU computation and communication are sublinear to the size of the blocks.

E. Altair Hard Fork and the Ethereum Sync Committees

Altair [24] is the first hard fork of Ethereum. This update brings two major changes to the previous mode of operating: 1) how rewards and penalties are calculated for validators, and 2) support for light clients, which will in turn add an additional reward type. This second change introduces a *sync committee* as the base layer to implement a light client protocol called *Altair Light Client* (ALC). The committee consists of 512 validators, randomly selected every sync committee period lasting 256 epochs ($\approx$ 27 hours). Nodes are given 512 epochs of prior notice before they become sync committee members. A list of the committee members is saved in the state of the Beacon chain. The sync committee signs new block headers, so Ethereum light clients can verify Ethereum's state using a Merkle proof. This process authenticates more recent block signatures using the sync committee's public keys. For more details on the Altair Hard Fork and the sync committee, please consult [13].

F. SNARKs

Broadly speaking, Succinct Non-Interactive Argument of Knowledge (SNARKs) are proofs that a statement is true [16], [27]. In recent years, their popularity increased as several key applications for blockchain were recognized, namely, for increasing scalability, interoperability, and privacy (zero-knowledge proofs [28], [29]). We explore the application of this technology in the context of blockchain interoperability. By proving the validity of a block header via a SNARK, cross-chain logic on a target blockchain (in the form of a decentralized application/smart contract) can now prove the state from a source blockchain using Merkle proofs that are verified against the state root of the validated beacon block header. Refer to [13] for a more detailed explanation of SNARKs in the context of blockchain interoperaiblity.

G. Cross-chain Transactions / Logic / State

We derive the definitions of cross-chain transactions and cross-chain state from recent research [1], [3]. A cross-chain transaction is a set of local transactions (e.g., one transaction in Ethereum and one transaction in Polkadot), governed by cross-chain rules. Cross-chain rules (or cross-chain logic, denoted by ζ) are the dependencies between local transactions. For example, a transaction with payload $p = (lock, amount, destinationaddress, proof)$ in Ethereum should have a transaction in Polkadot with payload $p' = (mint, amount, destinationaddress, proof)$. Cross-chain rules may be enforced by off-chain relayers and smart contracts. We call the state that a set of cross-chain transactions generate the *cross-chain state*. A cross-chain state is a key-value store that spawns across the blockchains that process cross-chain transactions. This state stores information useful to execute cross-chain logic. In our bridge example, the state would be a ledger storing $(p, p', ...)$ for bookkeeping. This work showcases the relevance of these concepts for our use case and how our system plays a role in realizing them.

III. The Harmonia Framework

In this section, we explain how the Harmonia framework works and how one can implement interoperable dApps on top of it. Harmonia is a framework to build reliable cross-chain applications, realizing the three interoperability modes [2]. At its core, it uses a light client protocol for the safety of interoperability. Our instantiation of Harmonia uses DendrETH as an on-chain SNARK-based light client that allows proving facts on the direction Ethereum $\rightarrow$ other chains, although Harmonia can support other combinations.

A. System Model and Components

Harmonia establishes a unidirectional communication channel between blockchains, which typically have different consensus mechanisms and trust models. A transaction has achieved finality if, for a block at index i, the difference between the head of the chain, block j, and block i is higher than a liveness parameter λ_l, i.e., $j - i > \lambda_l$. The liveness parameter of the source chain is particularly relevant to the destination blockchain because cross-chain logic depends on transactions from the source chain. In practice, the destination chain needs to wait at least λ_l.

Harmonia includes several agents. The simplified architecture is in Figure 1, and a more detailed version in Figure 2:

- **Source and Destination chains** : we assume chains support smart contracts.
- **Light Client Verifier Contract**: a smart contract deployed on the destination chain that verifies SNARKS created off-chain. It exposes a list of validated execution roots, optimistic roots, and finalized header roots that cross-chain applications can consume.
- **Application Verifier Contract**: a smart contract deployed on the destination chain that takes as input Merkle

proofs and verifies them against the latest validated block root made available by the Light Client Verifier Contract.
- **Cross-chain Logic Contracts**: a pair of smart contracts. The first one is deployed on the source chain and holds business logic belonging to that chain. The second contract is deployed on the second chain and enforces cross-chain rules based on the state of the first contract, having a dependency on the Application Light Client Verifier Contract for validating updates.
- **SNARK and Application Relayers**: relayers read the global state and require a full node (or blockchain client access). For a finalized ledger, every client will read the same state. Two types of relayers are considered. The SNARK relayer creates SNARKS from on-chain information on the source chain and relays it to the verifier contract on the destination chain. The Application Relayer creates collects relevant data to perform interoperation, accompanied by a proof (e.g., Merkle proof) - and submits it to the cross-chain logic contracts.

Agents send arbitrary, authenticated messages between one another, e.g., transaction payloads, ACKS, and encoded API calls. The system considers two *actors*: the end user who interacts with a frontend connected to an interoperable system and an adversary that tries to steal user funds by attacking the interoperable solution.

B. Threat and Network Model

The adversary $\mathcal{A}$ is a probabilistic polynomial time algorithm that can perform various actions, namely corrupting a subset of the validators before the protocol execution commences. Adversarial validators on the source blockchain are denoted by f and the total number of nodes by n (honest nodes are assumed to be $n-f$ and typically $n > 3f+1$, depending on the specific security threshold necessary for a chain to be considered secure). We rely of the hardness of well-known problems where SNARK technology relies [13], including but not limited to the discrete logarithm problem. We assume the cryptographic primitives of the source and target blockchains are secure (hash functions, signing algorithms, communication channels, public-key infrastructure). We assume the source and target blockchains are secure.

The network model considered is partially synchronous (for every message sent, an adversary can arbitrarily delay it up to a certain threshold). This implies no long network partitions, allowing the light client to have live updates. The threat model of Harmonia is tied to the security and crypto-economic models of the light client protocol and, if applicable, its light sync committee. Finally, we note that the security of our model is tied to the security of the proof system we use, which may rely on one or more of the referred cryptographic assumptions.

C. System Goals

Under the system, threat, and network models defined above, we propose a set of properties that the collective system actors known as the Harmonia system need to achieve (given the presented set of assumptions):

- On-chain **Safety**, $\mathcal{G}_1$: no invalid light client updates are validated.
- On-chain **Decentralization**, $\mathcal{G}_2$: anyone should be able to run an application relayer and a SNARK relayer. While bringing the decentralization advantages, it also introduces exploitation vectors, such as the extraction of value (MEV) [30].
- On-chain **Finality**, $\mathcal{G}_3$: once a message has been delivered and validated, it cannot be reverted.
- On-chain Off-chain **Liveness**, $\mathcal{G}_4$: valid updates are eventually accepted on-chain.
- On-chain Off-chain **Extensibility**, $\mathcal{G}_5$: refers to supporting execution environments other than the EVM (e.g., WASM). The solution should allow cross-chain use cases to expand seamlessly to other chains and reduce the risks of integrating new deployments.
- On-chain Off-chain **Flexibility**, $\mathcal{G}_6$: it is possible to integrate new blockchains, light client protocols, and SNARK schemes. In the context of SNARK-based bridges, this property is needed because new protocols will continue to emerge, while existing protocols might become defunct or undergo significant changes (e.g., forks).
- Off-chain **Safety (Off-Chain)**, $\mathcal{G}_7$: fake proofs of the construction of a block header on a source blockchain are computationally infeasible (this is given by the specific SNARK protocol with which we instantiate the light client).
- Off-chain **Censorship-Resistance**, $\mathcal{G}_8$: no entity should be able to prevent a valid light client from updating against our system.

Furthermore, we present two important performance metrics:

- On-chain **Cost**: the system shall minimize transaction fees and hardware expenses as much as possible, as they are a significant constraining factor.
- On-chain Off-chain **Latency**: the system shall be able to provably verify Ethereum's state in a reasonable amount of time.

D. Architecture

Figure 2 presents the architecture of Harmonia. The system is based on four building blocks:

1) Circuit generation block (steps 1-6): circuit generation components (steps to create and compile the circuit implementing the light client protocol). The circuit building block deals with the definition of the light client protocol as a set of circuits. A trusted setup ceremony is run, and the respective proving keys and circuits are generated. This is an expensive process that happens only once per circuit version.
2) SNARK generation and relaying (steps 7-11): SNARK generation components (fetching and giving inputs to the circuit, retrieving SNARK). In these steps, the SNARK relayer fetches the necessary information from the blockchain and inputs it into the circuit. This yields a valid SNARK for a specific input.

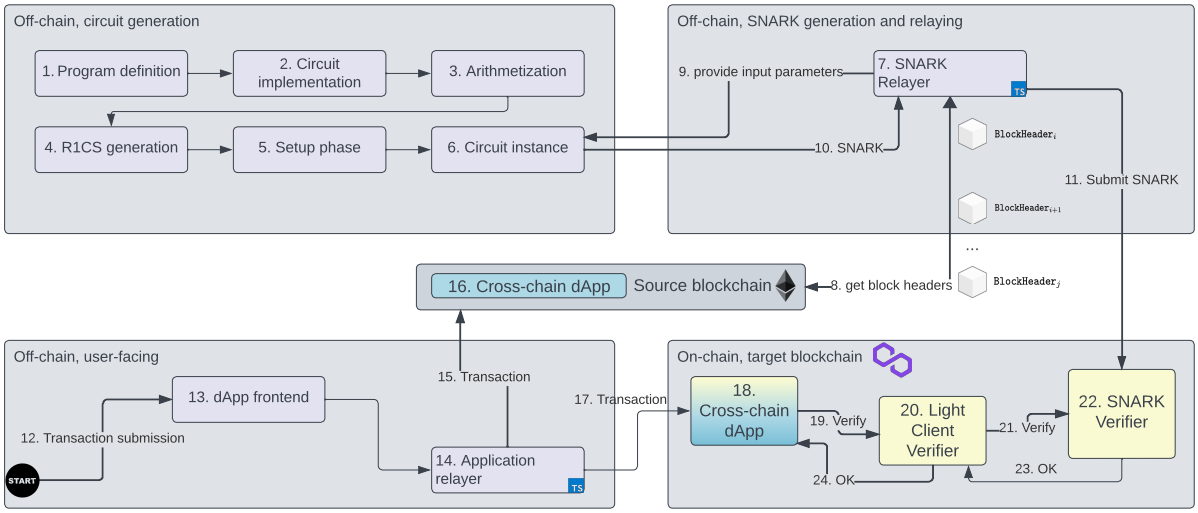


Fig. 2: Architecture of Harmonia. Off-chain components are represented by . Destination chain components . The cross-chain contract is represented by .

3) User-facing application (steps 12-16): user-facing side of the cross-chain dApp (logic on the source chain). Here, we define the logic of the source chain application. A user can interact with it through a user interface connected to the application relayer. The latter transacts against the source chain and fetches a Merkle proof for that transaction.
4) Destination chain cross-chain dApp (steps 17-23): cross-chain logic on the destination chain (logic on the target chain, based on proven facts on the source chain, via a SNARK). In the last steps, we prove facts on the source blockchain (e.g., a transaction on 16 was inserted in a block). The application relayer provides a Merkle proof and data in 17 (via a blockchain transaction). Step 20 validates that Merkle proof against a validated block header (namely, the execution state root that the SNARK verifier provides). After verification, the logic on 18 can be executed, with validated input from 17.

Contracts 16, 18,20, and 22 are deployed once per version (possible to deploy behind a smart contract proxy) by the set of administrators running those contracts (e.g., via a multisig address). The proposed architecture provides a reliable means to authenticate data in Ethereum. The attributes in Ethereum beacon chain block headers reference a "BeaconState" root hash, which points to a recent execution layer block header. The execution layer block header references the root hash of the execution layer state. Thus, if a chain of proofs is also supplied and verified against the light client contract state, it can be used to prove in the targeted blockchain the occurrence of any event in the Ethereum world starting from a Beacon block header, allowing users to build cross-chain applications using Ethereum's state.

E. Altair Light Client 1.0

This section presents the original Ethereum's light client proposal, published in the Altair fork, in a high-level way. A formal specification is present in the full version [13]. The Altair Light Client (ALC) relies on a sync committee mechanism, called Altair syncing protocol [31], deployed in the Altair hard fork [32]. Altair enables light clients to efficiently and securely construct the chain of beacon block headers. We discuss the low overhead for light clients, making the beacon chain light client-friendly for resource-constrained environments. After that, we present its limitations.

The algorithm works as follows. Figure 3 presents the process of a light client tracking the chain of recent beacon block headers. The process starts at the current block header. In this figure, a light client has a block header at slot s, with a well-defined state root (1) in period i. Let us consider current period i, current slot s, a sync committee at period i formed by 512 nodes $(N_1^i, N_2^i, ...N_{512}^i) = N_{[1:512]}^i = \mathcal{C}^i$. Each node has a private key $K_K^{N_i}$ and public key $K_P^{N_i}$ Let the block header at period i be $\texttt{BlockHeader}_i$.

The light client stores the current header so that the light client can authenticate information such as transactions and balances against the header (via Merkle proofs). The light client also stores the current and the next sync committees, obtainable via the state root (2). Using a Merkle proof, it verifies the next sync commmittee in the slot s post-state. This sync committee will sign block headers during period $P + 1$. After this, the protocol obtains the public keys of the light client sync committee (3). The keys of the nodes that participated in the aggregate signature of the sync committee are defined in a participation bit map (4). More concretely, this step creates a combined public key $K_P^{N_1,...,N_{512}}$ from the individual public keys of the sync committee subset that participated in the aggregate signature $\sigma_{N_1,...,N_{512}}$. An aggregated signature σ_{1-N} is calculated from the combined public key. That signature (pub) is compared with the aggregate signature downloaded from the block or the peer-to-peer network (sig). Lastly, the signature is verified against the combined public key and the newer block header, $\texttt{VERIFY}_{pub}(root, sig)$, in

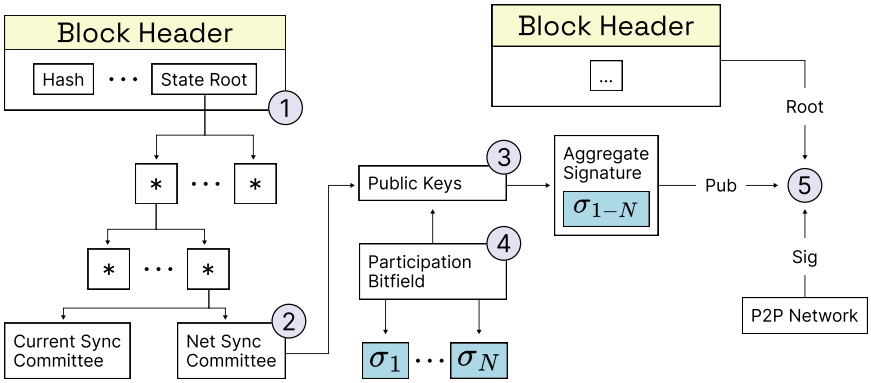


Fig. 3: Light Client Update algorithm (LCU) in Altair [31]

(5).

If the verification passes, the new block header has been successfully authenticated. The next sync committee will become the current sync committee of the next valid period (and, consequently, block) so that light client updates for the following block can be done in a chain. Taking advantage of this property, one can verify that blocks have been validated retroactively. The light client executes the light client update algorithm LCU to authenticate a block header in the following period.

Nevertheless, critical security issues have been identified with Altair [33], [34]. We will review two critical issues with the specification and provide the first solution to this problem. First, the sync committee is not held accountable for misbehaving since slashing is not enforced when sync committee members sign semantically invalid block headers. This incentivizes signers to blindly sign every block header (invalid or not) to reap the rewards, resulting in the loss of safety of the light client. Secondly, a light client sync committee might receive valid attestations but choose not to sign them, breaking the light client's liveness. Therefore, Altair 1.0 does not assure the generated SNARKs are semantically valid, allowing replay attacks and potential misuse.

F. DendrETH: Strengthening the Security of ALC

DendrETH is a smart contract implementation of ALC, which allows one to prove Ethereum facts on target blockchains. To motivate DendrETH, we formalize a new attack on ALC's sync committee, which explores its current lack of accountability, solving technical challenge $\mathcal{C}_3$.This attack is executed by bribing honest nodes and forcing them to generate valid proof for an invalid checkpoint (set of blocks up to a certain epoch). Hence, we call this attack *Ghost Checkpoint Attestation Attack*.

1) Threat Model: As the size of the sync committee is fixed for every 256 epochs, the time window for an attack to corrupt the committee is 512 epochs (54.6 hours). The economic security threshold k will be bounded, considering that each validator needs to stake 32 Eth. Since an adversary needs to control two-thirds of the sync committee to sign block headers, the number of nodes necessary to control would be 341 (collateral slashed would be approximately 20M USD). The following inequality can express this: $k > 512 \times 32 \times 1850$ USD/ETH $\times \frac{2}{3}$. At the current market rate, $k > 10,923$ ETH ($20,207,000$ USD). On the other hand, an adversary would have a time window of $54.6 + 27.3 \approx 82$ hours (nodes in the sync committee are notified 512 epochs in advance + 256 epochs where they operate) to corrupt two-thirds of this committee, where the capital necessary for it could be substantially lower. When the sum protected by the bridge is inferior to k, bribing the committee is not economically secure (assuming a slashing of 100%). However, the number of secured assets (ERC-20, NFTs) is often hundreds of millions of dollars, as we have seen by recently hacked bridges [3]. Note that slashing is proportional to the amount of stake performing the attack. Therefore, we provide an upper bound of the capital needed for an attack. A more lenient estimate shows that if the entire sync committee could be fully slashed down to 0 ETH, this would still cap the security level to the whole stake of the sync committee, or 16384 ETH $\approx$ 32 million USD. We put forward a slashing proposal and a future research direction that together increase the crypto-economic security of the system.

2) Ghost Checkpoint Attestation Attack: The idea behind this attack is that entities in the sync committee can create multiple valid histories of the Ethereum blockchain, sign them, and propagate them without being penalized. When the sync committee creates a valid block and submits it to the network of the targeted domain, there are no safety violations. However, a deceptive supermajority within the sync committee can mislead applications that depend on Ethereum's light client synchronization protocol into accepting a non-canonical (but valid) finalized header (this is, a proof pointing to a syntactically valid block header that is not included in the canonical chain). For instance, this message could be leveraged to compromise a bridge contract based on the light client sync protocol, reducing its trustworthiness. We describe a specific attack on SNARK-based bridges depending on Altair 1.0:

1) a corrupted subset of the sync committee creates an

invalid block.
2) it creates a SNARK for that invalid block.
3) the committee will attempt to delete evidence of their misbehavior and the invalid block is deleted to prevent accountability.
4) the (valid) SNARK attesting an invalid block is sent as proof to the destination chain.
5) arbitrary cross-chain logic is executed based on a non-canonical state.

Since the destination chain has no observability on the source chain by design (otherwise, it would not be a light client), there is no way for it to know the canonical block for this attack. The malicious sync committee can also perform an attack on the liveness of the protocol by synchronizing themselves (à la Flashbots [35]) not to attest block headers, effectively conducting a Denial of Service attack - this would have a cost of ≈ 51.2 ETH $\approx 99,300$ USD per sync period.

3) Sync Committee Slashing: The core of DendrETH's proposal is to make the sync committee accountable by creating and sharing evidence that can lead to slashing. The idea is if the sync committee signs and submits an alternative finalized history, the entire sync committee gets slashed. Only malicious behavior should be slashable - "but a validator tricked into syncing to an incorrect checkpoint should not be slashable even though it is participating on a non-canonical chain. Note that a slashing must be verifiable even without access to history, e.g., by a checkpoint synced beacon node" [36]. Due to space restrictions, DendrETH's algorithms `IdenfitySlashing` and `EnforceSlashing` are formally defined in [13].

On a high level, identify slashing checks that there exist two pieces of evidence with contradictory results. It checks the evidence points to the past, it checks that the periods of the evidence are sequential, and assesses the existence of conflicting block headers, which are submitted to the same slot. The enforce slashing algorithm identifies the parties to be slashed, validates the submitted evidence, and finally enforces the slashing. In conclusion, having two pieces of evidence provides a way to demonstrate and compare the conflicting actions of a validator. If a validator provides two contradicting pieces of information in a context where such a contradiction should not be possible, then the two pieces of evidence serve as proof of their wrongdoing, making them eligible for slashing.

G. Building Cross-Chain Applications

To simplify the use of Harmonia in different cross-chain applications, we adopt a modular design where we separate the verification logic (Beacon chain light client verifier) from the application logic (e.g., state sync, bridging). The application logic can consume a standard interface from the on-chain verifier that integrates the consumption of verifier block roots and Merkle trees onto its logic flow. The high-level idea is to use a SNARK-based light client as the source of truth of the source blockchain, providing regular, valid block header updates to an interoperability application. The actors involved in the process are those defined in the system model (Section III). In our implementation, we use DendrETH as the light client protocol and the Ethereum Beacon chain as the source blockchain. In greater detail, we define two sub-protocols that govern a cross-chain application using Harmonia: the SNARK Relayer protocol and the Application Relayer Protocol. The former is presented in the Protocol 1 listing, while the latter is formally defined here [13].

Protocol 1: SNARK Relayer Protocol

Input: Relayer private key K_k^r
Input: Full node list N, update period Δ_{proof}
Input: Light client verifier contract address $addr_c$
Input: Light client circuit $\mathcal{C}$ and Prover algorithm P
Data: Location of the latest block header, add_b
Data: Access to the source and target blockchains $\mathcal{B}_s$, $\mathcal{B}_t$
Result: Sends a SNARK to $\mathcal{B}_t$ that validates validity of a block header from $\mathcal{B}_s$

1 **Procedure** `SendSNARK`
2 $\texttt{LCU}_{data}[n] \leftarrow \emptyset$ ▷ `array with` n `values`
3 **for** *every* Δ_{proof} **do**
4 **for** *each* n *in* N **do**
5 $\texttt{BlockHeader} = n.read_{\mathcal{B}_s}(add_b)$
6 $\texttt{LCU}_{data}[index] =$ `GetLCUDataFromBlock(BlockHeader)`
7 **assert** $\forall i, j : 1 \leq i \leq n, \texttt{LCU}_{data}[i] = \texttt{LCU}_{data}[j]$ ▷ `responses from different nodes are consistent`
8 $(x_{\mathcal{C}}, w_{\mathcal{C}})$ = `GenerateProofInput(`$\texttt{LCU}_{data}$`)` ▷ `Generate input and witness`
9 $\pi_{SNARK} = P(\mathcal{C}, x_{\mathcal{C}}, w_{\mathcal{C}})$
$\sigma \leftarrow \texttt{SIGN}_{K_k^r}(\pi_{SNARK})$
10 $\texttt{store}_{\mathcal{B}_t}(add_c, (\pi_{SNARK}, \sigma))$

The SNARK relayer protocol runs every time interval defined (e.g., every 32 slots/blocks). To maximize resilience and stemming from the increasing usage of node-as-a-service companies such as Blockdaemon (maximizing efficiency but with security risks), the SNARK relayer should query several sources (line 4). In line 5, the relayer queries a full node and gets the latest block. The latest block contains the information necessary to build $\texttt{LCU}_{data}$ (line 6). In line 7, we assert that the responses from different node providers are the same. Otherwise, we abort the protocol.

In line 8, the SNARK relayer generates the necessary input and witness to be used as input to our light client circuit. In line 9, we generate the SNARK using the Prover algorithm P. It takes as input the light client circuit we developed and inputs a SNARK. The relayer signs the generated data in line 10 for accountability. Finally, in line 11, the SNARK is submitted to the Light Client Verifier contract on the target chain[2]. We note that any node can perform the role of the relayer, making the SNARK submission process permissionless and decentralized. An incentive mechanism can be built to incentivize the good

[2]Technically, we send the SNARK and the data we want the smart contract to record. We explain this process in detail in the implementation section.

functioning of the network. Details on the on-chain light client verifier can be found in the implementation section.

We now explain the Application Relayer Protocol. First, the application relayer takes as input the cross-chain state (the next step of the cross-chain logic). Then, the relayer creates a transaction that executes the next step on the source blockchain. After that, it submits the transaction on the source blockchain. Next, generates a Merkle proof proving the inclusion of tx_s in the source blockchain. The relayer then updates the cross-chain state. After that, the relayer creates a transaction according to the cross-chain logic aimed at the target blockchain. Then, it transacts against the target blockchain. Finally, the relayer updates the cross-chain state.

H. State Migration with Harmonia

This section presents an implementation of an application using Harmonia and DendrETH. Our proof of concept is based on SmartSync [37]. SmartSync implements state migration across EVM-based chains. The migration of state can be performed by interacting with pairs of smart contracts, one in each chain, and retrieving Merkle proofs for the updates on both contracts.

The starting point is a pair of (equal) smart contracts deployed on two EVM chains. A user or relayer will issue a transaction to one of the contracts and then reissue that same transaction to the other contract. The migration is verified by a third contract upon providing a pair of Merkle proofs (one for each transaction on each contract) and auxiliary data. The rationale is the following: consider that the storage hash is the Merkle root of the storage tree, which encodes a key-value store for each contract. If the contract updates on both ends yield the same storage hash and both Merkle proofs are valid, then both contracts have the exact same state at the block to which the Merkle proof refers. However, verifying Merkle proofs relies on a centralized, trusted relayer to provide the source of truth (block header roots). We remove this dependency by integrating our fork of SmartSync with Harmonia, showcasing a use case for interoperability other than asset transfers. The idea is based on four smart contracts. A cross-chain logic contract on the source chain, a cross-chain logic contract on the target chain, an application proof verifier contract, and a light client verifier contract.

Figure 4 shows the sequence diagram of the use case. In step 1, an application relayer or an end user interacts with the source contract, modifying its storage (update value v). A Merkle proof π for the storage modification is constructed (step 2). After that, the same transaction is made against the logic contract on the target chain (step 4), and a proof π' is generated (step 6). Next, the relayer submits a verification request that validates the migration. It takes as input the two generated Merkle proofs and requires the storage hashes of the contracts to be the same. The application proof verifier contract will interact with the Light client Verifier contract in step 8 to get the latest validated execution state root r, which was validated using a SNARK π_{SNARK}. After that, it will validate both Merkle proofs using the validated block header root. Upon verification, the state migration is complete.

IV. Implementation

In this section, we provide implementation details on the Harmonia components.

A. Relayers

The SNARK relayer efficiently generates proofs and publishes updates for all blockchains supported by Harmonia. We provide up-to-date Docker images and Nix environment configurations to simplify the process of running a relay. Our relayer has comprehensive setup instructions and is open source[3]. The initial slot for the relayer to start creating SNARKs is customizable by setting the finalization time interval `SLOTS_JUMP` (by default 64, i.e., 12.8 minutes). This interval will set a trade-off between liveness and cost: the shorter the interval, the more live the updates are (up to one update per epoch). However, a higher frequency of updates requires a higher workload for creating SNARKs and submitting them on-chain, raising operational costs. The relayer is implemented in Typescript (using NodeJS as the runtime environment) and is composed of multiple workers.

The SNARK relayer runs SNARKJs [38], where an initialization procedure occurs. The procedure includes performing a "Powers of Tau" ceremony compiling the circuit, conducting the setup with Groth16 (with the BLS12-381 curve), giving as input the witness received by the proof generation worker, and creating the proof. Upon completion of proof generation, the generated proof is saved in Redis. Multiple instances subscribing to this notification attempt to publish the proof on-chain to the verifier contract[4]). Due to the standardized light client verifier interface, the relay architecture allows for extensibility and includes different chains and transaction types. The three workers have been implemented in $\approx 2,400$ lines of code (LOC) of Typescript.

The application relayer is an off-chain server that has a dependency on our fork of SmartSync[5]. It receives calls from an end-user via a user interface and executes cross-chain logic: issue transactions to change the state on the source blockchain (e.g., storage of variable "A" of the cross-chain logic contract), fetches recent blocks from the source and destination chains, adding blocks to the relayer contract, creates Merkle proofs from on-chain transactions, obtains latest proofs and input from DendrETH, and conducts the state migration the overall flow is depicted in Figure 1. It contains an OpenAPI specification that generates the SDKs in different programming languages so that different stacks can use the application relayer. The relayer is parameterized with a wallet (so it can transact against the source and destination chains), node RPC endpoints for Ethereum Goerli and Polygon Mumbai and the addresses of the cross-chain logic contracts, the relay contract, and DendrETH.The relayer has been implemented in $\approx 2,150$ LOC of Typescript. Our fork of SmartSync has $\approx 1,500$ LOC of Typescript.

[3]see the relayer code, https://tinyurl.com/2hnauda6.
[4]an example of a successful update, https://tinyurl.com/4saa55fx.
[5]implementation available here, https://github.com/RafaelAPB/smart-sync.

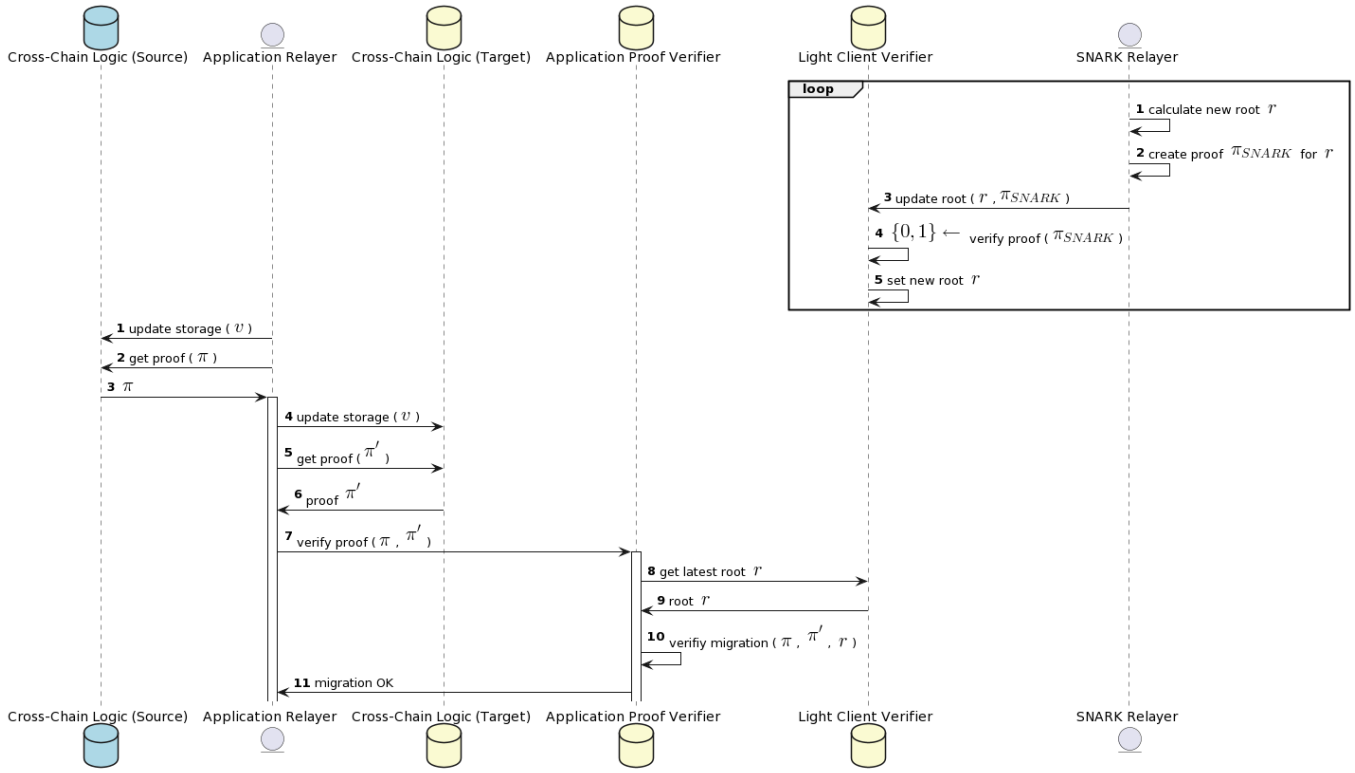


Fig. 4: Sequence flow of our fork of SmartSync integrated with DendrETH, using Harmnoia. The components of SmartSync are the cross-chain logic (source and target). The blue component is a contract deployed on the source chain. Yellow components are contracts deployed on the target chain. Purple components are off-chain components.

B. Cross-Chain Logic

The cross-chain logic contracts for our PoC are implemented using Solidity: a SimpleStorage contract, a Relay Contract, a Proxy Contract, and utility contracts. The SimpleStorage contract, along with the Proxy, implements the cross-chain logic. The Relay contract verifies Merkle proofs done against the Light Client Verifier contract. We utilized Foundry, a smart contract framework [39] to implement the integration with Harmonia instantiated with DendrETH (≈ 1,000 LOC of Solidity) and test our contracts (≈ 300 LOC of Solidity). We have made our open-source implementation available[6].

C. Light Client Verifier & Application Proof Verifier Contracts

In this section, we elaborate on the implementation details of the on-chain contracts for validating SNARKs and implementing the cross-chain logic.

1) Verifiers For EVM-based chains: The light client verifier contract has been implemented in Solidity for EVM-based chains (Ethereum Classic, Binance Smart Chain, Polygon, Avalanche, Celo, Theta, Hedera, Fantom). It is divided into the Beacon light client contract and the light client update verifier (SNARK verifier). The light client verifier exposes multiple public functions that consume the outcome of the verifying process, e.g., optimistic headers and execution state roots.

The light client verifier calls an auxiliary contract (SNARK Verifier) generated by SNARKJs, allowing us to verify SNARKS on-chain. The SNARK verifier uses a pairing library that provides functions for performing operations on elliptic curves. It provides functions for addition and scalar multiplication of points, the negation of points, and the bilinear map and pairing operations. The main function of our light client verifier is called `verifyProof`. It takes as the input a SNARK π_{SNARK} (parameters a,b,c that represent points in an elliptic curve) and the *public inputs*, that is, the data to be made available to be consumed by other applications (namely execution state root, optimistic header root, the attested header root, and the attested header slot). Listing **??** shows the code for the verifier contract. The contract computes a commitment to the input of the SNARK on-chain and passes that along to the SNARK verifier and SNARK proof.

2) Verifiers for non-EVM-based chains: For blockchains with a WebAssembly runtime, we developed a direct implementation of the light client syncing protocol based on the highly efficient BLS, SSZ, and Light client syncing libraries developed by Supranational and the Nimbus team. We do this by compiling Nim code that implements the light client syncing protocol to C, and then C code to WASM.

Furthermore, we provide two codebases we believe to be

[6] at Github, https://github.com/RafaelAPB/data-transfer-dendreth.

useful for researchers and practitioners: 1) a direct implementation of the light client protocol, which we adapted to run as a CosmWasm smart contract, and 2) a SNARK verifier in WASM which we also adapted to run as a CosmWasm smart contract.

D. Circuits

We implemented the circuits for DendrETH (without slashing), in 3,600 LOC. We used the Circom programming language [40], snarkJS, and the Groth16 proof system to generate our SNARKs. We chose Circom because it is the most production-ready language that compiles circuit descriptions into arithmetic circuits. The main circuit, `light_client.circom`[7], uses 12 additional auxiliary circuits (e.g., to compute the domain, verify a Merkle proof, calculate a supermajority).

V. Evaluation

This section evaluates the latency, throughput, and cost of the SNARK Relayer, the Application Relayer, the Light Client Verifier, and the Application Proof Verifier. Let us recall our performance goals: the system tries to minimize costs as much as possible, and it should be able to verify Ethereum's state in a reasonable amount of time. The exchange rates and gas prices are as of 14 July 2023.

A. Setup

We have launched several nodes to support connectivity to the blockchains we connect to. For the consensus layer of the Goerli/Prater network, we launched a Nimbus Client [41] paired with Geth v1.11.5 [42] and downloaded the Prater blockchain (around 60 GB). For Polygon Mumbai, we launched a node v0.3.7 [43]. The size of the blockchain is 260 GB. Both nodes are located in Amsterdam, Europe. When connection to other blockchains was needed, we leveraged the infrastructure provided by Blockdaemon to connect to Ethereum and Polygon. For the SNARK relayer, we deployed a server with 384 GB of RAM, 32-core, 1TB NVMe hard drive, configured with 500GB of swap space and an i9-13900 CPU. The Application Relayer was deployed on a 16 GB RAM, 1 TB SSD, and a ten-core 3.2GHz laptop.

B. Circuits

Our main circuit (`light_client.circom`) took 6 hours, 27 minutes, and 47 seconds to compile on the specified hardware, and the trusted setup phase took 26 hours, outputting a verification key of 55.6GB. It has 410 template instances, $\approx$ 90 million non-linear constraints, $\approx$ 5 million linear constraints, 0 public inputs, 2 public outputs, 20,961 private inputs, 0 private outputs, $\approx$ 93 million wires, and $\approx$ 470 million labels, being one of the most complex Circom circuits developed to date. We developed a test suite using snarkit2 [44] to evaluate the correctness of the sub-circuits that `light_client.circom` uses. For example, our tests for the `pow` circuit have five cases from which we illustrate two: 1) on input `base: 10, power: 3`, the output should be `1000`; 2) on input `base: 2, power: 10`, the output should be `1024`.

[7]available in the DendrETH repository, https://tinyurl.com/mtzutesj.

C. Latency

There are two latencies that we are interested in measuring. First, the latency of generating a SNARK proof (Δ_{proof}) to be submitted and validated on-chain. An attentive reader might notice that such a computationally intensive task can upperbound the total latency of our system, but we show this is not the case - instead, finality is the main responsible for latency. Secondly, we want to measure the latency of executing the cross-chain logic of our use case (represented by Δ_ζ). These two latencies add up to the total latency, or end-to-end latency (Δ_{total}) for a fact to be verified on a target chain, using Harmonia - this shows how applicable our proposal is in the real world. This includes measuring both the SNARK and the Application relayers (Δ_{snark} and Δ_{app}, respectively), issuing transactions against the source chain (Ethereum) and destination chain (Polygon) ($\Delta_{\texttt{store}}^{Ethereum}$ and $\Delta_{\texttt{store}}^{Polygon}$, respectively, note that this includes accommodating finalization times, which range from a few minutes to around twenty minutes). Note that generating a proof is included in the SNARK relayer operations, i.e., $\Delta_{snark} \geq \Delta_{proof}$. It is important to note that some operations depend on others, while some can be parallelized. In particular, the cross-chain logic rules ζ define that transactions on Ethereum happen before the ones in Polygon. Note that in our use case, we have one transaction in the source chain and two transactions on the destination chain. Generalizing for arbitrary cross-chain logic, the specific deltas depend on the numbers of transactions on the source and destination chains ($|tx|_s$ and $|tx|_d$, respectively). The end-to-end latency is given by $\Delta_{total} = \Delta_{snark} + \Delta_\zeta$. The latency Δ_{slack} sums the duration of one slot (12 seconds), which is the lag the relayer has with regard to the current block. This is because the current block (head block) aggregates the claims of the sync committee regarding the previous block: in the best-case scenario, we can start creating a SNARK of the penultimate block. Expanding the expression we obtain:

$$\begin{aligned}\Delta_{total} = \Delta_{snark} + \Delta_{slack} + \Delta_{tx_s} \\ + (|tx|_s \times \Delta_{\texttt{store}}^{source}) \\ + \Delta_{app} + (|tx|_d \times \Delta_{\texttt{store}}^{target}) \end{aligned} \quad (1)$$

Let us calculate Δ_{snark}. Consider a chain of block headers on the source chain $\texttt{HeaderChain}_i^j$ from block i to j. Let t_i be the time block i was finalized. Let t_j be the time block j was finalized. Let the finalization latency of the cross-chain logic transaction issued by the Application Relayer on the source chain (defined as tx_s) be denoted by Δtx_s. Let i be the block at which the SNARK relayer starts building a proof for block i. Let $j \pm \epsilon$ be the block where the SNARK relayer has finished building a SNARK for block i (ϵ accounts for small delays on the SNARK relayer software, for example, doing API calls and calling internal functions). Let Δ_{proof} be the time between i and j, i.e., the time the SNARK relayer needs to build a SNARK.

Figure 5 presents our latency model. In step ①, the user or application relayer on behalf of the user submits a transaction t_s to the source chain, following rule ζ_2. We have two possibilities: we either issue t_s before (②) or after i (②'). If issued before i, it means that the SNARK relayer will pick such transaction at block i for when it starts building a proof. Otherwise, it will be picked in the following block that will be verified (not every block is verified). The SNARK relayer picks block i, in step ③ to construct a SNARK. At block $j = i + \Delta_{proof} \pm \epsilon$ the tx_s is finalized, in ④. After the SNARK for block i is created, it can be verified in the destination chain, via t_j, in step ⑤, and eventually included in a block, ⑥. After that, the Application Relayer executes the rest of the cross-chain logic by broadcasting tx_d (executing rule ζ_3, in step ⑦). Formally:

$$\Delta_{snark} = \begin{cases} \Delta_{tx_s} + \Delta_{proof} & t_s < t_i \\ (\Delta_{proof} - \Delta_{tx_s}) + \Delta_{proof} & t_s \geq t_i \end{cases}$$

Typically, blocks are finalized in two to three epochs or approximately 12.8 to 19.2 minutes [45]. On average, a user transacts in the middle of an epoch. Thus, the last epoch only needs 66% of attestations, and thus, a transaction included there is finalized faster - averaging 14 minutes (16 slots from the first epoch + a full epoch, or 32 slots, + 66% of the last epoch, or 22 slots). Therefore, for use cases where an application on a third-party blockchain requires strong consistency on the Ethereum state, a safe buffer of around 10.8 to 14 minutes is expected.

We observed that creating a SNARK proof on our hardware takes 4 minutes and 25 seconds. That is the minimum theoretical latency, as one can generate a proof for the next block transition while generating the previous block proof. However, we defined the `SLOT_JUMPS` parameter on the worker to two epochs, or 64 slots (i.e., we batch transactions spanning two epochs). Since 64 slots $\times$ 12 seconds = 12.8 minutes, the latency introduced by the batching process supersedes the proving time. Therefore, $\Delta_{proof} = 12.8$ minutes. In practice, we could further reduce latency to 6.4 minutes (32 slot jumps) and even lower to around 4.5 minutes. Indeed, one can reduce latency by skipping fewer slots, at the peril of collecting transactions that will not be finalized and thus be included only in the next proof. However, the jumps should take longer than the proof generation time. It is worth noting that there is a trade-off between having smaller jumps and the time window for finalization: if a user submits a transaction in the second slot of the epoch (admitting the relayer will start creating a SNARK at the beginning of each epoch), the user will have to wait a full 31 slots until the prover starts with an input including that transaction. Conversely, the sooner a transaction is issued relative to being picked by the relayer, the more the user will have to wait for transaction finalization. Therefore, in minutes, $12.8 + \Delta_{tx_s}$ (minimum finalization time + a delta) $< \Delta_{snark} \lessapprox 25.6 + \epsilon$ (maximum finalization time + delays on relayer).

Empirically, this translates into a delay of around 50 blocks relative to the source chain: we have verified a transaction included at block i at block $i+50$. The target chain has a 150-180 block delay (will vary according to the specific destination chain): we can prove a fact at block header i in the source chain when around 150 blocks have been finalized in the target chain.

The Application Relayer constructs performs two reads from the blockchain state of the source and destination chains, creates two Merkle proofs, signs and broadcasts one transaction on the source chain, and signs and broadcasts three transactions on the destination chain, among other low-resource tasks. Since all of those happen concurrently with the SNARK Relayer, in a few seconds, Δ_{app} is statistically negligible. However, after the SNARK relayer has made the newest block execution state root available, the Application Relayer still needs to finalize the migration process according to our use case definition. The rest of the end-to-end process takes the time of one transaction confirmation. Since the SNARK latency will be the highest and parallel with the Application relayer and transaction finalization on different chains, we can approximate the overall latency: $\Delta_{total} \approx \Delta_{snark} + (|tx|_s \times \Delta_{\texttt{store}}^{Ethereum} + \Delta_{app} + |tx|_d \times \Delta_{\texttt{store}}^{Polygon}) \lessapprox \Delta_{snark} + (1 \times \Delta_{\texttt{store}}^{Ethereum} + \Delta_{app} + 3 \times \Delta_{\texttt{store}}^{Polygon}) \lessapprox 25.6$.

D. Transaction Fees and Costs

We analyze the transaction costs of the Light Client Verifier contract (Beacon Light Client), the Application Proof Verifier, and the cross-chain logic contracts regarding gas. We consider the costs of deploying the Light Client Verifier (1,399,127 gas) and performing one light client update (279,963 gas). We calculate the yearly costs in USD for issuing three light client updates per hour (1 every 20 minutes). This gives an overview of the transaction costs to operate the light client in several EVM-based chains (see Table I):

Regarding our cross-chain logic contracts, our integration adds 7,225 gas (around 40 cents), including one extra call to DendrETH to retrieve the execution state root. The values to obtain a storage variable are around 2,400 gas, and to set the variable, they are 5,400 gas. Deploying the Relay, Proxy, and simple storage contracts cost 1.6M, 3M, and 140K gas, respectively. Performing the migration process costs around 950K gas, costing around 0.2 MATIC (17 cents) in Polygon.

In the context of blockchains with a WASM runtime (e.g., Polkadot, Cosmos, Elrond, NEAR, EOS, Fantom), the verification costs of a SNARK is around 250K gas and over four times that for the direct verification (i.e., verifying consensus rules in the smart contract). Table II shows the gas costs for deployment and updates of our different implementations of the SNARK verifier.

E. Reproducibility

We provide tools for researchers to set up our project and reproduce our empirical results easily. First, we make available our codebase and results under a permissive open source license[8]. We provide tests and Docker containers following recommended engineering practices [46], [47]. Tests are available for the on-chain smart contracts, direct implementation,

[8]available at https://github.com/metacraft-labs/DendrETH

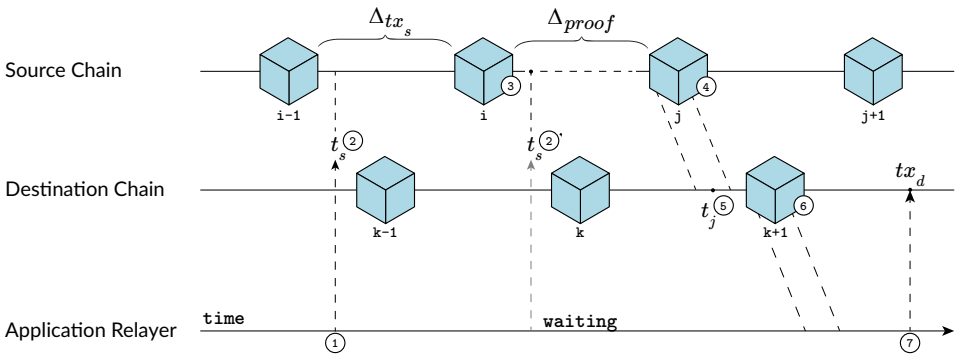


Fig. 5: Overall latency of a system built with Harmonia. It includes the latency to perform cross-chain transactions Δ_{tx_s} and Δ_{tx_d}, and the latency to generate a proof, Δ_{proof}.

	Price/Gas	Rate USD/Native Token	Light Client Verifier Deployment (USD)	Light Client Verifier Update (USD)	Cost/Year (USD)
Ethereum	29 Gwei	2010	81.61	16.33	429,152
Polygon	200 Gwei	0.85	0.24	0.05	1,314
Avalanche	26 nAVAX	14.79	0.54	0.11	2,891
EVMOS	30 nEVMOS	0.09	0.004	0.001	26.28
Optimism	0.001 Gwei	1.37	≈ 0	≈ 0	0.01
Arbitrum	0.1 Gwei	1.22	0.0002	≈ 0	0.9

TABLE I: Light Client Verifier deployment and Update costs

	Deployment	Initialize	Update
NIM-WASM Light Client	1,308,702	2,991,395	11,706,455
SNARK Verifier using nim-bncurve	1,302,849	447,436	1,812,337
SNARK Verifier using constantine	1,378,889	391,408	871,846

TABLE II: Deployment, Initialization, and Update Gas Costs

and Circom circuits. To facilitate our testing infrastructure, we maintain an archive of light client updates for each sync committee period since Altair, as produced by a fully-synced Nimbus node, available for the Ethereum mainnet and Prater [9]. For Prater/Goerli, the updates start on checkpoint 5601 823. Pre-generated proofs have been available since that checkpoint. Furthermore, to ease setting up the environment, we leverage Nix, a package management and system configuration tool that helps us showcase a reproducible, declarative, and reliable system. The deployment addresses of our contracts are available in our project's README.md file.

VI. Discussion and Qualitative Assessment

In this section, we present the discussion and qualitative evaluation of our solution.

A. Safety

Altair 1.0 is vulnerable to bribing. Although individual action by a small set of validators would be insufficient for a large-scale attack, the industry has seen centralized platforms for influencing validator behavior. An example is the Flashbots platform, which provides auction-based coordination to extract value from block reorganization [35]. Therefore, cooperation to exploit Ethereum's light client protocol could emerge sooner or later as an obvious first venue for conducting cross-chain attacks [3].

To harden the system's safety against these attacks, DendrETH is a necessary improvement to the state of the art, currently being standardized and implemented [48]. However, this security model might not work for smaller chains with much smaller total value locked (meaning bribes and colluding prices would be cheaper). Given that a high volume of traffic of a popular interoperability solution is non-EVM (order of tens of billions USD), this indicates the need to study the crypto-economic attacks possible to do for the light clients of those infrastructures. An example is the elevated traffic between layer two solutions, namely Arbitrum and Polygon, where the users want to transact between chains and not wait the large waiting queue periods to withdraw their funds from the L2 (typically using an optimistically verified method). Although this is not easily fixable, one could instead closely monitor misbehaving, forks, and slashing on these smaller chains using cross-chain models, and use circuit breakers for cross-chain logic if some suspicious behavior is detected.

Still, regarding safety, let us discuss long-range attacks on Ethereum light clients. In Proof of Stake Ethereum, nodes must verify block headers, account states, and balances throughout

[9] see light client updates, https://tinyurl.com/yampz8re.

the blockchain history or consider the risk of long-range attacks [26], [49]. The attack involves an adversary taking over the blockchain by creating an alternative (forked) chain starting from a point deep in the history of the legitimate chain. In PoS systems, the ability to create blocks is proportional to the amount of stake (tokens) one holds. However, in the past, it is possible that a large amount of stake was held in addresses that now hold little to no stake. If the private keys of these old, 'empty' addresses are obtained (perhaps they were discarded, sold, or are no longer secured because the stake was moved), an attacker could theoretically create a new chain starting from when those addresses had a significant stake. Given enough time, this alternative chain could grow longer than the original chain. In blockchain protocols, the longest chain is often considered the 'correct' one, so this could enable the attacker to overwrite the blockchain's history.

A long-range attack could trick light clients as they inherently trust the validity of the blockchain headers they receive and do not fully validate the entire blockchain. If an attacker successfully executes a long-range attack and creates an alternate, longer blockchain, they could send the light clients the headers from this fraudulent chain, and verify false facts on a destination chain, conducting different types of attacks on cross-chain logic, because the real blockchain history is indistinguishable from the forged one. The counter measure employed by Ethereum is to require all clients always to start syncing from a trusted recent checkpoint, which guarantees that the maximum number of exited validators will not be sufficient for carrying out an attack. This notion is known as weak subjectivity [50]. We now perform a detailed analysis of weak subjectivity from the perspective of light clients [51] to calculate the risk of long-range attacks in the full version of the paper [13].

B. Liveness

The liveness of our system is tied to the liveness of the light client sync committee and the liveness of off-chain parties. For the liveness of off-chain parties, techniques like crash-recovery for blockchain clients can be deployed [52], as well as having multiple instances deployed, mitigating the probability of unavailability.

We now explore on-chain liveness: let us consider the case where the light client receives a valid update containing a $finality\_header$ with at least two-thirds of the sync committee participating. However, there might be cases where part of the sync committee is unavailable (crash or attack). If the light client sees (no valid updates via the method for a one-sync committee period), it simply accepts the speculative header with the most signatures as finalized. This allows the light client to be live even during periods of extended non-finality, though at the cost of network latency of a period. This downtime period is not sufficiently long for long-range attacks.

Cross-chain protocols are conventionally structured to manage such situations efficiently by initiating retries following the resumption of services. These anomalies are deemed harmless up to a liveness parameter. Effectively, upon crashing, the relayer keeps track of the last update and submits all late updates to the chain when it is back online or when the chain resumes its operations, similarly to the crash fault recovery protocol of blockchain gateways [52], [53]. Effectively, this has happened a few times on our testnet deployments, as we can see in the subsequent transactions after this one[10]. The relayer provided all light client update transactions missing quickly after the incident.

C. Censorship Resistance

Attackers could attempt to block valid light client updates passing to the deployed chain, violating censorship resistance, by performing a denial of service on the off-chain relayers. A group of validators could attempt a denial of service, which would depend on the decentralization and security of the attacked network. Although a decentralized network of relayers would alleviate the likelihood of traditional denial-of-service attacks, an attacker could explore an alternative route. In particular, a denial of service of the whole blockchain could try to be performed. An attacker could buy the entire block space for the duration of an attack, which is around 5 ETH per block in July 2023[11]. To control the inclusion of blocks for a day, an attacker must spend at least 580 ETH, approximately 1 million USD. This would effectively censor 120 light client updates. Since the block space market is very dynamic and unpredictable [54], it is extremely unlikely that the adversary can select the gas fees to be higher than in every other transaction and the attack is economically viable. The worst-case scenario brings difficulties for an attacker. For example, the peak of gas price was ≈ 710 Gwei in July 2020, making a single Eth transfer cost 0.015 Eth. Since the gas limit in a block is 30 million [55], buying a single block could cost up to 21.3 Eth (42K USD). Since 7,200 blocks are created daily, this could account for $\approx 304,704,000$ USD to censor the entire network daily.

Note that this value could be considerably lower for weaker security blockchains. In Polygon, the daily block rewards are around 40K MATIC, around $26,800 USD. Thus, running a DoS for the light client where the source chain would be Polygon would probably cost substantially less (low fees are a feature of Layer 2 technologies). However, the specifics of an attack would have to be further investigated, as market forces could influence the gas fees consumed by a denial-of-service attack in unexpected ways [56].

VII. Related Work

The interoperability design space is increasingly diverse [1], [2], [57]–[60] and full of risks [4]. To better navigate the various classification frameworks available, one can aggregate the interoperability solutions by *interoperability mode*: data transfers, asset transfers, or asset exchanges. Harmonia allows realizing all modes, while it is instantiated with a data transfer use case. There has been extensive work on light client protocols for different types of blockchains. In [61], the

[10] see PolygonScan, Mumbai network, https://tinyurl.com/d9mtusny
[11] see Etherscan, https://etherscan.io/chart/blocks

authors propose smart-contract-based light clients. In [62], the authors propose fraud proofs to enhance light clients. Other optimizations have been proposed, e.g., using probabilistic block sampling [63]. For a comprehensive theoretical overview of this area, refer to [26], [64].

Westerkamp and Eberhardt [65] designed a SNARK-based verifier for Bitcoin, that yields validated block headers that can be consumed in Ethereum. However, this work focused solely on Bitcoin, and relied on a network of known peers. Recently, some projects have appeared to validate the consensus of more complex blockchains such as Ethereum. The following validate Ethereum's light client protocol[25], [66], [67], but these currently only work for EVM-based chains (except for zkBridge). While some of these have lower latency on the proof generation time, there is no practical benefit in reducing the latency beyond an Ethereum transaction's finalization time for a client application to consume a finalized root. We have not found code for relayers or on-chain verifier contracts for these projects. Furthermore, we have not found details on cross-chain applications built on top of these solutions, making it difficult for us to assess and compare these works systematically. Some circuit implementations prove other light client sync protocols (e.g., Bitcoin [68], ZCash [69]).

On the other hand, DendrETH can be classified as a natively verified system, where the "destination chain independently verifies that the received state is valid and final according to the source network's state transition and consensus rule." [1]. Compared to other approaches, this provides a higher level of security, since the security model is based on the sound cryptography of SNARK technology and not on external systems. Our system provides a new security model based on the soundness of the underlying SNARK protocol and crypto-economics, with minimal assumptions (liveness of the relayer network).

VIII. Conclusion

In this paper, we Harmonia, a framework to robustly build decentralized applications using zero-knowledge proofs. Our framework defines the existence of a set of relayers, cross-chain logic contracts, and proof verifiers. Harmonia proves block header generation correctness on any blockchain with a light client protocol. This way, state (e.g., transaction inclusion, storage variables) on a source blockchain can be proven correct and consumed by third-party chains, allowing the realization of arbitrary cross-chain logic use cases.

As Harmonia's core, we propose DendrETH, a decentralized, secure, and efficient light client that implements Ethereum's light client sync protocol. To implement DendrETH, we leverage Circom, a circuit meta-programming language that allows us to define the light sync protocol as an arithmetic circuit. From this circuit, we create zero-knowledge proofs that attest the correct execution of the light client protocol. During the development of our solution, we have proposed several extensions to the light client sync protocol that 1) guarantee accountability and slashing for misbehaving parties and 2) increase the crypto-economic security of the light client.

Harmonia was thoroughly evaluated. We show that the circuit size of DendrETH translates into an acceptable throughput and latency for applications to consume: a proof can be generated in around 4 minutes. Performing updates to systems built with Harmonia costs in the order of a few thousand US dollars per year in transaction fees, a small cost to potentially host an indefinite number of cross-chain applications consuming Ethereum's state. We show that the overall latency is upper-bounded at around 25 minutes for a full end-to-end state sync. To further validate our approach, we have developed a cross-chain application that implements data transfer, using Harmonia instantiated with DendrETH. In conclusion, Harmonia allows to design of secure, scalable, and simple cross-chain applications, hardened by the cryptography underlying SNARK technology.

Acknowledgments

We thank the open-source community for contributing to the software we used in our system. We warmly thank our colleagues in the IETF's working group Secure Asset Transfer Protocol (SATP) for fruitful discussions on interoperability security. We thank André Augusto, Guillaume Lethuillier, Freddy Zwanzger, Jan Süßenguth, Qi Feng, Peter Robinson, and Chris Spannos for providing comments that improved our paper. We thank Metacraft Labs' team, that helped implement DendrETH. We thank Dom Martinez and Iulia Mihaiu for their help in the graphical design of this paper. We thank the Nimbus team for contributing to the slashing proposal.This work was supported in part by the European Funds, "Recovery and Resilience Plan - Component 5: Agendas Mobilizadoras para a Inovação Empresarial," through project nr.51 "BLOCKCHAIN.PT - Agenda Descentralizar Portugal com Blockchain," included in the NextGenerationEU funding program, in part by the national funds through Fundação para a Ciência e a Tecnologia (FCT) under Grant UIDB/50021/2020 (INESC-ID) and Grant 2020.06837.BD, and in part by the European Commission under Grant 952226 (BIG).

References

[1] R. Belchior, J. Süßenguth, Q. Feng, T. Hardjono, A. Vasconcelos, and M. Correia, "A Brief History of Blockchain Interoperability," 6 2023. [Online]. Available: https://www.techrxiv.org/doi/full/10.36227/techrxiv.23418677.v3

[2] R. Belchior, L. Riley, T. Hardjono, A. Vasconcelos, and M. Correia, "Do you need a distributed ledger technology interoperability solution?" *Distributed Ledger Technologies: Research and Practice*, vol. 2, no. 1, pp. 1–37, 2023.

[3] R. Belchior, P. Somogyvari, J. Pfannschmid, A. Vasconcelos, and M. Correia, "Hephaestus: Modelling, analysis, and performance evaluation of cross-chain transactions," *TechRxiv preprint*, 2023, available at: https://tinyurl.com/muk64ve7.

[4] A. Augusto, R. Belchior, M. Correia, A. Vasconcelos, L. Zhang, and T. Hardjono, *SoK: Security and Privacy of Blockchain Interoperability*, 2023. [Online]. Available: https://www.techrxiv.org/users/687326/articles/691934-sok-security-and-privacy-of-blockchain-interoperability(more

[5] R. Browne, "$100 million worth of crypto has been stolen in another major hack," https://www.cnbc.com/2022/06/24/hackers-steal-100-million-in-crypto-from-harmonys-horizon-bridge.html, 2022, accessed: 21-June-2023.

[6] Rekt, "Rekt - Ronin Network," 2022. [Online]. Available: https://www.rekt.news/

[7] P. KidBold, "The Wormhole Bridge Attack Explained," Feb. 2022. [Online]. Available: https://kaicho.substack.com/p/the-wormhole-bridge-attack-explained

[8] L. Gudgeon, P. Moreno-Sanchez, S. Roos, P. McCorry, and A. Gervais, "Sok: Layer-two blockchain protocols," in *Financial Cryptography and Data Security: 24th International Conference, FC 2020, Kota Kinabalu, Malaysia, February 10–14, 2020 Revised Selected Papers 24*. Springer, 2020, pp. 201–226.

[9] S.-S. Lee, A. Murashkin, M. Derka, and J. Gorzny, "Sok: Not quite water under the bridge: Review of cross-chain bridge hacks," *arXiv preprint arXiv:2210.16209*, 2022.

[10] LiFi. (2023, Oct) LiFi's Declassified Bridge Survival Guide. Online Blog. Available online: https://lifi.substack.com/p/lifis-declassified-bridge-survival, last accessed on 8 October 2023. [Online]. Available: https://lifi.substack.com/p/lifis-declassified-bridge-survival

[11] A. Zamyatin, M. Al-Bassam, D. Zindros, E. Kokoris-Kogias, P. Moreno-Sanchez, A. Kiayias, and W. J. Knottenbelt, "Sok: Communication across distributed ledgers," in *Financial Cryptography and Data Security: 25th International Conference, FC 2021, Virtual Event, March 1–5, 2021, Revised Selected Papers, Part II 25*. Springer, 2021, pp. 3–36.

[12] R. Belchior, "Phd thesis proposal - blockchain interoperability," Instituto Superior Técnico, Departamento de Engenharia Informática, Tech. Rep., Sep 2021, available online: https://web.ist.utl.pt/~ist180970/papers/phd_cat_rafael_belchior.pdf. [Online]. Available: https://web.ist.utl.pt/~ist180970/papers/phd_cat_rafael_belchior.pdf

[13] R. Belchior, D. Dimov, Z. Karadjov, J. Pfannschmidt, A. Vasconcelos, and M. Correia, *Harmonia: Securing Cross-Chain Applications Using Zero-Knowledge Proofs*, Dec. 2023, citation Key: belchiorHarmoniaSecuringCrossChain2023. [Online]. Available: https://www.techrxiv.org/users/679023/articles/695183-harmonia-securing-cross-chain-applications-using-zero-knowledge-proofs?commit=8565e0802b73dd884f9a20de9edef48f4b1aa0fb

[14] E. Foundation, "The merge," https://ethereum.org/en/roadmap/merge/, 2023, accessed: 21-June-2023.

[15] E. Burger, B. Chiang, S. Chokshi, E. Lazzarin, J. Thaler, and A. Yahya, "The zero knowledge canon," https://a16zcrypto.com/posts/article/zero-knowledge-canon/, 2023, accessed: 6-July-2023.

[16] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof-systems," in *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*, 2019, pp. 203–225.

[17] N. Bitansky, R. Canetti, A. Chiesa, and E. Tromer, "From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again," in *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, 2012, pp. 326–349.

[18] R. Belchior, L. Torres, J. Pfannschmid, A. Vasconcelos, and M. Correia, "Can we share the same perspective? blockchain interoperability with views," Oct 2022. [Online]. Available: https://www.techrxiv.org/articles/preprint/Is_My_Perspective_Better_Than_Yours_Blockchain_Interoperability_with_Views/20025857/3

[19] H. Liu, X. Luo, H. Liu, and X. Xia, "Merkle tree: A fundamental component of blockchains," in *2021 International Conference on Electronic Information Engineering and Computer Science (EIECS)*. IEEE, 2021, pp. 556–561.

[20] J. Garay, A. Kiayias, and N. Leonardos, "The bitcoin backbone protocol: Analysis and applications," in *Annual international conference on the theory and applications of cryptographic techniques*. Springer, 2015, pp. 281–310.

[21] R. Belchior, L. Torres, J. Pfannschmid, A. Vasconcelos, and M. Correia, "Can We Share the Same Perspective? Blockchain Interoperability with Views," 10 2022. [Online]. Available: https://www.techrxiv.org/articles/preprint/Is_My_Perspective_Better_Than_Yours_Blockchain_Interoperability_with_Views/20025857

[22] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, "Aggregate and verifiably encrypted signatures from bilinear maps," in *Advances in Cryptology—EUROCRYPT 2003: International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4–8, 2003 Proceedings 22*. Springer, 2003, pp. 416–432.

[23] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2001, pp. 514–532.

[24] Eth2Book, "Altair: Part 2 - building blocks - signatures," https://eth2book.info/altair/part2/building_blocks/signatures/, 2023, accessed: 27-June-2023.

[25] T. Xie, J. Zhang, Z. Cheng, F. Zhang, Y. Zhang, Y. Jia, D. Boneh, and D. Song, "zkbridge: Trustless cross-chain bridges made practical," *arXiv preprint arXiv:2210.00264*, 2022.

[26] P. Chatzigiannis, F. Baldimtsi, and K. Chalkias, "Sok: Blockchain light clients," in *Financial Cryptography and Data Security: 26th International Conference, FC 2022, Grenada, May 2–6, 2022, Revised Selected Papers*. Springer, 2022, pp. 615–641.

[27] M. Bellare and O. Goldreich, "On defining proofs of knowledge," in *Annual International Cryptology Conference*. Springer, 1992, pp. 390–420.

[28] B. Bunz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell, "Bulletproofs: Short Proofs for Confidential Transactions and More," *Proceedings - IEEE Symposium on Security and Privacy*, vol. 2018-May, pp. 315–334, 2018, iSBN: 9781538643525.

[29] X. Sun, F. R. Yu, P. Zhang, Z. Sun, W. Xie, and X. Peng, "A survey on zero-knowledge proof in blockchain," *IEEE network*, vol. 35, no. 4, pp. 198–205, 2021.

[30] K. Qin, L. Zhou, and A. Gervais, "Quantifying blockchain extractable value: How dark is the forest?" in *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022, pp. 198–214.

[31] Ethereum, "Altair sync protocol - annotated spec," https://github.com/ethereum/annotated-spec/blob/master/altair/sync-protocol.md, 2023, accessed: 30-June-2023.

[32] D. H. Staff, "What does altair bring to ethereum 2.0?" https://dailyhodl.com/2021/11/04/what-does-altair-bring-to-ethereum-2-0/, 2021, accessed: 27-June-2023.

[33] t3rn, "Exploring Eth's Altair Light Client Protocol," 2023, available online: https://www.t3rn.io/blog/exploring-eths-altair-light-client-protocol-t3rns-vision, last accessed on 2023-09-18. [Online]. Available: https://www.t3rn.io/blog/exploring-eths-altair-light-client-protocol-t3rns-vision

[34] J. Prestwich, "Altair," https://prestwich.substack.com/p/altair, 2023, accessed: 29-July-2023.

[35] Flashbots, "Flashbots documentation," https://docs.flashbots.net/, 2023, accessed: 29-July-2023.

[36] Nimbus team. (2023) Sync committee slashing · Issue #3321 · ethereum/consensus-specs. Available online: https://github.com/ethereum/consensus-specs/issues/3321, last accessed on [your-access-date]. [Online]. Available: https://github.com/ethereum/consensus-specs/issues/3321

[37] M. Westerkamp and A. Küpper, "Smartsync: Cross-blockchain smart contract interaction and synchronization," in *2022 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, May 2022, p. 1–9.

[38] iden3, "snarkjs: a pure javascript zksnark library," https://github.com/iden3/snarkjs, 2023, accessed: 27-June-2023.

[39] Contributors. (2023) Foundry: A toolkit for ethereum application development. Accessed: 13 September 2023. [Online]. Available: https://github.com/foundry-rs

[40] (2023, Sep) Circom documentation. Accessed on 15 September 2023. [Online]. Available: https://docs.circom.io/

[41] "The nimbus guide," Sep 2023, accessed on 15 September 2023. [Online]. Available: https://nimbus.guide/

[42] "Documentation for the go-ethereum client," Sep 2023, accessed on 15 September 2023. [Online]. Available: https://geth.ethereum.org/docs

[43] (2023, Mar) Bor v0.3.7 - mainnet and mumbai release. Polygon Community Forum. Accessed on 15 September 2023. [Online]. Available: https://forum.polygon.technology/t/bor-v0-3-7-mainnet-and-mumbai-release/11559

[44] FluiDex. (2023, Apr) snarkit2: A toolkit to compile and debug circom circuit. GitHub repository. Accessed on 13 September 2023. [Online]. Available: https://github.com/fluidex/snarkit2

[45] E. Foundation, "Explanation of single slot finality," https://ethereum.org, 2023, accessed: 21-June-2023.

[46] J. R. F. Cacho and K. Taghva, "The state of reproducible research in computer science," in *17th International Conference on Information Technology–New Generations (ITNG 2020)*. Springer, 2020, pp. 519–524.

[47] J. Cito and H. C. Gall, "Using docker containers to improve reproducibility in software engineering research," in *Proceedings of the 38th international conference on software engineering companion*, 2016, pp. 906–907.

[48] M. Labs, "Issue 3321 from ethereum proof-of-stake consensus specifications," https://github.com/ethereum/consensus-specs/issues/3321, 2023, accessed: 29-July-2023.

[49] V. Buterin, "Proof of stake: How i learned to love weak subjectivity," https://blog.ethereum.org/2014/11/25/proof-stake-learned-love-weak-subjectivity/, 2014, accessed: 27-June-2023.

[50] A. Asgaonkar. (2020) Weak subjectivity in eth2.0. Available online: https://notes.ethereum.org/@adiasg/weak-subjectvity-eth2, last accessed on [your-access-date]. [Online]. Available: https://notes.ethereum.org/@adiasg/weak-subjectvity-eth2

[51] M. Labs. (2023) Safety considerations for long-range ethereum light client syncing. Available online: https://github.com/metacraft-labs/DendrETH/tree/main/docs/long-range-syncing#sync_committee_slashing-proposal, last accessed on 2023-09-18. [Online]. Available: https://github.com/metacraft-labs/DendrETH/tree/main/docs/long-range-syncing#sync_committee_slashing-proposal

[52] R. Belchior, M. Correia, A. Augusto, and T. Hardjono, "Satp gateway crash recovery mechanism," IETF, Technical Report, 2023, accessed: 29-July-2023. [Online]. Available: https://datatracker.ietf.org/doc/draft-belchior-satp-gateway-recovery/

[53] M. Hargreaves, T. Hardjono, and R. Belchior, "Secure Asset Transfer Protocol (SATP)," Internet Engineering Task Force, Internet Draft draft-hargreaves-sat-core-02. [Online]. Available: https://datatracker.ietf.org/doc/draft-hargreaves-sat-core

[54] P. Daian, S. Goldfeder, T. Kell, Y. Li, X. Zhao, I. Bentov, L. Breidenbach, and A. Juels, "Flash boys 2.0: Frontrunning, transaction reordering, and consensus instability in decentralized exchanges," *arXiv preprint arXiv:1904.05234*, 2019.

[55] Etherscan. Ethereum blocks on etherscan. Accessed on 5th October 2023. [Online]. Available: https://etherscan.io/blocks

[56] M. Vasek, M. Thornton, and T. Moore, "Empirical analysis of denial-of-service attacks in the bitcoin ecosystem," in *Financial Cryptography and Data Security*, R. Böhme, M. Brenner, T. Moore, and M. Smith, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 57–71.

[57] R. Belchior, A. Vasconcelos, S. Guerreiro, and M. Correia, "A survey on blockchain interoperability: Past, present, and future trends," *ACM Computing Surveys (CSUR)*, vol. 54, no. 8, pp. 1–41, 2021.

[58] Bhuptani, Arjun, "The Interoperability Trilemma: AKA Why Bridging Ethereum Domains is So Damn Difficult," 2021, available online: https://blog.connext.network/the-interoperability-trilemma-657c2cf69f17, last accessed on 2023-05-21. [Online]. Available: https://blog.connext.network/the-interoperability-trilemma-657c2cf69f17

[59] G. Caldarelli and J. Ellul, "The blockchain oracle problem in decentralized finance—A multivocal approach," *Applied Sciences (Switzerland)*, vol. 11, no. 16, 2021.

[60] Zarick, Ryan and Pellegrino, Bryan and Banister, Caleb, "LayerZero: Trustless Omnichain Interoperability Protocol," 2021, available online: https://layerzero.network/pdf/LayerZero_Whitepaper_Release.pdf, last accessed on 2023-05-22. [Online]. Available: https://layerzero.network/pdf/LayerZero_Whitepaper_Release.pdf

[61] D. Gruber, W. Li, and G. Karame, "Unifying lightweight blockchain client implementations," in *Proc. NDSS workshop decentralized IoT security stand*, 2018, pp. 1–7.

[62] M. Al-Bassam, A. Sonnino, and V. Buterin, "Fraud proofs: Maximising light client security and scaling blockchains with dishonest majorities," *arXiv preprint arXiv:1809.09044*, vol. 160, 2018.

[63] B. Bünz, L. Kiffer, L. Luu, and M. Zamani, "Flyclient: Super-light clients for cryptocurrencies," in *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2020, pp. 928–946.

[64] S. Paavolainen and C. Carr, "Security properties of light clients on the ethereum blockchain," *IEEE Access*, vol. 8, pp. 124 339–124 358, 2020.

[65] M. Westerkamp and J. Eberhardt, "zkrelay: Facilitating sidechains using zksnark-based chain-relays," in *2020 IEEE European Symposium on Security and Privacy Workshops (EuroSPW)*, Sep. 2020, p. 378–386, citation Key: westerkamp_zkrelay_2020.

[66] S. Labs, "Telepathy documentation," https://docs.telepathy.xyz/, 2023, accessed: 29-July-2023.

[67] (2023) Herodotus documentation. Available online: https://docs.herodotus.dev/herodotus-docs/, last accessed on 2023-09-18. [Online]. Available: https://docs.herodotus.dev/herodotus-docs/

[68] M. Westerkamp and J. Eberhardt, "zkrelay: Facilitating sidechains using zksnark-based chain-relays," in *2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, 2020, pp. 378–386.

[69] bunny sleepy, "private-bridge-zcash-ethereum," Jun 2023. [Online]. Available: https://github.com/bunny-sleepy/private-bridge-zcash-ethereum

Rafael Belchior completed his MSc in 2019 at Instituto Superior Técnico. He is a researcher at INESC-ID (Distributed Systems Group) and a Ph.D. student at Técnico Lisboa.

Dimo Dimov TBD

Zahary Karadjov TBD

Jonas Pfannschmidt has more than 15 years of professional experience in software engineering with a focus on Blockchain, Cloud, and Financial Services. In his current roles as Principal Blockchain Engineer, R&D lead, and Director of Blockdaemon Ltd. he builds institutional-grade Blockchain infrastructure to stake, deploy, and scale leading Blockchain networks.

André Vasconcelos is Assistant Professor (Professor Auxiliar) in the Department of Computer Science and Engineering, Instituto Superior Tecnico, Lisbon University, and researcher in Information and Decision Support Systems Lab at INESC-ID, in Enterprise Architecture domains, namely representation and modeling of Architectures of Information Systems, and Evaluation of Information Systems Architectures.

Miguel Correia is a Full Professor at Instituto Superior Tecnico (IST), Universidade de Lisboa, in Lisboa, Portugal. He is vice president for faculty at DEI. He is the coordinator of the Doctoral Program in Information Security at IST. He is a senior researcher at INESC-ID, and a member of the Distributed Systems Group (GSD). He is co-chair of the European Blockchain Partnership that is designing the European Blockchain Services Infrastructure (EBSI).

6 Hephaestus: Modelling, Analysis, and Performance Evaluation of Cross-Chain Transactions

The following chapter corresponds to the following publication [104]:

Status: Published ✓

Publication: IEEE Transactions on Reliability

Submitted: 30 August 2022

Accepted: 20 November 2023

Citation: R. Belchior, P. Somogyvari, J. Pfannschmidt, A. Vasconcelos and M. Correia, "Hephaestus: Modeling, Analysis, and Performance Evaluation of Cross-Chain Transactions," in IEEE Transactions on Reliability, 2023. doi: 10.1109/TR.2023.3336246.

Research Methodology: Design Science Research Methodology [145]

Evaluation Methodology: Empirical Evaluation [148], Qualitative Evaluation [149]

Journal Description: IEEE Transactions on Reliability is a refereed journal for the reliability and allied disciplines including, but not limited to, maintainability, [...], reliability for systems of systems, network availability, mission success, warranty, safety, and various measures of effectiveness. Topics eligible for publication range from hardware to software, from materials to systems, [...], from individual items to networks, from techniques for making things better to ways of predicting and measuring behavior in the field.

H-Index: 114 **Coverage**: 1963-present **Quartile**: Q1 **Scimago Journal Rank 2022**: 1.3

Hephaestus: Modelling, Analysis, and Performance Evaluation of Cross-Chain Transactions

Rafael Belchior*†, Peter Somogyvari ‡, Jonas Pfannschmidt †, André Vasconcelos*, Miguel Correia*
*INESC-ID, Instituto Superior Técnico, Universidade de Lisboa †Blockdaemon ‡Accenture

***Abstract*—Ecosystems of multiple blockchains are now a reality. Multi-chain applications and protocols are perceived as necessary to enable scalability, privacy, and composability. Despite being a promising emerging area, we have been witnessing devastating attacks on cross-chain bridges that have caused billions of dollars in losses, and no apparent solution seems to emerge from the ongoing chaos. In this paper, we present our contribution to minimizing bridge attacks, by monitoring a *cross-chain model*. In particular, we aggregate *cross-chain events* into *cross-chain transactions*, and verify if they follow a set of *cross-chain rules*, which then generate a model.**

We propose `Hephaestus`, the first cross-chain model generator that captures the operational complexity of cross-chain applications. `Hephaestus` can generate cross-chain models from local transactions in different ledgers, realizing arbitrary cross-chain use cases and allowing operators to monitor their applications. Monitoring helps identify outliers and malicious behavior, which can enable programmatically stopping attacks ("a circuit breaker"), including bridge hacks. We conduct a detailed evaluation of our system, where we implement a cross-chain bridge use case. Our experimental results show that `Hephaestus` can process 600 cross-chain transactions in less than 5.5 seconds in an environment with two blockchains using sublinear storage, paving the way for more resilient bridge designs.

I. Introduction

Recently, many initiatives and projects have appeared around the concept of blockchain interoperability (BI), where a multi-chain ecosystem is perceived as the enabler for a scalable and adaptable platform for various use cases [1]–[4]. To enable such an ecosystem, bespoke distributed ledger technology (DLT) interoperability solutions, such as cross-chain bridges (or simply bridges), are used to connect heterogeneous DLTs, i.e., DLTs with different privacy, security, decentralization, and scalability properties [5]. The total value locked (TVL) in bridges peaked in March 2022, at around $30 billion worth of assets locked just in Ethereum (as the chain receiving the transferred assets) [6], [7], effectively reflecting the synergistic effects of free flow of capital, as now users can use their capital on multiple blockchains. As of September 2023, the TVL is still significant, collecting around 9B USD, as Figure 1 shows. With more than 40 bridging projects [8], the trend is for projects to either mature by improving their security and usability or disappear.

Some examples of recent mediatic attacks include the Wormhole bridge, where the attacker stole around $325M [9], [10], and the most significant on-chain attack in the cryptocurrencies history, the Axie Infinity's Ronin Bridge [11], which caused around $625M in losses. In February 2022, the Wormhole bridge was attacked and resulted in $320M in damage [12]. In June 2022, the Harmony bridge was hacked, resulting in $100 million in losses [13]. Although hackers were offered $1 million to return the funds to the community, it seems that they have not complied [14]. In August 2022, the Nomad bridge collateral was stolen, resulting in the loss of $200M [15], despite the bridge being developed by an expert team and audited multiple times. More recently, in July 2023, Multichain was hacked and lost around $120 million [16].

Looking at the facts, many of the largest decentralized finance hacks in blockchain history were performed in bridges [17], [18], in a grand total of more than $2.5B in damages [19], [20]. The facts show that the community still has a long way to implement secure bridges. The trend for attackers to exploit bridges will likely not disappear soon, as the more value bridges they hold, the more incentive criminals will have to attack those systems [21].

To mitigate the presented issues, we start by formalizing the interactions between different systems (which we refer to as *domains*). Cross-chain transactions (*cctx*) occur across domains and consist of a set of transactions abstracted into a logical unit of work [22], or a single atomic transaction [23]. We refer to single atomic transactions by cross-chain events (*ccevent*)). These can take place in both off-chain and on-chain systems. A *ccmodel* is the set of *cross-chain rules* that define the conditions for *cctx*s to occur - originating a state (cross-chain state). If transactions do not follow the specified rules the *ccmodel* defines, the model is *incorrect*, and thus there are several options for the analyst to proceed. Either the model is under-specified, or there is "suspicious" behavior that caused the violation of rules (e.g., malicious, such as an attack, or non-malicious, such as a software bug). Effectively, a *ccmodel* allows one to have a baseline of expected behavior to compare ongoing *cctx*s with the baseline model, following a specification-based approach to security [24].

Capturing cross-chain logic for bridges would be helpful to formalize the protocols (and help identify bugs and bottlenecks), monitor them, and act upon specific triggers. For instance, if an attack on a bridge is detected, a monitoring smart contract may pause the withdrawals, limiting the scope and impact of the attack. However, defining cross-chain logic is difficult because the base systems to be dealt with are heterogeneous and decentralized, and the systems

built on top of them (e.g., decentralized applications) may have arbitrarily complex business logic. They can comprise multiple other systems (e.g., smart contracts). In a cross-chain setting, automating the discovery of *ccmodel*s and enabling its monitoring becomes very challenging, as there need to exist more tools to secure and monitor cross-chain applications. This is where our work fills the gap in current knowledge. In summary, we present the following contributions:

- We propose `Hephaestus`, a system that creates *ccmodel*s for fine-grain monitoring and auditing multiple blockchain use cases. Our system uses and extends a state-of-the-art BI solution, Hyperledger Cacti [25].
- To assess `Hephaestus`' capabilities, we provide a comprehensive evaluation of the system. In particular, we validate our contributions by generating a *ccmodel* of a bridge system that transfers tokens between heterogeneous blockchains. We tested cross-chain model generation and monitoring capabilities of `Hephaestus` according to a set of metrics (including scalability, latency, and cost) on different scenarios and workloads. After that, we present a qualitative evaluation and a discussion of the evaluation and the proposed system.
- As a technical contribution of independent interest that directly supports our contributions, we have developed and improved various Hyperledger Cacti components over several months, including blockchain connectors, several test ledgers, the RabbitMQ test server, and several Python notebooks that are available under an open-source license for the community to use.

This paper is organized as follows. Section II presents the background. In Section III we present the concept of cross-chain transaction, and then cross-chain model, in Section IV. Section V presents Hephaestus. After that, we present implementation details, in Section VI, and the experimental evaluation, in Section VII. Section VIII presents the related work and Section IX concludes the paper.

II. Background

This section presents the background necessary to understand the paper, that is, processes, BI, and *cctx*s.

A. Process Mining Background and Applications

Understanding core concepts around processes is important to construct a system that can analyze *cctx*s and thus create *ccmodel*s. A process is a set of activities (or tasks) to fulfill a specific goal [27]. For example, behind running a proof-of-stake blockchain, we have different processes a validator needs to run to achieve the end goal, the network's maintenance process, the consensus process, and others.

The techniques for creating, analyzing, and optimizing processes are called process mining techniques [28]. Process mining has two sub-areas that help us in our endeavors: process discovery and process conformance. Process discovery aims to infer a process from an event log, that is, from a sequence of related entries, typically represented in a table. The entries in this table are *events*. An event is an occurrence that targets an activity and a point in time, related to each other using a *case id*. Events point to an *activity* at a certain time, i.e., they have a *timestamp*. Activities are the operations that are executed within a process. Formally, an event e is a tuple $(act, caseId, timestamp, store)$, where act is the activity name, $caseId$ is the unique reference to the event, $timestamp$ refers to when the event was created, and a key-value $store$. The key-value is in the form $\{(a_1, v_1), \dots (a_n, v_m)\}$, where each a is an attribute of the event and v its value. The set of all events is $\mathcal{E}$.

The execution of a process produces what is called a *trace*, an ordered list of events with the same case id. Formally, a trace is a non-empty sequence $[e_1, \dots, e_n] \in [1, ..., n], e_i \in \mathcal{E} \wedge \forall i, j, [1, ..., n] : e_i.caseID = e_j.caseID$. An event log is a collection of traces that refer to one or more cases. Discovering a process model can be done in various ways (for a detailed overview of how to generate process models, see [29]). Process conformance checks if the incoming transactions, including their ordering (or event entries), conform (are expected) to an existing model, helping evaluate a property called replay fitness. Conformance is part of process monitoring, helping identify errors or deviations from expected behavior. Processes have different representations. Graphical representations include BPMN diagrams [30], a helpful notation for complex process semantics. In BPMN, events are denoted as circles, activities as rounded squares, and gateways as diamond squares.

B. Blockchain and Interoperability

A blockchain is a distributed protocol in which a group of nodes collectively maintains a ledger $\mathcal{L}$ of ordered transactions, possibly grouped into blocks [31]. Blockchains support two basic operations: reads and writes. Keys index information on databases; blockchains can be seen as key-value stores. A read operation obtains the value for a certain key, while a write operation on a key updates the value and returns true if successful; otherwise, it returns false. The history of each key's values is conserved by the blockchain data structure, which aggregates transactions (write requests) into cryptographically signed blocks. Reads are used to capture the part of the state relevant to interoperability processes. Reads and writes are often mediated by smart contracts, stateful, user-defined programs run by the nodes composing the blockchain network.

The blockchain properties that need to be satisfied for interoperability are *consistency* and *liveness*, widely documented in the literature [32]. Informally, consistency means that for a pair of honest nodes, at every round, the global state of one node (list of ordered transactions) is a prefix of the other node, or vice-versa. Liveness means that if an honest node receives a transaction in a certain round, it will be included in the ledger and available for all nodes.

BI is the problem of coordinating *local reads* and *local writes* such that they satisfy some cross-chain logic. That is, reads from ledger $\mathcal{L}_1$ can be composed with a write-on $\mathcal{L}_2$, realizing multiple use cases, such as data transfers, asset transfers, or asset exchanges [33]. Extensive work has been done in this area, including using two-phase commit to provide *cctx*s ACID [34] properties, where each local transaction executes successfully, or none at all [23]. We assume that there

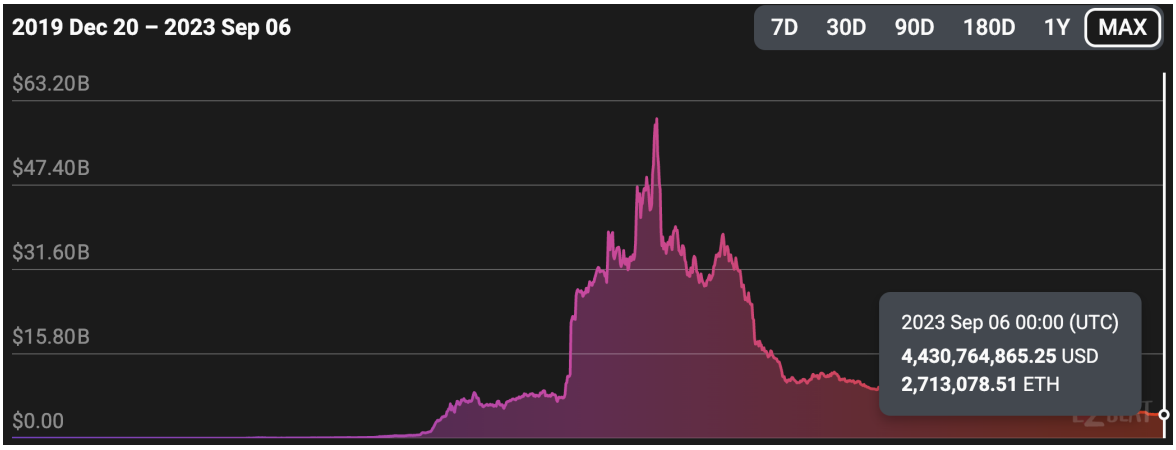


Fig. 1. Total value locked (TVL) in USD, on bridges, between 2019 Nov 15 – 2023 Sep 06. The green squares showcase relevant events in the bridging ecosystem, e.g., on 24 March 2023 (last green dot), the first ZK rollup with universal solidity support was launched. Source: [26].

is a cross-chain protocol deployed that orchestrates *cctx*s and defines the cross-chain rules that operate the use case. A *cctx* is an abstraction rooted in a set of local transactions from different systems (e.g., enterprise legacy system, centralized databases, blockchains), respecting a set of rules. Further ahead in the paper, we formally explain what these concepts are. We will map the concept of a *cctx* as a set of events (which represent local transactions) that constitute a trace over a process model (such as a ledger write). A local transaction is a transaction native to a given technological environment, called *domain*. Examples of domains are blockchains such as the Bitcoin network, the Ethereum main net, and centralized databases. Transactions trigger state changes and each state change is an event belonging to a trace. Transactions have different life cycles, data formats, and properties as a function of their domain.

Parameter	Type	Native
case ID	string	✗
receipt ID	string	✓
timestamp	Date	✓
blockchain ID	string	✗
invocation type	string	✓
method name	string	✓
parameters	string	✓
identity	string	✓
cost	number	✓
latency	number	✓

TABLE I
ccevent PARAMETERS, THEIR TYPE, AND NATURE (NATIVE ✓ OR NOT ✗).

III. Cross-chain Transactions

In this section, we define *cctx*s and their atomic units, the *cross-chain events* (*ccevent*s).

A *ccevent* extends a local transaction with metadata. We consider this metadata to be a set of non-native attributes (or parameters) $\{a_1, a_2, ..., a_n\}$ and their values $\{v_1, v_2, ..., v_n\}$. A *ccevent* e has native attributes (e.g., an *id*, a *timestamp*, the state key to which the transaction points (*target*), a *payload* (smart contract call) that will yield a state change, and a signature from the originator), and non-native attributes, obtained via a function add, i.e., $e = \texttt{add}_{t^d}(a, data)$, where add is a function that adds data item $data$ to an attribute a of a local transaction t from ledger l. Each $data$ item is a non-native parameter (marked with ✗ in Table I). The native parameters can be obtained from the underlying domains or systems, i.e., retrieved from the nodes supporting the blockchains without post-processing. Non-native parameters are externally obtained and are used to enrich local transactions. Native parameters may be used to calculate non-native parameters. For example, the carbon footprint depends on native parameters (e.g., on the native parameter cost (gas), in Ethereum).

A *cctx* is a set of n ordered events $\mathcal{E}$ from a subset of domains (e.g., ledgers) $\{d^1, ..., d^n\} \in \mathcal{D}$, i.e., $\mathcal{E} = \{e_1^{d^1 \in \mathcal{D}}, ..., e_k^{d^k \in \mathcal{D}}\}$, where k represents the number of events contained in a *cctx*. The events may follow a set of rules $\mathcal{R}$, the entity that logically connects events. Rules define conditions that must be verified to each event within a *cctx*; they depict the dependencies of each event on, for example, global time, local state, and third-party domain state. A rule is a datalog rule [35], [36]. A datalog rule contains a head $R_{\mathcal{E}}$ and a body, and is defined recursively. Given a set of predicates $\zeta = \{\zeta_1, \zeta_2, ..., \zeta_n\}$ over a set of events $\mathcal{E}$, we have that, for a certain time interval t_δ a rule is given in the form $R_{\mathcal{E},t_\delta} \longleftarrow \zeta(\mathcal{E})$ (we omit t_δ for simplicity of representation). The event set satisfying $R_{\mathcal{E}}$ are the intersection $\{\mathcal{E}|\zeta_1(\mathcal{E}) \wedge \zeta_2(\mathcal{E}) \wedge ... \wedge \zeta_n(\mathcal{E})\}$, this is, for an event set to satisfy a rule, it needs to satisfy all predicates. Each predicate ζ can define the conditions over transactions, i.e., temporal dependencies, the domain of a transaction, or a target function.

For example, consider the following rule (predicate set):

$$\zeta(\mathcal{E}) = \begin{cases} \zeta_1(e) = e^x \prec e^y & \text{order dependency} \\ \zeta_2(e) = \forall e : e^x \vee e^y & \text{included domains} \\ \zeta_3(e) = \exists e : e.cost < z & \text{event attributes} \\ \zeta_4(e) = e_w.target = e_{w+v}.target & \text{event payload} \end{cases} \quad (1)$$

In this predicate set, ζ_1 defines any event that occurs in the domain d_x precedes ($\prec$) any event happening in the domain d_y. The predicate ζ_2 defines events as part of domains x or y. Predicate ζ_3 states that there is at least one event in the event set, so its cost is less than z. Predicate ζ_4 states that the target of a transaction repeats every v transactions. Other predicates can be set for any of the attributes of a *ccevent*, in Table I. While we require each event to satisfy each sub-predicate of ζ, we can also set the validity of rules as the union $\{\mathcal{E}|\zeta_1(\mathcal{E}) \vee \zeta_2(\mathcal{E}) \vee ... \vee \zeta_n(\mathcal{E})\}$, or any other combination of predicates. We assume that there is an efficient way to transform a set of conjunction predicates into disjunctions or other formats. To capture this variability, we define a function `verifySatisfiability` that takes a predicate and an event and outputs true if the event satisfies the given predicate and false otherwise, i.e., $\texttt{verifySatisfiability}(e, \zeta) \longrightarrow \{0, 1\}$. We can then use this predicate for each event to assert a rule's validity.

To understand how this concept applies in practice, consider the following (simplified) rule that dictates the necessary events for a valid cross-chain asset transfer:

$$\zeta'(\mathcal{E}) = \begin{cases} \zeta'_1(e) = \underbrace{(\forall e(e^x \vee e^y)}_{\text{events happen in x or y}} \wedge \underbrace{\forall_{e \in (x,y)}(e^x \prec e^y)}_{\text{events on x happen before y}} \\ \zeta'_2(e) = \underbrace{e^x.target = exists(a) \wedge e^x.target = lock(a)}_{\text{asset can only be locked if exists}} \\ \zeta'_3(e) = \underbrace{\zeta'_2}_{\zeta_2\text{' is satisfied}} \wedge \underbrace{e^y.target = mint(a)}_{\text{a mint can occur in domain y}} \end{cases} \quad (2)$$

Let us define rule ζ, the disjunction of the ζ' predicate set. The predicate set ζ' defines a set of conditions for a cross-chain asset transfer to be valid. First, as determined by ζ'_1, events in domain x must happen before events in the domain y. This paves the way for a lock on a source blockchain to be done before a mint on a target blockchain. Predicate ζ'_2 states that an asset from the source blockchain must exist before it is locked. Predicate ζ'_3 states that before an asset is minted on the target blockchain, predicate ζ'_2 must be satisfied. One could add more rules, such as the time for a mint transaction has to be done before block b, i.e., $e.target = mint \wedge e.timestamp \prec b$. We illustrate a cross-chain use case that allows asset transfers, in finer detail, in Section VII-B.

IV. Cross-Chain Model

This section defines *ccmodel* and its artifact, *ccstate*. A *ccmodel* $\mathcal{M}$ is a tuple $(\mathcal{R}, cctx)$, where $\mathcal{R}$ is a set of cross-chain rules, $cctx$ is a set of *cctx*s. The *cctxs* originate a cross-chain state $\mathcal{S}$.

A. Properties

Cross-chain models have a set of properties:

- Verifiable correctness (safety property): a model is *valid* if all *ccevent* e in each $cctx$ respects the set of rules $\mathcal{R}$, i.e., $\forall cctx \in \mathcal{M} : \forall(e \in cctx) : \texttt{verifySatisfiability}(e, \mathcal{R}) \longrightarrow 1$.
- Liveness: the current cross-chain state $\mathcal{S}$ is updated no later than every t timesteps. Updating the *ccstate* implies checking the existing *cctx*s against the model rules.
- Probabilistic completeness: the larger the event log (i.e., the number of observed events and consequently *cctx*s), the higher the model completeness probability.
- Replay fitness: given an observation of the real-world use case, the matching between the events and the *ccmodel* is higher than a threshold probability p.

Cross-chain models are correct if each *ccevent* follows each rule, as suggested by [33]. Note that the `verifySatisfiability` predicate can be defined in several ways (e.g., a conjunction of rules, disjunction of rules, or a bespoke combination). We say a model is *incorrect* if there are events that do not satisfy the rules of the model. For example, the execution of *ccevent*s in an incorrect order as specified in the rules causes a model to be incorrect. In this case, the model is not secure. As *ccmodel*s capture security by evaluating a set of predicates on events and rules, we can capture different safety properties, such as atomicity, double-spend protection, and others commonly debated in the interoperability literature [1], [33], [37], [38].

Updating a model is, in practice, polling for new cross-chain events and matching them with the rules defined by the model. To this end, and as each domain has its own clock, measuring and tracking time across systems is important. Liveness states that models need to be updated every t timesteps - the time between updates is an attacker's time window.If a model cannot guarantee liveness within t timesteps, we say that the model is outdated (and thus security is not guaranteed). We will show that liveness is particularly important to detect cross-chain attacks.

Completeness is related to precision. A precise model avoids underfitting, a degree of measurement of how complete is the *ccmodel*. Replay fitness expresses the ability to explain on-chain behavior, i.e., how close the *ccmodel* is to reality. Other properties that are interesting in our context matter, but will be explored in future work. These generalizations measure whether the model is too tied to specific execution instances of a cross-chain use case. Simplicity measures whether the model is understandable by humans. Other aspects are omitted from the model generation, such as the task of minimizing noise, that is, minimizing behavior that is infrequent and does not represent the typical behavior of the process.

B. Cross-Chain State

The cross-chain state $\mathcal{S}$ is a key-value store that holds attributes relevant to the cross-chain use case, i.e., they are defined on a case-by-case basis. It is generated from the *cctxs* from the *ccmodel* and it is similar to the world state concept in Hyperledger Fabric. Essentially, the state contains the result

of executing all the *cctx*s, possibly enriched with metadata. For example, if the use case is a bridge, the state will record the assets locked in the source chain and the corresponding representations minted on the target chain, for each user, and some key metrics (see Section VI-C). Metrics are performance attributes of a set of *cctx*s [39] and provide meta-information about a cross-chain use case. These metrics indicate points of interest in a cross-chain use case. Metrics realize a meta state, where metrics about the formation and execution events that lead to that state are created.

Latency: We define latency as the time between a local transaction (via extended clients) and the creation of an *ccevent*. The total latency of a *cctx* ($\delta(cctx)$) is given by the latency of each event $\delta(e)$ from each local transaction, summed to the operational latency of the *ccmodel* generator ($\delta(op)$):

$$\delta(cctx) = \sum_{\substack{i=1,\dots,n \\ j=1,\dots,k}} \delta(e_i^{d_j}) + \delta(op) \qquad \forall d\, e \in cctx \quad (3)$$

The operational latency is the time the model generator takes to retrieve and process the local transactions.

The latency of a *ccmodel* is the sum of the latency of each *cctx*:

$$\delta(\mathcal{M}) = \sum_{i=1,\dots,n} \delta(cctx_i) \quad (4)$$

Throughput: The throughput of a *cctx* is defined as $\frac{1}{\delta(cctx)}$, and it counts the number of sets of events processed per unit of time. Effectively, the latency for each event compresses the issuance and processing of each local transaction (which can take a long time depending on the blockchain), plus operational costs. The slowest finalization time δ_{fmax} can be a valuable metric to complement throughput (and help identify bottlenecks in a *cctx*).

$$\delta_{fmax} = \max_{e_i \in \mathcal{E}, d \in \mathcal{D}} (\delta(e_i^d) \quad (5)$$

Cost and Revenue: Each local transaction might have a cost of transaction fees plus operation fees (in case a relayer or entity is transporting the local transaction payload across chains). Inspired by [39], we define the cost c of a $cctx$ events as the sum of variable costs (c_δ) plus operational costs (c_{op}):

$$c(cctx) = \sum_{\substack{i=1,\dots,n \\ j=1,\dots,k}} cost_\delta(e_i^{d_j}) + c_{op} \qquad \forall d\, e \in cctx \quad (6)$$

The environment (e.g., via our system) typically gives information about these costs. The revenue is calculated in a way similar to the above formula. We can then calculate the profit of each *cctx* by subtracting the costs from the revenue. The concept of revenue can be modeled as positive utility and cost as negative utility, which sometimes maps better to real-world applications.

V. Hephaestus: A Cross-Chain Model Generator

`Hephaestus`[1] is a software system that generates *ccmodel*s. It first captures local transactions (*ccevents*) and then generates *cctx*s. Those *cctx*s assist in generating the *ccmodel*, along with a process discovery algorithm. Derived from the *cctx*s we have the *ccstate*, which holds metrics of interest, and a key-value store that represents the system variables with regard to bespoke business logic.

The *ccmodel* holds a set of rules (as defined in Section III), and dictates which *cctx*s should be issued. In particular, rules define the order and dependencies of each *ccevent* that later form a *cctx*. Such models can be specified or learned from the environment. After that, such a model will continuously update its cross-chain state in real-time during the monitoring phase (Section V-D). This framework allows us to answer several questions: *What is the state of our cross-chain use case/protocol? Are there unexpected behaviors, i.e., deviations from the model? What are the current bottlenecks of a given cross-chain use case?* and *Is there suspicious behavior concerning my use case, for example, an attack?*

A. System Model

Cross-chain applications are structured as multi-step protocols with different types of agents. Agents are users (e.g., end-users, relayers [2], or protocol administrators) and smart contracts. Users take turns doing off-chain processing and interacting with one or more domains (e.g., submit transactions against smart contracts). Agents are considered Byzantine, i.e., they can attempt to deviate from a protocol. Smart contracts enforce cross-chain logic. Specifically, smart contracts running on different blockchains are trusted replicas in the state-machine replication literature (similarly, each domain is considered trusted, even if centralized). This assumption implies that if a domain cannot be trusted, then safety on cross-chain rule execution cannot be guaranteed. Domains can only learn the state of other domains and their changes using an agent. Of course, each domain must decide whether an agent is telling the truth. This can be achieved with a cross-chain protocol, where trust assumptions vary substantially [1]. Furthermore, we assume a partial synchrony model, i.e., there is some finite unknown upper bound t on the responses (e.g., event creation) from the underlying domains. `Hephaestus` runs a global clock, i.e., it can measure time against different domains, despite their clocks being different.

Our model assumes a cross-chain protocol that can provide a set of rules such that the set of rules is enforced by on-chain (e.g., smart contracts) and off-chain components (e.g., relayers). In particular, for us to generate a correct *ccmodel*, the execution of cross-chain transactions needs to result in a consistent cross-chain state for a certain set of rules. In our bridge example, the rules define that no double spending occurs. This is, it is impossible to mint an asset on a target blockchain without first locking it on the source blockchain; similarly, it is not possible to unlock such an asset on the source blockchain without first burning it on the target

[1]Hephaestus is the Greek god of metallurgy (that can connect different chains into a single useful artifact.)

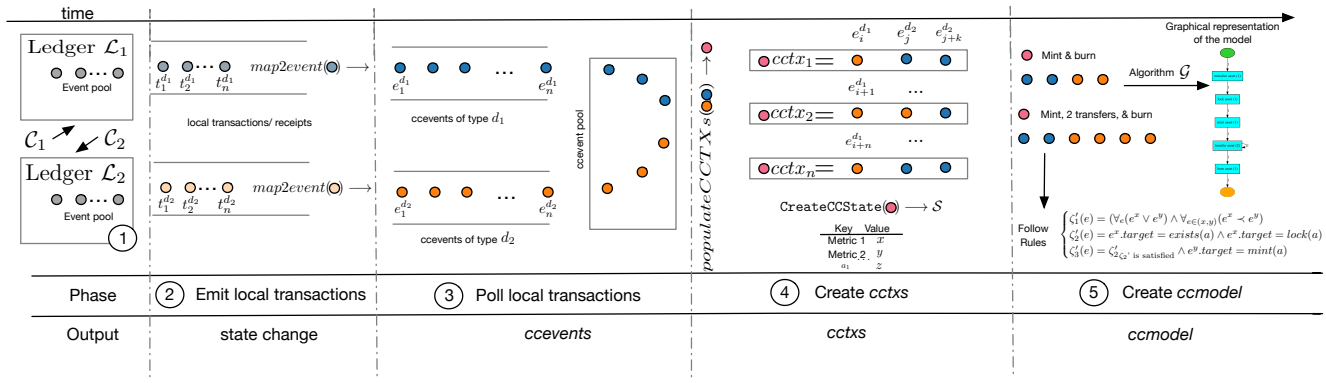


Fig. 2. Cross-chain model pipeline, spanning from phases 2 to 5.

blockchain. We will elaborate on this use case in Section VII. Liveness ensures that all cross-chain rules from a protocol are evaluated (executed) timely. In our example, liveness means that "conforming parties' assets cannot be locked up forever".

B. System Overview

After defining our domain scope (step ①), a set of modified blockchain clients, called *connectors* issue transactions against target blockchains via an application (step ②). These blockchains emit events (or transaction receipts) that our connectors capture. `Hephaestus` then collects the local transactions (also called transaction receipts) in step ③ and generates *ccevent*s enriched with metadata. After that, it generates a set of *cctx*s (④) and next, it generates a *ccmodel* (⑤). The next section will illustrate the steps in finer detail. Finally, the monitoring phase occurs (Section V-D), where events are constantly monitored and used to update the *ccstate*. Business logic can be defined to facilitate the integration with legacy systems or to implement audit or monitoring functionality.

C. Cross-Chain Model Generation

This section explains how to generate a model, focusing on phases ② to ⑤. Our pipeline is divided into phases (cf. Figure 2): ② *Emit Local Transactions*, ③ *Poll Local Transactions*, ④ *Create cctxs*, and ⑤ *Create ccmodel*. The monitoring phase is illustrated in the next section.

In phase ②, we start by listening to local transactions in our domain set $\mathcal{D}$. Each domain in our system has an accessible event pool from which we can fetch the events used to build the model. Without loss of generality, and to simplify our reasoning, we look for local transactions in the domains ledger $\mathcal{L}_1$ (), and ledger $\mathcal{L}_2$ (). The considered transactions are created and submitted by our connectors, $\mathcal{C}_1$ and $\mathcal{C}_2$, respectively. Transactions have a case id, meaning local transactions without this special identifier are not considered. Our clients capture the relevant transactions and send them to the *ccmodel* generator, starting the next phase. In phase ③, the raw transaction receipts enter a processor module that translates local transactions into a *ccevent* according to a function $map2event(tx_{\mathcal{L}}) \longrightarrow ccevent$.

The output of phase ③ are *ccevent*s coming from $\mathcal{L}_1$, $event_1$, and from ledger $\mathcal{L}_2$, $event_2$ which we aggregate onto a *ccevent* log, with events coming from different ledgers. Therefore, events implement a standardized data model. After constructing a set of events, we proceed to phase ④. In this phase, we receive an event log and output the cross-chain state and a set of *cctx*s (via a *createCCTX*s function).

Algorithm 1: `Cross-Chain State Update.` Creation of a cross-chain state from a set of *ccevent*s

Input: Set of events $\mathcal{E}$
Input: State update algorithm `createCCState`
Input: Cross-chain rules $\mathcal{R}$
Input: Cross-chain state $\mathcal{S}$
Output: Upon success returns cross-chain state $\mathcal{S}$, and a *SYNC MOVE*

```
require verifySatisfability(e, R, S) // Returns tuple (event, MOVE ON LOG ) if event do not conform to the rules, cross-chain state is invalid.
foreach e ∈ E do
    // For each event in retrieved event set
    if ∄S[e.caseID] then
        // each cross-chain state key is indexed by case ID.
        cc = populateCCTX(S[e.caseID], e)
    end if
    else
        cc = updateCCTx(S[e.caseID], e)
    end if
    S = S ∪ cc
    S' = createCCState(S, e.caseID)
     // Calculates the updated ccstate, algorithm is parametrizable
end foreach
return (S', SYNC MOVE )
```

Algorithm 1 illustrates step ④. It receives the processed *ccevent*s from step ③, a state update algorithm `createCCState`, a collection of rules (as specified in the use case), and the previous *ccstate* (could be empty). Note that the *ccevents* follow the data model specified in Section III. Algorithm 1 is responsible for monitoring that the events follow the rules, and is executed from time to time. Further-

more, it updates the cross-chain state (composed of metrics and a key-value store).

First, we collect every event retrieved dynamically by the connectors. A "require" check on line 1 verifies if the set of *ccevent*s respects the rules (see the defined rules for the bridge use case in Section III). Note that the rules evaluation might consider the current *ccstate*, which is why it is provided to the `verifiySatisfiability` primitive. If the check returns false, the algorithm returns an error and is handled by the incident management component. We will describe more details in the next subsection.

Otherwise, we iterate on the events from the set and aggregate them into *cctx*s (lines 3-10). Note that *cctx*s are indexed by caseID. Here, we create the metadata fields, namely the metrics (latency, cost, throughput, in line 5) or update them (in line 8) depending on if a *cctx* with identifier "caseID" already exists. The `populateCCTXs` function assigns the different metadata from *ccevent*s and attributes (e.g., payload) to the newly created *cctx*. The `updateCCTx` updates a *cctx* based on the new information a *ccevent* carries. For example, if the *ccevent* carries a cost, the new cost of the *cctx* will have its cost incremented by *e.metadata.cost*. The new or updated *cctx* is added or updated to our current *ccstate*. Now, we need to update specific cross-chain semantics with the function `createCCState`. We provide an example for `createCCState`, Algorithm 2. This algorithm verifies if the current event locks an asset on a source blockchain so it can be minted on a target blockchain. It sets a bit to one for each locked asset. Note that this function is illustrative and does not reflect a complex lock-unlock mechanism. For instance, the algorithm should check if an unlock with a newer timestamp happened regarding a locked asset and perform user management.

Having a cross-chain state, we initiate the *ccmodel* generation phase (5). In this phase, we generate a *ccmodel* using an algorithm $\mathcal{G}$ and the *ccstate*. Several graphical representations are possible, such as a process tree or a BPMN model. We provide details on this process in Section VI-D.

Algorithm 2: State update algorithm processCCState - Verification of a lock transaction referring to asset a

```
   Input: Cross-chain state S
   Input: Case id id referring to the lock
   Output: Updated cross-chain state S
1  foreach cctx ∈ S do
2      if cctx.caseID = id then
3          // current event?
4          if cctx.target == verifyLock ∧ e.target == a
             then
5              // if current event specifies a lock
6              S.lockedAssets[a] = 1  // then set asset a
                 to locked
7          end if
8      end if
9  end foreach
10 return S
```

D. Cross-Chain Transaction Monitoring

In this section, we explain how we monitor transactions and detect non-conformance behavior, alleviating on-chain bridge hacks. Non-conformance behavior can be one of three: outliers, malicious intent (bug exploitation/attack), or non-modeled behavior. The baseline for detecting non-conformance is a *ccmodel*, which corresponds to a specification of expected behavior. We define a set of traces belonging to the set of all possible traces $\{t_1, ..., t_n\} \in \mathcal{T}$ as a current execution of a cross-chain use case. For each trace being executed, we consider a set of steps $s_1, ..., s_n$. We then take the sequence of steps and perform alignment. Alignment-based replay aims to find one of the best alignment between the trace and the model. Each alignment creates a set of pairs (trace, transition) such that for each pair, one of the following can occur: 1) *SYNC MOVE* 🟢 - both the trace and the model advance in the same way during the replay, meaning we have a match, 2) *MOVE ON LOG* 🔴 - the trace that is not mimicked in the model. This means there is a deviation between our specification and the observed behavior.

The idea is now to retrieve incoming *ccevent*s at every t timesteps and build a trace. The current trace is constructed and encoded in the *ccstate*. For every incoming event, the next step of the trace is calculated and compared against the model (i.e., compared against the set rule $\mathcal{R}$). If a trace is *SYNC MOVE* 🟢,i.e., `verifySatisfability` returns true, then it is common behaviour (expected). The state is updated and returned. Otherwise, it is non-modeled behavior, and *MOVE ON LOG* 🔴, along with the point on the trace (current event) that originated the error. The lesser t, the better liveness a model provides and, consequently, the smaller the attack window. This error triggers an incident response framework. The framework defines how the incident should be investigated as it can result from under-modeling or malicious behavior (for example, an attack). The definition of an incident response framework for briges is out of scope and is left for future work. In any case, the end-user may inspect the event leading to the trace and understand which parameter has caused such behavior. Malicious behavior can come in different forms. The most common are smart contract vulnerabilities holding the business logic that realizes the use case. Many more attack vectors exist, such as smart contract framework vulnerability, dependency vulnerability, cryptographic vulnerability, network attacks such as denial of service or network partitioning, consensus manipulation, and others [40].

Upon detecting malicious activity, an incident response plan can be put into practice and halt the bridge until a patch is deployed, according to good practices [7], [41], [42]. The most prevalent attack in bridges is a cross-chain double spend [33], [43]: where the lock-unlock mechanism of such bridges is bypassed. In this paper, we focus on this type of attack, executed as a smart contract exploitation further elaborated in Section VII-D.

VI. Implementation

In this section, we present the implementation. The code is available on Github[2]. We developed our work as a Hyperledger Cacti (Cacti) [25] plugin. Cacti is a blockchain integration project supported by enterprises such as Blockdaemon, Accenture, IBM, and Fujitsu, with more than 270 stars and 80 contributors. Next, we detail this paper's relevant technical contributions and the implementation of `Hephaestus`.

Concept	Implementation	Lines of code
Domain	Fabric ($\mathcal{L}_1$), Besu ($\mathcal{L}_2$)	–
Domain logic	Bridge smart contracts	819
Ledger client	Fabric ($\mathcal{C}_{\mathcal{L}1}$), Besu ($\mathcal{C}_{\mathcal{L}2}$)	17k
Test ledger	Fabric and Besu test ledgers	1.8k
Model Generator	`Hephaestus`	5.9k
Process Discovery	pm4py	–
Process Conformance	pm4py	–

TABLE II
Implementation effort as the number of lines of code created, for each presented component

A. Connectors

We implemented two blockchain connectors to connect to multiple blockchains and retrieve transactions. Connectors are self-contained application programming interfaces that constitute the basis for interoperability functionality. The first connector binds our software to Hyperledger Fabric 2.2 — a permissioned blockchain system. Fabric is designed for enterprise-grade applications that benefit from decentralization. It supports smart contracts (called chaincode), that can be written in several general-purpose programming languages. The nodes execute proposals for transactions signed and sent to an orderer node. Orderer nodes reach consensus on the order of transactions, batch them into blocks, and link them, creating the blockchain. Then, new blocks are sent to the nodes on the network. Fabric has a key-value store that holds the most up-to-date values from the blockchain - a desirable programming model to implement a bridge; it allows chaincodes to retrieve state without reconstructing the blockchain. We implemented this connector, package name *cactus-plugin-ledger-connector-fabric* in Typescript, counting ≈5k lines of code. We wrote 16 integration tests, accounting for ≈4k lines of code. The connector supports functionality to issue transactions (*transact*), deploy smart contracts (*deployContract*), send transaction receipts to `Hephaestus`, and several administrative tasks (such as registering a new user).

The second connector connects to a Hyperledger Besu (Besu) 1.5.1 network. Besu is an open-source Ethereum client, that also has capabilities to span private EVM-runtime compatible networks. It allows for interacting with Ethereum networks, including participating in the consensus process, developing and deploying smart contracts and decentralized applications. Besu implements proof of authority algorithms such as IBFT (more suitable for private networks) and proof of work (Ethash). We implemented this connector, package name *cactus-plugin-ledger-connector-besu* in Typescript, counting ≈5k lines of code. We wrote 14 integration tests, accounting for ≈3k lines of code. The connector supports functionality to issue transactions (*transact*), deploy smart contracts (*deployContract*), send transaction receipts to `Hephaestus`, and several administrative tasks (such as obtaining a raw block from the network).

B. Test Ledgers

We implemented tools to programmatically create test networks for Fabric and Besu, allowing for reproducible tests and debugging of our application, namely the tools *besu-all-in-one* and *fabric-all-in-one*. These tools not only allow the reproducibility of our work but also ease the developers to create new applications and build on top of `Hephaestus`. The all-in-one test ledgers are divided into two parts: 1) a test ledger manager, a Typescript program that launches, administrates, stops, and destroys test ledgers by binding to a process running a Docker container, and 2) Dockerfiles defining the networks. The Fabric test ledger manager has ≈1k lines of code. The Besu test ledger manager has 428 lines of code. Other test ledgers such as *corda-all-in-one* and *substrate-all-in-one* are available for the research community.

C. Bridge and Smart Contracts

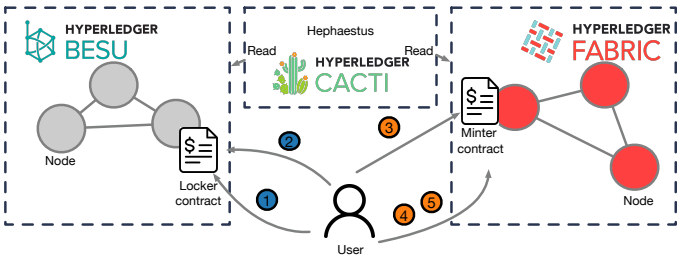


Fig. 3. Cross-chain bridge across Hyperledger Fabric and Hyperledger Besu

We implement a use case of asset transfers across a Hyperledger Besu network and a Hyperledger Fabric network as a foundation for testing `Hephaestus` capabilities. A cross-chain asset transfer triggered by an end-user generates a set of events representing an asset lock on a source blockchain (Besu) and an asset unlock on a target blockchain (Fabric). This lock-unlock mechanism assures that the representation of a minted asset is pegged to a locked asset. In asset transfers, there is typically a third-party actor called a relayer, which carries the proof of a lock to the target blockchain, so the mint can occur[3]. Alternatively, the user can provide proof. Without loss of generality, we use the later representation.

The use case is implemented as follows: on the Besu network, we have a Solidity smart contract "Locker" with two methods: *create asset*, and *lock asset*. First, a user must

[2]https://rafaelapb.page.link/code

[3]We could model the relayers behavior in our use case, by adding two activities: *submit proof*, which contains a payload that certifies that an asset was locked from the source chain, and *proof submission*, a payload consisting of a proof that validates the minting of an asset. These events would be added by a system other than the mock blockchain (e.g., relayer).

create an asset (step 1) and then lock it (step 2). On the Fabric network, the Typescript smart contract "Minter" allows a user to call the method *mint asset* (step 3), creating the representation of the locked asset. Afterward, a user can freely transfer that token to other users in exchange for other tokens, using *transfer asset* (optional, step 4). Finally, if users want to recover the original tokens, they run *burn asset representation* (step 5). This procedure would unlock the assets on the source chain (omitted for brevity).

D. Hephaestus Plugin

We implemented `Hephaestus` as a business logic plugin for Hyperledger Cacti, written in Typescript. Its latest version is version commit *8d8567e* (*stable branch)*, package name *cactus-plugin-cc-tx-visualization*. The main class is *CcTxVisualization*, which takes as input references different blockchain connectors. The plugin can be run as a web service, inspecting the event pools. After a local transaction is detected, the plugin adds them to a temporary queue. Transactions are transformed into events. From time to time, events are transformed into *cctx*s (batched for efficiency). The data model for events and metrics can be defined by the developer.

`Hephaestus` uses the *cctx*s to are used to build a cross-chain state, which is made available to the applicational layer. Models are built from *ccevent*s, given as input to a Python script (model generator) that, on its end, generates the *ccmodel*. The model generator used the open-source library pm4py version 2.2.20 [44]. We generate our model using the Inductive Miner algorithm [45], and generate the corresponding BPMN and process tree diagrams. Our plugin counts ≈5.5k lines of code. To identify misconformance, we used an alignment technique, available in the *conformance_diagnostics_alignments* function from the *pm4py* library, namely the Scipy linear solver tool.

VII. Evaluation

Goals: The goals of the experiments are as follows. 1) evaluate `Hephaestus` performance in terms of transaction throughput, latency, and storage required. We also evaluate the scaling capabilities concerning the number of local transactions, activities, and domains. Goal 2) is to evaluate the system's capability to identify misconformance, given a baseline *ccmodel*. In this section, we first conduct an experimental evaluation, followed by a qualitative analysis and discussion.

Experimental Setup: We deployed an instance of `Hephaestus` on Google Cloud (CPU with eight cores, 32Gb of RAM, SSD). The different event providers are our Hyperledger Fabric connector, and the Hyperledger Besu connector (version 1.0.0). We initialize a RabbitMQ server serving our event collector *rabbitmq-test-server*. Event emitters on the connectors are implemented as RabbitMQ clients. Every experiment was run 50 times (where we removed the first and last 10 runs, considering a total of 30 runs), and we report the average result, along with the standard deviation, for each run. We share the scripts to generate the plots and *ccmodel*s, making the evaluation process reproducible. Furthermore, we save the output of each evaluation scenario[4] and the generated cross-chain logs[5] and share it with the reader.

Metrics and Workloads: For each run, we capture the following metrics: throughput (*cctx*s) and their latency, storage cost, i.e., performance metrics. We test two scenarios under variable workloads, which we present later in this section. We characterize each scenario as a tuple *(interoperation mode, number of blockchains, event type, and workload)*. The interoperation mode states what cross-chain feature we are testing, asset transfers, asset exchanges, or data transfers. While intuitive, for space limitations, we refer to [33] for a detailed explanation. The number of domains reflects the number of ledgers or other systems emitting events in the scenario, namely Hyperledger Fabric, Hyperledger Besu, or a mock blockchain (essentially, we only model the message transmission). Finally, each workload contains details on the number of events, activities, and domains in that scenario. We implemented a workload generator that produces events across different blockchains. Events are then captured by `Hephaestus`.

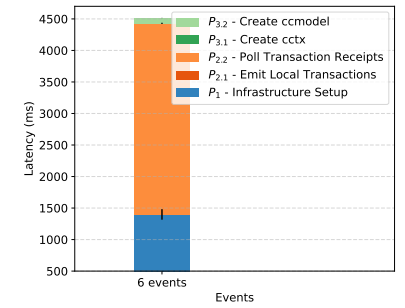
A. Baseline: Dummy Use Case with Test Receipts

In this section, we depict the evaluation of our system using a mock blockchain, interoperation mode asset transfer, within a single domain. The workload consists of 6 events, 6 activities, with *ccmodel* generation algorithm $\mathcal{G}$ = `inductive miner`.

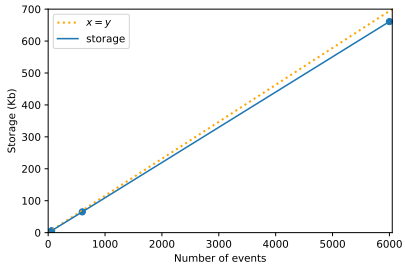
The dummy use case represents a *cctx* composed of 6 *ccevent*s. This transaction locks an asset from a source blockchain and unlocks a representation of the same asset on a target blockchain (typically using parties called relayers[6]). Instead of using blockchains to collect receipts, receipts are emitted by a single mock blockchain, which we call the *test blockchain*. The mock blockchain processes transactions as detailed in Section VI-C, namely *create asset*, *lock asset*, *mint asset*, *transfer asset* (optional), and *burn asset representation*.

We measure the performance of the following phases (see Figure 3a): the *Infrastructure Setup* (phase 1), the emission and polling of local transactions, as events *Emit Local Transactions* (phase 2.1), and *Poll Local Transactions* (phase 2.2), the creation of *cctx*s, *Create cctx* (phase 3.1) and the creation of the *ccmodel*, *Create ccmodel* (phase 3.2). The *infrastructure setup* includes setting the event emitters (connectors, including creating blockchain networks and initializing the connectors), setting the event collector (RabbitMQ server), and setting up `Hephaestus`. The *Emit Local Transactions* phase emits test events or issues transactions against the deployed ledgers. The *Poll Local Transactions* waits for the events and sends them to `Hephaestus` for processing. The *Create cctx* generation includes mapping the local transactions to *cctx*s,

[4]Online: https://rafaelapb.page.link/cctx-viz-output
[5]Online: https://rafaelapb.page.link/cctx-viz-csv
[6]We could model a third-party responsible for carrying proofs of on-chain execution, the relayer [1] - yielding two more events, in addition to the modeled six. Those two extra events are modeled as activities: *generate proof*, which contains a payload that certifies an asset was locked from the source chain (the proof), and *submit proof*, an event asserting a transaction with proof was submitted to be validated.



(a)



(b)

Fig. 4. Figure a) shows the latency, in milliseconds, for each phase of the baseline test scenario. Figure b) storage requirements, in kilobytes, for a variable number of events.

and calculating *cctx* metrics. Figure 4a) shows the latency phase breakdown for the emission of six events. We can observe that the setup phase takes around 1.5 seconds, and the most time-consuming phase takes approximately 3 seconds, despite being mock transactions. Figure 5 shows the same breakdown for a variable number of events. Table III supports this figure by reporting the mean end-to-end latency for each phase along with the standard deviation. Phases 1 and 2.2 remain practically constant. Phase 2.1 is sublinear. Phase 3.2's performance indicates that after a certain threshold (between 600 and 6000 transactions), the system starts bottlenecking.

We measure the storage required for generating, storing, and processing events into a *ccmodel*. Figure 4b) shows the required storage as a function of the number of events created. The RabbitMQ container and respective runtime data occupy 257Mb and 72.4kB, respectively. The storage requirements appear to be sublinear to the number of events - six events (one *cctx*) occupy 789 bytes, while six thousand events occupy around 6.6Mb. For a *cctx*, means each *cctx* occupies around 789 bytes + derived data (metrics, a few bytes). Since the metrics are five floats, a date, a string with 128 chars, and a list of events, each *cctx* occupies at least 937 bytes. Finally, the *cross-chain model generation phase* includes parsing the created *cctx*s and generating the *ccmodel*. The generated BPMN model for the dummy use case scenario is represented in Figure 5. Each *cctx* takes 985 milliseconds to build.

B. Use Case: Asset Transfer across Heterogeneous Networks

In this section, we depict the evaluation of our system using two blockchains. The interoperation mode is asset transfer

	Phase 1		Phase 2				Phase 3			
	——		Phase 2.1		Phase 2.2		Phase 3.1		Phase 3.2	
events	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
6	1397.53	83.06	0.47	0.51	3030.27	8.94	0.67	0.61	83.51	1.95
60	1387.93	20.09	2.20	0.66	3068.97	19.07	1.27	0.45	87.2	0.93
600	1388.33	22.26	13.03	3.02	3163.47	21.89	7.53	1.14	98.21	1.51
6000	1392.20	18.05	116.97	20.67	3459.37	18.36	26.5	3.42	265.61	4.18

TABLE III
END-TO-END PROCESS LATENCY MEAN (μ) AND STANDARD DEVIATION (σ), IN MILLISECONDS, AS A FUNCTION OF THE NUMBER OF EVENTS.

within two domains. The workload is composed of batches of 6 transactions (2 Besu plus 4 Fabric), and 6 activities, with *ccmodel* generation algorithm $\mathcal{G} =$ `inductive miner`.

Next, we illustrate an asset transfer between a private network running Hyperledger Besu and a private network running Hyperledger Fabric, implementing a cross-chain bridge. The rule set for a valid cross-chain asset transfer is illustrated in Equation 2, from Section III. The asset transfer process is the same as in the baseline scenario, i.e., a *cctx* is composed of 6 events, where two are local blockchain transactions. An `Hephaestus` instance is connected to a Fabric connector and a Besu connector. Each connector is connected to a Fabric network version 2.2 and Besu network version 21.1, respectively. The Fabric network consists of 2 peers and 1 orderer, using Raft as the consensus protocol of orderers and LevelDB to maintain the local storage in each node. The Besu network consists of a solo node network. The use case explored in this section follows the same transaction flow as the baseline, i.e., transactions *create asset*, *lock asset*, *mint asset*, *transfer asset* (two of them) and *burn asset representation* are issued in this order. This flow implements a simplified version of a cross-chain promissory note transfer [23] between Hyperledger Besu and Hyperledger Fabric. We used the smart contracts described in Section VI. Figure 6 depicts the normal functioning of the bridge, and also a scenario where double spending happens.

When testing the variable workload of 6-6000 events, the "Infrastructure Setup" and "Poll Transactions Receipt" phases take the most time, as expected. The median latency required for these phases is 1392 and 3469 seconds, respectively, for 6000 events. The infrastructure setup phase takes 90%, 56%, and 12% of the overall execution latency, while the transaction emission takes 7%, 42%, and 88%, for 6, 60, and 600 transactions, respectively. Since these phases take most of the execution time, we illustrate the breakdown of the remaining phases, "Poll Local Transactions", "Create cctx", and "Create ccmodel", in Figure 4. The storage requirements are similar to the baseline use case. For a 60-event execution, we obtain that each *cctx* (6 events) takes 2,04 seconds to construct.

C. Baseline Vs. Use Case

The bottleneck for both scenarios is the infrastructure setup (phase 1) and transaction emission phase (2.1) or polling transactions (phase 2.2). For the use case, the bottlenecks are the infrastructure setup (phase 1) and receipt emission phases (phase 2.1). We observe that the infrastructure setup

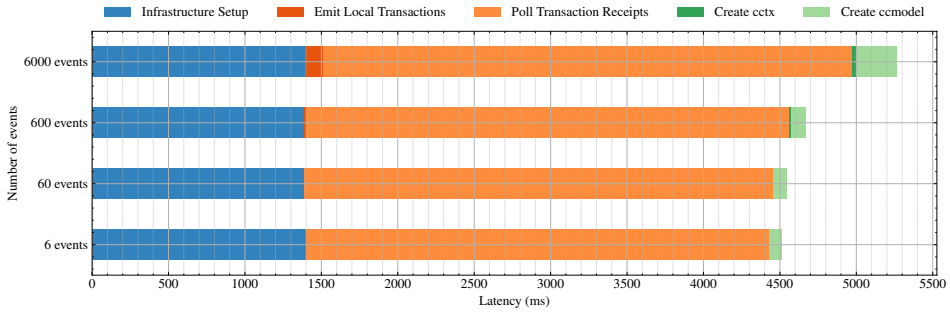


Fig. 5. Latency for each phase of the baseline test scenario for a variable number of events.

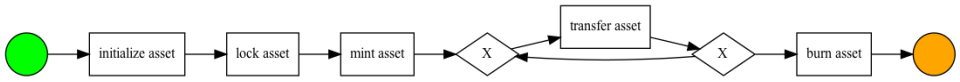


Fig. 6. Generated business process modelling notation model for the events generated in the baseline phase.

and transaction emission phases occupy around 97%, 98%, and practically 100% of the execution time, depending on if we are emitting 6,60, or 600 events, respectively. This is expected, and we conclude the bottleneck is the transaction execution and commitment. In the dummy use case creating receipts is a cheaper task than retrieving them; in this use case, the inverse happens because executing transactions on blockchains is generally an expensive task. In a production environment, the cross-chain throughput is limited by the finality speed of the underlying systems: the infrastructural part of `Hephaestus` is efficient in issuing the transactions and retrieving the respective receipts. Varying the number of domains/blockchains should not affect the creation of *cctx* as all transaction receipts are interpreted as *ccevent*s. However, a varying number of domains might influence the overall latency of the system depending on their transaction generation rate. In other words, the faster a domain is, the faster the "Emit local transactions" phase will be, and consequently, the faster *cctx*s and *ccstate* will be processed. The latency of a cross-chain transaction $cctx_i$, denoted by $\delta(cctx_i)$, will be the sum of the individual latencies (in case local transactions are sequential), or bounded by the slowest domain (paralellized): $\delta(cctx_i) = \max\{\delta(d_1), \ldots, \delta(d_n)\}$. Of course, this will depend on the specific cross-chain logic, as there are cases where some local transactions can be parallelized and others not. We leave experiments on real-world bridges, with a variable number of domains for future work.

The complexity of transforming the receipts into events may vary significantly, but our experiments show a very low overhead. Furthermore, the setup phase only needs to be performed once. We conclude that our system is scalable in terms of latency and extensibility.

D. Preventing Cross-Chain Double-Spends

In this section, we run experiments that allow us to evaluate if `Hephaestus` can detect deviations from expected behavior, namely detecting double-spends.

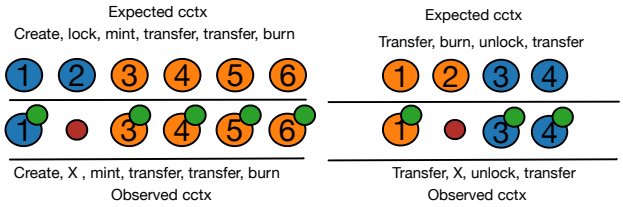


Fig. 7. Double spend detection. Figure a) shows a double spend direction source-to-target blockchain. Figure b) shows a double spend, direction target-to-source blockchain.

Expected behavior, or the specification, is given by the events we emit. After generating the *ccmodel*, we generate another set of events, this time in the following order: create asset, mint asset, transfer asset, burn asset. Note that the lock asset phase is not present - this emulates a user attempting to mint an asset without an appropriate lock (double spend case one). This mechanism models what happened in real-world bridge hacks, namely the PolyNetwork bridge (using a truncated function signature hash collisions and forced transaction inclusions) and the Meter bridge (via inconsistent deposit logic) [43]. Events originate on the source chain, or target chain. Events can lead to a *SYNC MOVE* or *MOVE ON LOG*. The latter indicates the point in the cross-chain model where an attack has happened.

Figure 7 shows the detection of double spending. We obtain detailed alignment information about transitions that did not execute correctly, namely the mint before the lock, according to Figure 7a). While it is possible to obtain a set of detailed metrics such as throughput and cost analysis of real-time flows, we stick to the conformance of the process (the fitness) for the sake of space. Fitness is a simple metric used in process mining that consists in measuring the ratio between *SYNC MOVE*s and *MOVE ON LOG*s. We obtain that a mint should occur after it has occurred (`('MintAsset', '>>')`), and fitness 82%. The trace generated by our implementation

and the expected traces differ, originating a *MOVE ON LOG* 🔴. We leave the study of applying different process mining algorithms to create cross-chain models of different real-world interoperability systems for future work. Along with this future work, we will study the different trade-offs between algorithms, including performance considerations (using metrics such as recall, precision, accuracy, F-Score, and error rate). Further research is needed to understand the optimal algorithms for each bridge and user cohort.

Double-spends can also happen in the contrary direction (Figure 7b). After a user bridged an asset, they can recover it by burning the bridged asset on the target blockchain and unlocking the original asset on the source blockchain. Double spending occurs when a false proof-of-burn is provided, making the user maintain the bridged token and the original token. This attack happened on the Polygon/Matic bridge (via incorrect proof-of-burn verification).

As soon as double-spends are detected, several defense mechanisms can be activated [43], limiting the scope of the attacks. If assets are frozen in due course, double spending can be not only alleviated but prevented. In conclusion, our system can detect several types of attacks directly mappable to real-world occurrences, without loss of generality.

E. Discussion

`Hephaestus` contributes to mitigating bridge hacks by 1) generating a *ccmodel* of the bridge protocols, allowing reasoning about the protocol flow, bottlenecks, and possible threats and vulnerabilities, and 2) minimizing the attack consequences by finding active monitoring and detecting suspicious behavior in real-time. Cross-chain models allow expressing complex cross-chain logic without having the protocol designer focus on timeouts, missing or corrupted information, and the technicalities of ad-hoc protocols. This allows the designer to focus instead on the business logic and its monitoring and achieve a separation of concerns.

Our tool can be extended to incorporate an incident framework that is activated upon detection of a *MOVE ON LOG* (e.g., freeze certain types of transactions), according to what is starting to be explored in the industry [42]. For this, adequate *ccmodel* representations are needed. We generate BPMN models, that are good for expressing the semantics of a cross-chain use case graphically, but research on what is an appropriate representation of cross-chain processes is still lacking. An important assumption is that the model is complete, i.e., models all the desired behavior. However, this is not always the case, and some *MOVE ON LOG* events can be false negatives. Creating robust models that tolerate noise and evaluating those models is an evolving, core challenge in the process mining area that would have repercussions in generating and maintaining *ccmodel*s [28]. A consideration of cross-chain security is that cross-chain models are deemed correct when certain predicates on incoming events are satisfied. To this end, designing the rules is of utmost importance, likely to be an interactive process involving different stakeholders; another consideration is that sometimes the events that are checked against rules are not controlled by the bridge operators - leaving a wide attack surface for hackers. Regarding privacy, there are multiple views on the interoperability literature [46]–[49]. It seems that the main property for cross-chain transactions is unlinkability: the inability of an external observer to link the lock to the mint transactions. However, our system does not provide asset transfer privacy. Further investigation on the security and privacy of *ccmodel*s is needed.

Finally, our system is modular - new blockchains can easily be supported. It has the potential to be integrated into cross-chain APIs for such purposes. The possibility of retrieving the cumulative metrics for all *cctx*s processed in the *ccmodel* allows enhanced and fine-grain monitoring of cross-chain logic. For example, the revenue and cost parameters can be adjusted according to the use case, allowing semantically enriching each transaction. Associating cost and revenue values to transaction receipts would help calculate capital profit taxes for a certain jurisdiction, for instance.

VIII. Related Work

`Hephaestus` is the result of an inter-disciplinary work that combines the fields of blockchain interoperability, on-chain analytics, and process mining applied to the blockchain.

A. Bridge Hacks and Monitoring

Lee et al. explored a systematization of cross-chain bridge hacks that supports our modeling of double spend [43]. Although some work on bridge security has been done recently [46], [49]–[54], the space still needs systematization. Our work puts forward the cross-chain model, a concept that unites several efforts in the area. The work by Zhang et al. [50] seems to be the most similar to ours. The authors create a tool to identify miss-conformance in the lock-unlock bridge mechanism. However, this work is directed specifically at bridges and not arbitrary cross-chain use cases. On the other hand, BUNGEE is a general-agnostic framework that inspires this work. In this paper, a tool that produces consolidated views over user activity on different blockchains [2] is proposed. `Hephaestus` can complement BUNGEE to generate metrics, protocol behavior patterns, and individual user activity. `Hephaestus` can be deployed over interoperability protocols such as ODAP/SAT [23], [55], [56], XCLAIM [37], and many others [1], to provide a monitoring layer.

B. On-chain Analytics

In the field of on-chain analytics, some industry solutions exist and are well-adopted: the Dune tool allows to Explore, create and share crypto analytics, including key metrics for DeFi, NFTs, and more, expressive queries, and the visualization of information in dashboards. `Hephaestus` would allow for the creation of a cross-chain Dune tool by cross-referencing transactions in multiple chains [57]. Chainanalysis provides a dashboard for investigation, compliance, and risk management tools to assert compliance with jurisdictions and fight fraud and illicit activities [58]. For example, it allows one to visualize the flow of funds and track movements across currencies. Our tool would provide possibilities to port

this monitoring for the cross-chain scenario. Metla finance [59], Morali [60], and Rokti [61] allows a unified view of user assets over different blockchains. Hephaestus would allow extending the views to support arbitrary states across blockchains. Certik provides a monitoring layer for analyzing and monitoring blockchain protocols and DeFi projects, but only from a security perspective [62]. Token Flow is the closest work to ours, an analytics tool to track cross-chain asset transfers [63]. However, Token Flow does not support arbitrary cross-chain use cases. On the academic side, we have several tools that allow on-chain analysis of smart contracts for security purposes [40], [64], [65], performance [66], [66], [67], compliance and anti-fraud [68], and others [69], [70] . However, such projects provide a sort of meta-view over user activity, do not provide specific information about interaction with protocols, and are not generalizable, contrarily to this work.

C. Process mining on blockchain

In the process mining area, some work has been done to apply it to the blockchain. In [71], the authors used process mining techniques to specify the behavior of the Augur protocol, discovering bottlenecks and proposing improvements. Some tools to automatize the creation of process models from blockchain protocols to facilitate multiple goals have been proposed [72]–[75], but none for the cross-chain scenario.

IX. Concluding Remarks

The need for multi-chain applications introduces additional challenges to end-users and developers, where we emphasize new attack vectors and a large attack surface. The exploitation of these threats leads to large-scale attacks on cross-chain bridges. We propose Hephaestus to address this problem.

Hephaestus generates cross-chain models from observed cross-chain events. By associating a set of transactions with a set of rules, transactional flow can be monitored and, therefore, deviations from the idealized process can be detected. This allows to timely act upon suspicious behavior that can indicate an attack. We implemented Hephaestus, several blockchain connectors, test ledgers, and a workload generator. Our evaluation includes creating a cross-chain use case on asset transfers (i.e., a bridge) composed of a pair of smart contracts and cross-chain logic. We tested our system with variable workloads to assess the performance and reliability of Hephaestus. We conclude that we have low latency in generating *ccmodel*s for the given use case and that our tool can scale with the number of blockchains and *cctx*s.

We pave the way to enable a better user experience for the end user and protocol operators by enabling the analysis, monitoring, and optimization of *ccmodel*s. In particular, Hephaestus can be applied over established blockchain interoperability protocols, serving as a monitoring and audit layer, providing better response capacity and thus enhanced proactive security. Use cases such as reconfiguring wallets across chains, better user interfaces for fund tracking across different chains, managing additional base layer tokens for gas, doing tax reports, and analyzing cross-chain maximal extractable value do not need to be complicated.

Acknowledgments

We thank the anonymous reviewers for the suggestions that immensely helped improve this paper's quality. This project was partially supported by *The Linux Foundation* as part of the *Hyperledger Summer Internships* program under the *Visualization and Analysis of Cross-chain Transactions* project. We thank Iulia Mihaiu for contributing with an initial exploration of the concepts in this paper. We thank André Augusto, Kevin Liao, Sabrina Scuri, and Nuno Nunes for suggestions that improved this paper. This work was partially supported by national funds through Fundação para a Ciência e a Tecnologia (FCT) with reference UIDB/50021/2020 (INESC-ID) and 2020.06837.BD, by the European Commission through contract 952226 (BIG), and by project nr.51 "BLOCKCHAIN.PT - Agenda Descentralizar Portugal com Blockchain", financed by European Funds, namely "Recovery and Resilience Plan - Component 5: Agendas Mobilizadoras para a Inovação Empresarial", included in the NextGenerationEU funding program.

References

[1] R. Belchior, A. Vasconcelos, S. Guerreiro, and M. Correia, "A Survey on Blockchain Interoperability: Past, Present, and Future Trends," *ACM Computing Surveys*, vol. 54, no. 8, pp. 1–41, May 2021. [Online]. Available: http://arxiv.org/abs/2005.14282

[2] R. Belchior, L. Torres, J. Pfannschmid, A. Vasconcelos, and M. Correia, "Is My Perspective Better Than Yours? Blockchain Interoperability with Views." TechRxiv, Jun. 2022. [Online]. Available: https://www.techrxiv.org/articles/preprint/Is_My_Perspective_Better_Than_Yours_Blockchain_Interoperability_with_Views/20025857/1

[3] B. Pillai, K. Biswas, Z. Hóu, and V. Muthukkumarasamy, "Cross-blockchain technology: integration framework and security assumptions," *IEEE Access*, 2022, publisher: IEEE.

[4] P. Robinson, "Survey of crosschain communications protocols," *Computer Networks*, vol. 200, p. 108488, Dec. 2021, publisher: Elsevier. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1389128621004321

[5] R. Belchior, J. Süßenguth, Q. Feng, T. Hardjono, A. Vasconcelos, and M. Correia, "A Brief History of Blockchain Interoperability," 6 2023. [Online]. Available: https://www.techrxiv.org/articles/preprint/A_Brief_History_of_Blockchain_Interoperability/23418677

[6] H. Qureshi, "Axelar, Bridges, and Blockchain Globalization," Jun. 2022. [Online]. Available: https://medium.com/dragonfly-research/axelar-bridges-and-blockchain-globalization-11ef3bbce9f1

[7] Ethereum Engineering Group, "Security of Crosschain Transactions and Bridges," Nov. 2022. [Online]. Available: https://www.youtube.com/watch?v=DJyEJVaXMNo

[8] D. Berenzon, "Blockchain Bridges," Sep. 2021. [Online]. Available: https://medium.com/1kxnetwork/blockchain-bridges-5db6afac44f8

[9] P. KidBold, "The Wormhole Bridge Attack Explained," Feb. 2022. [Online]. Available: https://kaicho.substack.com/p/the-wormhole-bridge-attack-explained

[10] C. Faife, "Wormhole cryptocurrency platform hacked for $325 million after error on GitHub," Feb. 2022. [Online]. Available: https://www.theverge.com/2022/2/3/22916111/wormhole-hack-github-error-325-million-theft-ethereum-solana

[11] Rekt, "Rekt - THORChain," 2022. [Online]. Available: https://www.rekt.news/

[12] R. Behnke, "Explained: The Wormhole Hack (February 2022)," Feb. 2022. [Online]. Available: https://halborn.com/explained-the-wormhole-hack-february-2022/

[13] FreddieChopin, "FYI, the hacker who exploited Harmony bridge for 100 M$ 3 days ago has already started sending stolen ETH to Tornado Cash mixer," Jun. 2022. [Online]. Available: www.reddit.com/r/CryptoCurrency/comments/vlt4xs/fyi_the_hacker_who_exploited_harmony_bridge_for/

[14] M. Barrett, "Harmony's Horizon Bridge Hack," Jun. 2022. [Online]. Available: https://medium.com/harmony-one/harmonys-horizon-bridge-hack-1e8d283b6d66

[15] C. Faife, "Nomad crypto bridge loses $200 million in "chaotic" hack," Aug. 2022. [Online]. Available: https://www.theverge.com/2022/8/2/23288785/nomad-bridge-200-million-chaotic-hack-smart-contract-cryptocurrency

[16] R. News. Multichain hacked for the third time - R3KT news. rekt. [Online]. Available: https://www.rekt.news/

[17] The Block Research, "Largest DeFi exploits," 2022. [Online]. Available: https://www.theblock.co/data/decentralized-finance/exploits/largest-defi-exploits

[18] Zach, Ally, "A Year of Bridge Exploits," Aug. 2022. [Online]. Available: https://messari.io/report/a-year-of-bridge-exploits

[19] The Straits Times, "Cryptocurrency-bridge hacks top $1.36 billion in little over a year," *The Straits Times*, Apr. 2022. [Online]. Available: https://www.straitstimes.com/tech/tech-news/cryptocurrency-bridge-hacks-top-136-billion-in-little-over-a-year

[20] N. Team, "The Road to Recovery," Aug. 2022. [Online]. Available: https://medium.com/nomad-xyz-blog/the-road-to-recovery-6abe5eec8ff1

[21] V. Buterin, "Vitalik Buterin on cross-chain bridges," 2022. [Online]. Available: www.reddit.com/r/ethereum/comments/rwojtk/ama_we_are_the_efs_research_team_pt_7_07_january/hrngyk8/

[22] D. Avrilionis and T. Hardjono, "Towards Blockchain-enabled Open Architectures for Scalable Digital Asset Platforms," Oct. 2021, publisher: ArXiv. [Online]. Available: https://www.scienceopen.com/document?vid=c60d84b9-911e-45a5-ab92-864ee24ec771

[23] R. Belchior, A. Vasconcelos, M. Correia, and T. Hardjono, "HERMES: Fault-Tolerant Middleware for Blockchain Interoperability," *Future Generation Computer Systems*, Mar. 2021.

[24] C. Ko, M. Ruschitzka, and K. Levitt, "Execution monitoring of security-critical programs in distributed systems: a specification-based approach," in *Proceedings. 1997 IEEE Symposium on Security and Privacy (Cat. No.97CB36097)*, May 1997, pp. 175–187, iSSN: 1081-6011.

[25] H. Montgomery, H. Borne-Pons, J. Hamilton, M. Bowman, P. Somogyvari, S. Fujimoto, T. Takeuchi, T. Kuhrt, and R. Belchior, "Hyperledger Cactus Whitepaper," Hyperledger Foundation, Tech. Rep., 2020. [Online]. Available: https://github.com/hyperledger/cactus/blob/master/docs/whitepaper/whitepaper.md

[26] L2BEAT – The state of the layer two ecosystem. [Online]. Available: https://l2beat.com/scaling/summary

[27] R. Belchior, S. Guerreiro, A. Vasconcelos, and M. Correia, "A survey on business process view integration: past, present and future applications to blockchain," *Business Process Management Journal*, vol. ahead-of-print, no. ahead-of-print, Jan. 2022. [Online]. Available: https://doi.org/10.1108/BPMJ-11-2020-0529

[28] W. Van Der Aalst, "Process mining: Overview and opportunities," *ACM Transactions on Management Information Systems (TMIS)*, vol. 3, no. 2, pp. 1–17, 2012, publisher: ACM New York, NY, USA.

[29] J. Küster, K. Ryndina, and H. Gall, "Generation of business process models for object life cycle compliance," in *International Conference on Business Process Management*, vol. 4714 LNCS. Springer, Berlin, 2007, pp. 165–181.

[30] R. M. Dijkman, M. Dumas, and C. Ouyang, "Semantics and analysis of business process models in BPMN," *Information and Software Technology*, vol. 50, no. 12, pp. 1281–1294, 2008. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950584908000323

[31] R. Belchior, L. Torres, J. Pfannschmid, A. Vasconcelos, and M. Correia, "Can We Share the Same Perspective? Blockchain Interoperability with Views," Oct. 2022. [Online]. Available: https://www.techrxiv.org/articles/preprint/Is_My_Perspective_Better_Than_Yours_Blockchain_Interoperability_with_Views/20025857/3

[32] J. Garay, A. Kiayias, and N. Leonardos, "The Bitcoin backbone protocol: Analysis and applications," in *Advances in Cryptology*, vol. 9057, 2015, pp. 281–310, iSSN: 16113349.

[33] R. Belchior, L. Riley, T. Hardjono, A. Vasconcelos, and M. Correia, "Do You Need a Distributed Ledger Technology Interoperability Solution?" *Distributed Ledger Technologies: Research and Practice*, Sep. 2022, just Accepted. [Online]. Available: https://doi.org/10.1145/3564532

[34] J. Han, H. E, G. Le, and J. Du, "Survey on NoSQL database," in *2011 6th International Conference on Pervasive Computing and Applications*, 2011, pp. 363–366.

[35] P. Alvaro, W. R. Marczak, N. Conway, J. M. Hellerstein, D. Maier, and R. Sears, "Dedalus: Datalog in Time and Space," in *Datalog Reloaded*, ser. Lecture Notes in Computer Science, O. de Moor, G. Gottlob, T. Furche, and A. Sellers, Eds. Berlin, Heidelberg: Springer, 2011, pp. 262–281.

[36] T. J. Green, S. S. Huang, B. T. Loo, and W. Zhou, "Datalog and Recursive Query Processing," *Foundations and Trends® in Databases*, vol. 5, no. 2, pp. 105–195, Nov. 2013, publisher: Now Publishers, Inc. [Online]. Available: https://www.nowpublishers.com/article/Details/DBS-017

[37] A. Zamyatin, D. Harz, J. Lind, P. Panayiotou, A. Gervais, and W. Knottenbelt, "Xclaim: Trustless, interoperable, cryptocurrency-backed assets," in *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019, pp. 193–210.

[38] A. Zamyatin, M. Al-Bassam, D. Zindros, E. Kokoris-Kogias, P. Moreno-Sanchez, A. Kiayias, and W. J. Knottenbelt, "Sok: Communication across distributed ledgers," in *International Conference on Financial Cryptography and Data Security*. Springer, 2021, pp. 3–36.

[39] I. Mihaiu, R. Belchior, S. Scuri, and N. Nunes, "A Framework to Evaluate Blockchain Interoperability Solutions," TechRxiv, Tech. Rep., Dec. 2021. [Online]. Available: https://www.techrxiv.org/articles/preprint/A_Framework_to_Evaluate_Blockchain_Interoperability_Solutions/17093039

[40] B. Putz and G. Pernul, "Detecting Blockchain Security Threats," in *2020 IEEE International Conference on Blockchain (Blockchain)*, Nov. 2020, pp. 313–320.

[41] C. Page, "Binance hit by $100 million blockchain bridge hack," Oct. 2022. [Online]. Available: https://techcrunch.com/2022/10/07/blockchain-bridge-hack/

[42] Chainlink, "Cross-Chain Interoperability Protocol (CCIP) | Chainlink," 2022. [Online]. Available: https://chain.link/cross-chain

[43] S.-S. Lee, A. Murashkin, M. Derka, and J. Gorzny, "SoK: Not Quite Water Under the Bridge: Review of Cross-Chain Bridge Hacks," Oct. 2022, arXiv:2210.16209 [cs]. [Online]. Available: http://arxiv.org/abs/2210.16209

[44] A. Berti, S. J. Van Zelst, and W. van der Aalst, "Process mining for python (PM4Py): bridging the gap between process-and data science," *arXiv preprint arXiv:1905.06169*, 2019.

[45] W. M. van der Aalst and A. Berti, "Discovering object-centric Petri nets," *Fundamenta informaticae*, vol. 175, no. 1-4, pp. 1–40, 2020, publisher: IOS Press.

[46] T. Haugum, B. Hoff, M. Alsadi, and J. Li, "Security and Privacy Challenges in Blockchain Interoperability - A Multivocal Literature Review," in *Proceedings of the International Conference on Evaluation and Assessment in Software Engineering 2022*, ser. EASE '22. New York, NY, USA: Association for Computing Machinery, Jun. 2022, pp. 347–356. [Online]. Available: https://doi.org/10.1145/3530019.3531345

[47] A. Deshpande and M. Herlihy, "Privacy-Preserving Cross-Chain Atomic Swaps," in *Financial Cryptography and Data Security*, ser. Lecture Notes in Computer Science, M. Bernhard, A. Bracciali, L. J. Camp, S. Matsuo, A. Maurushat, P. B. Rønne, and M. Sala, Eds. Cham: Springer International Publishing, 2020, pp. 540–549.

[48] Z. Yin, B. Zhang, J. Xu, K. Lu, and K. Ren, "Bool Network: An Open, Distributed, Secure Cross-Chain Notary Platform," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 3465–3478, 2022, conference Name: IEEE Transactions on Information Forensics and Security.

[49] J. Wang, J. Cheng, Y. Yuan, H. Li, and V. S. Sheng, "A Survey on Privacy Protection of Cross-Chain," in *Advances in Artificial Intelligence and Security*, ser. Communications in Computer and Information Science, X. Sun, X. Zhang, Z. Xia, and E. Bertino, Eds. Cham: Springer International Publishing, 2022, pp. 283–296.

[50] J. Zhang, J. Gao, Y. Li, Z. Chen, Z. Guan, and Z. Chen, "Xscope: Hunting for Cross-Chain Bridge Attacks," Aug. 2022, arXiv:2208.07119 [cs]. [Online]. Available: http://arxiv.org/abs/2208.07119

[51] Y. Zhang, Z. Ge, Y. Long, and D. Gu, "UCC: Universal and Committee-based Cross-chain Framework," in *Information Security Practice and Experience*, ser. Lecture Notes in Computer Science, C. Su, D. Gritzalis, and V. Piuri, Eds. Cham: Springer International Publishing, 2022, pp. 93–111.

[52] T. Xie, J. Zhang, Z. Cheng, F. Zhang, Y. Zhang, Y. Jia, D. Boneh, and D. Song, "zkBridge: Trustless Cross-chain Bridges Made Practical," Oct. 2022, arXiv:2210.00264 [cs]. [Online]. Available: http://arxiv.org/abs/2210.00264

[53] C. Pedreira, R. Belchior, M. Matos, and A. Vasconcelos, "Securing Cross-Chain Asset Transfers on Permissioned Blockchains," Jun. 2022. [Online]. Available: https://www.techrxiv.org/articles/preprint/Trustable_Blockchain_Interoperability_Securing_Asset_Transfers_on_Permissioned_Blockchains/19651248/3

[54] A. Augusto, R. Belchior, A. Vasconcelos, and T. Hardjono, "Resilient Gateway-Based N-N Cross-Chain Asset Transfers," Nov. 2022. [Online]. Available: https://www.techrxiv.org/articles/preprint/Resilient_Gateway-Based_N-N_Cross-Chain_Asset_Transfers/20016815/2

[55] M. Hargreaves, T. Hardjono, and R. Belchior, "Open Digital Asset Protocol draft 02," Internet Engineering Task Force, Tech.

Rep., 2021. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-hargreaves-odap-02
[56] R. Belchior, M. Correia, and T. Hardjono, "Gateway Crash Recovery Mechanism draft v1," IETF, Tech. Rep., 2021. [Online]. Available: https://datatracker.ietf.org/doc/draft-belchior-gateway-recovery/
[57] "Dune." [Online]. Available: https://dune.com/home
[58] Chainanalysis, "The Blockchain Data Platform - Chainalysis," 2022. [Online]. Available: https://www.chainalysis.com/
[59] Metla, "Metla - the ultimate crypto dashboard," 2022. [Online]. Available: https://metla.com/
[60] Moralis, "Moralis The Web3 Development Workflow," 2022. [Online]. Available: https://moralis.io/
[61] Rokti, "Rokti portfolio tracker," 2022. [Online]. Available: https://rotki.com
[62] Certik, "CertiK Blockchain Security Leaderboard," 2022. [Online]. Available: https://www.certik.com
[63] Token Flow, "Token Flow Insights," 2022. [Online]. Available: https://tokenflow.live
[64] L. Luu, D.-H. Chu, H. Olickel, P. Saxena, and A. Hobor, "Making smart contracts smarter," in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, 2016, pp. 254–269.
[65] B. Putz, F. Böhm, and G. Pernul, "HyperSec: Visual Analytics for blockchain security monitoring," in *IFIP International Conference on ICT Systems Security and Privacy Protection*. Springer, 2021, pp. 165–180.
[66] P. Zheng, Z. Zheng, X. Luo, X. Chen, and X. Liu, "A Detailed and Real-Time Performance Monitoring Framework for Blockchain Systems," in *2018 IEEE/ACM 40th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP)*, May 2018, pp. 134–143.
[67] M. Bartoletti, S. Lande, L. Pompianu, and A. Bracciali, "A general framework for blockchain analytics," in *Proceedings of the 1st Workshop on Scalable and Resilient Infrastructures for Distributed Ledgers*, 2017, pp. 1–6.
[68] D. N. Dillenberger, P. Novotny, Q. Zhang, P. Jayachandran, H. Gupta, S. Hans, D. Verma, S. Chakraborty, J. Thomas, M. Walli, and others, "Blockchain analytics and artificial intelligence," *IBM Journal of Research and Development*, vol. 63, no. 2/3, pp. 5–1, 2019, publisher: IBM.
[69] N. Tovanich, N. Soulié, N. Heulot, and P. Isenberg, "An Empirical Analysis of Pool Hopping Behavior in the Bitcoin Blockchain," in *2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, May 2021, pp. 1–9.
[70] B. Nasrulin, M. Muzammal, and Q. Qu, "Chainmob: Mobility analytics on blockchain," in *2018 19th IEEE International Conference on Mobile Data Management (MDM)*. IEEE, 2018, pp. 292–293.
[71] R. Hobeck, C. Klinkmüller, H. Bandara, I. Weber, and W. M. van der Aalst, "Process mining on blockchain data: a case study of Augur," in *International conference on business process management*. Springer, 2021, pp. 306–323.
[72] M. Müller and P. Ruppel, "Process Mining for Decentralized Applications," in *2019 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPCON)*, Apr. 2019, pp. 164–169.
[73] C. Klinkmüller, A. Ponomarev, A. B. Tran, I. Weber, and W. van der Aalst, "Mining Blockchain Processes: Extracting Process Mining Data from Blockchain Applications," in *Business Process Management: Blockchain and Central and Eastern Europe Forum*, ser. Lecture Notes in Business Information Processing, C. Di Ciccio, R. Gabryelczyk, L. García-Bañuelos, T. Hernaus, R. Hull, M. Indihar Štemberger, A. Kő, and M. Staples, Eds. Cham: Springer International Publishing, 2019, pp. 71–86.
[74] R. Mühlberger, S. Bachhofner, C. Di Ciccio, L. García-Bañuelos, and O. López-Pintado, "Extracting Event Logs for Process Mining from Data Stored on the Blockchain," in *Business Process Management Workshops*, ser. Lecture Notes in Business Information Processing, C. Di Francescomarino, R. Dijkman, and U. Zdun, Eds. Cham: Springer International Publishing, 2019, pp. 690–703.
[75] F. Corradini, F. Marcantoni, A. Morichetta, A. Polini, B. Re, and M. Sampaolo, "Enabling Auditing of Smart Contracts Through Process Mining," in *From Software Engineering to Formal Methods and Tools, and Back: Essays Dedicated to Stefania Gnesi on the Occasion of Her 65th Birthday*, ser. Lecture Notes in Computer Science, M. H. ter Beek, A. Fantechi, and L. Semini, Eds. Cham: Springer International Publishing, 2019, pp. 467–480. [Online]. Available: https://doi.org/10.1007/978-3-030-30985-5_27

Rafael Belchior is a researcher at INESC-ID (Distributed Systems Group) and a Ph.D. student at Técnico Lisboa. Studying the intersection of blockchain interoperability and security, he focuses on enabling interoperability across heterogeneous systems and building resilient blockchain infrastructure.

Peter Somogyvari is a technology architect manager at Accenture, where he is one of the maintainers of the top-level Hyperledger project called Cacti, which aims to be an enterprise-grade framework for blockchain integration/interoperability. He has spoken at Hyperledger Global Forum in 2020 and 2021 and several other technology conferences.

Jonas Pfannschmidt has more than 15 years of professional experience in software engineering with a focus on Blockchain, Cloud, and Financial Services. In his current roles as Principal Blockchain Engineer, R&D lead, and Director of Blockdaemon Ltd. he builds institutional-grade Blockchain infrastructure to stake, deploy, and scale leading Blockchain networks.

André Vasconcelos is Assistant Professor (Professor Auxiliar) in the Department of Computer Science and Engineering, Instituto Superior Tecnico, Lisbon University, and researcher in Information and Decision Support Systems Lab at INESC-ID, in Enterprise Architecture domains, namely representation and modeling of Architectures of Information Systems, and Evaluation of Information Systems Architectures.

Miguel Correia is a Full Professor at Instituto Superior Tecnico (IST), Universidade de Lisboa, in Lisboa, Portugal. He is vice president for faculty at DEI. He is the coordinator of the Doctoral Program in Information Security at IST. He is a senior researcher at INESC-ID, and a member of the Distributed Systems Group (GSD). He is co-chair of the European Blockchain Partnership that is designing the European Blockchain Services Infrastructure (EBSI).

7 Conclusions

This chapter concludes this thesis. Section 7.1 provides evidence of how the developed work validates the thesis statement. Section 7.2 summarizes and discusses our contributions and respective implications on the academia, industry, and society. Finally, we present open challenges and future research directions for the next generation of robust IMs.

Figure 7.1: Blockchain city allegory: each building is a blockchain, interconnected by blockchain interoperability protocols, the "roads". An interconnected ecosystem will sprout synergies that might change the course of the technology. Generated by DALL-E-3. Prompt: Vector image of the futuristic cityscape with the unique blockchain-shaped buildings.

7.1 Thesis Hypothesis Validation

Based on the theoretical and empirical observations collected from the contributions posed throughout this thesis, this section aims to test the proposed hypothesis: ***IMs providing interoperability across the technical, semantic, and organizational layers can securely implement the requirements of both centralized and decentralized organizations***. For this, let us focus on Table 7.1, which maps the research questions, respective associated requirements, contributions, and respective publications, serving as a reference point for reasoning about the thesis hypothesis:

RQ1 - How to assess the interoperability capabilities of an interoperability mechanism? - Addressed by **P1** [100]
R1.1 - Systematic classification of IMs
C1 - A unified conceptual model and classification framework for blockchain interoperability solutions;
R1.2 - Robust assessment of IMs
C2 - A framework to assess the interoperability capabilities of a system utilizing multiple DLTs [...]
R1.3 - Framework to choose an IM based on use case requirements
C3 - A pair of decision models that allow one to choose a blockchain interoperability solution [...]
RQ2 - How can IMs provide technical and semantic interoperability (and provide the basis for organizational interoperability)? - Addressed by **P2** [4], **P3** [83]
R2.1 - Privacy-preserving standardized data format for accountable interoperability;
C4 - A common data format for accountable and privacy-preserving interoperability - blockchain views.
R2.2 - Gateway-based interoperability framework and architecture;
R2.3 - Gateway reliability, decentralization security and privacy assessment
C5 - Blockchain gateway paradigm as the enterprise-grade IM for auditable, private, and reliable interoperability
R2.4 - Gateway interoperability and standardization
R2.5 - GuaranteeACIDC properties for gateway-operated transactions.
C6 - Technical architecture, protocols, and algorithms to guarantee ACIDC properties for transactions
RQ3 - How to add robustness to blockchain interoperability mechanisms from a security perspective? - Addressed by **P4** [151], **P5** [104]
R3.1 - Reduction of the attack surface for IMs and cross-chain applications
R3.2 - Secure decentralization of the IM
C7 - Protocols, algorithms [...] for decentralized, secure blockchain interoperability based on [...] SNARKs
R3.3 - Strategy for monitoring attacks to cross-chain application
C8 - Protocols, algorithms [...] for continuous monitoring and incident response of blockchain bridges.

Table 7.1: Contributions of this thesis per research question, respective requirements, and associated publications. Note that C5 addresses R2.2 and R2.3. Likewise, C6 addresses R2.4 and R2.5; C7 addresses R3.1 and R3.2.

Taking into consideration the addressed research questions, let us focus on the following views on the contributions of our work:

- in order to address the problem space and the **answer the formulated research questions**, we proposed theoretical artifacts (e.g., principles, conceptual models, classification frameworks, technical architectures, protocols), contributing to the theory of blockchain interoperability. Moreover, we provided a solution space that yielded practical contributions (e.g., algorithms, protocols, assessments, protocol instantiations, code), showing that the associated requirements can be satisfied in practice.
- evidence showcasing **the improvement of the status quo**. We proposed theoretical and practical contributions that show considerable improvements over the state of the art. We provided the first systematic framework to classify IMs, in Chapter 2. We proposed the blockchain view concept as a standard data format for blockchain interoperability, the first to cater to both centralized and decentralized systems, in Chapter 3. We proposed `Hermes` as a reliable and secure gateway-based system, catered to satisfy technical and semantic interoperability while providing the basis for organizational interoperability. `Hermes` is suitable for enterprise-grade needs (Chapter 4). We developed `Harmonia`, showcasing its advantages against what was previously the state-of-the-art frameworks for developing secure cross-chain applications suitable for permissionless blockchain use cases (Chapter 5). We developed the first method based on process mining to protect blockchain bridges, in Chapter 6.
- from the research *corpus*, we derive **practical implications** to the academia and the industry. We demonstrated the possibility of blockchain interoperability to be securely realized across heterogeneous infras-

tructure (centralized systems, private decentralized systems, public decentralized systems) with the hybrid blockchain approach. In particular, this category of solutions is enabled by the gateway paradigm, the first blockchain interoperability paradigm tailored for enterprise-grade integrations. Likewise, a new SNARK-based framework to build cross-chain applications, anchored by the cryptography behind zero-knowledge proofs enables connection to permissionless infrastructure without the disadvantage of centralization. Finally, we proposed new methods for vulnerability discovery and attack mitigations on cross-chain applications (with a focus on cross-chain bridges), in order to protect users that use hybrid applications.

We show that the proposed contributions satisfy the formulated research questions, improve the state-of-the-art, and unravel significant implications. Therefore, we believe these groups of arguments provide strong evidence to support the thesis statement. As a result, we expect increasing attention given to this field of research, the development and enhancement of incident response infrastructure, the development of organizational and legal interoperability in DLTs, and the flourishing of new use cases using hybrid blockchain applications, particularly where the thesis statement is verified.

7.2 Thesis Contributions and Implications

In the course of this thesis, we addressed different challenges associated with blockchain interoperability. Figure 7.2 showcases the technical architecture and their interdependencies of the solutions developed in this dissertation, following the C4 model [152]. The C4 model complements classical UML diagrams. C4 is known for its simplicity and expressiveness [153].

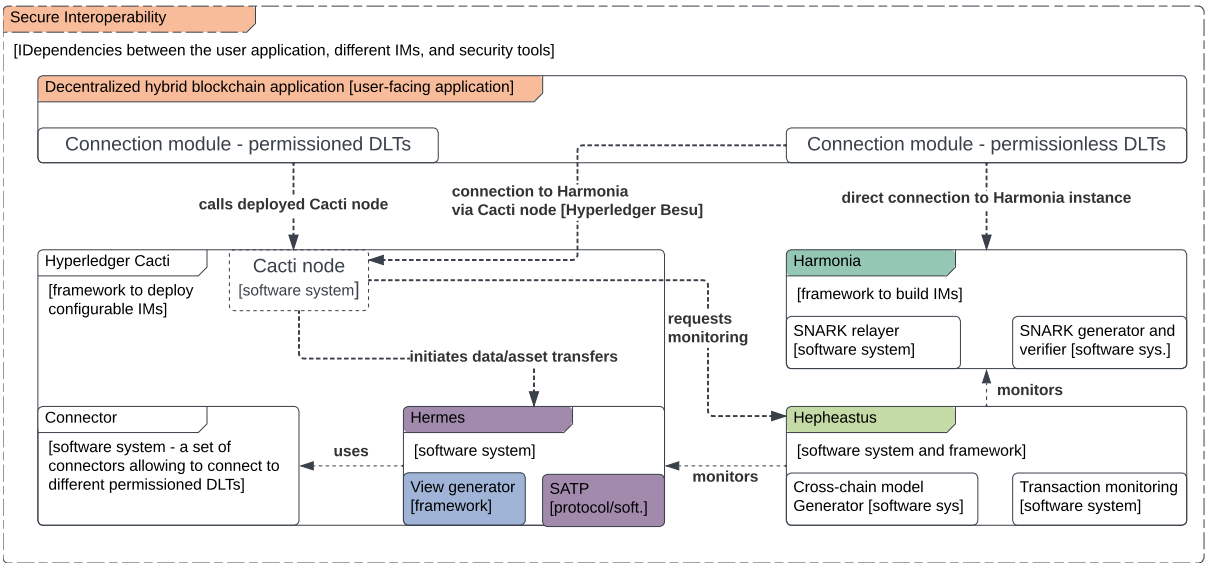


Figure 7.2: C4 model of the solutions developed in this dissertation and their dependencies.

The overall proposed set of systems to realize "secure interoperability" subsumes a set of five main components: `Hermes` [83], for powering interoperability for centralized systems and permissioned DLTs; BUNGEE, supporting the data format that `Hermes` uses; Hyperledger Cacti to deploy SATP, `Hephaestus`, and provide the ability to connect to permissioned blockchains; `Harmonia` [151], for powering interoperability for decentralized systems; and `Hephaestus` [104], to secure `Hermes` and `Harmonia`. The user can combine a subset of our proposed solutions to enable interoperability on their decentralized application. Instantiations include the SATP protocol based on `Hermes` and our crash-recovery draft [154] and hosted in Hyperledger Cacti, realizing diverse use cases

such as providing interoperability capabilities for central bank digital currencies [45]. Another example is the horizontal interoperability layer in the Portuguese Blockchain Agenda project that will leverage SATP as its interoperability protocol and `Hephaestus` as the monitoring layer. Both technologies will be made production-ready for this project. Our work on `Harmonia` has been deployed in the Gnosis cross-chain bridge called Hashi[1].

The conducted research gave rise to the following contributions and their respective implications. We elaborate on each contribution in the light of the research question it answers. Non-scientific contributions that were the output of this doctoral program can be found in Appendix A. The first three contributions answer **RQ1: How to assess the interoperability capabilities of an interoperability mechanism?**. To assess a blockchain interoperability solution, one needs to be able to systematically classify them (**C1**), assess them (**C2**), and have a decision model to perform the decision (**C3**).

C1: A unified conceptual model and classification framework for blockchain interoperability solutions

Every technology undergoes a series of steps (lifecycle). Blockchain can be considered to be in its growth/-maturation phase [155, 1, 156]. As such, many of its concepts, namely the ones connected to interoperability, are still being iterated. Such iterations bring clarity to adopters and provide a common language for reasoning about the technology. The creation of a conceptual model and classification framework serves as an indispensable tool for the systematic categorization and understanding of diverse concepts and methodologies in a research area.

This contribution delivered a unified conceptual model and a classification framework for blockchain interoperability solutions. The conceptual model was the refined aggregate of previous research that explained the same concept differently (for example, a transaction transferring data or assets across chains can be called cross-blockchain transaction [65, 157], cross-chain communication [64] or cross-chain transaction [111]). Our contribution at [100] provided a simplified conceptual model to reason about blockchain interoperability concepts, including the concepts of *DLT Protocols*, *DLT Networks and DLT subnetworks*, *interoperability* (vertical and horizontal), *interoperability mechanism*, *cross-chain application* (also known as multiple DLT decentralized application), and *cross-chain transactions*. Such a conceptual model is pivotal in establishing a coherent structure for the vast and rapidly evolving field, enabling researchers and practitioners to navigate, relate, and build upon existing knowledge effectively.

On the other hand, the classification framework proposed in the same work had several goals: 1) introduce the solution landscape in light of the different interoperability layers and interoperation modes; 2) capture the complexity and nascent solutions that previous work did not account for; 3) present the *types* of interoperability solutions, as opposed to focusing on technical details. This allows researchers to focus on the grand picture, leaving implementation details for developers. This yielded two groups of solutions that abstract technical details from previous work [65]: oracles [158, 159, 160] and cross-authentication [161, 162, 163, 164, 165]. The proposed types are presented in detail regarding their architecture, protocols, advantages, and disadvantages, providing a holistic view for the researcher to reason about specific implementations.

Implications

A unified conceptual model and classification framework allows researchers and practitioners alike to share a common understanding of the area and for them to reason about specific implementations. A sound reasoning about the available solutions allows the researcher to understand potential classification frameworks.

[1] https://github.com/gnosis/hashi/pull/15

C2: A framework to assess the interoperability capabilities of a system utilizing multiple DLTs (in terms of potentiality, compatibility, and performance), based on the defined conceptual model

Little work has been done on systematically comparing interoperability solutions due to the considerable challenges that a lack of a uniform API and concrete benchmark datasets brings. However, the first steps are taken: we provided a framework for assessing interoperability solutions systematically [166, 100]. While writing this dissertation, benchmark studies started appearing, albeit scarce [167, 168], with promising results. In particular, Chervinski et al.'s work illustrates how systematic evaluation of IMs can show problems with transaction confirmation latency, bottlenecks in the blockchain's node implementation, and concurrency issues, paving the way for cost optimization. Our work shows that the studied interoperability solutions (e.g., Lifi [169]) show some interesting insights. First, the fees involved in a type of IMs called bridge aggregators appear to be a function of the gas price on the destination network, contrarily to the source. Second, the latency of a cross-chain transaction appears to be at least twice of the source chain (up to around ten times its latency). Finally, some solutions are optimistic, meaning they act on the target blockchain before confirmation on the source chain. Consequences for cross-chain logic relative to reorganizations of the source chain prior to the target chain transaction being executed are not obvious. To the best of our knowledge, a method does not exist to rebalance the cross-chain state after it has entered an inconsistent state.

Furthermore, methodology and empirical studies to assess components around cross-chain solutions, which contribute to the understanding of the first, have been proposed. For instance, systematic benchmarks of cryptographic primitives [170], libraries, compilers [171], and interoperability-related technologies such as SNARKs, STARKs, [172] have been done. Studying interoperability solutions in the Web3 world will also give back to traditional interoperability research, as we collect insights on integrating centralized with decentralized systems.

Implications

The provided guidelines and assessment framework allow researchers to systematically assess interoperability solutions. This unravels new research directions (e.g., benchmarks, metrics). It helps researchers uncover bottlenecks and to optimize their solutions. Furthermore, it allows for the dissemination of relevant information that can be used not only to classify the IM, but to assess its viability as a solution to a business problem.

C3: A framework that allows one to choose a blockchain interoperability solution, considering a set of criteria defined by our blockchain interoperability model

Following a systematization of concepts in the field, and respective techniques to compare solutions, we now provide a decision model for researchers to choose an interoperability solution, based on several criteria. These criteria stem from computer science interoperability research [89]: potentiality, compatibility, and performance (PCP). While all criteria were well-documented in the literature, specific applications for blockchain interoperability were missing, hence our contribution. In particular, we proposed two decision models that assist in selecting the appropriate functionality and infrastructure according to the PCP assessment criteria. The necessity of providing two different decision models stems from a good engineering practice of decoupling responsibilities. In fact, the deployment and operations management of an IM and their functionality can be decoupled and delegated to different parties. This is particularly useful because many companies in the space delegate node operations to infrastructure providers such as Blockdaemon [87]. Although most solutions are still not systematically evaluating their solutions, we highlight the following example that evaluates an IM in detail, according to our framework (namely performance-wise) [168].

Implications

The two proposed decision models assist researchers in choosing the infrastructure and functionality for a specific implementation of a blockchain interoperability solution, considering their need in terms of potentiality, compatibility, and performance. This streamlines projects integrating centralized with decentralized systems.

Contributions C4-C6 answer **RQ2: How can IMs provide technical and semantic interoperability (and provide the basis for organizational interoperability)?**. For an IM to be able to provide organizational interoperability, support for technical and semantic interoperability needs to be delivered (although necessary, they may not be sufficient conditions). Furthermore, we put forward a new set of conditions that we expect to respect the minimum necessary conditions for organizational interoperability. First, IMs need to be able to communicate data in a privacy-preserving manner, promoting accountable interoperability (**C4**). Next, we propose an interoperability paradigm based on the internet gateway model (**C5**). Finally, we contribute algorithms and protocols for guaranteeing atomicity and other desirable properties of cross-chain transactions across gateways (**C6**).

C4: A common data format for accountable and privacy-preserving interoperability - blockchain views.

BUNGEE is the protocol proposing a common data format for accountable and privacy-preserving interoperability. To understand the contribution of BUNGEE, it is useful to recall some Internet protocols. HTTP (Hypertext Transfer Protocol) is the foundation protocol for data communication on the World Wide Web. It defines the protocol for transferring hypertext requests and information on the internet, being adopted worldwide. It allows the exchange of information in a predictable and standardized way. Such protocols allow for defining application programming interfaces (APIs). APIs provide a set of rules and specifications that allow different software applications to communicate with each other. By defining the methods and data formats that should be used for interaction, APIs facilitate the integration of disparate systems, enabling them to work together to achieve a common goal.

The infrastructure created by these technologies can be augmented to the blockchain scenario. Many blockchain protocols define APIs using the architectural styles REST or Remote procedure call (RPC), allowing developers to create communication protocols on top of it. This contribution defines the interfaces and data formats (technical interoperability) to realize protocols such as SATP (semantic interoperability). We call the data format that represents arbitrary state of any blockchain a blockchain view. Specific implementations of blockchain views can use different architectural styles, catering to permissioned DLTs, permissionless DLTs, and centralized systems. Thus, the view is a format that can be used in the interoperability context by overriding data formats from heterogeneous systems. Moreover, such data format must cater to flexible privacy requirements since centralized organizations might require them. By defining algorithms and interfaces for merging and processing views according to different privacy policies, one can define and enforce privacy requirements, allowing privacy-preserving interoperability, a necessary condition for some hybrid blockchain applications. Moreover, the merge, integration, and processing of views are standardized to be made in an accountable way (namely, BUNGEE signs operations done over the view, ensuring accountability).

Implications

A privacy-preserving friendly data format and standard to communicate arbitrary data across centralized and decentralized systems. It allows interoperability with less engineering overhead, while providing the necessary assurances for decentralized systems.

C5: Blockchain gateway paradigm as the enterprise-grade IM for auditable, private, and reliable interoperability.

As new regulations about tokenized assets come into force [173], there is renewed interest in developing the Web3 ecosystem that enables the *Internet of Value*. While public blockchains remain fundamental building blocks for Web3 asset networks, private DLT infrastructure is being tailored for centralized infrastructures. As a consequence, the same software stacks are beginning to be re-used in these permissioned environments. As such, enterprises struggle to guarantee certain properties that blockchains traditionally do not grant: privacy, strong accountability [4], and the ability to roll-back transactions [83]. This challenge is particularly notorious in the cross-chain setting [174]. This contribution introduces the blockchain gateway paradigm, an answer to the reconciliation between centralized and decentralized technologies. In this paradigm, blockchain gateways mediate the interoperability across network boundaries, much like gateway routers in the internet architecture, being able to provide certain degrees of privacy and accountability. Gateways operate as logical network boundaries, which can coincide with jurisdictional boundaries - such border control is appropriate to manage tokenized regulated assets. SATP-powered gateways allow voluntary asset transfers of digital assets from the origin network to a destination network in such a way that evidence of the transfer can be verified by a third-party audit in the case of disputes. The origin and destination networks are assumed to share a common understanding of the digital asset, which requires a degree of organizational interoperability in production.

This kind of work also briefly touches legal interoperability. In particular, common regulations that virtual asset service providers need to abide by are the travel rule [175] and FAFT recommendation 15 [176]. In practice, the implication of these regulations, among others, is that crypto-exchanges and other types of virtual asset service providers is that they must be able to share the originator and beneficiary information for virtual asset transactions. This process – also known as the Travel Rule – originates from "under the US Bank Secrecy Act (BSA - 31 USC 5311 - 5330), which mandates that financial institutions deliver certain types of information to the next financial institution when a funds transmittal event involves more than one financial institution. This customer information includes (i) originator's name; (ii) originator's account number (e.g. at the Originator's VASP); (iii) originator's geographical address, or national identity number, or customer identification number (or date and place of birth); (iv) beneficiary's name; (v) beneficiary account number (e.g. at the Beneficiary-VASP)" [177]. Therefore, we expect blockchain gateway standardization work to accelerate research not only on organizational interoperability in the medium term, but also legal interoperability, in the medium-long term.

Implications

Our blockchain gateway paradigm provides, for the first time, the technical basis for enterprises to systematically implement recommendations and abide by the law when it comes to manage their activities in the Web3 world. Gateways allow to guarantee privacy and accountability in interactions across centralized and decentralized systems, by transacting with blockchain views and enforcing an audit log. Furthermore, by means of connecting gateways operating in different blockchains, we can now perform cross-chain asset transfers in a traceable, auditable, and privacy-preserving way[2]. Such a model is in the inception of standardization efforts such as the Secure Asset Transfer Protocol at the Internet Engineering Task Force [178].

[2]Blockchain gateways operating on privacy-preserving blockchains are the currently known best candidates for private data and asset transfers. However, more research is needed [121].

C6: Technical architecture, protocols, and algorithms to guarantee ACIDC properties for transactions amongst gateway-based IMs.

The previous contribution alleviates privacy and accountability problems for interoperability involving centralized systems with decentralized systems, or involving private DLTs. However, the need to roll-back transactions [83] is typically not provided by most IMs. This is due to several reasons, the most notorious being the fact that blockchains are tamper-resistant systems that do not allow rollbacks in normal circumstances. The present contribution provides a fault-tolerant middleware that assures atomicity, consistency, isolation, durability, and accountability of cross-chain transactions. This solves a problem that has been dealt optimistically. Imagine a cross-chain transaction when A happens, B happens. However, A got reverted. Most solutions ensure this does not happen by waiting for enough confirmations (thus raising the probability that transaction A will not get reverted) [116, 1]. However, sometimes, that does happen [179], and the state of cross-chain applications is left in an inconsistent state. This contribution adds a safety guarantee by defining and enforcing rollback logic, which is desirable for companies to use IMs in their software stack. Furthermore, we extend blockchain gateways with a crash recovery mechanism, which handles the rollback logic as a function of when a crash occurred in a gateway. By choosing the specific recovery model, the developer can tailor the recovery process to their organization's technical infrastructure, assuring that gateways keep a consistent cross-chain state even in the presence of crashes.

Implications: For Researchers

Our work on atomic cross-chain interoperability was one of the first, in 2021, inspiring multiple improvements by the academia over the years. For example, [180, 181, 182] generalizes our asset transfer to smart contract invocations. We foresee that an integration of such methods will lead to more performant, secure ways to realize atomic general purpose interoperability.

Implications: For Practitioners

While we cannot provide transaction atomicity across chains in the literal sense of the word, the best we can currently do are abstractions that provide ACID properties under certain conditions (probabilistic atomicity using rollbacks). Such solutions are instrumental for the industry to implement dependable bridges and rollback if necessary [174]. This way, one can implement transaction rollback (e.g., in case of a malicious transaction), redactions across blockchains [174], and rebalancing of invalid cross-chain state.

Contributions C7 and C8 answer **RQ3: How to add robustness to blockchain interoperability mechanisms from a security perspective?**. These two contributions focuses on increasing the resiliency of current solutions, in face of the numerous attacks [121]. First, we study the applicability of verifiable computation (SNARK) technology to create a novel interoperability solution class, as a means to reduce the attack surface that attackers exploit. We explore its secure decentralization as a means to increase its robustness (**C7**). As all solutions are exploitable, potentially including our own at contribution C7, we design a novel strategy for monitoring attacks on cross-chain applications. The rationale is to proactively secure applications by responding as soon as an attack commences, limiting damages (**C8**).

C7: Protocols, algorithms, technical architecture, and frameworks for decentralized, secure blockchain interoperability based on the cryptography behind zero-knowledge proofs (namely SNARKs).

The interoperability research field until 2023 contemplated mostly local verification, trusted third parties, local verification, and native state verification (mostly inclusion proofs). In 2023, we proposed, shortly after a similar project [183] a validity proof-based interoperability mechanism using SNARKs for heterogeneous chains, `Harmonia`. The key intuition of the paper is that SNARKs can be used to prove that the consensus rules of a source chain are correctly executed, and then verify them on a third-party blockchain, succintly. This yields the first practical, cost-effective, secure, and decentralized interoperability mechanism. A key contribution of `Harmonia` is reducing the attack vector for applications relying on the Ethereum blockchain state in several ways. First, it reduces dependencies on trusted third parties, generally agreed to be the most effective way to secure IMs [121], and thus, no external trust assumptions are introduced (external trust assumptions accounted for the majority of cross-chain bridge hacks [121]). The security of the interoperability processes is anchored on the consensus of the source chain and the cryptography behind SNARKs. Second, any node has the autonomy to connect with the network, relay block headers, produce proofs, and secure rewards, and thus the solution is decentralized and resilient to censorship resistance. The second key contribution is the improvements proposed to Ethereum's light client protocol that has critical flaws. While interesting in itself, this proposal carries significant implications as it increases the robustness of Ethereum as the source chain for interoperability, until the Ethereum protocol does not incorporate the security of the whole validator set for the light sync committee.

From a user perspective, `Harmonia` provides an adaptable and extensible design. `Harmonia`-powered applications can introduce their unique verification processes and functionalities, enhancing their breadth of application (e.g., transaction validation via supplementary Merkle proof). By distinguishing the interoperability mechanism from the cross-chain logic, `Harmonia` simplifies the integration of decentralized applications with several blockchains that can support the verification of SNARKs. Our research shows that the decentralization, latency, and monetary costs are appropriate, given the security benefits.

Implications

`Harmonia` provides the first comprehensive framework to deploy cross-chain applications with strong security, liveness, and decentralization, while providing acceptable costs. We demonstrate the suitability of this paradigm via our instantiation of the framework with DendrETH. We foresee that SNARK-based bridges will slowly dominate the industry, with possible privacy extensions [121]. Some solutions are already looking to migrate from more centralized schemes to this paradigm [184, 185].

C8: Protocols, algorithms, technical architecture, and frameworks for continuous monitoring and incident response of blockchain bridges.

Recent years have witnessed significant advancements in cross-chain technology. However, the field faces two pressing challenges when it comes to security. Hacks on cross-chain bridges have led to monetary losses of around 3 billion USD [121], highlighting flaws in security models governing interoperability mechanisms (IMs) and the ineffectiveness of incident response frameworks. The authors understand that no solution is impervious to attacks despite best efforts to diminish its likelihood. For example, several interoperability solutions marketed themselves as "security-first", and despite best-efforts, they were exploited [104, 186, 121]. Therefore, the authors understand that the solutions based on the delivered contributions are susceptible to exploitation. As such, the present contribution focuses on actively preventing and, if not possible, swiftly reacting and recovering from

attacks. We propose, for the first time, a process mining-based technique to infer bridge processes automatically and detect deviations from the normal process. This is done by either modeling and inputting the cross-chain rules to `Hephaestus` or letting `Hephaestus` deduct the cross-chain rules by observing cross-chain transactions. Either way, a cross-chain model is constructed that contains the rules that are expected to be verified with regard to transactional flow. Response to deviations from those rules can be programmed. A rule of thumb we propose is for alerts to be emitted to trigger an incident response mechanism, which takes proactive steps to defend it. While there is little work on incident response frameworks for cross-chain bridges [121, 187], we hypothesize that such frameworks would follow traditional cybersecurity incident response practices [188].

Implications: For Practitioners

The present contribution models the security of a bridge based on local events. Modeling security allows developers to define security metrics and their monitoring based on a wide array of implementations (process mining [104]), formal methods [189], and already large language models [190]). The present work opens up future research directions on studying tradeoffs between different incident response mechanisms and processes.

7.3 Future Work

Interoperability is a field in continuous evolution and has been an important topic of interest since the 1980s. Accompanying this evolution, the notion, underlying concepts, and frameworks have evolved to cater to new technologies, techniques, and application domains. We predict that the field will continue to evolve to cater to future needs, as corroborated by other researchers. In each chapter of this dissertation, we highlight the limitations of the research publication it points to, as well as future research directions. Nonetheless, we update the future research directions of each publication as of January 2024. Below, we highlight potentially relevant research for future work according to the chapters presented in this dissertation.

The listed directions for future research can be essentially divided according to their aim: 1) application Application, 2) extension Extension, and 3) exploration Exploration.

The application category states that the developed principles, frameworks, decision models, and algorithms can be used to enable interoperability, can be applied in alternative real-world use cases to realize semantic interoperability. Furthermore, the presented technical contributions enable organizations to cooperate via the proposed IMs and achieve organizational interoperability. The "Extension" is a placeholder for the work that can be refined and augmented to 1) guarantee their further improvement in terms of performance, privacy, connectivity, security assurances, and decentralization, and 2) solve new problems (e.g., using interoperability for security robustness, the so-called blockchain of blockchains approach [65]; using interoperability for increased performance; using interoperability for risk management, and so on). Finally, the proposed work can be leveraged to explore, evaluate, and validate existing cross-chain applications or interoperability mechanisms, through "Exploration".

7.3.1 Do You Need a Distributed Ledger Technology Interoperability Solution?

- Extension create a database of existing DLT Protocols, DLT Networks, and DLT Subnetworks (similar to chainlist.org but also supporting permissioned infrastructure);
- Extension identifying the technical capabilities of networks and subnetworks (e.g., supported hash functions, signature schemes, asset types);

- Extension incorporate security, privacy, and decentralization on the decision model (similarly to how it is depicted in [167]);
- Extension formalization of technical requirements for DLTs to be interoperable;
- Extension include standardization support on the decision model;
- Extension provide fine-grain evaluation of compatibility by defining sub-levels for the latter;
- Exploration apply the compatibility assessment for enterprise-grade IMs to assess compliance with standards and/or regulations;
- Exploration on the performance assessment, empirically assess what is "acceptable" latency, throughput, transaction costs, and carbon footprint, possibly for specific groups of IMs;
- Exploration create a comprehensive benchmark test suite for comparing IMs and cross-chain dApps;
- Exploration create a comprehensive database of classified IMs against the proposed framework;

7.3.2 BUNGEE: Dependable Blockchain Views for Interoperability

- Application provide more complex merging algorithms for real-world use cases;
- Application using views to support cross-chain state creation, management, and visualization.
- Application views for blockchain state migration;
- Extension support Byzantine-fault tolerant blockchain view generators for fault tolerance and reliability;
- Extension support zero-knowledge proofs for 1) enhanced privacy of the view generation, view processing, and view integration processes and 2) trustless verifiability of the view generation, view processing, and view integration processes;
- Exploration extend blockchain views used in SATP to support centralized systems;
- Exploration study privacy and other compliance requirements for specific regulations;

7.3.3 Hermes: Fault-Tolerant Middleware for Blockchain Interoperability

- Application leverage SATP for asset transfer IMs that implement organizational interoperability;
- Extension support optimistic and pessimistic commitments for SATP and respective rollback strategies;
- Extension support coordinator gateway crashes (i.e., SATP using three-phase commit [191]);
- Extension support for aggregating asset and data transfers in a single transaction for cost reduction;
- Exploration determine what is the right level of privacy for storing on-chain versus off-chain gateway logs (this includes studying organizational interoperability);
- Exploration high-scale performance assessment of gateway networks, similarly to [168];
- Exploration explore the definition of regulations and laws for blockchain interoperability, for specific use cases (legal interoperability);

7.3.4 Harmonia: Securing Cross-Chain Applications Using Zero-Knowledge Proofs

- Application bi-directional SNARK-based bridges for heterogeneous public blockchains;
- Application bridges between private and public blockchains leveraging a common data format (e.g., verifying SNARKs on a blockchain view);
- Application integrate SATP and `Harmonia` to create bridges between centralized and decentralized systems;
- Application create trustable blockchain migrations using `Harmonia` and blockchain views;
- Extension extend `Harmonia` to support privacy-preserving cross-chain applications (asset transfers, data transfers);
- Extension extend DendrETH to implement light client protocols from other chains, including private ones;
- Extension support different verifiable computation proof generation schemes, e.g., STARKs [192, 193], Bulletproofs [194], PLONK [195], and more recent schemes [196, 197, 198];
- Extension study the costs for censorship resistance in blockchains other than Ethereum;
- Extension reduce latency by exploring parallelism in sub-circuits that compose DendrETH;
- Exploration study the trade-offs between latency and performance in real-world applications;
- Exploration conduct a detailed risk assessment of specific cross-chain applications using `Harmonia`;
- Exploration reduce the SNARK relayer latency by distributing proof generation;
- Exploration reduce the average costs of generating a SNARK through proof aggregation;
- Exploration benchmark `Harmonia` and DendrETH with different proof types, relayer infrastructures, and incentive infrastructures;
- Exploration explore quantum-resistant zero-knowledge cryptography;

7.3.5 Hephaestus: Modelling, Analysis, and Performance Evaluation of Cross-Chain Transactions

- Application real-time monitoring of deployed cross-chain applications with emphasis on bridges;
- Application real-time monitoring of hybrid applications (centralized, public decentralized, and private decentralized systems).
- Application tax report generator based on different legal frameworks (organizational and legal interoperability);
- Application visualization functionality and fund tracking for cross-chain wallets;
- Extension support programmable response incident response frameworks (e.g., define custom circuit breakers);
- Exploration create a database of cross-chain models for main cross-chain applications;
- Exploration study whether main cross-chain applications always followed their respective cross-chain models;
- Exploration study deviations from cross-chain models (e.g., for identification of outliers, bugs, attacks);
- Exploration mechanisms for defending bridge aggregator systems [1];
- Exploration create a repository of cross-chain models for benchmarking and cost optimization.

7.3.6 Summary and Future Landscape

We lay down future work that is transversal to all chapters in this thesis and summarizes existing research directions. The ones we believe are the most prominent as of January 2024, and common to the ones presented, are the challenges inherent to the maturation of a technology, institutional adoption, privacy, benchmarking, and security monitoring and enforcement.

Maturation challenges

The continous evolution of interoperability brings an orthogonal problem to the area: a lack of uniformization of terms and vocabulary. Although addressed by a few standards [22, 199, 200], extensively in the academia, and in this thesis, some (more novel) terms are still used inconsistently. We expect that the current maturation of the area brings about a uniform vocabulary that can be used across verticals and professional roles.

This trend is potentiated by the emergence of new interoperability types, new frameworks, conceptual models, ontologies, architectures, techniques, and implementations will be needed. A recent study showed that much of the interoperability research focuses on semantic interoperability, leaving many interoperability types not accounted for [90]. As interoperability needs to be considered from different perspectives, solving social-technical interoperability in the blockchain space could be a necessary venue for future research. More concretely, legalinteroperability is addressed insufficiently, e.g., leaving decentralized autonomous organizations not communicating across smart contracts and blockchains. This new area of research would motivate structures and incentive mechanisms to solve consensus not solely on technical aspects (e.g., transaction ordering in proof of stake blockchains), but also on cross-border organizational consensus. This challenge is exacerbated by decentralized autonomous organizations (DAOs) potentially having members from different countries following different legal frameworks. The validating nodes that include the transactions done against the DAO by its members could also abide by very different legal frameworks. We need to rethink how organizational and legal interoperability can be solved in contexts like these. One of the key challenges would be assuring compliance with privacy regulations such as the GDPR [143], technically achievable by redacting [174] or rollbacking [83] state in a cross-chain setting.

Institutional Adoption

We have assisted a trend where most use cases for cross-chain applications are data transfers (using oracles) and asset transfers (using cross-chain bridges) [100]. More recently, cross-chain MEV [201, 202] and cross-chain arbitrage [203, 204] have been proposed, but in limited scope [121]. However, we do believe the trend is shifting. In particular, the industry is evolving towards an equilibrium where not only use cases using permissionless blockchains will have a great impact on society.

More and more enterprises are using blockchain technology, either to decentralize their processes or to support cryptocurrencies in their service and product offerings [205, 206, 207]. Interestingly, in a Forbes article, it is stated ”BlackRock’s Larry Fink has come out strongly supporting tokenization as the next generation for markets. BlackRock estimates that tokenization of private market assets will open markets worth $290 trillion. The Boston Consulting Group predicts that some $16 trillion worth of assets, most of which are illiquid, will be tokenized by 2030.” [208]. This trend suggests major financial market players will eventually need to transfer blockchain tokens (e.g., cryptocurrencies, digital twins, securities, non-fungible tokens) to each other in a legally binding, auditable, and regulated way. We emphasize our work in this area with the Secure Asset Transfer Protocol, which

is tailored for institutional needs and supports different jurisdictions and legal frameworks [178]. According to SATP's charter, "assets in one network cannot be moved easily to another network. The problem is more acute in the case of private asset networks, where external entities have no visibility into the state of an asset in the private network. An example is regulated digital representations of real-world private assets, such as property ownership certificates, and regulated government-issued digital currencies. The goal of the Secure Asset Transfer Protocol (SATP) working group will be to develop a standard protocol that operates between two peer gateways to transfer digital assets between an originator in the origin network to a beneficiary in a destination network. The resulting protocol shall be agnostic with respect to the type of asset being transferred." [178]. We predict institutional adoption to keep increasing, with interoperability work as an instrumental component. SATP is one of the first protocols addressing the issues inherent to enterprise-grade interoperability.

Cross-chain privacy

Institutional adoption will only come with certain guarantees privacy-wise. It is generally agreed upon that the properties of anonymity (in terms of unlinkability), confidentiality, and indistinguishability of transactions are beneficial privacy properties in the cross-chain context [209]. An anonymous asset transfer (or exchange) will hide the identities of the parties involved in the transfer. Confidentiality will hide the number of transferred tokens. Indistinguishability means an external observer cannot say whether or not the transaction is part of a swap. Researchers and practitioners alike have done work in cross-chain, specifically in the areas of asset transfers (namely between privacy-enhanced blockchains, as the source, and public blockchains, as the target [210], leveraging promising technologies such as zero-knowledge proofs [151]). Although there is a long way ahead, existing work seems to suggest that in scenarios where at least one confidential blockchain is involved (by confidential, we mean permissioned or privacy-enabled by default like Hyperledger Fabric, ZCash, or Monero, e.g., confidential to confidential), preserving the property of unlinkability is possible, therefore achieving some level of anonymity (and possible some confidentiality depending on the blockchain, as ZCash would allow).

Privacy on asset exchanges has also been studied [211] (see, for example, an implementation of a cross-chain private asset exchange here[3]). Privacy on asset exchanges looks more straightforward than other interoperability modes: HTLCs share secrets only understandable by the involved parties, so it becomes harder to draw direct associations between transactions. Of course, by analyzing certain heuristics (simpler: amount locked, cryptographic parameters such as the prime field for a private HTLC; more complex: time intervals for swaps, user activity interactions, crossing with off-chain data), one could deanonymize the actors behind cross-chain transactions [212]. Thus, more research is needed. An interesting direction proposed recently is the creation of techniques to make the referred heuristics harder to calculate, for example, using mixers [213] in the cross-chain scenario.

On the other hand, privacy on data transfers is studied only partially: some authors worked on the concept of self-sovereign identity to facilitate cross-chain interactions [214, 37, 215]. In fact, interoperation for data sharing between blockchains requires the networks' ability to authenticate requests using well-defined access control policies (thus increasing confidentiality) and validating proofs. While the first steps have been taken, no practical implementations of this idea exist. Furthermore, to the best of our knowledge, no empirical studies are studying cross-chain privacy. We emphasize that there is a trade-off between privacy and accountability: revocation of privacy could be conditional (e.g., the user moves funds above the established limit), dependent on the interoperability mechanism architecture.

[3] https://github.com/RafaelAPB/blockchain-integration-framework/tree/private-htlcs

Security Monitoring and Enforcement

We provided several research directions to increase the robustness of IMs, namely their safety, liveness, privacy, and decentralization, yielding improvements in the overall security of these systems. However, bugs and exploits are not only inevitable but abundant [216, 217, 121], so one must invest in monitoring and quick response to contain damages [218]. Monitoring interoperability protocols and the sophisticated, and sometimes fragile relationships between ecosystems quickly becomes hard, because the systems to be dealt with are heterogeneous and decentralized, and the systems built on top of them (e.g., decentralized applications) may have arbitrarily complex business logic [104]. Imagine a simple case: an application on blockchain A depends on the consensus of blockchain B. What happens if blockchain B forks, is attacked (e.g., 51%), suffers any of the many possible cross-chain attacks [120], or even collapses?

This last possibility was a reality for the Terra blockchain, with implications for the Cosmos and Ethereum ecosystem, as the Osmosis bridge connected them. In the Terra blockchain collapse, exploiters created a destabilization of the stablecoin hosted by Terra. This destabilization caused liquidation cascading, possibly the main cause for a new crypto crash [179]. The collapse of economic security on Luna posed dangers for the Cosmos hub Osmosis, a decentralized exchange bridged to Ethereum. In Osmosis, there were $66 million dollars of OSMO tokens in the UST/OSMO pool, where UST is the Terra blockchain, that could be stolen over the bridge by an attacker with voting power equal to two-thirds of the staked LUNA. A solution to this problem was for bridge operators to manually shut down bridges, causing impermanent losses to users holding the TERRA token. The monitoring of the operations underlying this particular use case could have prevented such a tragic outcome and helped mitigate loss. However, in a cross-chain setting, automating the discovery of cross-chain models and enabling their monitoring becomes very challenging, as there is a lack of tools to secure and monitor cross-chain applications. Solutions based on formal modeling by specification [104] or alternatives based on large language models [190] (despite yielding a high rate of false positives and false negatives) could be interesting directions for future work, possibly enabling novel and effective incident response techniques .

Bibliography

[1] R. Belchior, J. Süßenguth, Q. Feng, T. Hardjono, A. Vasconcelos, M. Correia, A brief history of blockchain interoperability, Commun. ACMOnline First (Sep. 2024). doi:10.1145/3648607.
URL https://doi.org/10.1145/3648607

[2] N. Popper, Digital Gold: Bitcoin and the Inside Story of the Misfits and Millionaires Trying to Reinvent Money, 2016.

[3] G. Davies, History of Money, University of Wales Press, 2002, google-Books-ID: cU2uBwAAQBAJ.

[4] R. Belchior, L. Torres, J. Pfannschmidt, A. Vasconcelos, M. Correia, BUNGEE: Dependable Blockchain Views for Interoperability, ACM DLT (Jan. 2024). doi:10.1145/3643689.
URL https://doi.org/10.1145/3643689

[5] C. Cachin, M. Vukolić, Blockchain consensus protocols in the wild, arXiv e-prints 91, publisher: Schloss Dagstuhl- Leibniz-Zentrum fur Informatik GmbH, Dagstuhl Publishing tex.arxivid: 1707.01873 (Jul. 2017).
URL http://arxiv.org/abs/1707.01873

[6] B. Alpern, F. B. Schneider, Defining liveness, Information processing letters 21 (4) (1985) 181–185, publisher: Elsevier.

[7] M. Correia, From byzantine consensus to blockchain consensus, Essentials of Blockchain Technology (2019) 41.

[8] S. Nakamoto, Bitcoin: A peer-to-peer electronic cash system (2008).
URL http://bitcoin.org/bitcoin.pdf

[9] V. Cermak, Can Bitcoin Become a Viable Alternative to Fiat Currencies? An Empirical Analysis of Bitcoin's Volatility Based on a GARCH Model (May 2017). doi:10.2139/ssrn.2961405.
URL https://papers.ssrn.com/abstract=2961405

[10] L. Levulytė, A. Šapkauskienė, Cryptocurrency in context of fiat money functions, The Quarterly Review of Economics and Finance 82 (2021) 44–54. doi:10.1016/j.qref.2021.07.003.
URL https://www.sciencedirect.com/science/article/pii/S1062976921001265

[11] DeVries, Peter, Analysis of Cryptocurrency, Bitcoin and the Future (2016).
URL https://journal.formosapublisher.org/index.php/eajmr/article/view/803

[12] J. Bouoiyour, R. Selmi, M. E. Wohar, Bitcoin: competitor or complement to gold?, Economics Bulletin 39 (1) (2019) 186–191, publisher: Economics Bulletin.
URL https://hal.science/hal-01994187

[13] Z. J. Zhang, Cryptopricing: Whence comes the value for cryptocurrencies and NFTs?, International Journal of Research in Marketing 40 (1) (2023) 22–29, publisher: Elsevier.

[14] A. Rejeb, K. Rejeb, J. G. Keogh, Cryptocurrencies in modern finance: a literature review, Etikonomi 20 (1) (2021) 93–118.

[15] U. Bindseil, G. Pantelopoulos, Towards the holy grail of cross-border paymentsPublisher: ECB Working Paper (2022).
URL http://tinyurl.com/bdzbfx2j

[16] N. Papadis, L. Tassiulas, Blockchain-based payment channel networks: Challenges and recent advances, IEEE access : practical innovations, open solutions 8 (2020) 227596–227609, publisher: IEEE.

[17] J. Poon, T. Dryja, The bitcoin lightning network: Scalable off-chain instant payments (2016).
URL https://nakamotoinstitute.org/research/lightning-network/

[18] S. P. Yadav, K. K. Agrawal, B. S. Bhati, F. Al-Turjman, L. Mostarda, Blockchain-based cryptocurrency regulation: An overview, Computational Economics 59 (4) (2022) 1659–1675, publisher: Springer.

[19] B. D. Feinstein, K. Werbach, The impact of cryptocurrency regulation on trading markets, Journal of Financial Regulation 7 (1) (2021) 48–99, publisher: Oxford University Press.

[20] V. Buterin, Ethereum: A next-generation smart contract and decentralized application platform (2014).
URL https://github.com/ethereum/wiki/wiki/White-Paper

[21] G. Wood, Ethereum: A secure decentralised generalised transaction ledger. Byzantium version 7e819ec, Tech. rep., Ethereum Foundation (2019).
URL https://ethereum.github.io/yellowpaper/paper.pdf

[22] ISO, ISO/TC 307/WG 1 FDIS 22739 (E), Tech. rep., International Organization for Standardization (ISO) (2020).
URL https://www.iso.org/committee/6266604.html

[23] P. W. Abreu, M. Aparicio, C. J. Costa, Blockchain technology in the auditing environment, in: 2018 13th iberian conference on information systems and technologies (CISTI), 2018, pp. 1–6. doi:10.23919/CISTI.2018.8399460.

[24] R. Belchior, M. Correia, A. Vasconcelos, JusticeChain: Using Blockchain To Protect Justice Logs, in: CoopIS 2019: 27th International Conference on Cooperative Information Systems, 2019.

[25] R. Belchior, A. Vasconcelos, M. Correia, Towards Secure, Decentralized, and Automatic Audits with Blockchain, in: European Conference on Information Systems, 2020.

[26] H. Pagnia, F. C. Gärtner, others, On the impossibility of fair exchange without a trusted third party, Tech. rep., Citeseer (1999).

[27] S. Peterson, Centralized Exchanges Have a Major Problem (Jul. 2021).
URL https://coingape.com/centralized-exchanges-major-problem/

[28] What Was Mt. Gox? Definition, History, Collapse, and Future (2023).
URL https://www.investopedia.com/terms/m/mt-gox.asp

[29] B. Schulz, R. Mcdermid, What is the FTX scandal? How the celebrity-endorsed crypto giant collapsed into chaos (2023).
URL https://www.usatoday.com/story/money/2022/11/16/ftx-bankman-frieds-crypto-bankruptcy/10710734002/

[30] Expresso, Mais de 100 investidores em Portugal perderam €6,5 milhões com o colapso da empresa de criptoativos FTX (Aug. 2023).
URL http://tinyurl.com/yb9d5ta4

[31] L. Peng, W. Feng, Z. Yan, Y. Li, X. Zhou, S. Shimizu, Privacy preservation in permissionless blockchain: A survey, Digital Communications and Networks 7 (3) (2021) 295–307, publisher: Elsevier.

[32] C. Keville, R. Belchior, How to Improve Web3 Privacy and Regulatory Compliance — Blockdaemon Blog (2023).
URL https://www.blockdaemon.com/blog/how-to-improve-web3-privacy-and-regulatory-compliance

[33] B. Markey-Towler, Anarchy, Blockchain and Utopia: A theory of political-socioeconomic systems organised using Blockchain, Available at SSRN 3095343 (2018).

[34] A. Zohar, Bitcoin: Under the hood, Communications of the ACM 58 (9) (2015) 104–113, publisher: Association for Computing Machinery tex.mendeley-tags: Support.

[35] F. Ul Hassan, A. Ali, S. Latif, J. Qadir, S. Kanhere, J. Singh, J. Crowcroft, Blockchain and the future of the internet: A comprehensive review, arXiv e-printsArXiv: arXiv:1904.00733 tex.arxivid: arXiv:1904.00733 tex.mendeley-tags: INCLUDED,INCLUDED$_G$REY$_L$ITERATURE,Support (2019).
URL https://arxiv.org/abs/1904.00733

[36] S. Rouhani, R. Belchior, R. S. Cruz, R. Deters, Distributed attribute-based access control system using permissioned blockchain, World Wide Web (2021). doi:10.1007/s11280-021-00874-7.

[37] R. Belchior, B. Putz, G. Pernul, M. Correia, A. Vasconcelos, S. Guerreiro, SSIBAC : Self-Sovereign Identity Based Access Control, in: The 3rd International Workshop on Blockchain Systems and Applications (part of IEEE TrustCom 2020), IEEE, 2020.

[38] P. Dunphy, F. A. Petitcolas, A first look at identity management schemes on the blockchain, IEEE Security and Privacy 16 (4) (2018) 20–29, publisher: IEEE.

[39] Z. Wenhua, F. Qamar, T.-A. N. Abdali, R. Hassan, S. T. A. Jafri, Q. N. Nguyen, Blockchain technology: security issues, healthcare applications, challenges and future trends, Electronicsweek 12 (3) (2023) 546, publisher: MDPI.

[40] C. C. Agbo, Q. H. Mahmoud, J. M. Eklund, Blockchain technology in healthcare: a systematic review, in: Healthcare, Vol. 7, MDPI, 2019, p. 56, issue: 2.

[41] F. Casino, T. K. Dasaklis, C. Patsakis, A systematic literature review of blockchain-based applications: Current status, classification and open issues, Telematics and Informatics 36 (2019) 55–81. doi:10.1016/j.tele.2018.11.006.

[42] Blockchain platforms reviews 2021 | gartner peer insights (2021).
URL https://www.gartner.com/reviews/market/blockchain-platforms

[43] Coin Market Cap, Coin Market Cap - All coins (2023).
URL https://coinmarketcap.com/coins/views/all/

[44] R. Khalil, A. Gervais, Revive: Rebalancing off-blockchain payment networks, in: Proceedings of the 2017 acm sigsac conference on computer and communications security, 2017, pp. 439–453.

[45] A. Augusto, R. Belchior, A. Vasconcelos, I. Kocsis, G. László, M. Correia, CBDC bridging between Hyperledger Fabric and permissioned EVM-based blockchains, in: IEEE ICBC'23 - CrossChain workshop, IEEE, 2023.

[46] X. Han, Y. Yuan, F.-Y. Wang, A blockchain-based framework for central bank digital currency, in: 2019 IEEE International conference on service operations and logistics, and informatics (SOLI), IEEE, 2019, pp. 263–268.

[47] F. Schär, Decentralized finance: On blockchain-and smart contract-based financial markets, FRB of St. Louis Review (2021).

[48] S. Werner, D. Perez, L. Gudgeon, A. Klages-Mundt, D. Harz, W. Knottenbelt, Sok: Decentralized finance (defi), in: Proceedings of the 4th ACM conference on advances in financial technologies, 2022, pp. 30–46.

[49] World Bank Group, Blockchain interoperability (2020).
URL https://www.ft.com/content/1cfb6d46-5d5a-11e9-939a-341f5ada9d40

[50] I. Mikhalev, K. Burchardi, I. Struchkov, B. Song, J. Gross, Central bank digital currency (CBDC) tracker (2021).
URL https://cbdctracker.org/cbdc-tracker-whitepaper.pdf

[51] M. Hargreaves, T. Hardjono, Implementing a CBDC: the challenges and a solution, Tech. rep. (2021).
URL https://www.quant.network/insights/implementing-a-cbdc-the-challenges-and-a-solution

[52] A. Pentland, A. Lipton, T. Hardjono, Time for a new, digital Bretton Woods, Barron's (Jun. 2021).
URL https://rb.gy/yj31vq

[53] L. Pawczuk, R. Massey, J. Holdowsky, Deloitte's 2019 global blockchain survey (2019).

[54] A. Henninger, A. Mashatan, Distributed interoperable records: The key to better supply chain management, Computers 2021, Vol. 10, Page 89 10 (7) (2021) 89, publisher: Multidisciplinary Digital Publishing Institute. doi:10.3390/COMPUTERS10070089.
URL https://www.mdpi.com/2073-431X/10/7/89/htmhttps://www.mdpi.com/2073-431X/10/7/89

[55] Binance, Binance launches $1B binance smart chain fund to reach one billion crypto users (2021).
URL http://tinyurl.com/yat6b39d

[56] Blockchain Grants, Defi & NFT blockchain grants (2023).
URL https://www.blockchaingrants.org/home

[57] Interchain Foundation, Funding | interchain foundation (2023).
URL https://medium.com/the-interchain-foundation/icf-funding-program-2024-3928d3b59e2f

[58] Hyperledger, Hyperledger mentorship program (2023).
URL https://wiki.hyperledger.org/display/INTERN/Hyperledger+Mentorship+Program

[59] J. Kolb, M. Abdelbaky, R. H. Katz, D. E. Culler, Core concepts, challenges, and future directions in blockchain: A centralized tutorial, ACM Computing Surveys 53 (1), publisher: Association for Computing Machinery (Feb. 2020). doi:10.1145/3366370.

[60] O. Rikken, M. Janssen, Z. Kwee, Governance challenges of blockchain and decentralized autonomous organizations, Information Polity 24 (4) (2019) 397–417, publisher: IOS Press.

[61] H. Tam Vo, Z. Wang, D. Karunamoorthy, J. Wagner, E. Abebe, M. Mohania, Internet of blockchains: Techniques and challenges ahead, in: 2018 IEEE international conference on internet of things (iThings) and IEEE green computing and communications (GreenCom) and IEEE cyber, physical and social computing (CPSCom) and IEEE smart data (SmartData), 2018, pp. 1574–1581.

[62] B. K. Mohanta, D. Jena, S. S. Panda, S. Sobhanayak, Blockchain technology: A survey on applications and security privacy challenges, Internet of Things 8 (2019) 100107, publisher: Elsevier.

[63] B. Wen, Y. Wang, Y. Ding, H. Zheng, B. Qin, C. Yang, Security and privacy protection technologies in securing blockchain applications, Information Sciences (2023) 119322Publisher: Elsevier.

[64] A. Zamyatin, M. Al-Bassam, D. Zindros, E. Kokoris-Kogias, P. Moreno-Sanchez, A. Kiayias, W. J. Knottenbelt, Sok: Communication across distributed ledgers, in: Financial cryptography and data security: 25th international conference, FC 2021, virtual event, march 1–5, 2021, revised selected papers, part II 25, Springer, 2021, pp. 3–36.

[65] R. Belchior, A. Vasconcelos, S. Guerreiro, M. Correia, A Survey on Blockchain Interoperability: Past, Present, and Future Trends, ACM Computing Surveys 54 (8) (2021) 1–41.

[66] Z. Zheng, S. Xie, H.-N. Dai, X. Chen, H. Wang, Blockchain challenges and opportunities: A survey, International journal of web and grid services 14 (4) (2018) 352–375, publisher: Inderscience Publishers (IEL).

[67] A. A. Monrat, O. Schelén, K. Andersson, A survey of blockchain from the perspectives of applications, challenges, and opportunities, IEEE access : practical innovations, open solutions 7 (2019) 117134–117151, publisher: IEEE.

[68] D. Chen, G. Doumeingts, F. Vernadat, Architectures for enterprise integration and interoperability: Past, present and future, Computers in Industry 59 (7) (2008) 647–659, publisher: Elsevier. doi:10.1016/J.COMPIND.2007.12.016.

[69] World Economic Forum, Bridging the Governance Gap: Interoperability for blockchain and legacy systems, Tech. rep. (2020).

[70] World Economic Forum, Defining interoperability – digital currency governance consortium white paper series, Tech. rep. (Nov. 2021).

[71] L. Pawczuk, M. Gogh, N. Hewett, Inclusive deployment of blockchain for supply chains: Part 6 – a framework for blockchain interoperability in collaboration with deloitte, Tech. rep., World Economic Forum (2020).

[72] N. Atzei, M. Bartoletti, T. Cimoli, A survey of attacks on ethereum smart contracts, in: International conference on principles of security and trust, Association for Computing Machinery, 2017, pp. 164–186, tex.mendeley-tags: Support.

[73] R. Belchior, PhD Thesis Proposal - Blockchain Interoperability, Tech. rep., Instituto Superior Técnico (Sep. 2021).

[74] R. Belchior, S. Guerreiro, A. Vasconcelos, M. Correia, A Survey on Business Process View Integration: Past, Present and Future Applications to Blockchain, Business Process Management JournalTex.copyright: All rights reserved (Jan. 2022). doi:10.1108/BPMJ-11-2020-0529.

[75] K. Sandor, E. Genç, The fall of terra: A timeline of the meteoric rise and crash of UST and LUNA (2023). URL http://tinyurl.com/5aemybfc

[76] S. Heiler, Semantic interoperability, ACM Computing Surveys (CSUR) (1995).

[77] P. Wegner, Interoperability, ACM Computing Surveys 28 (1) (1996).

[78] W. Litwin, L. Mark, N. Roussopoulos, Interoperability of multiple autonomous databases, ACM Computing Surveys (CSUR) 22 (3) (1990) 267–293, publisher: ACM PUB27 New York, NY, USA. doi:10.1145/96602.96608.
URL https://dl.acm.org/doi/abs/10.1145/96602.96608

[79] A. Tolk, J. A. Muguira, The levels of conceptual interoperability model, in: Simulation interoperability workshop, 2003.

[80] N. Ide, J. Pustejovsky, What does interoperability mean, anyway? Toward an operational definition of interoperability for language technology, Conference on Global Interoperability (2010).
URL http://www.language-archives.org

[81] T. Hardjono, A. Lipton, A. Pentland, Toward an interoperability architecture for blockchain autonomous systems, IEEE Transactions on Engineering ManagementTex.mendeley-groups: Survey Block Int Backup,Fenix: Gateways tex.mendeley-tags: 3_BLOCKCHAIN$_C$ONNECTOR,INCLUDED,REVIEWED (2019).

[82] J. Manyika, C. Roxburgh, The great transformer: The impact of the Internet on economic growth and prosperity, Report, McKinsey Global Institute, issue: 0360-8581 (2011).

[83] R. Belchior, A. Vasconcelos, M. Correia, T. Hardjono, HERMES: Fault-Tolerant Middleware for Blockchain Interoperability, Future Generation Computer SystemsTex.copyright: All rights reserved (Apr. 2022).

[84] G. Falazi, U. Breitenbücher, F. Daniel, A. Lamparelli, F. Leymann, V. Yussupov, Smart contract invocation protocol (SCIP): A protocol for the uniform integration of heterogeneous blockchain smart contracts, CAiSE 2020 1 (2020) 134–149, iSBN: 9783030494353. doi:10.1007/978-3-030-49435-3.

[85] G. Verdian, P. Tasca, C. Paterson, G. Mondelli, Quant overledger whitepaper v0.1, Tech. rep., Quant (2018).

[86] E. J. Scheid, T. Hegnauer, B. Rodrigues, B. Stiller, Bifröst: a modular blockchain interoperability API, in: IEEE 44th conference on local computer networks, Institute of Electrical and Electronics Engineers (IEEE), 2019, pp. 332–339.

[87] Blockdaemon, One blockchain API to access multiple protocols. Blockdaemon provides Fully indexed API access to blockchain data using REST or Websockets (2023).
URL https://www.blockdaemon.com/nodes/universal-api

[88] ISO, ISO 11354-1:2011 - Advanced automation technologies and their applications — Requirements for establishing manufacturing enterprise process interoperability — Part 1: Framework for enterprise interoperability (2011).

[89] G. d. S. S. Leal, W. Guédria, H. Panetto, Interoperability assessment: A systematic literature review, Computers in Industry 106 (2019) 111–132, publisher: Elsevier.

[90] R. S. P. Maciel, P. H. Valle, K. S. Santos, E. Y. Nakagawa, Systems Interoperability Types: A Tertiary Study, arXiv:2310.19999 [cs] (Oct. 2023). doi:10.48550/arXiv.2310.19999.
URL http://arxiv.org/abs/2310.19999

[91] M. Hargreaves, T. Hardjono, R. Belchior, Secure Asset Transfer Protocol (SATP), Internet Draft draft-ietf-satp-core-02, Internet Engineering Task Force, num Pages: 37 (Jul. 2023).
URL https://datatracker.ietf.org/doc/draft-ietf-satp-core

[92] S. Gilbert, N. Lynch, Perspectives on the CAP theorem, Computer 45 (2) (2012) 30–36, publisher: IEEE.

[93] G.-T. Nguyen, K. Kim, A survey about consensus algorithms used in blockchain., Journal of Information processing systems 14 (1) (2018).

[94] J. Barreiro-Gomez, H. Tembine, Blockchain token economics: A mean-field-type game perspective, IEEE access : practical innovations, open solutions 7 (2019) 64603–64613, publisher: IEEE.

[95] R. Zhang, R. Xue, L. Liu, Security and privacy on blockchain, ACM Computing Surveys (CSUR) 52 (3) (2019) 1–34, publisher: ACM New York, NY, USA.

[96] G. Wang, Z. Jerry, M. Nixon, SoK : Sharding on blockchain, in: ACM conference on advances in financial technologies, 2019, tex.mendeley-tags: INCLUDED,REVIEWED,Support.

[97] L. Gudgeon, P. Moreno-Sanchez, S. Roos, P. McCorry, A. Gervais, Sok: Layer-two blockchain protocols, in: Financial cryptography and data security: 24th international conference, FC 2020, kota kinabalu, malaysia, february 10–14, 2020 revised selected papers 24, Springer, 2020, pp. 201–226.

[98] Web3Foundation, Architecture · Polkadot Wiki (Jan. 2024).
URL https://wiki.polkadot.network/docs/learn-architecture

[99] Cosmos, Introduction | Cosmos Hub (2024).
URL http://tinyurl.com/3s86zshc

[100] R. Belchior, L. Riley, T. Hardjono, A. Vasconcelos, M. Correia, Do You Need a Distributed Ledger Technology Interoperability Solution?, Distributed Ledger Technologies: Research and Practice (ACM DLT) 2 (1) (2022) 1–37. doi:10.1145/3564532.

[101] G. Wang, Z. J. Shi, M. Nixon, S. Han, Sok: Sharding on blockchain, in: Proceedings of the 1st ACM conference on advances in financial technologies, 2019, pp. 41–61.

[102] H. Montgomery, H. Borne-Pons, J. Hamilton, M. Bowman, P. Somogyvari, S. Fujimoto, T. Takeuchi, T. Kuhrt, R. Belchior, Hyperledger Cactus Whitepaper, Tech. rep., Hyperledger, issue: 2 tex.copyright: All rights reserved (2022).

[103] A. Augusto, R. Belchior, T. Hardjono, A. Vasconcelos, M. Correia, Multi-Party Cross-Chain Asset Transfers (Jun. 2023). doi:10.36227/techrxiv.20016815.v4.
URL https://www.techrxiv.org/articles/preprint/Resilient_Gateway-Based_N-N_Cross-Chain_Asset_Transfers/20016815/4

[104] R. Belchior, P. Somogyvari, J. Pfannschmidt, A. Vasconcelos, M. Correia, Hephaestus: Modeling, Analysis, and Performance Evaluation of Cross-Chain Transactions, IEEE Transactions on Reliability (2023) 1–15doi:10.1109/TR.2023.3336246.
URL https://ieeexplore.ieee.org/document/10363680/

[105] M. M. Lankhorst, H. A. Proper, H. Jonkers, The architecture of the archimate language, in: International workshop on business process modeling, development and support, Springer, 2009, pp. 367–380.

[106] V. Buterin, R3 report - chain interoperability, Tech. rep., R3 Corda (2016).
URL https://www.r3.com/wp-content/uploads/2017/06/chain_interoperability_r3.pdf

[107] T. Koens, E. Poll, Assessing interoperability solutions for distributed ledgers, Pervasive and Mobile Computing 59 (2019) 101079, publisher: Elsevier. doi:10.1016/J.PMCJ.2019.101079.

[108] A. Singh, K. Click, R. M. Parizi, Q. Zhang, A. Dehghantanha, K. K. R. Choo, Sidechain technologies in blockchain networks: An examination and state-of-the-art review, Journal of Network and Computer Applications 149 (2020).

[109] P. Robinson, Survey of crosschain communications protocols, Computer Networks 200 (2021) 108488, publisher: Elsevier. doi:10.1016/J.COMNET.2021.108488.
URL https://linkinghub.elsevier.com/retrieve/pii/S1389128621004321

[110] I. A. Qasse, M. A. Talib, Q. Nasir, Inter blockchain communication: A survey, in: ArabWIC 6th annual international conference research track, 2019.

[111] S. Johnson, P. Robinson, J. Brainard, Sidechains and interoperability, arXiv preprint arXiv:1903.04077 (2019).

[112] V. A. Siris, P. Nikander, S. Voulgaris, N. Fotiou, D. Lagutin, G. C. Polyzos, Interledger approaches, IEEE access : practical innovations, open solutions 7 (2019) 89948–89966, publisher: Institute of Electrical and Electronics Engineers Inc.

[113] N. Kannengießer, M. Pfister, M. Greulich, S. Lins, A. Sunyaev, Bridges between islands: Cross-chain technology for distributed ledger technology, Hawaii International Conference on System Sciences (2020).

[114] M. Bishnoi, R. Bhatia, Interoperability solutions for blockchain, Proceedings of the International Conference on Smart Technologies in Computing, Electrical and Electronics, ICSTCEE 2020 (2020) 381–385.

[115] A. Lohachab, S. Garg, B. Kang, M. Bilal, J. Lee, S. Chen, X. Xu, Towards interconnected blockchains: A comprehensive review of the role of interoperability among disparate blockchains, ACM Computing Surveys (CSUR) 54 (7) (2021) 1–39, publisher: ACM PUB27 New York, NY, USA. doi:10.1145/3460287.
URL https://dl.acm.org/doi/abs/10.1145/3460287

[116] A. Zamyatin, D. Harz, J. Lind, P. Panayiotou, A. Gervais, W. Knottenbelt, XCLAIM: Trustless, Interoperable, Cryptocurrency-Backed Assets, in: 2019 IEEE Symposium on Security and Privacy (SP), 2019, pp. 193–210, iSSN: 2375-1207. doi:10.1109/SP.2019.00085.
URL https://ieeexplore.ieee.org/document/8835387

[117] Q. Zhao, Y. Wang, B. Yang, K. Shang, M. Sun, H. Wang, Z. Yang, X. He, A comprehensive overview of security vulnerability penetration methods in blockchain cross-chain bridges, Authorea (Authorea) (Oct. 2023). doi:https://doi.org/10.22541/au.169760541.13864334/v1.
URL http://tinyurl.com/4mnrs676

[118] L. Duan, Y. Sun, W. Ni, W. Ding, J. Liu, W. Wang, Attacks against cross-chain systems and defense approaches: A contemporary survey, IEEE/CAA Journal of Automatica Sinica 10 (8) (2023) 1643–1663, publisher: IEEE/CAA Journal of Automatica Sinica.

[119] T. Haugum, B. Hoff, M. Alsadi, J. Li, Security and Privacy Challenges in Blockchain Interoperability - A Multivocal Literature Review, in: Proceedings of the International Conference on Evaluation and Assessment in Software Engineering 2022, EASE '22, Association for Computing Machinery, New York, NY, USA, 2022, pp. 347–356. doi:10.1145/3530019.3531345.
URL https://doi.org/10.1145/3530019.3531345

[120] S.-S. Lee, A. Murashkin, M. Derka, J. Gorzny, SoK: Not quite water under the bridge: Review of cross-chain bridge hacks, arXiv preprint arXiv:2210.16209 (2022).

[121] Augusto, Andre, Belchior, Rafael, Correia, Miguel, Vasconcelos, Andre, Zhang, Luyao, Hardjono, Thomas, SoK: Security and Privacy of Blockchain Interoperability (2023).
URL http://tinyurl.com/soksp

[122] V. Buterin, vitalik.eth on Twitter, arguments for a multi-chain future (2022).

[123] E. Androulaki, A. Barger, V. Bortnikov, S. Muralidharan, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Murthy, C. Ferris, G. Laventman, Y. Manevich, B. Nguyen, M. Sethi, G. Singh, K. Smith, A. Sorniotti, C. Stathakopoulou, M. Vukolić, S. W. Cocco, J. Yellick, Hyperledger fabric: A distributed operating system

for permissioned blockchains, in: Proceedings of the 13th EuroSys conference, EuroSys 2018, Vol. 2018-Janua, Association for Computing Machinery, Inc, New York, New York, USA, 2018, pp. 1–15.

[124] J. Kwon, E. Buchman, Cosmos whitepaper, Tech. rep., Cosmos Foundation (2016).

[125] G. Wood, Polkadot: Vision for a heterogeneous multi-chain framework, White paper 21 (2327) (2016) 4662.

[126] V. Buterin, Vitalik Buterin on cross-chain bridges (2022).
URL https://www.reddit.com/r/ethereum/comments/rwojtk/ama_we_are_the_efs_research_team_pt_7_07_january/hrngyk8/

[127] H. Qureshi, Axelar, bridges, and blockchain globalization (Jun. 2022).
URL https://medium.com/dragonfly-research/axelar-bridges-and-blockchain-globalization-11ef3bbce9f1

[128] Ethereum Engineering Group, Security of crosschain transactions and bridges (Nov. 2022).
URL https://www.youtube.com/watch?v=DJyEJVaXMNo

[129] L2BEAT – The state of the layer two ecosystem (2023).
URL https://l2beat.com/scaling/summary

[130] P. KidBold, The wormhole bridge attack explained (Feb. 2022).
URL https://kaicho.substack.com/p/the-wormhole-bridge-attack-explained

[131] C. Faife, Wormhole cryptocurrency platform hacked for $325 million after error on GitHub (Feb. 2022).
URL http://tinyurl.com/vv83wyvr

[132] Rekt, Rekt - ronin network (2022).
URL https://www.rekt.news/

[133] R. Behnke, Explained: The wormhole hack (february 2022) (Feb. 2022).
URL https://halborn.com/explained-the-wormhole-hack-february-2022/

[134] FreddieChopin, Harmony hack (Jun. 2022).
URL https://www.reddit.com/r/CryptoCurrency/comments/vlt4xs/fyi_the_hacker_who_exploited_harmony_bridge_for/

[135] M. Barrett, Harmony's horizon bridge hack (Jun. 2022).
URL https://medium.com/harmony-one/harmonys-horizon-bridge-hack-1e8d283b6d66

[136] C. Faife, Nomad crypto bridge loses 200 million in chaotic hack (Aug. 2022).
URL http://tinyurl.com/2hcx67vm

[137] B. Liu, $80M lost in first hack of 2024, section: DeFi (Jan. 2024).
URL https://blockworks.co/news/80-million-lost-orbit-bridge

[138] The Straits Times, Cryptocurrency-bridge hacks top 1.36 billion in little over a year (Apr. 2022).
URL http://tinyurl.com/bddxa4zw

[139] The Block Research, Largest DeFi exploits (2022).
URL https://www.theblock.co/data/decentralized-finance/exploits/largest-defi-exploits

[140] J. Yerushalmy, The SEC has approved bitcoin ETFs. What are they and what does it mean for investors?, The Guardian (Jan. 2024).
URL http://tinyurl.com/y5xw5j5y

[141] M. Herlihy, Atomic cross-chain swaps, in: Proceedings of the 2018 ACM symposium on principles of distributed computing, 2018, pp. 245–254.

[142] H. H. Perritt Jr, Format and content standards for the electronic exchange of legal information, Jurimetrics J. 33 (1992) 265, publisher: HeinOnline.

[143] P. Voigt, A. Von dem Bussche, The eu general data protection regulation (gdpr), A Practical Guide, 1st Ed., Cham: Springer International Publishing 10 (3152676) (2017) 10–5555, publisher: Springer.

[144] H. Li, L. Yu, W. He, The impact of GDPR on global technology development, issue: 1 Pages: 1–6 Volume: 22 (2019).

[145] K. Peffers, T. Tuunanen, M. A. Rothenberger, S. Chatterjee, A design science research methodology for information systems research, Journal of management information systems 24 (3) (2007) 45–77, publisher: Taylor & Francis.

[146] A. Hevner, S. Chatterjee, A. Hevner, S. Chatterjee, Design science research in information systems, Design research in information systems: theory and practice (2010) 9–22Publisher: Springer.

[147] R. C. Moore, Making the transition to formal proof, Educational Studies in mathematics 27 (3) (1994) 249–266, publisher: Springer.

[148] R. Wieringa, Empirical research methods for technology validation: Scaling up to practice, Journal of systems and software 95 (2014) 19–31, publisher: Elsevier.

[149] B. Kaplan, J. A. Maxwell, Qualitative research methods for evaluating computer information systems, in: Evaluating the organizational impact of healthcare information systems, Springer, 2005, pp. 30–55.

[150] U. Dayal, H.-Y. Hwang, View definition and generalization for database integration in a multidatabase system, IEEE Transactions on Software Engineering (6) (1984) 628–645, publisher: IEEE.

[151] R. Belchior, D. Dimov, Z. Karadjov, J. Pfannschmidt, A. Vasconcelos, M. Correia, Harmonia: Securing Cross-Chain Applications Using Zero-Knowledge Proofs, Submitted to IEEE Transactions on Dependable and Secure Computing (Dec. 2023). doi:10.36227/techrxiv.170327806.66007684/v1.
URL http://tinyurl.com/zkharmonia

[152] A. Vázquez-Ingelmo, A. García-Holgado, F. J. García-Peñalvo, C4 model in a Software Engineering subject to ease the comprehension of UML and the software, in: 2020 IEEE global engineering education conference (EDUCON), 2020, pp. 919–924. doi:10.1109/EDUCON45650.2020.9125335.

[153] C. Kobryn, Modeling components and frameworks with UML, Communications of the ACM 43 (10) (2000) 31–38, publisher: ACM New York, NY, USA.

[154] R. Belchior, M. Correia, A. Augusto, T. Hardjono, SATP Gateway Crash Recovery Mechanism, Internet Draft draft-belchior-satp-gateway-recovery-00, Internet Engineering Task Force, num Pages: 28 (Jul. 2023).
URL https://datatracker.ietf.org/doc/draft-belchior-satp-gateway-recovery

[155] Y. Jardi, The Unified Future of AI, Blockchain and Virtual Worlds in 2024, section: Consensus Magazine (Dec. 2023).
URL http://tinyurl.com/5ysvdvfv

[156] C. Nyman, Ten Blockchain and Digital Assets trends to expect in 2024 eBook (Jan. 2024).
URL https://www.bpm.com/insights/ten-blockchain-and-digital-assets-trends-to-expect-in-2024/

[157] M. Borkowski, C. Ritzer, D. McDonald, S. Schulte, Caught in chains: Claim-first transactions for cross-blockchain asset transfers, authority: Technische Universität Wien (2018).
URL http://tinyurl.com/4z6wax99

[158] G. Caldarelli, J. Ellul, The blockchain oracle problem in decentralized finance—A multivocal approach, Applied Sciences (Switzerland) 11 (16) (2021). doi:10.3390/app11167572.

[159] S. Eskandari, M. Salehi, W. Catherine Gu, J. Clark, SoK: Oracles from the ground truth to market manipulation; SoK: Oracles from the ground truth to market manipulation, arxiv 2106.00667ISBN: 9781450390828 tex.arxivid: 2106.00667v2 (2021). doi:10.1145/3479722.3480994.
URL https://doi.org/10.1145/3479722.3480994

[160] R. Mühlberger, S. Bachhofner, E. Castelló Ferrer, C. Di Ciccio, I. Weber, M. Wöhrer, U. Zdun, Foundational oracle patterns: Connecting blockchain to the off-chain world, in: Lecture notes in business information processing, Vol. 393 LNBIP, Springer Science and Business Media Deutschland GmbH, 2020, pp. 35–51, iSSN: 18651356 tex.arxivid: 2007.14946. doi:10.1007/978-3-030-58779-6{_}3.
URL https://link.springer.com/chapter/10.1007/978-3-030-58779-6_3

[161] T. Li, P. Niu, Y. Wang, S. Zeng, X. Wang, W. Susilo, HT2REP: A fair cross-chain atomic exchange protocol under UC framework based on HTLCs and TRE, Computer Standards & Interfaces (2024) 103834doi:10.1016/j.csi.2024.103834.
URL https://www.sciencedirect.com/science/article/pii/S0920548924000035

[162] I. Tsabary, M. Yechieli, A. Manuskin, I. Eyal, MAD-HTLC: because HTLC is crazy-cheap to attack, in: 2021 IEEE symposium on security and privacy (SP), IEEE, 2021, pp. 1230–1248.

[163] J. Cai, Y. Zhou, T. Hu, B. Li, PTLC: Protect the identity privacy during cross-chain asset transaction more effectively, in: 2022 IEEE 22nd international conference on software quality, reliability, and security companion (QRS-C), IEEE, 2022, pp. 70–78.

[164] F. Barbàra, C. Schifanella, MP-HTLC: Enabling blockchain interoperability through a multiparty implementation of the hash time-lock contract, Concurrency and Computation: Practice and Experience 35 (9) (2023) e7656, publisher: Wiley Online Library.

[165] Y. Xue, D. Jin, M. Herlihy, Fault-tolerant and expressive cross-chain swaps, in: Proceedings of the 24th international conference on distributed computing and networking, 2023, pp. 28–37.

[166] R. Belchior, S. Scuri, I. Mihaiu, N. Nunes, T. Hardjono, Towards a Common Standard Framework for Blockchain Interoperability - A Position Paper (Aug. 2023). doi:10.36227/techrxiv.17093039.v3.
URL http://tinyurl.com/upcwuutb

[167] Subramanian,Shankar, Augusto, André, Belchior , Rafael, Benchmarking Bridge Aggregators, Tech. rep., Techrxiv (2024).
URL https://www.techrxiv.org/users/687326/articles/697988-benchmarking-bridge-aggregators

[168] J. O. Chervinski, D. Kreutz, X. Xu, J. Yu, Analyzing the performance of the inter-blockchain communication protocol, arXiv preprint arXiv:2303.10844 (2023).

[169] Welcome to the LI.FI API (2024).
URL https://apidocs.li.fi/reference/welcome-to-the-lifinance-api

[170] G. Hatzivasilis, I. Papaefstathiou, C. Manifavas, Password hashing competition-survey and benchmark, Cryptology ePrint Archive (2015).

[171] M. Bellés-Muñoz, M. Isabel, J. L. Muñoz-Tapia, A. Rubio, J. Baylina, Circom: A circuit description language for building zero-knowledge applications, IEEE Transactions on Dependable and Secure ComputingPublisher: IEEE (2022).

[172] J. Ernstberger, S. Chaliasos, G. Kadianakis, S. Steinhorst, P. Jovanovic, A. Gervais, B. Livshits, M. Orrù, zk-Bench: A Toolset for Comparative Evaluation and Performance Benchmarking of SNARKs, publication info: Preprint. (2023).
URL https://eprint.iacr.org/2023/1503

[173] EC, Markets in Crypto-Assets Regulation (MiCA) (2023).
URL https://www.esma.europa.eu/esmas-activities/digital-finance-and-innovation/markets-crypto-assets-regulation-mica

[174] S. Hu, M. Li, J. Weng, J.-N. Liu, J. Weng, Z. Li, IvyRedaction: Enabling Atomic, Consistent and Accountable Cross-Chain Rewriting, IEEE Transactions on Dependable and Secure Computing (2023) 1–18Conference Name: IEEE Transactions on Dependable and Secure Computing. doi:10.1109/TDSC.2023.3339675.
URL https://ieeexplore.ieee.org/abstract/document/10342807

[175] A. Sood, The FinCEN Travel Rule (Feb. 2021).
URL https://alessa.com/blog/fincen-travel-rule/

[176] FAFT, Targeted Update on Implementation of FATF's Standards on VAs and VASPs (2023).
URL https://www.fatf-gafi.org/en/publications/Fatfrecommendations/Targeted-update-virtual-assets-vasps.html

[177] T. Hardjono, A. Lipton, A. Pentland, On standardized service interfaces for the interoperability of tokenized asset networks (2023).
URL http://tinyurl.com/5n76za2u

[178] M. Hargreaves, T. Hardjono, R. Belchior, Secure Asset Transfer Protocol (SATP), Internet Draft draft-hargreaves-sat-core-02, Internet Engineering Task Force, num Pages: 29 (Mar. 2023).
URL https://datatracker.ietf.org/doc/draft-hargreaves-sat-core

[179] Economy, Business, Luna crypto crash wipes out savings of thousands of investors, sparking fears for sector | Economy and Business | EL PAÍS English (2022).
URL http://tinyurl.com/27tp8cj4

[180] Y. Chen, A. Asheralieva, X. Wei, AtomCI: A new system for the atomic cross-chain smart contract invocation spanning heterogeneous blockchains, IEEE Transactions on Network Science and EngineeringPublisher: IEEE (2024).

[181] P. Robinson, R. Ramesh, S. Johnson, Atomic crosschain transactions for ethereum private sidechains, Blockchain: Research and Applications 3 (1) (2022) 100030, publisher: Elsevier.

[182] G. Falazi, U. Breitenbücher, F. Leymann, S. Schulte, V. Yussupov, Transactional cross-chain smart contract invocations, Distributed Ledger Technologies: Research and PracticePublisher: ACM New York, NY (2023).

[183] T. Xie, J. Zhang, Z. Cheng, F. Zhang, Y. Zhang, Y. Jia, D. Boneh, D. Song, zkBridge: Trustless Cross-chain Bridges Made Practical, in: Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS '22, Association for Computing Machinery, New York, NY, USA, 2022, pp. 3003–3017. doi:10.1145/3548606.3560652.
URL https://dl.acm.org/doi/10.1145/3548606.3560652

[184] NEAR, NEAR Selects Wormhole as an Official Bridge and Launches ZK-Enabled Cross-Chain Communication with Ethereum (May 2023).
URL https://wormhole.com/near-selects-wormhole-as-an-official-bridge/

[185] Wormhole Labs emerges from stealth to accelerate innovation on Wormhole (Nov. 2023).
URL https://finance.yahoo.com/news/wormhole-labs-emerges-stealth-accelerate-140000115.html

[186] Nomad Announces $22.4 Million Seed Round to Build More Secure Cross-Chain Interoperability (Apr. 2022).
URL https://www.businesswire.com/news/home/20220414005497/en/%C2%A0Nomad-Announces-22.4-Million-Seed-Round-to-Build-More-Secure-Cross-Chain-Interoperability

[187] Uniswap, Notion – The all-in-one workspace for your notes, tasks, wikis, and databases. (2023).
URL https://www.notion.so

[188] B. Schneier, The future of incident response, IEEE Security & Privacy 12 (5) (2014) 96–96, publisher: IEEE.

[189] R. Ganguly, Y. Xue, A. Jonckheere, P. Ljung, B. Schornstein, B. Bonakdarpour, M. Herlihy, Distributed Runtime Verification of Metric Temporal Properties for Cross-Chain Protocols, arXiv:2204.09796 [cs] (Apr. 2022).
URL http://arxiv.org/abs/2204.09796

[190] Y. Gai, L. Zhou, K. Qin, D. Song, A. Gervais, Blockchain large language models, arXiv preprint arXiv:2304.12749 (2023).

[191] M. Atif, Analysis and verification of two-phase commit & three-phase commit protocols, in: 2009 international conference on emerging technologies, IEEE, 2009, pp. 326–331.

[192] E. Ben-Sasson, A. Chiesa, M. Riabzev, N. Spooner, M. Virza, N. P. Ward, Aurora: Transparent succinct arguments for R1CS, in: Advances in Cryptology–EUROCRYPT 2019: 38th annual international conference on the theory and applications of cryptographic techniques, darmstadt, germany, may 19–23, 2019, proceedings, part I 38, Springer, 2019, pp. 103–128.

[193] D. Boneh, B. Bünz, B. Fisch, Batching techniques for accumulators with applications to IOPs and stateless blockchains, in: Advances in Cryptology–CRYPTO 2019: 39th annual international cryptology conference, santa barbara, CA, USA, august 18–22, 2019, proceedings, part I 39, Springer, 2019, pp. 561–586.

[194] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, G. Maxwell, Bulletproofs: Short proofs for confidential transactions and more, in: 2018 IEEE symposium on security and privacy (SP), IEEE, 2018, pp. 315–334.

[195] A. Gabizon, Z. J. Williamson, O. Ciobotaru, Plonk: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge, Cryptology ePrint Archive (2019).

[196] Aayush, An opinionated overview of ZK tooling and proof systems right now (Sep. 2023).
URL https://blog.aayushg.com/posts/zk/

[197] D. Čapko, S. Vukmirović, N. Nedić, State of the art of zero-knowledge proofs in blockchain, in: 2022 30th telecommunications forum (TELFOR), IEEE, 2022, pp. 1–4.

[198] J. Ernstberger, S. Chaliasos, L. Zhou, P. Jovanovic, A. Gervais, Do You Need a Zero Knowledge Proof?, publication info: Published elsewhere. CfC St. Moritz Academic Research Track 2024 (2024).
URL https://eprint.iacr.org/2024/050

[199] EEA resources, authority: Enterprise Ethereum Alliance (2023).
URL https://entethalliance.org/resources/

[200] Blockchain community group, authority: World Wide Web Consortium (W3C) (2023).
URL https://www.w3.org/community/blockchain/

[201] A. Obadia, A. Salles, L. Sankar, T. Chitra, V. Chellani, P. Daian, Unity is Strength: A Formalization of Cross-Domain Maximal Extractable Value, arXiv:2112.01472 [cs] (Dec. 2021). doi:10.48550/arXiv.2112.01472.
URL http://arxiv.org/abs/2112.01472

[202] C. McMenamin, SoK: Cross-Domain MEV, arXiv:2308.04159 [cs] (Aug. 2023).
URL http://arxiv.org/abs/2308.04159

[203] J. H. Sjursen, W. Meng, W.-Y. Chiu, Towards Quantifying Cross-Domain Maximal Extractable Value for Blockchain Decentralisation, in: Information and Communications Security: 25th International Conference, ICICS 2023, Tianjin, China, November 18–20, 2023, Proceedings, Springer-Verlag, Berlin, Heidelberg, 2023, pp. 627–644. doi:10.1007/978-981-99-7356-9_37.
URL https://doi.org/10.1007/978-981-99-7356-9_37

[204] O. Mazor, O. Rottenstreich, An Empirical Study of Cross-chain Arbitrage in Decentralized Exchanges, publication info: Published elsewhere. Minor revision. International Conference on Communication Systems and Networks (COMSNETS) 2024 (2023).
URL https://eprint.iacr.org/2023/1898

[205] S. Chhina, M. Chadhar, S. Firmin, A. Tatnall, Understanding blockchain adoption from an institutional theory perspective (2023).

[206] R. Auer, M. Farag, U. Lewrick, L. Orazem, M. Zoss, Banking in the Shadow of Bitcoin? The Institutional Adoption of Cryptocurrencies (2023). doi:10.2139/ssrn.4416784.
URL https://papers.ssrn.com/abstract=4416784

[207] B. O. Ansah, W. Voss, K. O. Asiama, I. Y. Wuni, A systematic review of the institutional success factors for blockchain-based land administration, Land Use Policy 125 (2023) 106473, publisher: Elsevier.

[208] L. Wintermeyer, Big Financial Institutions Are Adopting Crypto And Blockchain — What Does The Technology Offer SMBs?, section: Fintech.
URL http://tinyurl.com/4pc7y6ka

[209] R. Yin, Z. Yan, X. Liang, H. Xie, Z. Wan, A survey on privacy preservation techniques for blockchain interoperability, Journal of Systems Architecture (2023) 102892Publisher: Elsevier.

[210] A. Sanchez, A. Stewart, F. Shirazi, Bridging sapling: Private cross-chain transfers, in: 2022 IEEE crosschain workshop (ICBC-CROSS), IEEE, 2022, pp. 1–9.

[211] A. Deshpande, M. Herlihy, Privacy-preserving cross-chain atomic swaps, in: Financial cryptography and data security: FC 2020 international workshops, AsiaUSEC, CoDeFi, VOTING, and WTSC, kota kinabalu, malaysia, february 14, 2020, revised selected papers, Springer, 2020, pp. 540–549.

[212] Cai,Jingyuan, Zhou, Ying, Hu, Tianyuan, Li, Bixin, PTLC: Protect the Identity Privacy during Cross-Chain Asset Transaction More Effectively, 2022.
URL https://ieeexplore.ieee.org/abstract/document/10077065

[213] Z. Wang, S. Chaliasos, K. Qin, L. Zhou, L. Gao, P. Berrang, B. Livshits, A. Gervais, On how zero-knowledge proof blockchain mixers improve, and worsen user privacy, in: Proceedings of the ACM web conference 2023, 2023, pp. 2022–2032.

[214] B. C. Ghosh, V. Ramakrishna, C. Govindarajan, D. Behl, D. Karunamoorthy, E. Abebe, S. Chakraborty, Decentralized cross-network identity management for blockchain interoperation, in: 2021 IEEE international conference on blockchain and cryptocurrency (ICBC), IEEE, 2021, pp. 1–9.

[215] Zecchini,Marco, Sober, Michael, Schulte, Stefan, Vitaletti,Andrea, Building a Cross-Chain Identity: A Self-Sovereign Identity-Based Framework, 2023.
URL https://ieeexplore.ieee.org/document/10237010

[216] J. S. Brown, K. VanLehn, Towards a generative theory of "bugs", in: Addition and subtraction, Routledge, 2020, pp. 117–135.

[217] T. Holt, A. Bossler, Cybercrime in progress: Theory and prevention of technology-enabled offenses, Routledge, 2015.

[218] I. Ghafir, V. Prenosil, J. Svoboda, M. Hammoudeh, A survey on network security monitoring systems, in: 2016 IEEE 4th international conference on future internet of things and cloud workshops (FiCloudW), IEEE, 2016, pp. 77–82.

[219] Hyperledger Cacti, original-date: 2019-10-21T16:06:38Z (Oct. 2023).
URL https://github.com/hyperledger/cacti

[220] RFCs, authority: Internet Engineering Task Force (IETF) (2023).
URL https://www.ietf.org/standards/rfcs/

[221] S. Ghaemi, S. Rouhani, R. Belchior, R. S. Cruz, H. Khazaei, P. Musilek, A Pub-Sub Architecture to Promote Blockchain Interoperability, arXiv:2101.12331 [cs] version: 1 (Jan. 2021). doi:10.48550/arXiv.2101.12331.
URL http://arxiv.org/abs/2101.12331

[222] K. Wüst, A. Gervais, Do you need a blockchain?, in: 2018 crypto valley conference on blockchain technology (CVCBT), IEEE, 2018, pp. 45–54.

[223] M. Nofer, P. Gomber, O. Hinz, D. Schiereck, Blockchain, Business & Information Systems Engineering 59 (2017) 183–187, publisher: Springer.

[224] N. Szabo, Formalizing and securing relationships on public networks 2 (9), publisher: First Monday (1997).

[225] A. M. Antonopoulos, G. Wood, Mastering {Ethereum}: building smart contracts and dapps, O'Reilly Media, 2018.

[226] Z. Zheng, S. Xie, H. Dai, X. Chen, H. Wang, An overview of blockchain technology: Architecture, consensus, and future trends, in: Proceedings - 2017 IEEE 6th international congress on big data, BigData congress 2017, Institute of Electrical and Electronics Engineers Inc., 2017, pp. 557–564.

[227] R. Brown, The corda platform: An introduction white paper (2018).
URL https://www.r3.com/reports/the-corda-platform-an-introduction-whitepaper/

[228] JP Morgan, Quorum white paper (2017).
URL https://github.com/jpmorganchase/quorum/blob/master/docs/QuorumWhitepaperv0.2.pdf

[229] M. Conti, K. E. Sandeep, C. Lal, S. Ruj, A survey on security and privacy issues of bitcoin, IEEE Communications Surveys and Tutorials 20 (4) (2018) 3416–3452, publisher: Institute of Electrical and Electronics Engineers Inc.

[230] S. Heiler, Semantic interoperability, ACM Computing Surveys (CSUR) 27 (2) (1995) 271–273, publisher: ACM New York, NY, USA.

[231] D. Chen, N. Daclin, Framework for enterprise interoperability, in: Interoperability for enterprise software and applications: Proceedings of the workshops and the doctorial symposium of the second IFAC/IFIP I-ESA international conference: EI2N, WSI, IS-TSPQ 2006, Wiley Online Library, 2006, pp. 77–88.

[232] S. P. Hufnagel, Interoperability, Military medicine 174 (suppl_5) (2009) 43–50, publisher: Oxford University Press.

[233] M. L. Zeng, Interoperability, KO Knowledge Organization 46 (2) (2019) 122–146, publisher: Nomos Verlagsgesellschaft mbH & Co. KG.

[234] I. P. Zarko, S. Soursos, I. Gojmerac, E. G. Ostermann, G. Insolvibile, M. Plociennik, P. Reichl, G. Bianchi, Towards an IoT framework for semantic and organizational interoperability, GIoTS 2017 - Global Internet of Things Summit, ProceedingsISBN: 9781509058730 Publisher: Institute of Electrical and Electronics Engineers Inc. (Aug. 2017). doi:10.1109/GIOTS.2017.8016253.

[235] M. Thabet, M. Boufaida, F. Kordon, An approach for developing an interoperability mechanism between cloud providers, International Journal of Space-Based and Situated Computing 4 (2) (2014) 88, publisher: Inderscience Publishers. doi:10.1504/IJSSC.2014.062469.

[236] K. Kaur, S. Sharma, K. S. Kahlon, Interoperability and portability approaches in inter-connected clouds: A review, ACM Computing Surveys 50 (4) (2017). doi:10.1145/3092698.

[237] W. Li, L. Ping, Trust model to enhance security and interoperability of cloud environment, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 5931 LNCS (2009) 69–79, iSBN: 3642106641 Publisher: Springer, Berlin, Heidelberg. doi:10.1007/978-3-642-10665-1{_}7.
URL https://link.springer.com/chapter/10.1007/978-3-642-10665-1_7

[238] H. L'Amrani, B. E. Berroukech, Y. El Bouzekri El Idrissi, R. Ajhoun, Toward interoperability approach between federated systems, in: Proceedings of the 2nd international conference on big data, cloud and applications, 2017. doi:10.1145/3090354.3090391.

[239] R. Klischewski, A. Tagamoa, A. Khames, Information integration or process integration? How to achieve interoperability in administration, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 3183 (2004) 57–65, iSBN: 3540229167 Publisher: Springer, Berlin, Heidelberg. doi:10.1007/978-3-540-30078-6{_}10.
URL https://link.springer.com/chapter/10.1007/978-3-540-30078-6_10

[240] J. Ofoeda, R. Boateng, J. Effah, Application programming interface (API) research: A review of the past to inform the future, International Journal of Enterprise Information Systems (IJEIS) 15 (3) (2019) 76–95, publisher: IGI Global.

[241] K. J. Sullivan, W. G. Griswold, Y. Cai, B. Hallen, The structure and value of modularity in software design, ACM SIGSOFT Software Engineering Notes 26 (5) (2001) 99–108, publisher: ACM New York, NY, USA.

[242] A. Tolk, S. Y. Diallo, C. D. Turnitsa, Applying the levels of conceptual interoperability model in support of integratability, interoperability, and composability for system-of-systems engineering, Journal of Systems, Cybernetics, and Informatics 5 (5) (2007).

[243] C. Casiano Flores, A. P. Rodriguez Müller, V. Albrecht, J. Crompvoets, T. Steen, E. Tambouris, Towards the inclusion of co-creation in the european interoperability framework, in: Proceedings of the 14th international conference on theory and practice of electronic governance, 2021, pp. 538–540.

[244] A. Kouroubali, D. G. Katehakis, The new European interoperability framework as a facilitator of digital transformation for citizen empowerment, Journal of biomedical informatics 94 (2019) 103166, publisher: Elsevier.

[245] R. Belchior, DLT Interoperability and More 28 — SoK: Cross-Domain MEV (Sep. 2023).
URL https://rafaelbelchior.medium.com/dlt-interoperability-and-more-%EF%B8%8F-28-sok-cross-domain-mev-%EF%B8%8F-477971a4887e

[246] I. L. Traiger, J. Gray, C. A. Galtieri, B. G. Lindsay, Transactions and Consistency in Distributed Database Systems, IBM Research Report RJ2555 (1979).

[247] F. Rowe, What literature review is not: diversity, boundaries and recommendations, European Journal of Information Systems 23 (3) (2014) 241–255, publisher: Taylor & Francis.

[248] B. Kitchenham, S. Charters, Guidelines for performing systematic literature reviews in software engineering version 2.3, Tech. rep., Keele University and University of Durham (2007).
URL https://www.elsevier.com/__data/promis_misc/525444systematicreviewsguide.pdf

[249] H. Sun, H. Mao, X. Bai, Z. Chen, K. Hu, W. Yu, Multi-blockchain model for central bank digital currency, in: Parallel and distributed computing, applications and technologies, PDCAT proceedings, Vol. 2017-Decem, IEEE Computer Society, 2018, pp. 360–367, tex.mendeley-tags: Support. doi:10.1109/PDCAT.2017.00066.

[250] M. Christodorescu, W. C. Gu, R. Kumaresan, M. Minaei, M. Ozdayi, B. Price, S. Raghuraman, M. Saad, C. Sheffield, M. Xu, M. Zamani, Towards a two-tier hierarchical infrastructure: An offline payment system for central bank digital currencies, arXiv 2012.08003 (2020) 1–21ArXiv: 2012.08003 tex.arxivid: 2012.08003 tex.mendeley-groups: 0 - To read.
URL http://arxiv.org/abs/2012.08003

[251] BIS Innovation Hub, Central bank digital currencies: foundational principles and core features (Oct. 2020).
URL https://www.bis.org/publ/othp33.pdf

[252] A. Bechtel, A. Ferreira, J. Gross, P. Sandner, The future of payments in a DLT-based european economy: A roadmap, in: The future of financial systems in the digital age, Springer, Singapore, 2022, pp. 89–116.

[253] U. Emanuele, B. Alessia, C. Daniele, C. Angela, C. Marco, F. Simone, G. Giuseppe, G. Giancarlo, M. Gabriele, T. Pietro, V. Alessia, A digital euro: a contribution to the discussion on technical design choices, Mercati, infrastrutture, sistemi di pagamento (10) (2021).
URL https://www.bancaditalia.it/pubblicazioni/mercati-infrastrutture-e-sistemi-di-pagamento/questioni-istituzionali/2021-010/N.10-MISP.pdf

[254] Banque de France, Digital euro experiment, combined feasibility – tiered model (Jul. 2021).
URL https://www.banque-france.fr/sites/default/files/media/2021/08/02/821220_digital_euro_en.pdf

[255] R. Auer, P. Haene, H. Holden, others, Multi-CBDC arrangements and the future of cross-border payments, BIS PapersPublisher: Bank for International Settlements (2021).

[256] V. Ramakrishna, T. Hardjono, draft-ietf-satp-usecases-01 (2023).

[257] V. Ramakrishna, T. Hardjono, Secure Asset Transfer (SAT) Use Cases, Internet Draft draft-ramakrishna-sat-use-cases-02, Internet Engineering Task Force, num Pages: 12 (Mar. 2023).
URL https://datatracker.ietf.org/doc/draft-ramakrishna-sat-use-cases

[258] J. S. Waterman, The promissory note as a substitute for money, Minnesota Law Review 14 (1929) 313, publisher: HeinOnline tex.mendeley-groups: Fenix: Gateways.

[259] FQX, eNI™ infrastructure - fqx.ch - electronic negotiable instruments - FQX, tex.mendeley-groups: Fenix: Gateways (2020).
URL https://fqx.ch/

[260] FQX, Transatlantic shipment of metals financed via FQX eNote, tex.mendeley-groups: Fenix: Gateways (2021).
URL https://treasury-management.com/news/transatlantic-shipment-of-metals-financed-via-fqx-enote/

[261] C. E. W. Group, hyperledger-labs/blockchain-carbon-accounting (2021).
URL https://github.com/hyperledger-labs/blockchain-carbon-accounting

[262] Carbon Accounting and Certification WG, Carbon accounting and certification WG - climate action SIG - hyperledger foundation (2021).
URL https://wiki.hyperledger.org/display/CASIG/Carbon+Accounting+and+Certification+WG

[263] Caldarelli, Giulio, Before Ethereum. The Origin and Evolution of Blockchain Oracles (2023).
URL https://ieeexplore.ieee.org/abstract/document/10131932

[264] P. Daian, S. Goldfeder, T. Kell, Y. Li, X. Zhao, I. Bentov, L. Breidenbach, A. Juels, Flash boys 2.0: Frontrunning in decentralized exchanges, miner extractable value, and consensus instability, Proceedings - IEEE Symposium on Security and Privacy 2020-May (2020) 1106–1120, iSBN: 9781728134970 Publisher: Institute of Electrical and Electronics Engineers Inc. doi:10.1109/SP40000.2020.00040.

[265] Chainlink, What Is an Oracle in Blockchain? (2023).
URL https://chain.link/education/blockchain-oracles

[266] Band, Band Protocol - Cross-Chain Data Oracle (2023).
URL https://bandprotocol.com

[267] API3, API3 | The Web3 API Economy (2023).
URL https://api3.org/

Other contributions

During the course of this thesis, we have provided several non-scientific contributions. These main ones can be categorized into software, standardization, and education contributions. This list does not count service to the community as a reviewer of academic papers, talks, panels, participation in podcasts and interviews, contributions to newspapers, appearances on media (examples here[1]), and events attended. Those can be found here[2].

A.1 Software - Open Source Technology for Common Good

Counting almost 300 stars and over 240 forks, Hyperledger Cacti [219] is the flagship open-source enterprise interoperability project, backed by Hyperledger, Accenture, Fujitsu, IBM, and others. The code of the proof of concepts from most of our scientific contributions is incorporated into the main codebase of Cacti, being available for researchers and practitioners alike. We also contributed to managing the project, increasing its reach, recruiting contributors, organizing workshops, and adding diverse features, mostly on the infrastructure side. Our contributions are tracked in the issues and pull request lists in Github[3].

Another collaboration is with IGFEJ - Instituto de Gestão Financeira e Equipamentos da Justiça IP, at the Portuguese government, where we developed Justicechain, a blockchain-based solution to standardize and secure justice data. This project yielded the following publication [25]. Other collaborations include participation in the H2020 European Project: Qualichain: Decentralised Qualifications on the Blockchain, where we developed the identity management layer[4], and the "Consortium and Certificate Management module"[5]. Taking a step further, we investigate decentralized access control mechanisms, yielding SSIBAC [37]. We are currently contributing to a Portuguese PRR project called Blockchain.pt[6].

[1] https://spectrum.ieee.org/blockchain-interoperability, and https://www.publico.pt/autor/rafael-belchior
[2] https://rafaelapb.github.io//academic
[3] https://github.com/hyperledger/cacti/issues/created_by/RafaelAPB
[4] https://github.com/QualiChain/identity-access-management
[5] https://github.com/QualiChain/consortium
[6] https://blockchain.void.pt/

A.2 Standardization - IETF

On the other hand, our work on Hermes [83] yielded several drafts in the context of a forming working group at the Internet Engineering Task Force, a standardization organization responsible for TLS, HTTPS, TCP/IP and others [220]. Two IETF drafts are the direct outcome of our work: the SATP core draft [91] and the Crash Recovery draft [154]. We manage these and related drafts in collaboration with MIT Connection Science and other institutions on a GitHub repository[7]. We hope that our standardization effort, which takes academics and practitioners, can yield RFCs[8], paving the way for standardization in the space.

A.3 Education, Volunteering, and Mentorship

During my PhD, I taught several courses as an Assistant Professor at Instituto Superior Técnico and was a guest lecturer at NOVA IMS Information Management School. In collaboration with Quant Overledger, we authored the cryptography section of the blockchain course "A Beginner's Guide to Becoming a Blockchain Developer with Overledger", a course counting with more than 2650 students[9]. We collaborated with the Linux Foundation and Hyperledger Foundation on blockchain research. We created the first free Hyperledger Cacti workshop[10] in both English and Portuguese, counting now with more than 110 participants and 1,500 views. With the same institutions, we created a university-level course on blockchain called "Enterprise Blockchain Technologies", focusing on Hyperledger Fabric[11]. This course had more than 80 students. We did over 20 talks throughout these PhD studies, listed at `https://rafaelapb.github.io//academic`.

We provide other educational content to the community via our research blog series: "DLT Interoperability And More", available on Medium[12]. So far, we have analyzed more than 30 academic papers in this series, with a focus on blockchain interoperability, the theme of this thesis. We served as mentors in Hyperledger-backed projects[13], namely:

- The Hyperledger Fabric-Based Access Control project[14]: The goal is to create a blockchain-based access control system using Hyperledger Fabric. This project yielded our paper *Distributed attribute-based access control system using permissioned blockchain* [36], published in the journal World Wide Web.
- The Towards Blockchain Interoperability with Hyperledger[15]: the goal is to create a pub-sub architecture promoting blockchain interoperability. This project yielded our technical report *A Pub-Sub Architecture to Promote Blockchain Interoperability* [221].
- The Visualization and Analysis of Cross-chain Transactions[16]: the goal is to create a system to visualize and analyze cross-chain transactions. The findings of this paper inspired Hephaestus [104].
- Technical Deep Dive Workshop Content Creation for Hyperledger Cactus[17]: the goal is to create example applications to support the Hyperledger Cacti workshop.

[7] https://github.com/ietf-satp
[8] https://www.ietf.org/standards/rfcs/
[9] https://www.futurelearn.com/courses/become-a-blockchain-developer-foundations
[10] https://old.hyperledger.org/event/blockchain-interoperability-with-hyperledger-cacti-workshop
[11] https://github.com/hyperledger-labs/university-course
[12] https://medium.com/@rafaelbelchior
[13] see our mentor profile here: https://mentorship.lfx.linuxfoundation.org/mentor/0fdc2b17-5b9e-48e6-a4aa-17c2468e4829
[14] https://wiki.hyperledger.org/display/INTERN/Hyperledger+Fabric+Based+Access+Control
[15] https://wiki.hyperledger.org/display/INTERN/Towards+Blockchain+Interoperability+with+Hyperledger
[16] https://wiki.hyperledger.org/display/INTERN/Visualization+and+Analysis+of+Cross-chain+Transactions
[17] https://wiki.hyperledger.org/display/INTERN/Technical+Deep+Dive+Workshop+Content+Creation+for+Hyperledger+Cactus

- Business Logic Plugins for Hyperledger Cactus[18]: the goal is to create example applications to support developer adoption. This project contributed to SATP's first implementation.
- Cross-Chain State Modelling and Analysis[19]: the goal is to develop applications that manage cross-chain state visualization.
- Benchmarking Cross-Chain Bridges[20]: the goal is to benchmark interoperability mechanisms. This work contributed to the first cross-chain bridge benchmark available [167].
- Cacti: Implement Standardized Secure Asset Transfer Protocol[21]: the goal is to provide a second reference implementation of SATP in Rust, contributing to the diversity of implementations of the protocol.

A.4 Service to the Community

During this doctoral thesis, we were co-chairs of ICBC Crosschain 2024, the IEEE International Conference on Blockchain and Cryptocurrency Crosschain workshop[22], fourth edition.

The objectives of the workshop are to:

- Provide researchers with a forum to publish peer-reviewed papers on cross-chain communications.
- Ensure that the researchers working on cross-chain will be aware of each other's work.
- Help developers stay across the latest research.
- Provide a forum for practitioners to talk about how cross-chain is being used and the challenges that they face.
- Have a cross-pollination of ideas. Help us all gain a better understanding of the trade-offs of the various approaches.

During this doctoral dissertation, we were members of the technical program committees of several international conferences and journals, namely, IEEE NCA, ECIS, IEEE ICBC, Hyperledger Global Forum, Symposium on Distributed Ledger Technology, IEEE DSN; ACM DLT, Future Generation Computing Systems, IEEE Access, Business Process Management Journal, Frontiers in Blockchain, IEEE Transactions on Network and Service Management, IEEE Transactions on Reliability, International Journal of Information Security, Tech4Good workshop affiliated with ICDCS).

[18] https://wiki.hyperledger.org/display/INTERN/Cactus-samples+-+Business+Logic+Plugins+for+Hyperledger+Cactus
[19] https://wiki.hyperledger.org/display/INTERN/Cross-Chain+State+Modelling+and+Analysis
[20] https://wiki.hyperledger.org/display/INTERN/Benchmarking+Cross-Chain+Bridges
[21] https://wiki.hyperledger.org/display/INTERN/Cacti%3A+Implement+Standardized+Secure+Asset+Transfer+Protocol
[22] https://icbc2024.ieee-icbc.org/workshop/crosschain

B Complementary Background

This Appendix explains technical terms and background in greater detail.

B.1 Introduction to Blockchain

The term *blockchain* has at least two different meanings: a type of system and a type of data structure. In this document, we use the term blockchain to denominate a class of distributed systems. A blockchain maintains a shared state, specifically a replicated data structure that we denominate *distributed ledger*. This ledger is maintained by a set of machines with computational and storage resources, called nodes (or peers or participants). Nodes are not trusted individually to maintain the distributed ledger; they are trusted as a group due to their number and diversity. A blockchain can also be considered a *deterministic state machine* that provides a certain service, given existing incentives that the network can reward. The first blockchain was part of the Bitcoin system and provided as service transactions of a cryptocurrency, a digital currency, also designated Bitcoin. The service provided by Bitcoin is the execution of transactions of bitcoins. Many technical descriptions of blockchain technology exist [222, 223, 66].

B.2 Smart Contracts

Most blockchains are programmable, i.e., their state machine is extensible with user programs. These programs are often designated *smart contracts* [224, 20] and their execution is caused by calls also designated *transactions*. Smart contracts are executed in a virtual machine, e.g., in the Ethereum Virtual Machine (EVM) in Ethereum and other blockchains that adopted the EVM for compatibility (that we designate *EVM-based blockchains*). Smart contracts are often used to implement *tokens*, i.e., blockchain-based abstractions that can be owned and represent currency, resources, assets, access, equity, identity, collectibles, etc. [225]. There are several standard token formats, e.g., ERC-20 and ERC-721. These tokens are fungible and non-fungible assets, respectively. A fungible asset is interchangeable with another asset of the same type. Conversely, a non-fungible asset is an asset that is unique and has specific properties.

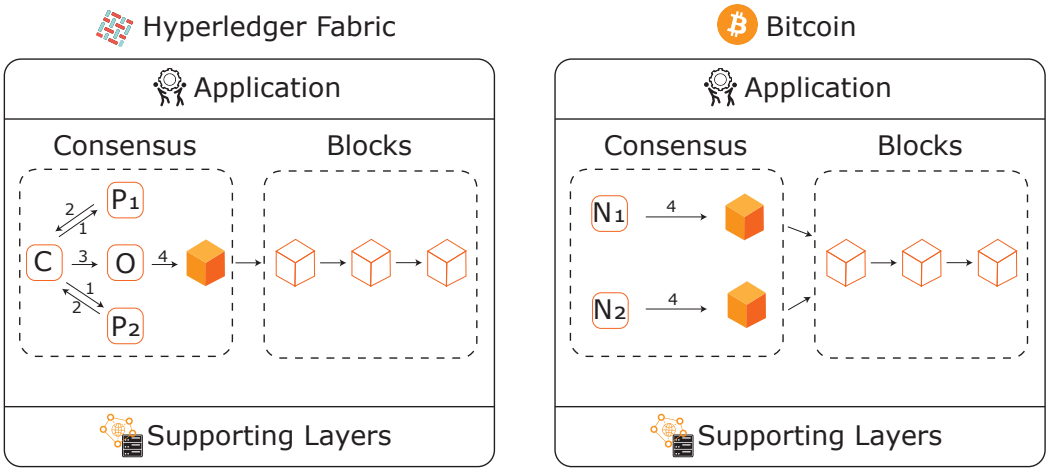


Figure B.1: Two blockchains: Hyperledger Fabric, and Bitcoin.

In many blockchains, transactions are aggregated in *blocks*, linked by the previous block's cryptographic hash. Hence those data structures are also called blockchains - often viewed as deterministic state machines.

Blockchain systems ought to be resilient to faults (e.g., *crash fault-tolerant* or *Byzantine fault-tolerant*), as there may be crashes or malicious nodes on the network [7]. They run a consensus algorithm to create agreement on a global ledger state in the presence of Byzantine faults. Consensus algorithms are important because they define the behavior of blockchain nodes and their interaction [7, 226], and the security assumptions of each blockchain. They, therefore, affect how blockchain peers communicate and operate with each other: in Bitcoin's Proof-of-Work (PoW), peers have to compute a cryptographic challenge to validate transactions, competing with each other. Another blockchain, Tendermint, uses a Byzantine fault-tolerant state machine replication (BFT) algorithm [124], supporting up to a third less one of faulty participants. In Hyperledger Fabric, a widely-used private blockchain platform, a consensus algorithm allows higher transaction throughput than PoW by allowing a subset of nodes to execute and endorse transactions (called endorser peers) and by typically using a weaker consensus (only crash fault-tolerant). The variety of blockchain infrastructures makes it challenging to categorize blockchains, and their interoperability solutions, as there are no *de facto* blockchain interoperability or blockchain architecture standards.

Apart from differences in the consensus, blockchains can be deemed *public* (also called permissionless) or *private* (also called permissioned). Permissionless blockchains do not require authentication for participants to access the ledger. *Bitcoin* [8] and *Ethereum* [20, 21] are examples of such blockchains. Permissioned blockchains are blockchains in which users are authenticated and can be held accountable according to a governance model suitable for enterprise and governmental needs. Hyperledger Fabric [123], Corda [227], Quorum [228], and Tendermint [124] are examples of permissioned blockchains.

Figure B.1 depicts two blockchains: Hyperledger Fabric, a permissioned blockchain; and Bitcoin, a permissionless blockchain. The supporting layers (e.g., networking, storage, encryption) provide a basis for the consensus engine, which orders transactions and appends them to the chain of blocks. In Hyperledger Fabric, the consensus is modular, based on endorsement policies. In Fabric, a client (C) sends a transaction proposal to the peer nodes (P) and obtains a signed transaction, called an endorsement (steps 1 and 2). An orderer validates the endorsements and builds a block with valid transactions, appending it to the ledger (steps 3 and 4). In Bitcoin, the consensus is based on the notion of Proof-of-Work (PoW), a cryptographic puzzle that mining nodes need to solve to build a valid block. This corresponds roughly to Fabric's steps 1-3. After a node finds a solution to PoW, it then can propose a block of transactions to be appended to the ledger (step 4).

Blockchain trust is based on the incentive models that guide the behavior of the nodes. For instance, in Bitcoin, nodes have the incentive to produce blocks of transactions and support the network because they are rewarded Bitcoins. Conversely, nodes are not incentivized to disrespect the protocol, as attacks are expensive and nodes can get punished [229]. In Hyperledger Fabric, where nodes are identified, they have the business incentive to follow the protocol because parties cooperate towards a common goal, and misbehavior can be punished according to the law or applicable governance model. Decentralization, different goals, and incentives support trust in the blockchain- parties can share the ledger without relying on a trusted, centralized party.

The ability to distribute trust on a global state fostered the appearance of *decentralized applications* (*dApps*) [225]. A dApp is a computer program running on a decentralized peer-to-peer network. Thus, dApps are based on smart contracts running on a blockchain, but they also have other components that should equally be decentralized.

B.3 Interoperability of Traditional Software Systems versus Blockchain

Coming with a detailed and comprehensive review of the interoperability field is a challenging task[1], as interoperability is a key factor for the extensibility, modularity, and performance of software systems. Interoperability of software systems has been studied extensively in the last decades [77, 230, 231, 232], with more up-to-date studies [233, 89] focusing on software systems for specific domains. Other works evaluate interoperability for IoT [234], cloud providers [235, 236, 237], and more generic ones [238, 239]. The latest comprehensive study, from Maciel et al. [90], surveys the literature from 2012 to 2023, focusing on blockchain and the Internet of Things (IoT).

Interoperability is a non-functional requirement of software systems that refers to the process of communication and inter-cooperation among systems, components, or modules without generating a technological dependency among them [77], in the same machine or over the internet. This cooperation can be realized by establishing well-defined interfaces for functions, modules, web services, and application programming interfaces, a common procedure in software engineering [240, 241]. The heterogeneity of interoperability pushes beyond well-defined interfaces: it has to deal with systems implemented in different programming languages, technologies, frameworks, standards, communication protocols, and platforms. With the potential to refer to different dimensions that a software system spawns across, interoperability can be decomposed into different layers, according to its interoperability model (this is, a set of well-defined rules to classify interoperability layers, for example, [242]). A multitude of frameworks exist [90]. As an example, there are interoperability dimensions that consider the interoperability across data formats (synthetic), across programs or logic (semantic), across business processes (organizational), and across jurisdictions (legal) - interoperability is multidimensional and spawns across different research communities and application domains. An important reference in this field is the European Interoperability Framework [243, 244], where we based much of our work [65, 100].

Interoperability challenges in DLT systems differ significantly from those in traditional software environments due to the unique characteristics of distributed ledger technologies. We emphasize a few: 1) in blockchain networks, the decentralized nature and the absence of a central orchestrator create a fundamentally different landscape for achieving interoperability. Unlike traditional software systems where interoperability often involves standardized protocols and centralized integration points [245], blockchain interoperability must navigate

[1] as a reference, the number of Google Scholar results for the prompt "interoperability" is around 1,160,000 results, while the prompt "interoperability software engineering" returns around half a million results. There are at least 38 literature reviews in the area of interoperability [90].

a complex web of autonomous networks [178], each with its own consensus mechanisms, governance models, and cryptographic primitives. This decentralized architecture means establishing interoperability involves aligning incentives, ensuring security across multiple chains, and maintaining the core principles of decentralization and trust minimization - a more coordinated effort between all interoperability layers needs to be done [245]. Moreover, interoperability at the semantic layer (e.g., cross-chain transaction for asset transfers) must be verifiable, often requiring proof to ensure the integrity of cross-chain operations [4].

B.4 Cross-Chain Transactions

To realize transactions across distributed ledgers, we envision a distributed ledger system as an abstract representation of a distributed database. In this design, multiple replicas maintain a global state using a consensus algorithm. The global state is changed via user-submitted transactions, similar to conventional databases. Changing the state is subject to transactions adhering to specific consistency rules. Consistency rules are enforced in each database (i.e., blockchain) locally, but, additionally, have more restrictions (i.e., consistency rules) coming from the cross-chain logic [104, 141].

In practice, these consistency rules are restrictions in a sequence of read and write operations, orchestrated across different chains. However, unlike traditional databases, a distributed shared ledger lacks a singular or unitary entity that can be relied upon for reading from or writing to it. Instead, the internal consensus protocol assumes the responsibility of ensuring safety and liveness. Typically, cross-chain transactions respect a set of properties equivalent to ACID [246, 83]. Different techniques to provide ACID-like properties to cross-chain transactions enforce the correctness of cross-chain protocols, namely that cross-chain transactions are atomic: either all the local transactions are executed correctly and committed to the underlying ledger, or none are. The underlying technical challenge is *how to ensure that two or more distributed ledgers mutually agree on a specific ledger state within a defined time limit, unidirectionally or bidirectionally?*. The more researchers worked on this problem, the clearer the solutions: proving state with cryptographic proofs [151], the usage of timelocks [211], the use of state-locking [83].

C A Survey on Blockchain Interoperability: Past, Present, and Future Trends

The following chapter corresponds to the following publication [65]. The respective online appendix can be found in the ACM Library[1]:

Status: Published ✓

Publication: ACM Computing Surveys

Submitted: 1 June 2020

Accepted: 31 May 2021

Citation: R. Belchior, A. Vasconcelos, S. Guerreiro, and M. Correia, "A Survey on Blockchain Interoperability: Past, Present, and Future Trends," ACM Computing Surveys, vol. 54, no. 8, pp. 1–41, May 2021.

Research Methodology: Systematic literature review [247, 248]

Journal Description: Comprehensive, readable surveys and tutorial papers give guided tours through the literature and explain topics to those who seek to learn the basics of areas outside their specialties in an accessible way. The carefully planned and presented introductions in Computing Surveys (CSUR) are also an excellent way for researchers and professionals to develop perspectives on, and identify trends in complex technologies. Contributions which bridge existing and emerging technologies with a variety of science and engineering domains in a novel and interesting way are also welcomed.

H-Index: 190 **Coverage**: 1969-present **Quartile**: Q1 **Scimago Journal Rank 2022**: 4.46

[1] https://dl.acm.org/doi/10.1145/3471140

A Survey on Blockchain Interoperability: Past, Present, and Future Trends

RAFAEL BELCHIOR, ANDRÉ VASCONCELOS, SÉRGIO GUERREIRO, and MIGUEL CORREIA, INESC-ID, Instituto Superior Técnico, Universidade de Lisboa, Portugal

Blockchain interoperability is emerging as one of the crucial features of blockchain technology, but the knowledge necessary for achieving it is fragmented. This fact makes it challenging for academics and the industry to achieve interoperability among blockchains seamlessly. Given this new domain's novelty and potential, we conduct a literature review on blockchain interoperability by collecting 284 papers and 120 grey literature documents, constituting a corpus of 404 documents. From those 404 documents, we systematically analyzed and discussed 102 documents, including peer-reviewed papers and grey literature. Our review classifies studies in three categories: Public Connectors, Blockchain of Blockchains, and Hybrid Connectors. Each category is further divided into sub-categories based on defined criteria. We classify 67 existing solutions in one sub-category using the Blockchain Interoperability Framework, providing a holistic overview of blockchain interoperability. Our findings show that blockchain interoperability has a much broader spectrum than cryptocurrencies and cross-chain asset transfers. Finally, this article discusses supporting technologies, standards, use cases, open challenges, and future research directions, paving the way for research in the area.

CCS Concepts: • **Computer systems organization → Dependable and fault-tolerant systems and networks**; **Peer-to-peer architectures;**

Additional Key Words and Phrases: Survey, blockchain interoperability, standards, interconnected blockchains, cross-chain transactions, cross-blockchain communication

ACM Reference format:
Rafael Belchior, André Vasconcelos, Sérgio Guerreiro, and Miguel Correia. 2021. A Survey on Blockchain Interoperability: Past, Present, and Future Trends. *ACM Comput. Surv.* 54, 8, Article 168 (October 2021), 41 pages.
https://doi.org/10.1145/3471140

1 INTRODUCTION

Blockchain technology is maturing at a fast pace. The development of real-world applications shows real interest from both industry and academia [176, 201]. For instance, applications have been developed in the areas of public administration [18, 22], access control [20, 152], and others [47]. Additionally, blockchain is progressing toward the performance of centralized systems: for

The authors express their gratitude to the Linux Foundation for providing the Diversity & Inclusion scholarship. This work was partially supported by the EC through project 822404 (QualiChain), and by national funds through Fundação para a Ciência e a Tecnologia (FCT) with reference UIDB/50021/2020 (INESC-ID).
Authors' address: R. Belchior, A. Vasconcelos, S. Guerreiro, and M. Correia, INESC-ID, Instituto Superior Técnico, Universidade de Lisboa, Rua Alves Redol, 9, Lisboa 1000-029, Portugal; emails: {rafael.belchior, andre.vasconcelos, sergio.guerreiro, miguel.p.correia}@tecnico.ulisboa.pt.

0360-0300/2021/10-ART168 $15.00
https://doi.org/10.1145/3471140

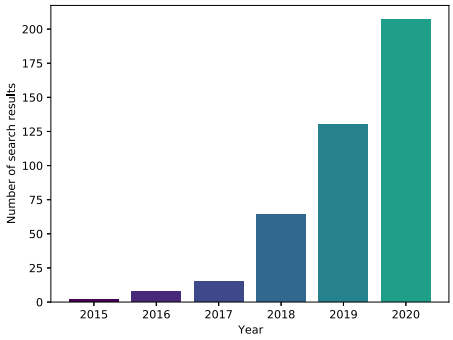


Fig. 1. Research trends on blockchain interoperability.

example, the Hyperledger Fabric blockchain is predicted to achieve 50,000 *transactions per second* [79, 80]. Figure 1 depicts the number of search results per year for "blockchain interoperability" that Google Scholar returned. In 2015, only two documents were related to blockchain interoperability. In 2016, 2017, 2018, 2019, and 2020, the results were 8, 15, 64, 130, and 207, respectively, showing a steep increase regarding interest in this research area.

Serving multiple use cases and stakeholders requires various blockchain features and capabilities [185]. The need for adaptability is a motivating factor for creating different blockchains, leading to a heterogeneous ecosystem [84, 139, 191]. Choosing new blockchains allows researchers and developers to implement new use case scenarios and keep up with recent endeavors. However, each blockchain has its security risks, as the technology is still maturing, the user base is limited (e.g., in comparison to the web or databases), and there are uncovered bugs, and security flaws [12]. Therefore, developers and researchers have to choose between novelty and stability, leading to a vast diversity of choices [6]. This diversity leads to *fragmentation*: there are many *immature* blockchain solutions (e.g., without extensive testing). Until recently, blockchains did not consider the need for interoperability, as each one focused on resolving specific challenges, leading to *data and value silos* [1, 99, 171].

Moreover, what if the blockchain in which a particular service is running becomes obsolete, vulnerable, or is shut down? If the user requirements or circumstances change over time, a different blockchain might be more appropriate for a specific use case [124]. What if the service to serve is so crucial that it requires seamless dependability? Furthermore, if we want to reproduce our use case to another blockchain, how can we increase *portability*?

In 1996, Wegner stated that "interoperability is the ability of two or more software components to cooperate despite differences in language, interface, and execution platform" [186]. In that context, Wegner established a bridge between the concept of interoperability and existing standards. As authors were influenced by the standards existing at that time, authors nowadays are influenced by the Internet architecture and concepts, in what concerns blockchain interoperability [86, 170]. Thus, reflecting on the Internet's architecture seems like a good starting point to understand how blockchains can interoperate. Thus, it is important to solve the *blockchain interoperability* challenge, i.e., to provide interoperability between blockchains in order to explore synergies between different solutions, scale the existing ones, and create new use cases (see Section 2.3). For example, a user should be able to transfer their assets from a blockchain to another or build *cross-blockchain decentralized applications*.

While information systems evolve, so do the meaning and scope of interoperability. According to the **National Interoperability Framework Observatory** (**NIFO**), endorsed by the European Commission, there are several interoperability layers [133]: *technical interoperability*, *semantic interoperability*, *organizational interoperability*, *legal interoperability*, *integrated public*

service governance, and *interoperability governance*. For instance, technical interoperability regards the technical mechanisms that enable integration among blockchains, while semantic interoperability concerns whether the application-specific semantics can be conserved across blockchains. Despite interoperability having an extensive scope, we mainly focus on *technical interoperability*, and *semantic interoperability* as most blockchain interoperability work is concentrated. We leave the study of other interoperability layers for future work.

Interoperability does not only conflate flexibility and application portability. It also has the potential to solve some of the biggest blockchain research challenges. In particular, interoperability promotes blockchain *scalability*, as it provides a way to offload transactions to other blockchains, e.g., via sharding [75, 181], it can promote privacy (by allowing the end-user to use different blockchain for data objects with different privacy requirements), and creates new business opportunities. Given the complexity of this research area, we attempt to answer three research questions:

RQ1: What is the current landscape concerning blockchain interoperability, both in industry and academia?
RQ2: Are technological requirements for blockchain interoperability currently satisfied?
RQ3: Are there real use cases requiring blockchain interoperability?

1.1 Contributions

As a systematization of knowledge on blockchain interoperability, this article yields fourfold contributions:

—Introduce the blockchain interoperability research area, presenting the necessary background and highlighting definitions tailored both for industry and academia. We define blockchain interoperability and discuss different blockchain interoperability architectures and standards.
—Propose the **Blockchain Interoperability Framework** (**BIF**), a framework defining criteria to assess blockchain interoperability solutions.
—Present a *systematic literature review*, where we identify and discuss blockchain interoperability solutions, according*a* to BIF, in three categories: *Public Connectors*, *Blockchain of Blockchains*, and *Hybrid Connectors*. In particular, our analysis is based on several sources (e.g., peer-reviewed papers, whitepapers, blog posts, technical reports), enabling an in-depth understanding of each solution's current state and its *roadmap*, i.e., its creator's plans. To achieve this, *we systematically contacted the authors of grey literature papers and industrial solutions*: this is our innovative attempt to provide the reader with high-quality information in this rapidly emerging research area. This method allows us to obtain up-to-date, reliable information that often is cumbersome to obtain.
—We identify and propose use cases that benefit from a multiple blockchain approach, pinpoint challenges and obstacles to the development of blockchain interoperability solutions and standards, and propose future research directions, paving the way for systematic research in this area.

1.2 Organization

Section 2 provides background on blockchain consensus algorithms, previous results on blockchain interoperability, and blockchain interoperability definitions and architecture. Next, Section 3 presents and discusses related literature reviews, while Section 4 introduces the BIF. Next, a systematic review and analysis of blockchain interoperability categories is conducted, distributed across three categories, in Section 5: Public Connectors (Section 5.1), Blockchain of Blockchains (Section 5.2), and Hybrid Connectors (Section 5.3). For each category, we provide a detailed

analysis and discussion. To provide a holistic view of the blockchain interoperability landscape, we promote a general discussion in Section 6. This discussion compares solutions across categories (Section 6.1), presents standardization efforts (Section 6.2), informs readers regarding use case scenarios with multiple blockchains (Section 6.3), answers to the research questions (Section 6.4), and indicates challenges related to interoperability (Section 6.5). We present research directions (Section 7), and, finally, we conclude the article (Section 8). Six appendices complement this survey. Appendix A presents the methodology employed. Appendix B presents an architecture for blockchain interoperability, reviewing the various efforts on that topic. Appendices C, D, and E present a description of the surveyed public connectors, blockchain of blockchains, and hybrid connector approaches, respectively. Finally, Appendix F complements the use case section, by presenting more cross-blockchain use cases.

2 BACKGROUND

In this section, we provide the necessary background to the understanding of this survey.

2.1 A Primer on Blockchain Technology

The term *blockchain* has at least two different meanings: a type of system and a type of data structure. In this article, we use the term blockchain to denominate a class of distributed systems. A blockchain maintains a shared state, specifically a replicated data structure that we denominate *distributed ledger*. This ledger is maintained by a set of machines with computational and storage resources, called nodes (or peers or participants). Nodes are not trusted individually to maintain the distributed ledger; they are trusted as a group due to their number and diversity [46]. A blockchain can also be considered a *deterministic state machine* that provides a certain service, given existing incentives that the network can reward. The first blockchain was part of the Bitcoin system and provided as service transactions of a cryptocurrency, a digital currency, also designated Bitcoin [132]. The service provided by Bitcoin is the execution of transactions of bitcoins.

Most blockchains are programmable, i.e., their state machine is extensible with user programs. These programs are often designated *smart contracts* [43, 168] and their execution is caused by calls also designated *transactions*. Smart contracts are executed in a virtual machine, e.g., in the **Ethereum Virtual Machine** (**EVM**) in Ethereum and other blockchains that adopted the EVM for compatibility (that we designate *EVM-based blockchains*). Smart contracts are often used to implement *tokens*, i.e., blockchain-based abstractions that can be owned and represent currency, resources, assets, access, equity, identity, collectibles, and so on [8]. There are several standard token formats, e.g., ERC-20 and ERC-721. These tokens are fungible and non-fungible assets, respectively. A fungible asset is interchangeable with another asset of the same type. Conversely, a non-fungible asset is an asset that is unique and has specific properties.

In many blockchains, transactions are aggregated in *blocks*, linked by the previous block's cryptographic hash. Hence, those data structures are also called blockchains—often viewed as deterministic state machines.

Blockchain systems ought to be resilient to faults (e.g., *crash fault-tolerant* or *Byzantine fault-tolerant*), as there may be crashes or malicious nodes on the network [54]. They run a consensus algorithm to create agreement on a global ledger state in the presence of Byzantine faults. Consensus algorithms are important because they define the behavior of blockchain nodes and their interaction [54, 199], and the security assumptions of each blockchain. They, therefore, affect how blockchain peers communicate and operate with each other: in Bitcoin's **Proof-of-Work** (**PoW**), peers have to compute a cryptographic challenge to validate transactions, competing with each other. Another blockchain, Tendermint, uses a **Byzantine fault-tolerant** state machine

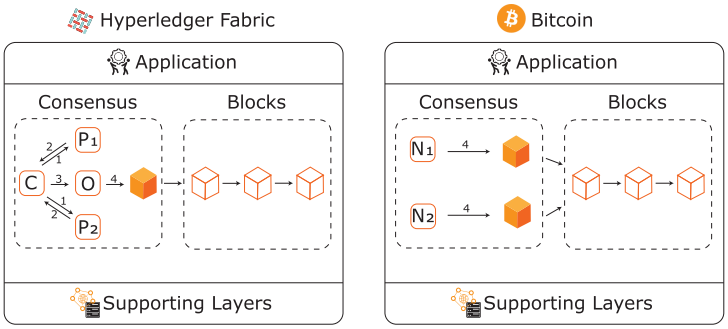


Fig. 2. Representation of two blockchains, Hyperledger Fabric [7], and Bitcoin [132].

replication (**BFT**) algorithm [109], supporting up to a third less one of faulty participants. In Hyperledger Fabric, a widely used private blockchain platform, a consensus algorithm allows higher transaction throughput than PoW by allowing a subset of nodes to execute and endorse transactions (called endorser peers) and by typically using a weaker consensus (only crash fault-tolerant). The variety of blockchain infrastructures makes it challenging to categorize blockchains, and their interoperability solutions, as there are no de facto blockchain interoperability or blockchain architecture standards.

Apart from differences in the consensus, blockchains can be deemed *public* (also called permissionless) or *private* (also called permissioned). Permissionless blockchains do not require authentication for participants to access the ledger. *Bitcoin* [132] and *Ethereum* [43, 189] are examples of such blockchains. Permissioned blockchains are blockchains in which users are authenticated and can be held accountable according to a governance model suitable for enterprise and governmental needs. Hyperledger Fabric [7], Corda [39], Quorum [101], Tendermint [109], and Multichain [81] are examples of permissioned blockchains.

Figure 2 depicts two blockchains: Hyperledger Fabric, a permissioned blockchain; and Bitcoin, a permissionless blockchain. The supporting layers (e.g., networking, storage, encryption) [102] provide a basis for the consensus engine, which orders transactions and appends them to the chain of blocks. In Hyperledger Fabric, the consensus is modular, based on endorsement policies. In Fabric, a client (C) sends a transaction proposal to the peer nodes (P), and obtains a signed transaction, called an endorsement (steps 1 and 2). An orderer validates the endorsements and builds a block with valid transactions, appending it to the ledger (steps 3 and 4). In Bitcoin, the consensus is based on the notion of PoW, a cryptographic puzzle that mining nodes need to solve in order to build a valid block. This corresponds roughly to Fabric's steps 1–3. After a node finds a solution to PoW, it then can propose a block of transactions to be appended to the ledger (step 4).

Blockchain trust is based on the incentive models that guide the behavior of the nodes. For instance, in Bitcoin, nodes have the incentive to produce blocks of transactions and support the network because they are rewarded Bitcoins. Conversely, nodes do not have the incentive to disrespect the protocol, as attacks are expensive and nodes can get punished [53]. In Hyperledger Fabric, where nodes are identified, they have the business incentive to follow the protocol because parties cooperate toward a common goal, and misbehavior can be punished according to the law or applicable governance model. Decentralization, different goals, and incentives support the trust on the blockchain—parties can share the ledger without relying on a trusted, centralized party.

The ability to distribute trust on a global state fostered the appearance of ***decentralized applications*** (***dApps***) [8]. A dApp is a computer program running on a decentralized peer-to-peer

network. For example, Steemit[1] is a social blogging dApp that rewards content creators with cryptocurrency. Thus, dApps are based on smart contracts running on a blockchain, but they also have other components that should equally be decentralized.

2.2 Cross-Blockchain Communication

Cross-blockchain communication involves two blockchains: a *source blockchain*, and a *target blockchain*. The source blockchain is the blockchain in which the transaction is initiated to be executed on a target blockchain. While general-purpose interoperability comes down to a blockchain exposing its internal state to others, cross-chain asset transfers rely on an atomic three-phase procedure: (1) locking (or extinguishing) of an asset on a source blockchain; (2) blockchain transfer commitment, and (3) creation of a representation of the asset on a target blockchain [23, 75, 88]. This procedure, later explained in detail, relies on a ***cross-chain communication protocol*** (**CCCP**).

A CCCP defines the process by which a pair of blockchains interact to synchronize cross-chain transactions correctly. Hence, a CCCP allows *homogeneous* blockchains to communicate. For instance, sidechains typically use a CCCP (e.g., Zendoo allows communication between Bitcoin-like blockchains systems [76]). Conversely, a ***cross-blockchain communication protocol*** (**CBCP**) defines the process by which a pair of blockchains interact to synchronize cross-blockchain transactions correctly. CBCPs allow *heterogeneous* blockchains to communicate (e.g., the Interledger Protocol allows any blockchains that implement the protocol to exchange "money packets" [97]). The differentiation between CCCPs and CBCPs is important because CCCPs typically can leverage the interoperating blockchains' constructs and functionality (e.g., utilize smart contracts to implement a relay [110]), whereas CBCPs normally require blockchains to be adapted. However, CBCPs may leverage specific functionalities of both blockchains [66]. Cross-blockchain, or cross-chain communication, is a requirement for blockchain interoperability. This section provides a few theoretical results regarding cross-blockchain communication, and thus also blockchain interoperability.

Zamyatin et al. [194] prove that "there exists no asynchronous **CCC** (**cross-chain communication**) protocol tolerant against misbehaving nodes." The authors use a reduction to the fair exchange problem [11] to prove that correct cross-chain communication is as hard as the fair exchange problem. As a consequence of the presented theorem, the authors state that "there exists no CCC protocol tolerant against misbehaving nodes without a trusted third party." A trusted third party can be centralized or decentralized. Centralized trusted parties are, for example, trusted validators [129]. A decentralized trusted party can be another blockchain, in which their participants agree on the global ledger state via a consensus algorithm. However, the trusted party has to ensure that most participants are honest, guaranteeing the correctness of the process is guaranteed. Cross-chain protocols therefore "use the consensus of the distributed ledgers as an abstraction for a trusted third party" [194]. Borkowski et al. [37] derive the "lemma of rooted blockchains" that states that a source blockchain cannot verify the existence of data on a target blockchain with practical effort. In particular, the source blockchain would need to be able to mimic consensus from the target blockchain, and it would have to store a (potentially large) subset of the target blockchain's block history. On a recent endeavor, Lafourcade and Lombard-Platet [112] formalize the blockchain interoperability problem, arguing that fully decentralized blockchain interoperability is not possible. More specifically, there is no protocol assuming a full client that can realize its interoperability functions, such as asset transfer, without a third party's aid. However, a blockchain with two ledgers offers the possibility of interoperability (there is, in fact, the possibility of moving assets from one ledger to the other). This study applies mainly to public blockchains.

[1] https://steemit.com/.

The results above are relevant because they lead to an important consideration: *cross-blockchain transactions are not feasible in practice without the participation of a trusted third party*. In other words, although trust assumptions vary greatly from permissionless to permissioned networks, cross-blockchain transactions, as well as cross-chain transactions, require a trusted third party to assure the correctness of the underlying protocol. Most solutions presented throughout this article present at least one decentralized trust anchor.

2.3 Blockchain Interoperability Definitions

In this section, we define additional technical terms for an understanding of this study.

Vernadat defines interoperability among enterprise systems as [179]: "a measure of the ability to perform interoperation between [...] entities (software, processes, systems, business units...). The challenge relies on facilitating communication, cooperation, and coordination among these processes and units." Abebe et al. propose a general communication protocol as an alternative approach to the "point-to-point" blockchain interoperability approach [1]. Interoperability is defined as "the semantic dependence between distinct ledgers to transfer or exchange data or value, with assurances of validity." Pillai and Biswas refer that "cross-communication is not intended to make direct state changes to another blockchain system. Instead, cross-communication should trigger some set of functionalities on the other system that is expected to operate within its own network" [138].

A technical report from the **National Institute of Standards and Technology** (**NIST**) defines blockchain interoperability as "a composition of distinguishable blockchain systems, each representing a unique distributed data ledger, where atomic transaction execution may span multiple heterogeneous blockchain systems, and where data recorded in one blockchain are reachable, verifiable, and *referable* by another possibly foreign transaction in a semantically compatible manner" [192]. Hardjono et al. define blockchain survivability as "the completion (confirmation) of an application-level transaction [composed of subtransactions] independent of blockchain systems involved in achieving the completion of the transaction"[86]. The concept of transactions and subtransactions relates to "*best effort delivery*," that applications must comply to, by ensuring that transactions and their *subtransactions* are completed (i.e., committed) within a certain time frame.

Regarding types of blockchain interoperability, Besançon et al. highlight three [27]: interoperability between different blockchains, interoperability between dApps using the same blockchain, and interoperability blockchain and other technologies (such as integration with enterprise systems). While different definitions tackle different dimensions of interoperability, there is room for improvement. We define several terms that encompass the whole scope of technical interoperability to later provide a holistic definition of technical interoperability (see Figure 3). To recall the definition presented in Section 2.2, a source blockchain is a blockchain that issues transactions against a target blockchain. A *source node* is a node from the source blockchain, and a target node belongs to the target blockchain. When several participants elect a source node and a target node, we achieve decentralization in the context of interoperability [99].

A ***Cross-Chain Transaction*** (**CC-Tx**), where "CC" stands for *cross-chain*, and "Tx" for transaction, is a transaction between different chains, which belong to the same blockchain system (homogeneous blockchains), for example, between EVM-based blockchains. We use the *Cross-Chain Transaction* (CC-Tx), *inter-chain transaction*, and *inter-blockchain transaction* terms interchangeably. A ***Cross-Blockchain Transaction*** (**CB-Tx**) is a transaction between different blockchains (heterogeneous blockchains), for example, between Hyperledger Fabric and Bitcoin. Note that the terms CC-Tx and *Cross-Blockchain Transaction* (CB-Tx) are used as synonyms in the industry, as currently, most solutions connect homogeneous blockchains. A ***Cross-Chain Decentralized Application*** (**CC-dApp**) is a dApp that leverages cross-blockchain transactions to implement

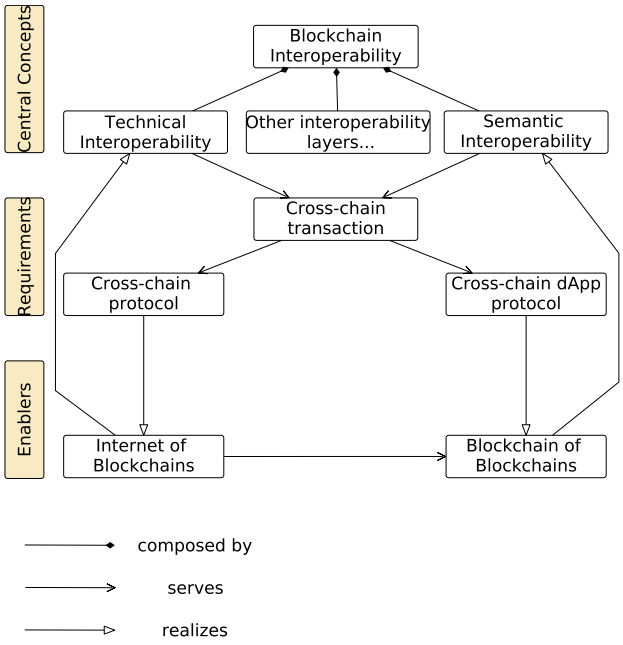


Fig. 3. Concept map, illustrating the relationship between different concepts related to blockchain interoperability

its business logic. We use the terms *Cross-Chain Decentralized Application* (CC-dApp) and ***cross-blockchain decentralized application*** (**CB-dApp**) interchangeably. Other terms with the same meaning in the literature are inter-chain decentralized application and inter-blockchain decentralized application.

An ***Internet of Blockchains*** (**IoB**) is a system "where homogeneous and heterogeneous decentralized networks communicate to facilitate cross-chain transactions of value" [170]. We use this definition of IoB throughout this article.

The term ***Blockchain of Blockchains*** (BoB) is not used consistently [120, 178]. Verdian et al. use it to describe the structure that aggregates blocks from different blockchains into "meta blocks," organized through a consensus mechanism using *posets* (partially ordered sets) and total order theory [178], thus producing a blockchain of blockchains. A poset consists of a set of elements and their binary relationships that are ordered according to a specific set of rules [28].

Influenced by those authors, we define a *Blockchain of Blockchains* (BoB) *as a system in which a consensus protocol organizes blocks that contain a set of transactions belonging to CC-dApps, spread across multiple blockchains. Such a system should provide accountability for the parties issuing transactions on the various blockchains and providing a holistic, updated view of each underlying blockchain.* Note that BoB solutions belong to the category with the same name. Therefore, the notion of *Internet of Blockchains* (IoB) directly refers to the connection relationships among blockchains, whereas the term BoB refers to an architecture made possible by IoB. BoB approaches are concerned with the validation and management of cross-blockchain transactions.

Figure 3 shows the relationship between the different concepts concerning blockchain interoperability. A CC-dApp realizes the blockchain of blockchains approach. This approach can provide the semantic level interoperability (i.e., concerned at transmitting the meaning of the data, which corresponds to the value level interoperability) required by organizations, mappable by the applicational layer. However, it relies on the existence of an IoB—a network of blockchains. For an IoB to exist, technical interoperability (or mechanical interoperability) is required. In the context of

Table 1. Survey Comparison Criteria, Description, and Sub-Criteria

Criteria	Description	Sub-criteria 1	Sub-criteria 2	Sub-criteria 3
Public Connectors (PCs)	Addresses public connectors	Sidechains	Hash lock contracts	Notary Schemes
Blockchain of Blockchains (BoBs)	Addresses BoBs	Describes solutions	Detailed comparison	N/A
Hybrid Connectors (HCs)	Addresses Hybrid Connectors	Trusted Relays	Blockchain agnostic protocols	Blockchain migrators
Architecture (AR)	Addresses architectures enabling CCCPs	Proposes architecture	Presents related work	N/A
Cross-chain Standards (ST)	Addresses standards for interoperability	Present standards	Relate standards to solutions	N/A
Cross-analysis (CC)	Compares across categories	Compare categories	Compare sub-categories	N/A
Use Cases (UCs)	Presents use cases using an IoB or BoB	Existing use cases	Predicted use cases	N/A
Open Issues (OIs)	Challenges on interoperability	Research directions	Relate interoperability to other issues	N/A

a CC-dApp, cross-chain transactions are ordered by a *cross-chain dApp protocol*. Such protocols should assure transaction atomicity and resolve possible conflicts in transactions spawning across homogeneous and heterogeneous blockchains.

From the several definitions we encountered during our research, we envision *blockchain interoperability* as *the ability of a source blockchain to change the state of a target blockchain (or vice versa), enabled by cross-chain or cross-blockchain transactions, spanning across a composition of homogeneous and heterogeneous blockchain systems, the IoB.* IoB transactions are delivered via a cross-blockchain communication protocol, thereby granting technical interoperability, enabling CC-dApps. CC-dApps provide semantic interoperability via the BoB. The BoB approach is realized by a cross-blockchain dApp protocol, which provides consensus over a set of cross-chain transactions, thus enabling cross-chain dApps.

3 RELATED LITERATURE REVIEWS

Due to the novelty and large breadth of this research area, few updated surveys cover aspects of blockchain interoperability. We compare existing surveys based on the *criteria* and *sub-criteria* shown in Table 1. For example, in the first row, the criteria "public connector" evaluates if a study addresses its sub-criteria: work on sidechains, hash-lock time contracts, and notary schemes. On the second row, the criteria Blockchain of Blockchains evaluates if a study describes BoB solutions (1) and if it performs a detailed comparison, including consensus, security, validators, and performance.

Buterin presents a survey on public connector solutions, including notary schemes, sidechains, and hash-time locking techniques [44]. Similarly, other surveys focus on public connectors [36, 106, 163, 194], with a focus on sidechains and hash-lock time contracts. Vo et al. present work mostly on architecture for interoperability, presenting some BoB and HC solutions [170], while Qasse et al. organize solutions across sidechains, blockchain routers, smart contracts, and industrial solutions [145]. Johnson et al. focus on Ethereum as the infrastructure enabling interoperability across several categories of solutions [100]. Siris et al. [164], Kannengießer et al. [103], and Bishnoi and Bhatia [29] tackle a wider range of solutions.

We aim at providing a solid, throughout and comprehensive foundation on which researchers can rely upon as a starting point in this field, including a description of the related surveys, which illuminated our research. In contrast to most of the works mentioned above, this article provides a holistic view of blockchain interoperability by focusing not only on public connectors but also on BoBs and hybrid connectors. By including updated grey literature and focusing on private blockchain interoperability, a comprehensive discussion on standards, use cases, and architecture for interoperability was possible. Table 2 shows a comparison of related literature reviews, based on multiple criteria.

Table 2. Comparison of Related Literature Reviews: PC (Public Connectors), Blockchain of Blockchains (BoBs), HCs (Hybrid Connectors), AR (architectures for blockchain interoperability), ST (standards), CC (cross-comparison), UC (use cases), OIs (open issues)

	Solution Category			Detailed Analysis				
Reference	PC	BoB	HC	AR	ST	CC	UC	OI
Buterin [44], 2016	+	−	−	−	−	±	+	+
Vo et al. [170], 2018	−	±	±	+	±	±	±	+
Borkowski et al. [35], 2018	+	−	−	−	−	±	−	+
Qasse et al. [145], 2019	±	±	±	−	−	±	±	±
Johnson et al. [100], 2019	±	±	±	−	−	−	−	−
Zamyatin et al. [194], 2019	+	−	−	−	−	±	−	+
Siris et al. [164], 2019	±	±	±	±	−	+	−	−
Koens and Poll [106], 2019	+	+	−	−	−	±	−	+
Singh et al. [163], 2020	+	−	−	−	−	−	+	+
Kannengießer et al. [103], 2020	+	±	±	−	−	±	−	−
Bishnoi and Bhatia [29], 2020	+	±	±	−	−	−	−	−
This survey	+	+	+	+	+	+	+	+

Each criterion can be "fulfilled" ("+" in green background), "partially fulfilled" ("±" in orange background), or "not fulfilled" ("-" in red background), if it addresses all, between one and all, or none of its sub-criteria, respectively.

4 BLOCKCHAIN INTEROPERABILITY FRAMEWORK

This section presents the BIF, a framework classifying solutions collected through our methodology. To drive criteria for assessing the categories (and specific solutions) of blockchain interoperability, we analyzed the solution space using the six "W" questions: Who, What, Where, When, Why, and How. The "Why" was determined irrelevant to our analysis because its purpose is constant—connecting different chains (CC-Txs), different blockchains (CB-Txs), or even to arbitrary systems (e.g., enterprise legacy systems). This is instead addressed by the "where" question.

4.1 Deriving Evaluation Criteria

The "what" refers to the *assets* exchanged. An interoperability solution can handle different data objects or assets. Hence, it is important to know which data representations a solution supports [186]. Assets can be treated as data (arbitrary payloads), as fungible assets, or non-fungible assets [16, 129, 139]. Arbitrary data is often represented via a key-value pair, being the preferred representation of some blockchains [7, 46, 93]. The key-value is also useful to represent the contents of account-based blockchains [48, 65, 101]. Payment tokens are fungible tokens [138]. Utility tokens include tokens used to access a service or application, such as non-fungible tokens (e.g., ERC20 tokens). Finally, asset tokens represent real-world physical or digital instruments, such as blockchain-based promissory notes, regulated by the Swiss Financial Market Supervisory Authority [155] (see more details in Section 6.3), or bonds [16]. An asset has different maturity levels. In

particular, an asset may be standardized (e.g., ERC tokens[180], standardized schema for utility tokens, ERC1400, a security token [159, 160]) and/or regulated [126, 167, 177]. Regulated digital assets are backed by legal frameworks. We consider all asset tokens to be regulated. We envision utility tokens as standardized and asset tokens as standardized and regulated (i.e., asset tokens are emitted by legal entities).

The "who" question refers to whom controls the CC-Tx process and thus accounts for trust establishment [77, 194] It can be the end-user (e.g., [71, 129]), a consortium (e.g., [13, 158]), or a trusted third party (e.g., cloud services, centralized notary schemes). Some solutions allow different levels of control.

The "where" refers to what are the source and target ledgers, as well as what is the support of conducting the CC process. Solutions can support public blockchains (P) or non-public blockchains (NP). We use NP to designate private blockchains, other **decentralized ledger technology** (**DLT**) systems, and centralized systems (e.g., VISA payment network). The supported systems of each solution matter since communication may happen unidirectionally or bi-directionally [129]. Blockchain oracles apart, it often is not feasible to have a solution based on a blockchain system connected to a centralized system (e.g., providing insurance data). A smart contract may be the one conducting an asset transfer (on-chain channel, with on-chain CC-Tx validation) versus an off-chain settlement, e.g., techniques using commitment schemes [2, 76], or via a (semi-)centralized system (off-chain channel). Typically, on-chain channels offer more resiliency, but off-chain channels are more scalable. Combinations between off-chain and on-chain channels also exist (e.g., payment networks [144]). Offline channels depend on different proof generation mechanisms [2, 76, 77].

The "when" refers to the set of processes (e.g., executing CC-Txs) that are defined at *design time* or *runtime*. *Design-time customization* decisions affect the punctual behavior of a CC-dApp concerning when it is executed. At design time, a user defines the behavior of the solution *a priori*. If a change is needed, a new instance of the solution needs to be deployed. Conversely, *runtime customization* decisions are flexible, allowing the end-user to adjust the conditions defined by business logic as needed. Solutions in which business logic is changed at runtime are called *flexible approaches*, allowing one to adjust business logic and conditions that trigger the execution of a CB-Tx or CC-Tx by a CC-dApp. Most literature reviews focus on design-time approaches and public blockchains, leaving a vast range of recent solutions out of scope. In this survey, we also consider private-private and public-private blockchain interoperability, focusing on flexible approaches.

The "how" regards the realization of cross-chain transactions: how are CC-Txs realized on the underlying DLTs? Often, these transactions can be performed using *cross claims*, i.e., by locking/burning an asset on the source blockchain and unlocking/creating its representation on the target blockchain. Cross-claims require two nodes from different blockchains, where one performs one operation in a source blockchain in exchange for its counterparty performing other operations on a target blockchain—each party logs the operation in case a dispute is needed. Typically, cross-claims operate in semi-trusted environments (e.g., private blockchain, regulated blockchain), and can be operated via a (semi) trusted third party [17, 88, 129]. Escrowed cross-claims are the standard mechanism for asset transfers, operating similarly to cross-claims, but in an untrusted environment, leveraging dispute-resolution mechanisms (e.g., via smart contracts requiring inclusion proofs [2]) or by parties holding custody of assets and collateral [38, 45, 195]. Inclusion proofs include applying Merkle tree proofs to block header transfer via a coordinating blockchain, block header transfer, or direct signing [151]. Collateralization is the process in which a party performing the transfer of assets provides a certain amount of their assets as a guarantee of following the protocol (e.g., not to steal assets from the end-user). If a party misbehaves (e.g., steals assets), the deposit is given to the victim party. Finally, a mediated CC-Tx includes (an offline) trusted party [129]. In case of a dispute about an asset transfer between a public blockchain and a private

blockchain (P-NP) or a public blockchain and an enterprise system (also P-NP), there needs to be a dispute-resolution mechanism. This is due to NP systems' private nature, although several mechanisms exist to prove internal state belonging to private blockchains. Hence, CC-Txs have a tradeoff risk performance: the less centralization there is on the CC-Tx settlement, the worst the performance, but the lesser the risk.

The "how" also relates to the extent to which the implementation of the solution is tested. Solutions might be implemented, tested, and validated (application to a real-world scenario). Testing regards *correctness guarantees*: *behavioral correctness* or *formal correctness*. Behavioral correctness is the ability to guarantee that CC-Txs are issued as intended, without unintended consequences (e.g., asset lock, asset theft). While in practice, behavioral correctness depends on formal correctness, we say a solution has behavioral correctness if it has a suite of test cases [131]. Formal correctness assures that an algorithm is correct with respect to a specification. Formal verification checks the correctness of algorithms against a specification using, for instance, formal methods. Smart contract verification tools allow developers to reduce the probability of creating bugs, thus incurring penalties, as smart contracts are generally difficult to update once deployed [64]. Another point of providing trust to the user is the solution to have an open source implementation, where the code can be peer-reviewed and corrected if needed.

4.2 Evaluation Criteria

Having discussed the survey's scope, we next define the set of criteria we use to characterize the interoperability solutions. Similarly to Section 3, each criterion can be "fulfilled," "partially fulfilled," or "not fulfilled." If a criterion is a yes/no question (e.g., does the solution support asset type "data"?), we do not explicitly refer to the fulfillment conditions as they are evident. Next, we detail the criteria type (first-level), criteria sub-type (second level), and criteria from BIF:

- Asset: this category refers to properties of an asset involved in a CC-Tx.
  - Type: What type of assets does the solution support?
    (1) Data: Ccan the solution manipulate arbitrary data?
    (2) Payment tokens: Can the solution manipulate cryptocurrencies? This criterion is partially fulfilled if the asset is only used as collateral or to reward a service's operational maintenance.
    (3) Utility tokens: Can the solution manipulate utility tokens? This criterion is partially fulfilled if the asset is used only as collateral or to reward a service's operational maintenance.
    (4) Asset tokens: Can the solution manipulate utility tokens?
  - Infrastructure: What are the systems involved?
    (1) P: This criterion is fully fulfilled if more than two public blockchains are supported. It is partially fulfilled if one or two public blockchains are supported.
    (2) NP: This criterion is fully fulfilled if more than two non-public blockchains are supported. It is partially fulfilled if one or two non-public blockchains are supported.
- Trust Establishment: This category refers to how a solution provides trust to the users.
  - Decentralization: Who operates the solution instance?
    (1) End-user
    (2) Consortium
    (3) Trusted (third) party
    If multiple criteria are selected, it indicates a solution supports more than one mode of operation.
  - Channel: Where are CC-Tx validated?
    (1) On-chain: This criteria is partially fulfilled if proofs are created on-chain but validation occurs off-chain.
    (2) Off-chain: This criteria is partially fulfilled if proofs are created off-chain but validation occurs on-chain.

- CC-Tx Realization: This category refers to how and where a CC-Tx is settled.
  - Mechanism: How are CC-Txs agreed-upon multiple parties?
    (1) Cross-claim
    (2) Escrowed cross-claim
    (3) Mediated
- Extra-functional: This category refers to the design of the solution itself.
  (1) Tests: The approach provides a set of test cases.
  (2) Implementation: The approach provides an open source implementation and is validated in the industry. This criterion is partially fulfilled if the implementation is closed source.
  (3) Validation: The approach is validated in an actual use case scenario.
  (4) Runtime: The business logic of the solution can be changed dynamically, as needed. This criterion is considered not fulfilled if logic is settled when the solution is instantiated, i.e., changing logic requires a new instance.

5 OVERVIEW OF BLOCKCHAIN INTEROPERABILITY APPROACHES

We conducted a systematic literature review following the protocol described in Appendix A, yielding 80 relevant documents out of the initial 330. By grouping the publications and grey literature, a pattern arises: these works are either about interoperability across public blockchains holding cryptocurrencies, application-specific blockchain generators with interoperability capabilities, or protocols connecting heterogeneous blockchains. We thus classify each study into one of the following categories: *Public Connectors* (Section 5.1), *Blockchain of Blockchains* (Section 5.2), and *Hybrid Connectors* (Section 5.3). Each category is further divided into sub-categories. Table 3 summarizes the work conducted.

5.1 Public Connectors

The first family of blockchain interoperability solutions aimed to provide interoperability between cryptocurrency systems, as stated by Buterin [44]. This category identifies and defines different chain interoperability strategies across public blockchains supporting cryptocurrencies, including sidechain approaches, notary schemes, and hash time hash locks. Some solutions share characteristics of more than one sub-category, and thus they can be considered hybrid. We introduce each sub-category, presenting only two illustrative examples of each one for the sake of space. Appendix C depicts a list of Public Connectors approaches. After that, a summarized evaluation table is presented using the BIF. These tables are later discussed in Section 5.1.4.

5.1.1 Sidechains and Relays. A *sidechain* (or *secondary chain*, or *satellite chain*, or *child chain*) is a mechanism for two existing blockchains to interoperate [13, 77], scale (e.g., via blockchain sharding [107]), and be upgraded [196] in which one blockchain (*main chain* or mainchain) considers another blockchain as an extension of itself (the sidechain). The mainchain maintains a ledger of assets and is connected to the sidechain, a separate system attached to the main chain via a cross-chain communication protocol [76]. An example is a *two-way peg*, a mechanism for transferring assets between the main chain and the sidechain [163]. Main chains can be sidechains of each other [44], creating each chain's possibility to connect to others. Sidechains are considered layer-1 solutions (built on top of layer-0 solutions—blockchains) to implement layer-2 solutions, such as payment channels [104]. The second layer allows off-chain transactions between users through the exchange of messages tethered to a sidechain [82]. A sidechain is then a construct that allows for offloading transactions from the mainchain, processes it, and can redirect the outcome of such processing back to the main chain.

For instance, state channels are off-chain sidechains used to implement, for example, payment channels, by offloading transactions of the blockchain [144]. In a payment channel, participants

Table 3. Evaluation of Blockchain Interoperability Solutions by Sub-Category According to The Blockchain Interoperability Framework

	Asset			Trust Establishment										
	Type			Infra.		Decentral.			Channel		CC-Realization			
Sub-Category	D	P	U	P	NP	U	C	TTP	OC	OF	CC	ECC	M	**References**
Sidechains & Relays	+	±	-	±	-	-	+	-	+	+	-	+	-	[143, 182]
	+	±	-	±	-	+	+	-	+	-	-	+	-	[15, 66, 72, 73, 110]
	-	+	+	+	-	+	+	-	+	-	-	+	-	[9, 98]
	-	+	+	±	-	+	+	-	+	+	-	+	-	[13, 63, 104, 121, 140]
	+	+	-	±	-	-	+	-	+	-	-	+	-	[56, 76, 113, 114]
	-	+	+	±	-	-	+	-	+	±	-	+	-	[24, 59, 83, 150, 161, 162]
	-	+	-	+	-	+	+	+	-	+	+	-	+	[94, 97]
Notary Scheme	-	+	+	+	-	-	-	+	±	-	-	-	+	See Section 5.1.2
	-	+	+	+	-	+	+	-	+	-	-	+	-	[125, 174, 184]
HLTC	-	+	+	±	-	-	+	-	+	-	-	+	-	[38, 52, 57, 74, 122, 153, 195]
Blockchain of Blockchains	+	+	+	±	-	+	+	-	+	-	-	+	-	[108, 109, 188]
	+	+	+	+	+	-	+	+	+	-	-	+	+	[10, 147, 165]
Trusted Relays	+	-	-	±	±	+	-	-	-	+	-	-	+	[40, 68, 102, 134]
	+	+	+	±	+	+	+	-	+	±	+	-	+	[17, 23, 86, 88, 183, 190, 198]
B. Agnostic Protocols	+	+	+	+	+	+	+	-	-	+	+	+	-	[1, 2, 129, 146]
	+	+	+	±	±	+	+	-	+	-	-	+	-	[50, 62, 120, 136, 139, 151]
Blockchain Migrators	+	-	-	±	-	+	-	-	-	+	N/A	N/A	N/A	[71, 156, 187]
	+	+	+	±	±	+	+	-	+	-	-	+	-	[75]

N/A stands for not applicable. Public connectors are represented in green, blockchain of blockchains in orange, and hybrid connectors in red.

interact, collecting cryptographically signed messages. Those messages update the current state without publishing it to the mainchain. When the payment channel is closed, the final state is published onto the main chain, where an on-chain dispute/closure phase may occur [104]. Payment channels are appropriated for use cases requiring several transactions that can be combined in a single one.

Main chains communicate with sidechains via a CCP, often tightly coupled with the functionality of both chains. The basic components of sidechain design are the mainchain consensus protocol, the sidechain consensus protocol, and the cross-chain communication protocol [76]. Sidechains allow different interactions between participating blockchains, being the most common the transfer of assets between the main chain and the sidechain (two-way peg) [105, 163]. A *two-way peg* works in the following manner: a user, operating on the mainchain, sends X tokens to a custom address that locks assets. Those funds are locked on the mainchain, and a corresponding number of tokens are created on the sidechain. The user can now use the tokens on the sidechain. Eventually, the user can transfer back the tokens to the main chain, which causes assets on the sidechain to be locked or destroyed, depending on the implementation. There are three major types of two-way pegs: simplified payment verification, centralized two-way pegs, and federated two-way pegs. ***Simplified payment verification*** **(SPV)** [33, 132] is done by *light clients*, which consist of blockchain clients that can verify transactions on the blockchain without having its entire state. The SPV client only needs the block headers; verifying that a transaction is in a block is to request a Merkle tree proof

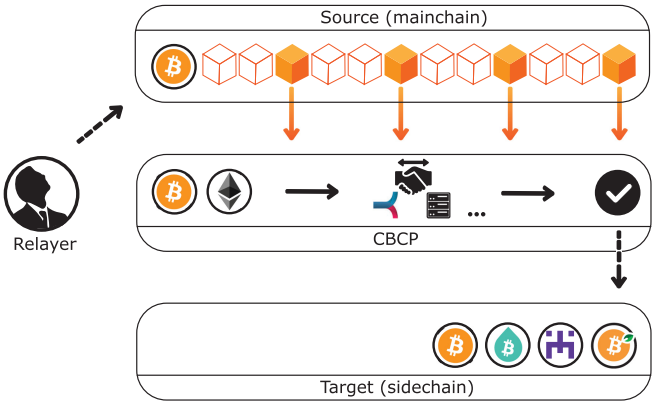


Fig. 4. A general sidechain system [66].

[169] including that transaction. In particular, transactions are represented as Merkle tree leaves. Given a leaf node as a target and a path comprised of nodes and its siblings to the target, verifying a Merkle tree proof of including the target is to reconstruct a partial Merkle tree root.

A relay solution is an SPV client for a source blockchain running on a target blockchain, enabling verification of transactions [72]. This verification enables conditional logic to occur on a target blockchain. Since relays are between blockchains and those blockchains are using behavior from others (bidirectionally or unidirectionally), relays include the presence of sidechains. This is saying, without a sidechain, there are no relay solutions.

Centralized two-way pegs, on the contrary, trust a central entity, benefiting in terms of efficiency. An example is an *exchange*, an organization, typically a company, that trades cryptocurrencies on behalf of its clients. However, Exchanges are a Notary Scheme, so we defer their explanation to Section 5.1.2. Disadvantages include a single point of failure and centralization. *Federated two-way pegs* try to decentralize the previous solution. In this solution, a group is responsible for locking and unlocking funds instead of just one. Standard implementations rely on multi-signature schemes, in which a quorum of entities must sign transactions to be deemed valid by the network. Although a better option, it does not eliminate centralization.

Figure 4 depicts a system based on the BTC Relay [66]. In *BTC Relay*, parties called *relayers* keep track of the block headers of the main chain (the Bitcoin network in the figure), and input them to the BTC Relay smart contract, hosted on Ethereum. This procedure builds a pool of Bitcoin headers that can be used (via their stored Merkle trees) to verify on-chain information, including the presence of transactions. This way, any party can request a transaction to be verified by the smart contract that holds the headers' knowledge (via SPV). Transaction validation can be relayed to deployed Ethereum smart contracts, allowing several use cases, for example, the issuance of tokens.

Zendoo is a cross-chain transfer protocol that realizes a decentralized, verifiable blockchain system for payments [76]. The authors consider a parent-child relationship, where nodes from the sidechain can observe the mainchain's state, but the main chain can only observe the sidechains via cryptographically authenticated certificates. Zk-SNARKSs enable the authentication, validation, and integrity of the information provided by the sidechains via verifiable proofs [25]. Such proofs are used to generate certificate proofs for the mainchain, enabling a secure verification scheme.

5.1.2 Notary Schemes. A notary scheme involves a *notary* that is an entity that monitors multiple chains, triggering transactions in a chain upon an event (e.g., a smart contract is deployed)

taking place on another chain [44]. Notary schemes are, in practice, instantiated as centralized **exchanges** (**EXs**) or **decentralized exchanges** (**DEXs**). The most popular centralized exchanges, by volume, as of the 8th of February 2021 are Binance,[2] Coinbase,[3] and Huobi Global.[4] Exchanges facilitate signaling between market participants by managing an order book and matching buyers and sellers. If the trust anchor is put on a centralized party, where it holds users' private keys or funds, the notary is a centralized exchange. Otherwise, if exchanges do not execute the trades on behalf of the users, only providing a matching service, they are considered decentralized exchanges. We present the protocols of two decentralized exchanges: 0x [184], and Uniswap [3].

0x implements a decentralized exchange as a set of smart contracts (called automated market makers), replacing an on-chain order book with a real-time price-adjustment model. 0x uses a hybrid implementation, "off-chain order relay with on-chain settlement," combining the idea of a state channel with settlement smart contracts. Two parties participate: *makers* and *takers*. Makers place orders on the exchange, providing liquidity for the network (a set of decentralized exchanges), while takers place orders matched with the makers' orders. 0x uses the ZRX token and the Ethereum blockchain to incentivize users to host and maintain order books (provide liquidity). In exchange, 0x makers choose the rewards they obtain for each trade—although they have to comply with the DEX policies under the possibility of the order not being disseminated. This approach relies on a smart contract set (smart contract) and several smart contracts representing the different tokens supported. First, a maker creates an order to exchange token A for B, at a given rate, right after it approves a DEX to access its balance of token A. A taker discovers this order and wishes to trade its tokens B for tokens A. The taker grants permission to the DEX to access its tokens, and the DEX performs the exchange after several validations (e.g., the order has not expired, and it has not been filled).

Uniswap is a set of smart contracts implementing an automated liquidity pool, serving as a decentralized exchange [3]. Each Uniswap pool provides liquidity for two assets based on the constant set as the reserves' product. Prices for each asset are provided by an on-chain price oracle smart contract. Uniswap can support ERC-20 to ERC-20 trades and even flash loans, a theme explored in the decentralized finance area. A flash loan is a type of loan that does not require collateral, as the debt is repaid within the transaction. Flash loans work because the borrowed asset to be paid is within the transaction requesting it [3].

5.1.3 Hashed Time-Lock Contracts. **Hashed time-lock contracts** (**HTLCs**) initially appeared as an alternative to centralized exchanges, as they enable cross-chain atomic operations [30]. HTLCs techniques use hash locks [31] and timelocks [32] to enforce atomicity of operations, normally between two parties. A trader commits to making the transaction by providing a cryptographic proof before a timeout to the other. This scheme allows for the creation of multiple outputs (such as multiple payments), depending on solely one hash lock. HTLCs are used in Bitcoin for conditional payments, or cross-chain payments (Bitcoin-Ethereum), i.e., *atomic swaps* [34, 58, 90]. Atomic swaps can be thought as a form of distributed commitment resilient to Byzantine adversaries. Thus, an atomic cross-chain swap is a distributed atomic transaction [91], settled on-chain.

Several projects implement HTLCs differently, providing different correctness guarantees. However, the general algorithm is quite similar in most of the solutions. Let us consider an HTLC-supported atomic swap between Alice (holding assets of type a in blockchain $\mathcal{B}_a$) and Bob (holding assets of type b in blockchain $\mathcal{B}_b$). An atomic swap can be realized as follows [24, 197]: (1) Alice generates and hashes a secret s, yielding h. The protection of a smart contract with hash

[2] https://www.binance.com/en.
[3] https://www.coinbase.com/.
[4] https://www.huobi.com/.

h is called a hash lock because it will lock a smart contract—only parties with knowledge of secret s can know it since secure hash functions are pre-image resistant (i.e., a hash function cannot be inverted). Alice also creates a timelock t_b, corresponding to an upper bound in which the created hash lock can be unlocked, i.e., Bob can unlock the smart contract up to t_b, where t_b corresponds to a specified future time or block height. (2) Alice publishes the smart contract in $\mathcal{B}_a$. Bob verifies the deployment, and records h and t_b. (3) Bob publishes a smart contract in $\mathcal{B}_b$ locking b with hash lock h, but with timelock ta such that $t_a < t_b$, i.e., Alice can claim b before t_a. (4) Alice checks that Bob's smart contract has been published and gives as input secret s, before t_a, acquiring asset b. In practice, this triggers a transfer. (5) Bob now sends s to Alice's smart contract in the interval $[t_a, t_b]$, acquiring a. Note that if Bob issues the transaction after t_b, Bob will not obtain access to b. Some solutions utilize the notion of HLTC and enhance it, providing an additional on-chain trust anchor. In particular, two solutions are presented: XCLAIM [195] and the **Lightning Network** (**LN**) [144].

XClaim uses a combination of HLTCs, collateralization, and escrow parties, realizing non-interactive cross-chain atomic swaps [195]. This protocol includes several actors: the requester, the sender, the receiver, the redeemer, the backing vault, and the issuing smart contract. Requesters lock coins to issue tokens, while the redeemer burns tokens to receive coins. The sender sends tokens, while the receiver receives them. After that, the vault smart contract fulfills requests of asset backing and ensures correct redeeming. An issuing smart contract issues and exchanges representations of a token (cryptocurrency-backed assets) and enforces the vault's correct behavior. Considering a transaction between Bitcoin and Ethereum, firstly, the vault locks collateral in Ethereum smart contracts. This collateral defines the amount of CBA that the vault can issue. A user that wants to issue Bitcoin-backed tokens sends Bitcoin to the vault. User A then sends a proof of transaction submitted to the Bitcoin mainchain to a chain relay, e.g., BTC Relay. The chain relay verifies the submitted transaction and alerts the issuing smart contract. The smart contract releases the Bitcoin-backed assets to the user. On the other hand, a user issues a transaction against the smart contract, locking/burning its backed tokens. The vault releases the Bitcoin to the user, and it submits a proof of the involved operations to the chain relay. The chain relay verifies the proof and only then releases the collateral to the vault. XClaim currently supports exchanges between Bitcoin and Ethereum.[5] The protocol execution consumes substantially lower Ether than traditional HTLCs.

Ligthning Network (LN) enables high-volume, low-latency micro-payments on the Bitcoin network [144]. LN is a payment scheme (i.e., an off-chain sidechain). LN allows several parties to open a payment channel, transact amongst them, and when all the intermediary payments are completed, the final output is sent to the mainchain. LN works as follows: (1) funds are placed into a multi-signature Bitcoin address (two-party multi-signature if only two people are transacting). In order for funds to be changed, two signatures are required. After that, the funds will be managed off-chain via commitment transactions (i.e., a commitment to pay part of the available funds to the other party). (2) Parties can now transact offline under the regime they choose. (3) To settle the payments performed off-chain, both parties sign a new exit transaction. Note that parties can unilaterally close the payment channel in case of conflict. LN is considered a precursor of HLTCs because its bi-directional payment channels allow payments to be routed across multiple payment channels using HLTCs.

5.1.4 Discussion on Public Connectors. Public Connectors started emerging as early as 2015 [202], when researchers and practitioners alike saw the potential in cross-chain transactions to

[5] https://github.com/crossclaim/xclaim-sol.

support, for instance, atomic swaps [34, 58, 90] and payment channels [144]. Sidechains are the solutions increasing the main network's scalability by processing and batching large amounts of transactions before submission on the main blockchain [114, 143, 163]. Relays can fetch block headers from sidechains, enabling data verification [66, 110, 111]. While sidechains are mainly used on public blockchains, there are also permissioned blockchain sidechains [115]. We note that some sidechains may have a cross-chain mechanism realization HLTC, being a solution belonging to multiple categories (e.g., [144]).

Most sidechains use Ethereum and have a sidechain consensus mechanism, which is allusive to bidirectional transfers [76]. Simple relay schemes, which verify transactions on other chains, such as BTC Relay, have a simple sidechain consensus, as the information flow is unidirectional [66]. In particular, validators can sign events that happened on the source chain (if validation happens across EVM-based chains) or transfer block headers (via users or aggregation chains) [151]. Liquid [63] and POA [9] rely on a consortium of validators running trusted hardware to execute smart contracts and validate transactions. Other solutions, such as Wanchain [70] rely on a trusted consortium, but without running trusted hardware.

However, sidechains suffer from several limitations. Safe cross-chain interactions are rooted in the assumption that the main chain is secure, i.e., the network cannot be successfully attacked. Compromising the main chain would invalidate the sidechain logic. Conversely, centralization in sidechains tends to exist to a higher degree than on mainchains, because typically there is a trade-off between decentralization-performance (e.g., lesser validating nodes versus higher throughput). Consequently, if an attacker can obtain control on a (potentially small) set of validators, funds can be stolen from users. Therefore, it is important to have different stakeholders with different incentives, diminish the likelihood of collusion, and rely on a reasonable quorum of validators to sign each transaction (e.g., 8 out of 11 versus 3 out of 10). If a sidechain has a strong security model, it may lead to a slow transaction settlement, stalling assets, and lowering liquidity. For example, the RSK sidechain [114] takes approximately the time to confirm 100 Bitcoin blocks (around 15 hours) to convert BTC to RBTC.[6] Finally, sidechains typically do not allow for arbitrary code to specify conditions on the pegging mechanism, thus not empowering them to develop more complex applications.

Notaries on the Public Connectors category are cryptocurrency exchanges. EXs have the majority of the market share, comparatively to DEXs. While EXs provide services to the end-user, decentralized exchanges tend to provide better exchange fees and security. The tradeoff is, therefore, comfort and speed—security. This sub-category provides great flexibility at runtime because EXs and smart contracts that DEXs support triggers (e.g., stop-loss orders).

Notary schemes have to capture the logic of smart contracts in both chains. Although they can capture the full spectrum of interoperability—both at the value and the mechanical levels (see Section 5.3), practical applications are limited. In summary, notary schemes are intermediaries between blockchains. EXs are notaries because they execute actions on behalf of the end-user (e.g., buy cryptocurrencies conditionally). DEXs are notaries because they provide matching for the end-users by pinning and advertising trade offers encoded in smart contracts.

The HTLCs category was the first one to allow asset exchange in a trustless way. HTLCs allow atomic swaps between different blockchains, funding bidirectional payment channels. HTLCs are flexible because they can be chained after each other [197], and therefore enable trades even if there is no direct connection between the trading parties. As they serve as programmable escrows, they represent the most trustless and practical approach of the three. However, hashed time locks might lead to capital retention and unfair trade, as the trader issuing a cross-blockchain asset

[6] https://developers.rsk.co/rsk/.

transfer may only provide the secret on specific conditions (exploring the spread of the cryptocurrency exchange rate) [195]. Many solutions are hybrid, sharing characteristics of HTLCs and sidechains, either exploring collateralization-punishment schemes rooted on smart contracts ([154, 161, 195], or locking-in and locking-out assets [38, 74, 127, 128]. HLTCs are practical solutions across public blockchains. HLTCs could also provide asset transfers between private blockchains, but only under the participation of a third party blockchain and a semi-trust environment [84], or if both parties belong to both private blockchains. Current efforts to address these limitations include Hyperledger Cactus [129].

Concluding, Public Connectors are the best approach to perform cryptocurrency trades and moving fungible and non-fungible assets across public blockchains. We encourage the reader to refer to some related surveys focusing on sidechains to complement this survey (see Section 3).

5.2 Blockchain of Blockchains

Blockchain of Blockchains are frameworks that *provide reusable data, network, consensus, incentive, and contract layers for the creation of application-specific blockchains (customized blockchains) that interoperate between each other.* We briefly present Polkadot [41, 188] and Cosmos [109], the most widely adopted Blockchain of Blockchains in terms of market capitalization.[7] A detailed comparison between Polkadot, Cosmos, and Ethereum 2.0 (the baseline) is deferred to Appendix D. Other Blockchain of Blockchains include Ark [10], Komodo [108], and AION [165].

Wood proposes *Polkadot*, a network that aims to connect blockchain networks [188]. Polkadot provides the foundation for *parachains*, i.e., "globally coherent dynamic data structures" hosted side-by-side. Parachains are, thus, the parallelized chains that participate in the Polkadot network. Specialized parachains called bridges link independent chains [188]. Polkadot is based on *Substrate*, a framework for creating cryptocurrencies and other decentralized systems. It guarantees cross-language support with WebAssembly, a light client, and off-chain workers, allowing for integration with other technologies.

Polkadot enables interoperability based on state transition validation, done by the chain-relay validators. Parachains communicate through the **Cross-chain Message Passing Protocol** (**XCMP**), a queuing communication mechanism based on a Merkle tree [141]. Communicating state transition proofs from parachain to relay chain is achieved via an erasure-coding scheme. Polkadot scales by connecting up to 100 parachains directly to the relay chain in the short-medium term. A long-term solution is being studied, where second and third-level parachains are added in parallel.

Cosmos is a decentralized network of independent parallel blockchains, called *zones* [109]. The zones are essentially Tendermint blockchains [172]. Zones can transfer data to other zones directly or via *hubs*. Hubs minimize the number of connections between zones and avoid double spendings. For example, zone A can connect to zone B via Hub C and receive tokens from zone B. Zone A would need to trust the tokens from zone B and Hub C. This scheme allows zones to maintain a reduced number of connections. Both ways utilize the inter-**blockchain communication protocol** (**IBC**) [95].

IBC resembles the Internet network layer as it routes arbitrary data packets to a target blockchain. A target blockchain can know that a certain ordered packet with arbitrary data came from another blockchain. By handling transportation and order, the protocol has several steps to achieve cross-zone transactions. First, each chain involved tracks the headers of the others, acting as a light client. When a transfer is initiated, the protocol locks the assets on the origin chain. After that, the proof is sent to the target blockchain, which then represents the locked assets. A similar

[7] USD 22.1B and USD 3.6B, respectively, as of February 2021.

Table 4. Comparison of Blockchain Engine interoperability solutions [109, 137]

	Communication		Properties						Community	
	Cross-chain Protocol	Cross-blockchain interoperability	Consensus Mechanism	Security assumption	Validator number	Maximum Throughput	Number of instances	Smart Contracts	Launch	Roadmap
Polkadot [188] ✓	XCMP	●	BABE and GRANDPA	SM	197	10^3	200	WASM	November 2019	Main network launch
Cosmos [109] ✓	IBC Protocol	◐	Tendermint	SM	125	10^3	> 70	WASM	March 2019	Governance updates
ARK [10] ✓	SmartBridge	◐	Delegated proof of stake	M	51	18.5	Unlimited	WASM*	May 2019	ARK Swap Market
AION [165] ✓	Interchain transactions	○	Proof of intelligence	M	×	×	×	Aion Language	April 2018	Market assimilation

✓ our description was endorsed by the authors/team.
× not known.
∗ some languages compilable to WASM, such as Go and .NET, but not all of them.
● can interoperate with instances of the same blockchain engine. Interoperate with more than two heterogeneous blockchains.
◐ can interoperate with instances of the same blockchain engine. Interoperate with up to two heterogeneous blockchains.
○ can interoperate with instances of the same blockchain engine.

mechanism is used to recover the original tokens. This scheme allows for interoperability among Tendermint blockchains. Other kinds of blockchains can interoperate with a Cosmos chain via peg zones. Peg zones resemble the pegged sidechain mechanism [13], in which a representation of the locked token of the source blockchain is created on the target blockchain.

Cosmos abstracts the development of a blockchain into three layers: networking, consensus, and application. Tendermint BFT realizes the networking and consensus layers. The Tendermint BFT engine is connected to the application layer by a protocol called the **Application Blockchain Interface** (**ABCI**). The Cosmos SDK realizes the applicational layer, allowing developers to develop smart contracts in languages that can be compiled to WASM.[8]

5.2.1 Discussion on Blockchain of Blockchains. Blockchain of Blockchains implementations are similar to relays and sidechains, as there is typically the main chain (often called relay chain) that connects the secondary chains, which can be application-specific blockchains. This scheme allows high throughput and flexibility to the end-users, providing interoperability capabilities between their platform instances. For example, Cosmos's Tendermint-based blockchains interoperate (instant finality assured), while Polkadot provides interoperability on Substrate-based blockchains (for instance, via Cumulus,[9] a tool for connecting a blockchain to Polkadot). To connect to other chains, Cosmos, Polkadot, and AION, utilize a mechanism similar to pegged sidechains or hash lock time contracts (ARK [10]) to interact with other blockchains, commonly called bridges.

Table 4 maps out the current blockchain engine landscape by extracting and evaluating their main characteristics. Some information was not possible to obtain due to the lack of details on the whitepapers. It is possible to observe that Blockchain of Blockchains is very recent: Polkadot's test network, Kusama [142], was released in November 2019; Cosmos' main network was launched in March 2019. ARK launched in May 2019. AION launched in April 2018. Blockchain of Blockchains has different cross-chain communication protocol, e.g., in Polkadot, cross-chain message passing[10]; in Cosmos, the inter-blockchain communication protocol [109]. Cosmos and Polkadot have some differences regarding their approach: in Cosmos, the idea is to provide blockchains tailored to specific applications. IBC is more generic than XCMP, letting users customize their zones with

[8] https://blog.cosmos.network/announcing-the-launch-of-cosmwasm-cc426ab88e12.
[9] https://wiki.polkadot.network/docs/en/build-cumulus.
[10] https://wiki.polkadot.network/docs/en/learn-crosschain.

higher freedom: security and validation are decided per zone. Polkadot restricts this customization but offers a shared security layer, making a trade-off security customization.

The security assumptions criteria depict the number of nodes assumed to be honest. A **supermajority** (**SM**) assumes that at least two-thirds of the nodes are honest, a common condition required by Byzantine fault-tolerant consensus algorithms ($n > \frac{2}{3}$), while the **majority** (**M**) assumes at least half of the nodes are honest ($>\frac{1}{2}$). The validator number on a network comes with a tradeoff: while a higher number is generally better for decentralization and robustness, it comes with an increase of latency toward block production, and consequently lower throughput. Polkadot currently has around 297 validators, and this number is gradually increasing in the short term to support up to 100 first-level parachains. At the time of writing, Polkadot is developing bridges for Bitcoin [195], Tendermint, Libra [117], and Ethereum. Interoperability between parachains is provided by Substrate.

Currently, Cosmos has 125 validators. The number of validators can rise to 300. Currently, there are around 70 zones, and "the number is growing.' While Cosmos does hold a limit for zones (as each zone is self-sovereign), there is no limit for how many zones can be attached to a Hub. Cosmos can interoperate with Etheruem. The Cosmos SDK provides interoperability between zones. Cosmos supports multiple peg zone implementations for Bitcoin and one for Ethereum. ARK has 51 validators, which can validate the transactions of a number of blockchains bound to the company's physical resources (instances managed by ARK). ARK can send and receive ERC-20 tokens to the Ethereum blockchain. We found no information regarding AION's validator number, throughput, or maximum sub-chains [165]. The theoretical throughput of the presented solutions varies: Polkadot's relay chain supports around 1,000 transactions per second, considering that a block can include around 7,000 transactions at a 6-second block time (considering current weights, March 2021). Cosmos theoretical throughput can achieve up to dozens of thousands of transactions per second (tps) with two validators. With 64 validators, it falls into several thousand transactions per second. ARK can achieve around 18.5 transactions per second, relying on a proof of work consensus. The number of validators is set to 51. ARK is not a completely decentralized solution, as it manages instances of ARK blockchains. There is no theoretical limit of bridge chains, except the service provider resources. Several optimizations are being done in Cosmos, Polkadot, and ARK, to increase the throughput. The AION project looks deprecated and stalled. As stated, the "white paper is both ambitious and experimental" [165]. AION is now a part of a larger project called the ***Open Application Network*** (**OAN**).

Cosmos and Polkadot support smart contracts in languages compilable to **WASM** (**Web Assembly**), which means developers can write them in languages such as Go, C++, and JavaScript. AION would support domain-specific languages—Aion language. Blockchain of Blockchains instances achieve inter-chain interoperability by a common point of contact, the "connector," analogous with Hyperledger Fabric channels [7]. The connectors are the relay chain, the Cosmos Hub, the AION-1 blockchain, and the ARK main net if the technology is Polkadot, Cosmos Network, AION, or ARK, respectively. In Polkadot, the connector provides shared security. The relay chain (the chain that coordinates consensus and communication between parachains and external blockchains) connects parachains and parachains to bridges. In Cosmos, the connector is loosely coupled to blockchains, providing greater flexibility than Polkadot. We could not extract meaningful considerations about AION's connector. In ARK, it looks like the connector is centralized at the expense of developability and ease of use. Concerning cross-blockchain interoperability, all solutions rely on *bridges* or *adapters* that route transactions from a particular blockchain type to another.

While the provided features can be desirable for end-users, blockchain engines do not interoperate with each other. In light of this fact, end-users are obligated to choose between existing solutions, leading to sub-optimal leveraging of available resources. Therefore, participant networks

have constraints on interoperability, ending at relying on a single blockchain engine solution. Some authors defend that blockchain engine approaches are not universally accepted and cannot eliminate fragmentation [1]. Some solutions are even centralized, in the sense that its code is not open source, and the end-user needs to use an SDK to access core functionalities (e.g., [10, 165]). However, ongoing work on building a Tendermint light client for GRANDPA, which would allow Polkadot to interact with Cosmos, may allow blockchain engine interoperability in the short-medium term. Thus, in theory, interoperability across Blockchain of Blockchains can also be achieved via the relay chain technique (i.e., a blockchain engine can be a sidechain of other blockchain engines; validation can happen via SPV).

Moreover, Blockchain of Blockchains requires transaction fees to keep the network operating. Given enterprise blockchain systems, a question could be posed: at which point shall an organization pay fees to sustain its business model across several blockchains? While Cosmos can provide flexibility configuring a zone, on Polkadot this can be harder. Therefore, Blockchain of Blockchains can provide an optimal leveraging for public infrastructures, but that is not necessarily the case for private blockchains.

5.3 Hybrid Connectors

The *Hybrid Connector* category is composed of interoperability solutions that are not Public Connectors or Blockchain of Blockchains. Directed to both public and private blockchains, Hybrid Connectors attempt at delivering a "blockchain abstraction layer" [185], capable of exposing a set of uniform operations allowing a dApp to interact with blockchains without the need of using different APIs [68]. We derived a set of sub-categories from the studies available: *Trusted Relays*, *Blockchain Agnostic Protocols* (including *Blockchain of Blockchains*), and *Blockchain Migrators*. Trusted relays are directed to environments where a blockchain registry facilitates the discovery of the target blockchains. Typically, such a scheme appears in a permissioned blockchain environment, where trusted escrow parties route cross-blockchain transactions. As the name suggests, Blockchain-agnostic protocols provide technology-agnostic protocols for interoperation between distributed ledger systems but do not guarantee backward compatibility. In other words, to use such protocols, their source code has to be changed to existing blockchains to use such protocols. Solutions from the blockchain of blockchains category aim to provide mechanisms for developers to build cross-chain dApps. The blockchain migrators sub-category aggregates solutions that perform data migration across blockchains, which resemble the notary schemes discussed in Section 5.1.2 (as there is typically a centralized party mediating the migration process).

We introduce each sub-category, presenting only one illustrative example of each for the sake of space. Appendix E depicts a complete list of Hybrid Connectors. Evaluation tables for each sub-category are discussed in Section 5.3.4.

5.3.1 Trusted Relays. Trusted relays are trusted parties that redirect transactions from a source blockchain to a target blockchain, allowing end-users to define arbitrary business logic. These solutions imply managing different APIs, in which cross-chain consensus may be modular.

Hyperledger Cactus (Cactus), previously known as Blockchain Integration Framework, uses an interoperability validator network that validates cross-chain transactions, optionally using a trusted escrow party [129]. However, decentralized validators are implemented as well, making this project move toward a decentralized trusted relay. Cactus allows a party or a set of parties to issue transactions against several ledgers, similarly to some notary scheme solutions [89, 157]. The interoperability is enabled through a set of *interoperability validators*, which are participants from the source and target blockchains. Such validators collect cross-chain transaction requests, sign, and deliver them. A CB-Tx is deemed valid, given that a quorum of validators signs them. It is then assumed that the blockchains participating in the network know how to address each

other. However, trusted escrows can be replaced by decentralized parties. Currently, Hyperledger Cactus[11] supports Hyperledger technologies (e.g., Fabric, Besu), Corda, and Quorum. The roadmap predicts integration with public blockchains and blockchain migration capabilities.

5.3.2 Blockchain-Agnostic Protocols. Blockchain-agnostic protocols enable cross-blockchain or cross-chain communication between arbitrarily distributed ledger technologies by providing a blockchain abstraction layer. These solutions enable BoBs, "a system in which a consensus protocol organizes blocks that contain a set of transactions belonging to CC-dApps, spread across multiple blockchains. Such system should provide accountability for the parties issuing transactions on the various blockchains and providing a holistic, updated view of each underlying blockchain" (Section 2.3). Typically, the cross-chain consensus is fixed, and business logic is more restricted.

The ***Interledger Protocol*** (**ILP**) can be considered a decentralized, peer-to-peer payment network [173]. It firstly adopted a generalized hash locking scheme to enable asset transfers, and it was directed to cryptocurrency transfers. Nowadays, ILP is technology-agnostic, defining a "lowest unit common denominator" across distributed ledgers, blockchains, fiat payment networks, and the ILP packet.

ILP sends payment information in packets by leveraging a network of connectors, which route such packets. At the core of Interledger is the Interledger Protocol (ILPv4) [97], which defines how senders, routers (or node, or connector), and receivers interact. Typically, the connector is a money packet router. The root of trust is then the connector, which has to be trusted: companies can settle payments via the routers, given that clearance of such payments is done afterward while being protected by the law. A sender is an entity that initiates a value transfer. A router applies currency exchange and forwards packets of value. The receiver obtains the value transmitted. ILPv4 is a request/response protocol enabled by ILPv4 packets. Each packet contains transaction information, and can be divided into *prepare*, *fulfill*, and *reject* packets. A sender node initiates an exchange of value by sending a *prepare* ILPv4 packet to a receiver. When a receiver obtains the prepared packet, it sends the response back to the sender via routers. The response may be a *fulfill* packet, whereby a transaction has been successfully executed, or a reject packet.

Several specifications for Interledger and related protocols are available.[12] The ILP is discussed by a W3C community group[13] and has a proposal that "describes data structures and formats, and a simple processing model, to facilitate payments on the Web."[14] The ILP cannot integrate with existing blockchains: each one must be adapted to use ILP. A disadvantage is that Interledger does not support the transfer of non-fungible tokens (such as ERC-721[15] tokens).

5.3.3 Blockchain Migrators. Blockchain migrators allow an end-user to migrate the state of a blockchain to another. Currently, it is only possible to migrate data across blockchains, although moving smart contracts is also predicted [129].

Fynn et al. present an abstraction for smart contracts to switch to another blockchain consistently, moving the state required by the transaction to the target blockchain and execute it [75]. The authors call such abstraction the *Move* operation. The operation works as follows: first, it locks a smart contract on the source blockchain; next, the Move protocol recreates the smart contract in the target blockchain. This method allows arbitrary states to be transferred between blockchains. For example, it allows transferring cryptocurrencies by creating tokens on the target blockchain backed up by locked tokens on the source blockchain (similarly to pegged sidechains). This method

[11] https://github.com/hyperledger/cactus.
[12] https://github.com/interledger/rfcs.
[13] https://www.w3.org/community/interledger/.
[14] https://w3c.github.io/webpayments/proposals/interledger/.
[15] http://erc721.org/.

was tested on Ethereum and Hyperledger Burrow (based on Ethereum). The solution assumes the same cross-blockchain smart contracts utilize the same virtual machine, which can be limiting. Furthermore, for such a solution to be deployed, it requires Solidity changes and possibly a soft fork on Ethereum.

5.3.4 Discussion on Hybrid Connectors. This section defined the hybrid connector category and its sub-categories: trusted relays, blockchain-agnostic protocols, and blockchain migrators.

Regarding centralization, almost all adopt a decentralized model. Permissioned blockchain solutions are less flexible, as all involved participants are identified. In particular, trusted relays endorse connections made in a peer-to-peer fashion, upon previous agreement [1, 78]. However, Abebe et al.'s work poses some limitations: interoperating networks require *a priori* knowledge of each other's identities and configurations, hence being static. A discovery service could be implemented using a blockchain registry or a pub-sub mechanism [78], in which networks could be added and removed. In trusted relays, it is not completely clear the mechanisms to minimize malicious relay services, apart from replication (whereby the risk of a censorship attack is reduced but not erased). Hyperledger Cactus could be a true enabler of interoperability, given that a (decentralized) trusted blockchain registry would be deployed, and public escrow parties could replace the overlay of trusted parties. Cactus could, therefore, make the transition between a trusted relay to a semi-trusted relay or even a trustless relay.

Blockchain-agnostic protocols will be better positioned to offer interoperability to existing and yet-to-exist blockchains, but most do not grant backward compatibility and lack the flexibility to define business logic. This inflexibility is inherent to the provided homogeneous interfaces (containing roles, methods, data, message formats, for instance [68]); at least such solutions scale slowly, as adding methods compatible with all the supported blockchains incur in a polynomial effort. However, this category might resemble some of the trusted relay solutions. In particular, both Cactus [129] and SCIP [68] rely on connectors and validators and gateways to access the underlying blockchains. The gateway paradigm implies a (semi-) trusted gateway having read/write access to the shared ledger of the blockchain, and often they are expected to participate in the consensus mechanism of the blockchain [84]. While there is a higher trust requirement, gateway approaches might be the most suitable to solve interoperability across private blockchains if gateways are framed in a legal and regulatory framework. Enterprise-grade solutions require infrastructure that include support for authentication, authorization, accountability, and a set of connectors. Cactus can provide such features for gateways.

From the blockchain of the blockchains category, we highlight Hyperservice, a peer-reviewed paper, and Overledger. Hyperservice tries to achieve full dApp atomicity by introducing the concept of *stateless smart contracts*. Using a stateless smart contract, a CC-dApp can load a clean state for a contract, using a valid block. While it can partially solve forks in the underlying blockchains a CC-dApp utilizes, the application of this concept paves a direction to decouple smart contract execution from the consensus layer [120]. Overledger is a sorted list of messages that are interpreted as the state of a cross-blockchain application. While this is an exciting approach to blockchain interoperability, the solution is proprietary, hindering community efforts for more complex solutions.

Blockchain migrators respond to an enterprise need: migration in case of disaster or performance issues [14, 19]. The two presented solutions can only provide data portability across a small set of public blockchains. It is currently impossible to reproduce the chain of events via smart contracts, as that requires a smart-contract translator functionality.

A limitation that we identified in the context of Hybrid Connectors is that most solutions do not support hard forks (i.e., the separation of a blockchain into two different blockchains) or propose

a solution for eventual forks, unlike some public connectors (most HTLCs and notary schemes). Forks do not happen regularly, and some solutions offer a quick analysis of the problem and acknowledge their importance [98, 129, 178]. However, this is still a problem that can affect the dependability of cross-chain dApps; dealing with forks is still an open issue. For instance, the protocol used in Hyperservice is unable to revert any state update to smart contracts when a dApp terminates prematurely, i.e., it does not grant atomicity. If one does not have atomicity guarantees, it forces the cross-blockchain application into an inconsistent state when a fork occurs. This can put at risk the purpose of the project: functional cross-blockchain applications. The same problem applies to, for instance, Overleder [147].

While one might be tempted to conclude that standardization could improve cross-blockchain API design, some argue that APIs are unlikely to generalize well across radically different technologies. Blockchain-agnostic protocols are more likely to be standardized than APIs, as shown historically by successful standards efforts such as HTTP or the TCP/IP family. Finally, solutions that prove cross-smart contract capabilities are emerging, but are still in development [1, 98, 120, 156, 178].

6 DISCUSSION, USE CASES, AND RESEARCH QUESTIONS

This section presents a comprehensive summary of each blockchain interoperability category we extracted and our considerations about blockchain interoperability. Then it presents use cases and finishes with answers to the research questions we proposed.

6.1 Discussion

Although blockchain interoperability is a complex technology, connecting blockchains ends up being a manageable approach, despite differences in, for example, data structures, digital signature schemes, transmission protocols, verification mechanisms, consensus mechanisms, token issue mechanisms, and smart contract language. However, "there is a scant effort today to address the standardization of the various infrastructure building blocks—messages, data formats, and flows—to support the interoperability across blockchains" [84].

Different categories of solutions approach the interoperability problem differently. Our article firstly introduced Public Connectors in Section 5.1 and stressed their importance. Token exchange is arguably no longer the whole scope of blockchain interoperability [120]. Instead, various interoperability approaches emerged in the last years, whereby many of them aimed at generalizing blockchain interoperability. In particular, emerging solutions can be categorized as Hybrid Connectors, which provide cross-blockchain communication, and Blockchain of Blockchains, which allow an end-user to create customized, interoperable blockchains at the expense of vendor lock-in.

Public connectors are the most cited among industry and academia, as they provide practical solutions to real-world problems: asset transfers. As these were the first solutions to emerge, not surprisingly, some may not succeed. It seems that the merge of sidechain and protocols relying on an escrow party (enforced by smart contracts) are the most suitable solutions for interoperability among public blockchains. We argue that the flexibility, decentralization, and security of such proposals can be utilized for secure interoperability. However, creating and maintaining a decentralized application using several blockchains was difficult—and hence the Blockchain of Blockchains solutions appeared. Those can facilitate blockchain adoption while providing built-in interoperability among instances of the same platform, whereas variations of the solutions mentioned above can be used to bridge Blockchain of Blockchains to other blockchains.

While Blockchain of Blockchains, such as Cosmos or Polkadot, provide a consensus engine and a security infrastructure to build blockchains, blockchain of blockchains aims at developing solutions using different infrastructures. In particular, Cosmos and Polkadot might progress toward

homogeneity, as they support only the creation of Tendermint-based blockchains and Substrate-based blockchains, respectively. While they provide interoperability capabilities, mainly on the chains relying on their technology and other desirable features (shared layer of security, decentralization, governance, better scalability), the end-users choice will be tied to specific implementations. Paradoxically, such solutions might contribute to data and value silos, as solutions built with them cannot connect with an arbitrary blockchain [1]. Despite this fact, one could argue that this problem can be alleviated by building bridges/adapters. These solutions are promising but are challenging to integrate with legacy systems and, generally, private blockchains, and hence the hybrid connectors started appearing.

Hybrid Connectors, specifically blockchain migrators and blockchain of blockchains, progress toward a user-centric, blockchain-agnostic view, enabling enterprise-connected CC-dApps. Arguably, the most suitable solution for connecting private blockchains is the usage of blockchain-agnostic protocols; however, they do not grant backward compatibility (as all previous solutions have to be adapted to integrate the adopted communication protocol). To overcome this fact, the short-medium-term solution would be using trusted relays. An interesting way for trusted relays to venture is by decentralizing the escrow party: from a set of trusted validators to a network of public nodes. It then follows from this survey that one could perceive trusted relays and blockchain-agnostic protocols to be good solutions to link private blockchains; and sidechain, smart-contract-based protocols suitable to solve interoperability among public blockchains.

A network of blockchain engine-powered blockchains can be leveraged using Hybrid Connectors. For instance, there is a possible synergy between Cosmos and the Interledger Protocol: when a user wants to make an in-app payment with fiat currency (e.g., dollars) within a Cosmos zone, he or she can rely on the interledger protocol as a payment rail. If using cryptocurrencies to pay (e.g., Bitcoin), the interledger router can route the transactions for a payment channel (e.g., LN), providing more trustful interaction. To connect this ecosystem to private blockchains, bridges have to be developed. To make such bridges trustable, a possible solution would be to elect a group of validator nodes, via an overlay network, that participates in the consensus of public blockchains and private blockchains. This way, cross-chain and cross-blockchain transactions can be endorsed.

It is worth mentioning that several cross-chain programming languages are appearing, such as the Hyperservice Language [119] and DAML [61]. DAML provides a unified Blockchain programming model by abstracting the underlying blockchains and exposing a higher-level abstract ledger on top, similarly to HSL. DAML has different integration degrees: DAML as an application on the target platform; and integration where the DAML runtime engine validates transactions. Programs compiled on such languages can run on top of a BoB platform.

To conclude this discussion, we recall to the reader that blockchain development has been done in silos since its inception. New solutions for blockchain interoperability started emerging as of 2017, and, perhaps not surprisingly, such solutions are also being adopted in silos. While Public Connectors methods are commonly used nowadays, we focus on Blockchain of Blockchains and Hybrid Connectors. Blockchain of Blockchains and Hybrid Connectors allows interoperability between blockchains and other distributed ledger technologies and enterprise systems in the medium term. This promotes the development of blockchain interoperability standards. While blockchain matures, industries will tend to incorporate this technology into their business processes. Then, we predict that mass adoption will follow.

6.2 Supporting Technologies and Standards

Besides the presented solutions, there is work toward the support and standardization of blockchain interoperability. Blockchain interoperability standards attempt to create a "standardized transaction format and syntax," which is common to all blockchains, and secondly, a "standardized

minimal operations set," common to all blockchains [86]. In particular, a standardized format is important because while fungible and non-fungible assets have a single, well-defined representation in each blockchain, arbitrary data can be manipulated freely. First, we introduce indirect contributions that promote blockchain interoperability and then the existing standards.

Recent efforts are visible in enabling heterogeneous smart contract integration through service orientation [69], allowing external consumer applications to invoke smart contract functions uniformly. A cross-blockchain data storage solution becomes a feasible solution to achieve application interoperability, whereby applications rely on one blockchain. Some dApps[16] already leverage the **InterPlanetary File System** (**IPFS**) [26] to create a common storage, adjacent to the blockchain. The IPFS provides a peer-to-peer network for storing and delivering arbitrary data in a distributed file system, potentially facilitating the transfer of data across blockchains [21]. Organizations are working on standardizing digital assets. The Token Taxonomy Initiative[17] is a consortium dedicated to digital token standardization. It proposes a standard to identify tokens' behavior, properties, and control interfaces according to a token classification hierarchy. This project allows application developers to utilize a standard code set for interacting with tokens regardless of the blockchain platform, thus incentivizing blockchain interoperability. In the context of general interoperability, the Ethereum ERCs are a de facto standard.[18]

Oracles are mechanisms that software systems provide as an external source of truth for blockchains [130], and they can be centralized or decentralized [4]. Typically, centralized oracles are not as dependable as decentralized oracles, as they constitute a single point of failure.

Hyperledger Avalon [123] defers intensive processing from the main blockchain to an off-chain channel to support centralized yet trustable oracles (by using trusted execution environments). Since multiple blockchains can use the same data, it fosters interoperability.

Open source projects like Hyperledger Indy[19] and Hyperledger Aries[20] operate in the field of digital identity and self-sovereign identity. Central concepts of self-sovereign identity are **decentralized identifiers** (**DIDs**) [149] and verifiable credentials [42]. DIDs can be created, managed, and shared using **Zero-Knowledge Proofs** (**ZKPs**) mechanism, even allowing one to create new access control models [20]. Such technologies allow for identity portability, enabling cross-blockchain identities [92].

So far, the presented standards are called DLT/Blockchain Enabling Technology Standards because they focus on standardizing elements that blockchains can use, as opposed to DLT/Blockchain Generic Framework Standards [118]. These refer to standardization of blockchain interoperability data and metadata formats, identity, and protocols, namely, the IETF, ISO, Enterprise Ethereum Alliance, IEEE, The EU Blockchain Observatory & Forum, and W3C.

At the **Internet Engineering Task Force** (**IETF**), work is being done defining a set of drafts that guide the implementation of ODAP, a protocol using gateways [17, 85, 88]. The ISO Technical Committee 307 works toward the "standardization of blockchain and distributed ledger technologies,"[21] but did not produce any standard yet. Subgroup 7 (ISO/TC/SG7) focuses specifically on interoperability. The Enterprise Ethereum Client Specification, currently on its seventh version, "defines the implementation requirements for Enterprise Ethereum clients, including the interfaces to external-facing components of Enterprise Ethereum and how they are intended to be

[16] https://ethlance.com/.
[17] https://tokentaxonomy.org/.
[18] https://eips.ethereum.org/erc.
[19] https://www.hyperledger.org/projects/hyperledger-indy.
[20] https://www.hyperledger.org/projects/hyperledger-aries.
[21] https://www.iso.org/committee/6266604.html.

used," including cross-chain interoperability [5]. The IEEE Blockchain Initiative[22] and the IEEE Standards Association,[23] through the IEEE Standards P3203, P3204, and P3205,[24] work at providing "interfaces and protocols of data authentication and communication for homogeneous and heterogeneous blockchain interoperability." The EU Blockchain Observatory & Forum by the European Commission aims to (1) monitor blockchain activities in Europe, (2) manage the source of blockchain knowledge, (3) create a forum for sharing information, and (4) create recommendations on the role the EU could play in blockchain [67]. The same entity points out the likelihood of an increasing number of standards and adoption within governments [55]. The W3C, via the Interledger Payments Community Group,[25] is connecting payment networks, including decentralized ledgers. Other organizations working in the area include BIA, BiTA, BRIBA, BSI, CESI, DCSA, EBP, GS1, and MOBI [185].

Standardization efforts focused on a specific blockchain (DLT/Blockchain Platform-Specific Standards) are, for example, the 0302 Aries Interop Profile[26] and the Hyperledger Fabric Interoperability working group.[27]

Multiple standards will likely arise and be used, for each vertical industry, as there is a lack of generalized interoperability standards. Standards are then reused across industries (e.g., IEEE P2418.5). Solving interoperability in a specific sector would then pave the way for standards in other industries because the main requirement is domain expertise (ontologies are good starting points for standardization) [118]. The heterogeneity created by standards will pose a regulation challenge, as blockchains may spread across different jurisdictions [23].

6.3 Use Cases with Multiple Blockchains

In this section, we present use cases with multiple blockchains. More use cases can be found in Appendix F.

The industry is still applying blockchain to use cases using only one blockchain. Consequently, it is expected that use cases with multiple blockchains are rare. Notwithstanding, according to the existing resources, it seems that there is considerable interest in use cases using multiple blockchains. As long as the technologies mature, novel, disruptive use cases may be found. For the sake of space, we present some general use cases involving an IoB [155]. We refer readers to Appendix F for more use cases relative to Public Connectors, Hybrid Connectors, and Blockchain of Blockchains.

The first big IoB use case is asset transfers, where users can transfer assets from one blockchain to another. While some approaches implement this use case in an ad-hoc way, the emergence of **central bank digital currencies (CBDCs)** [126, 166], requires further efforts and standardization [49]. A CBDC is a digital version of a sovereign currency of a nation. A CBDC is issued by central banks, where each unit represents a claim on the value held by such central bank. Many blockchains features are appealing to implement CBDCs, particularly the offered immutability, transparency, and trust distribution. Some central banks are already experimenting with blockchain, including the Monetary Authority of Singapore and the Bank of Canada [155]. As each CBDC can be implemented with a blockchain, and each central bank might choose a different technology, interoperability between them is achieved using an IoB or even a BoB.

[22] https://blockchain.ieee.org/standards.
[23] https://standards.ieee.org/.
[24] https://blockchain.ieee.org/standards.
[25] https://www.w3.org/community/interledger/.
[26] https://github.com/hyperledger/aries-rfcs/tree/master/concepts/0302-aries-interop-profile.
[27] https://wiki.hyperledger.org/display/fabric/Fabric+Interop+Working+Group.

Another major use case is interoperability across supply chains [129, 155]. A supply chain is a chain of value transfer between parties, from the raw product (physical or intellectual) to its finalized version. Managing a supply chain is a complex process because it includes many non-trusting stakeholders (e.g., enterprises, regulators). As many markets are open and fluid, enterprises do not take the time to build trust, and instead, rely on a paper trail that logs the state of an object in the supply chain. This paper trail is needed for auditability and typically can be tampered with, leading to blockchain's suitability to address these problems [185]. A key challenge of blockchain-based supply chains is to interoperate with other DLT systems. Interoperability granted each participant of the supply chain (e.g., supplier, manufacturer, retailer) can participate at several supply chains (and thus several blockchains) using a single endpoint, simplifying the interaction process while reducing costs. Other use cases comprise connecting Hyperledger Fabric and Ethereum with Singapore Exchange and Monetary Authority of Singapore via node integration and EVRYTHNG, a product connecting multiple chains via API to digitize products [185].

Finally, identity and data portability can be provided by an IoB approach. Identity paradigms like self-sovereign identity [20] can increase identity portability by providing users control of their identities. Typically, this is achieved by rooting user credentials in a blockchain. Hence, if blockchains can communicate with identity providers that are blockchains, one can use the same identity in different blockchains. Data portability complies with blockchains, allowing blockchain users to use their data outside of a blockchain without requiring significant effort.

6.4 Answers to the Research Questions

In this section, we provide answers to the presented research questions (further elaborated in Appendix A1).

(1) **What is the current landscape concerning blockchain interoperability solutions, both from the industry and the academia? That is, what is the current state of blockchain interoperability solutions?**
The first step toward blockchain interoperability has been creating mechanisms allowing the exchange of tokens (e.g., cryptocurrencies). We categorized such solutions as Public Connectors (Section 5.1). Such category comprises Sidechains and Relays (Section 5.1.1), Notary Schemes (Section 5.1.2), and Hash Time-Lock Contracts (Section 5.1.3). This category provides an idea of the emergence of blockchain interoperability, but this area no longer applies solely to token exchanges between homogeneous blockchains.

Novel blockchain interoperability approaches are Blockchain of Blockchains (see Section 5.2) and Hybrid Connectors (Section 5.3). Hybrid Connectors fall into three subcategories: trusted relays (Section 5.3.1), blockchain-agnostic protocols (Section 5.3.2), and blockchain migrators (Section 5.3.3). We also analyzed related literature reviews on blockchain interoperability in Section 3.

(2) **Is the set of technological requirements for blockchain interoperability currently satisfied?**
There are two requirements for realizing technical interoperability [44]: a pair of sufficiently mature blockchains to build artifacts that promote interoperability and "some application or need that cannot be served by implementing it on a single blockchain." There are several blockchains that can be considered mature enough to support applications built on top of them [7, 80, 109, 188]. On the other hand, interoperability regarding blockchain needs to have the necessary infrastructure and facilitating technologies. In particular, the production of standards [92, 175] technologies like decentralized identifiers [149], verifiable credentials [42], cross-blockchain communication protocols [38, 194, 195], and the representation

of blockchain smart contracts [92] can foster the likelihood for blockchain interoperability standards and solutions, as they remove considerable barriers to blockchain interoperability.

On the other hand, there is a set of cross-blockchain use cases that validate the need for interoperability, which will inevitably foster it [23]. In conclusion, the set of critical requirements for blockchain interoperability is currently satisfied, but there is still work to be done at standardization and interoperability across public-private and private-private blockchains.

(3) **Are there real use cases enabling a value chain coming from blockchain interoperability?**
Regarding the third research question, some authors defend that blockchain interoperability is important and crucial for the survivability of this technology [23, 86, 120, 138]. Standards are paving the way for blockchain adoption [60, 92]. It is likely that "forward-looking interoperability standards are most likely to result in successful standards creation and facilitate industry growth" [92]. Conversely, standardization is a requirement for mass adoption that is being developed. Given the multiple blockchain interoperability solutions, both Hybrid Connectors and Blockchain of Blockchains, some of them with considerable weight in the industry, we believe this is a very likely scenario. In Section 6.3, we expose multiple use-cases that may benefit from cross-blockchain technology, which can foster adoption by enthusiasts and enterprises. In conclusion, we envision reliable solutions and standards emerging in the following years and a steady increase in blockchain adoption by enterprises and individuals alike.

As a value enhancer and maturing key factor, interoperability will ultimately decide the survival of this technology. Based on the evidence collected and analyzed, we foresee increased attention to this research area, with blockchain interoperability gaining traction among the academia and the industry.

6.5 Open Issues and Challenges

In this section, we present open issues and challenges regarding blockchain interoperability and, in a more general sense, the adoption of blockchain.

Nowadays, solutions available to build decentralized applications lack interoperability, thwarting scalability [27]. As Liu et al. note, "it is very challenging to enforce correct executions in a full trust-free manner where no trusted authority is allowed to coordinate the executions on different blockchains" [120]. Although interesting and notorious recent advances in this area make interoperability a reality, there is still a gap between theory and practice, as much of the existing work is mostly conceptual.

Given the vast amount of blockchain platforms, fragmentation regarding solutions and their approach to interoperability is strongly present, for example, in IoT scenarios [200]. A combination of multiple platforms tailored for specific purposes, which can be public, private, or consortium, adds an overhead to manage workflows. In particular, this concern is intensified when multiple blockchains are serving a specific application.

Concerning blockchain scalability, the internet of blockchains can be realized upon improvements to current performance, both in public and private blockchains. Techniques such as implicit consensus and data sharding can improve transaction throughput and storage [107]. However, blockchain sharding requires solving cross-blockchain transaction routing and retrieval and asset referencing (also known as the discoverability problem).

It is challenging to coordinate transactions from different blockchains to support a cross-chain dApp, as different blockchains have different properties (e.g., architecture, protocols [1], service

discovery, access control, between others). In particular, reverting a transaction that depended on another can be cumbersome, especially given different transaction finalities from different blockchains). Some solutions have proposed a mechanism to overcome such a challenge (blockchain of blockchains) [120, 178]. Although a promising approach, it is still unclear the applicability of these solutions to arbitrarily complex cross-blockchain dApp. More research is required to confirm the feasibility of this approach.

Some authors [170] highlight problems related to the **General Data Protection Regulation (GDPR)**[28], such as security, trust, confidentiality, and data privacy issues. In particular, security threats are exacerbated by the presence of multiple blockchains and possible multiple administrators. Regarding privacy, the authors underline problems with the right-to-forget, in which a user can ask his or her data to be deleted from the blockchain. Currently, most blockchains do not provide effective mechanisms that can respond to this request. Blockchain fine-grain access control is appointed as a key requirement to minimize information leakage and confidentiality risk.

Blockchain interoperability reduces dependencies on a single blockchain, and consequently, risk (e.g., the blockchain is attacked) [38]; it does not eliminate the inherent risks. It is worth underscoring that the multiple blockchain approach is more complicated than the sum of its parts, as there is extra complexity underlying the cross-chain communication. This adds challenges to governance: whereas a private consortia can use Hybrid Connectors at will to interoperate systems (decentralized and centralized), the governance model is not straightforward within community projects, supported by public blockchains.

In short, the most relevant open issues toward blockchain interoperability are as follows:

—The gap between theory and practice, including the lack of standardization and implementations [84, 200].
—Discoverability [1, 120, 178].
—Privacy and Security [170, 173, 188, 194].
—Governance [86, 87, 145, 185].

Notwithstanding, security [116, 148], privacy [47], and scalability (e.g., using sharding [193] or novel blockchain systems [135]) remain the most prominent areas to be improved in the blockchain space.

7 RESEARCH DIRECTIONS

New tools, frameworks, standard proposals, and even programming models are emerging and need further development. Programming models such as Polkadot and Cosmos offer developers a way to create their blockchains effectively and connect them to other blockchains. Protocols such as ILP and UIP allow cross-blockchain transactions. Programming languages such as HSL and DAML aim at embedding different blockchain models, providing an abstraction for cross-blockchain dApps.

Although one can have good reasons to utilize blockchain interoperability solutions for public or private blockchains, few solutions are available for connecting them. The problem of obtaining state from permissioned blockchains effectively [2] makes interoperating with private blockchains a challenge [96, 185]. Thus, connecting public and private blockchains bidirectionally remains an open problem.

One of the problems that bidirectional communication across permissioned and permissionless ledgers poses is semantic compatibility. Technical interoperability does provide the technical foundation that realizes interoperability but does not grant semantic interoperability per se [86]. There is, therefore, a gap: how can we effectively combine both blockchain types to enable new use cases?

[28] https://gdpr.eu/.

How to make sure a solution complies with the goals of all involved stakeholders? Disciplines as view integration can help to provide an answer [19]. View integration is the process that combines views of the same business process into a consolidated one by combining the different views of the stakeholders participating in different blockchains.

Another considerable obstacle for blockchain adoption is its fast-paced development. The development of blockchain interoperability standards may provide a way for more flexibility regarding backward compatibility.

In the light of the present study and the identified open issues and challenges, we propose research directions based on some sections of our survey: research on architecture for enabling blockchain interoperability, Public Connectors, Blockchain of Blockchains, Hybrid Connectors, and supporting technologies, standards, use cases, and others.

Architecture for Blockchain Interoperability (further elaborated in Appendix B):

—Define a blockchain interoperability maturity model, modeling interoperability at its various layers (e.g., technological, semantic, organizational).
—Model the different views on the various types of interoperability, according to different stakeholders (e.g., the provider's technical view on a cross-blockchain dApp vs. the semantic view of the end-user on the same cross-blockchain dApp).
—Study blockchain interoperability semantics by exploring, for example, the research area of view integration [51].

Public Connectors (Section 5.1):

—Research on how permissioned blockchains can benefit from sidechains to improve scalability and privacy.
—Develop protocols to allow fiat money exchange, higher liquidity on decentralized exchanges. Conversely, improve the level of privacy and security of centralized exchanges.

Blockchain of Blockchains (Section 5.2):

—Integration of existing blockchain systems with Blockchain of Blockchains.
—Study how Blockchain of Blockchains can provide a reliable interoperability scheme bridging permissioned blockchains and permissionless blockchains.
—Connect Blockchain of Blockchains to both centralized systems and decentralized ledger systems (e.g., connect Polkadot to Visa).

Hybrid Connectors (Section 5.3):

—Decentralize the trust of trusted relays by integrating them with public blockchains (e.g., by submitting the state periodically to a public blockchain).
—Study how blockchain-agnostic protocols can be easily adapted to existing ledgers.
—Explore the blockchain of blockchains approach as an advance in dependable blockchain-based applications.
—Improve atomicity and consistency guarantees on cross-blockchain decentralized applications.
—Explore blockchain migration across public and permissioned ledgers. Such migration schemes can be decentralized and adapt to functional and non-functional requirements imposed by stakeholders.
—Explore blockchain migration via non-trusted relays (e.g., using a set of public escrow nodes following a protocol).
—Develop frameworks for multiple blockchain management. Such frameworks should respond to multiple stakeholder needs, decentralizing trust.

—Model integration abstraction layers that enable the development of universally connected contracts.
—Research on the visualization of CC-Txs.

Supporting technologies and standards, use cases, and others (Section 6.1):

—Work along with regulators and standardizing bodies to come up with blockchain interoperability standards across industries.
—Research on blockchain interoperability programming languages, supporting tools, and standards, including but not limited to cross-blockchain programming languages and frameworks, decentralized identifiers and verifiable credentials, and blockchain interoperability standards for enterprise blockchains.
—Explore new use cases using multiple blockchains, the "value-level" interoperability [129].
—Research synergies between cryptocurrency-based interoperability approaches, Blockchain of Blockchains, and Hybrid Connectors.
—Study security aspects of blockchain interoperability.
—Understand the implications of the different interoperability layers (value, semantic, organizational, among others).

8 CONCLUSION

In this article, we performed a systematic literature review on blockchain interoperability. We systematically analyzed, compared, and discussed 80 documents, corresponding to 45 blockchain interoperability solutions. By including grey literature, we expect to thwart intrinsic limitations regarding the blockchain interoperability research area, such as a considerable presence of the industry. By exploring each solution methodologically, this study provides interesting insights, distributed across three categories: Public Connectors, Blockchain of Blockchains, and Hybrid Connectors. Despite sidechain and HLTC solutions are gaining traction in the industry, blockchain interoperabilities are not solely Public Connectors solutions. New approaches started emerging since 2017. HCs provide a varied landscape of solutions, adapted for the majority of the use cases. They are likely to be used to produce cross-blockchain dApps. Blockchain of Blockchains are likely to be adopted by the industry in the short-medium term, by leveraging easy-to-produce, customizable blockchains.

Our findings allow us to conclude that conditions to research on blockchain interoperability are fulfilled, allowing a multitude of new use cases. Thus, we expect interest in this research area to raise considerably.

This work is toward making the blockchain ecosystem more practical, by easing work for developers and researchers. We expect that this study provides a robust and dependable starting point whereby developers and researchers can work in the blockchain interoperability research area.

ACKNOWLEDGMENTS

The authors would like to thank to the anonymous reviewers that constructively provided suggestions that significantly improved this article. Thanks to Peter Somogyvari, Paul DiMarzio, Jag Sidhu, Sergio Lerner, Andy Leung, Travis Walker, Bill Laboon, Josh Lee, Austin King, Oliver Birch, Thomas Hardjono, and Miguel Matos for fruitful discussions regarding blockchain interoperability. We thank Daniel Hardman and Ken Elbert for constructive discussions about DIDs and verifiable credentials. Special thanks go to Iulia Mihaiu, Cláudio Correia, Benedikt Putz, Francisco Braga, Gavin Wood, João Ferreira, Miguel Pardal, Jonas Gehrlein, and Dilum Bandara for constructive comments and suggestions that greatly contributed to improving the article.

REFERENCES

[1] Ermyas Abebe, Dushyant Behl, Chander Govindarajan, Yining Hu, Dileban Karunamoorthy, Petr Novotny, Vinayaka Pandit, Venkatraman Ramakrishna, and Christian Vecchiola. 2019. Enabling enterprise blockchain interoperability with trusted data transfer. In *Proceedings of the 20th International Middleware Conference Industrial Track*. Association for Computing Machinery, 29–35.

[2] Ermyas Abebe, Dileban Karunamoorthy, Jiangshan Yu, Yining Hu, Vinayaka Pandit, Allison Irvin, and Venkatraman Ramakrishna. 2021. Verifiable observation of permissioned ledgers. arXiv (2021). Retrieved on 2 February, 2021 from https://uniswap.org/whitepaper.pdf.

[3] Hayden Adams, Noah Zinsmeister, and Dan Robinson. 2020. *Uniswap v2 Core*. Technical Report.

[4] John Adler, Ryan Berryhill, Andreas Veneris, Zissis Poulos, Neil Veira, and Anastasia Kastania. 2018. ASTRAEA: A decentralized blockchain oracle. In *2018 IEEE International Conference on Internet of Things (iThings), IEEE Green Computing and Communications (GreenCom), IEEE Cyber, Physical and Social Computing (CPSCom), and IEEE Smart Data (SmartData)* (2018), 1349–1354.

[5] Enterprise Ethereum Alliance. 2021. *Enterprise Ethereum Alliance Client Specification v7*. Technical Report. Ethereum. Retrieved on 3 March, 2020 from https://entethalliance.github.io/client-spec/spec.html.

[6] Emmanuelle Anceaume, Antonella Del Pozzo, Romaric Ludinard, Maria Potop-Butucaru, and Sara Tucci-Piergiovanni. 2018. Blockchain Abstract Data Type. http://arxiv.org/abs/1802.09877

[7] Elli Androulaki, Artem Barger, Vita Bortnikov, Srinivasan Muralidharan, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Chet Murthy, Christopher Ferris, Gennady Laventman, Yacov Manevich, Binh Nguyen, Manish Sethi, Gari Singh, Keith Smith, Alessandro Sorniotti, Chrysoula Stathakopoulou, Marko Vukolić, Sharon Weed Cocco, and Jason Yellick. 2018. Hyperledger fabric: A distributed operating system for permissioned blockchains. In *Proceedings of the 13th EuroSys Conference, EuroSys 2018*, Vol. 2018-Jan. Association for Computing Machinery, Inc., New York, New York, 1–15.

[8] Andreas M. Antonopoulos and Gavin Wood. 2018. *Mastering {Ethereum}: Building smart contracts and dApps*. O'Reilly Media.

[9] V. Arasev. 2017. *POA Network Whitepaper*. Technical Report. POA Network. Retrieved on 15 May, 2020 from https://www.poa.network/for-users/whitepaper.

[10] ARK. 2019. ARK Whitepaper Version 2.1.0. Retrieved on 15 May, 2020 from https://whitepaper.ark.io/prologue.

[11] N. Asokan, Victor Shoup, and Michael Waidner. 1998. Optimistic fair exchange of digital signatures. In *International Conference on the Theory and Applications of Cryptographic Techniques*, Vol. 1403. Springer-Verlag, 591–606.

[12] Nicola Atzei, Massimo Bartoletti, and Tiziana Cimoli. 2017. A survey of attacks on ethereum smart contracts. In *International Conference on Principles of Security and Trust*. Association for Computing Machinery, 164–186.

[13] Adam Back, Matt Corallo, Luke Dashjr, Mark Friedenbach, Gregory Maxwell, Andrew Miller, Andrew Poelstra, Jorge Timón, and Pieter Wuille. 2014. *Enabling Blockchain Innovations with Pegged Sidechains*. Technical Report. Blockstream.

[14] HMN Dilum Bandara, Xiwei Xu, and Ingo Weber. 2019. Patterns for blockchain data migration. (June 2019). http://arxiv.org/abs/1906.00239

[15] Tal Baneth. 2019. Waterloo—a Decentralized Practical Bridge between EOS and Ethereum. Retrieved on 15 May, 2020 from https://blog.kyber.network/waterloo-a-decentralized-practical-bridge-between-eos-and-ethereum-1c230ac65524.

[16] Richard Barnes. 2020. Factors in the portability of tokenized assets on distributed ledgers. arXiv pre-prints (May 2020). http://arxiv.org/abs/2005.07461

[17] Rafael Belchior, Miguel Correia, and Thomas Hardjono. 2021. *DLT Gateway Crash Recovery Mechanism (draft-belchior-gateway-recovery-00)*. Technical Report. IETF. https://datatracker.ietf.org/doc/draft-belchior-gateway-recovery/.

[18] Rafael Belchior, Miguel Correia, and André Vasconcelos. 2019. JusticeChain: Using blockchain to protect justice logs. In *27th International Conference on Cooperative Information Systems*. Springer, Cham.

[19] Rafael Belchior, Sérgio Guerreiro, André Vasconcelos, and Miguel Correia. 2020. A survey on business process view integration. (Nov. 2020). http://arxiv.org/abs/2011.14465

[20] Rafael Belchior, Benedikt Putz, Guenther Pernul, Miguel Correia, André Vasconcelos, and Sérgio Guerreiro. 2020. SSIBAC: Self-sovereign identity based access control. In *The 3rd International Workshop on Blockchain Systems and Applications*. IEEE.

[21] Rafael Belchior, André Vasconcelos, and André Correia. 2021. Creating, Merging, and Processing Blockchain Views With BUNGEE. *to appear* (2021).

[22] Rafael Belchior, André Vasconcelos, and Miguel Correia. 2020. Towards secure, decentralized, and automatic audits with blockchain. In *European Conference on Information Systems*. Association for Information Systems.

[23] Rafael Belchior, André Vasconcelos, Miguel Correia, and Thomas Hardjono. 2021. HERMES: Fault-tolerant middleware for blockchain interoperability. *TechrXiv 14120291/1* (March 2021). https://doi.org/10.36227/TECHRXIV.14120291.V1
[24] Marianna Belotti, Stefano Moretti, Maria Potop-Butucaru, and Stefano Secci. 2020. *Game Theoretical Analysis of Atomic Cross-Chain Swaps*. Technical Report. https://hal.archives-ouvertes.fr/hal-02414356
[25] Eli Ben-Sasson Technion Alessandro Chiesa, Eran Tromer, and Madars Virza. 2014. Succinct non-interactive zero knowledge for a von Neumann architecture. In *USENIX Security*. USENIX Association.
[26] Juan Benet. 2014. *IPFS - Content Addressed, Versioned, P2P File System*. Technical Report Draft 3. IPFS.
[27] L Besançon, Catarina Silva, and Parisa Ghodous. 2019. Towards blockchain interoperability: Improving video games data exchange. In *2019 IEEE International Conference on Blockchain and Cryptocurrency*.
[28] Garrett Birkhoff. 1940. *Lattice Theory, Volume 25, Part 2*. American Mathematical Society. 418 pages. https://books.google.com/books?id=0Y8d-MdtVwkC&pgis=1.
[29] Monika Bishnoi and Rajesh Bhatia. 2020. Interoperability solutions for blockchain. In *Proceedings of the International Conference on Smart Technologies in Computing, Electrical and Electronics (ICSTCEE'20)*, 381–385.
[30] Bitcoin Wiki. 2016. Hash Time Locked Contracts. https://en.bitcoin.it/wiki/Hash_Time_Locked_Contracts.
[31] Bitcoin Wiki. 2016. Hashlock. https://en.bitcoin.it/wiki/Hashlock.
[32] Bitcoin Wiki. 2016. Timelock. https://en.bitcoin.it/wiki/Timelock.
[33] Bitcoin Wiki. 2017. Simplified Payment Verification (SPV). https://en.bitcoinwiki.org/wiki/Simplified_Payment_Verification.
[34] Matthew Black, Tingwei Liu, and Tony Cai. 2019. Atomic Loans: Cryptocurrency Debt Instruments. http://arxiv.org/abs/1901.05117
[35] Michael Borkowski, Philipp Frauenthaler, Marten Sigwart, Taneli Hukkinen, Oskar Hladky, and Stefan Schulte. 2019. Cross-Blockchain Technologies: Review, State of the Art, and Outlook. Retrieved on 23 January, 2020 from https://dsg.tuwien.ac.at/projects/tast/pub/tast-white-paper-4.pdf.
[36] Michael Borkowski, Daniel Mcdonald, Christoph Ritzer, and Stefan Schulte. 2018. Towards Atomic Cross-Chain Token Transfers: State of the Art and Open Questions within TAST, 10 pages. http://www.infosys.tuwien.ac.at/tast/.
[37] Michael Borkowski, Christoph Ritzer, Daniel Mcdonald, and Stefan Schulte. 2018. Caught in Chains: Claim-First Transactions for Cross-Blockchain Asset Transfers. Retrieved on 23 January, 2020 from http://www.infosys.tuwien.ac.at/tast/.
[38] Michael Borkowski, Marten Sigwart, Philipp Frauenthaler, Taneli Hukkinen, and Stefan Schulte. 2019. DeXTT: Deterministic cross-blockchain token transfers. *IEEE Access* 7 (8 2019), 111030–111042.
[39] Richard Brown. 2018. The Corda Platform: An Introduction White Paper. Retrieved on 23 January, 2020 from https://www.r3.com/reports/the-corda-platform-an-introduction-whitepaper/.
[40] Gewu Bu, Riane Haouara, Thanh Son Lam Nguyen, and Maria Potop-Butucaru. 2020. Cross hyperledger fabric transactions. In *CRYBLOCK 2020 - Proceedings of the 3rd Workshop on Cryptocurrencies and blockchains for distributed systems, part of MobiCom 2020*. Association for Computing Machinery, 35–40. https://doi.org/10.1145/3410699.3413796
[41] Jeff Burdges, Alfonso Cevallos, Peter Czaban, Rob Habermeier, Syed Hosseini, Fabio Lama, Handan Kıhnç Alper, Ximin Luo, Fatemeh Shirazi, Alistair Stewart, and Gavin Wood. 2020. Overview of polkadot and its design considerations. arXiv 2005.13456. https://arxiv.org/abs/2005.13456.
[42] Daniel Burnett and Matt Stone. 2017. *Verifiable Credentials Working Group*. Technical Report. W3C. Retrieved on 20 February, 2020 from https://www.w3.org/2017/vc/WG/.
[43] Vitalik Buterin. 2014. Ethereum: A next-generation smart contract and decentralized application platform. Retrieved on 20 February, 2020 from https://github.com/ethereum/wiki/wiki/White-Paper.
[44] Vitalik Buterin. 2016. *R3 Report - Chain Interoperability*. Technical Report. R3 Corda. Retrieved on 15 January, 2020 from https://www.r3.com/wp-content/uploads/2017/06/chain_interoperability_r3.pdf.
[45] Vitalik Buterin. 2021. An Incomplete Guide to Rollups. Retrieved on 20 February, 2020 from https://vitalik.ca/general/2021/01/05/rollup.html.
[46] Christian Cachin and Marko Vukolić. 2017. Blockchain consensus protocols in the wild. *arXiv e-prints* 91 (July 2017). http://arxiv.org/abs/1707.01873
[47] Fran Casino, Thomas Dasaklis, and Constantinos Patsakis. 2019. A systematic literature review of blockchain-based applications: Current status, classification and open issues. *Telematics and Informatics* 36 (March 2019), 55–81.
[48] Jing Chen and Silvio Micali. 2016. Algorand. (July 2016), 51–68. http://arxiv.org/abs/1607.01341
[49] Mihai Christodorescu, Catherine Gu, Ranjit Kumaresan, Mohsen Minaei, Mustafa Ozdayi, Benjamin Price, Srinivasan Raghuraman, Muhammad Saad, Cuy Sheeeld, Minghua Xu, and Mahdi Zamani. 2020. Towards a two-tier hierarchical infrastructure: An ooine payment system for central bank digital currencies. https://arxiv.org/abs/2012.08003.
[50] Clearmatics. 2018. Ion Interoperability Framework v2. Retrieved on 14 Febraury, 2020 from https://github.com/clearmatics/ion.

[51] João Colaço and Pedro Sousa. 2017. View integration of business process models. In *Lecture Notes in Business Information Processing*, Vol. 299. Springer-Verlag, 619–632.
[52] K. Sai and D. Tipper. 2019. Disincentivizing double spend attacks across interoperable blockchains. In *First IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA'19)*. 36–45.
[53] Mauro Conti, Kumar E. Sandeep, Chhagan Lal, and Sushmita Ruj. 2018. A survey on security and privacy issues of bitcoin. *IEEE Communications Surveys and Tutorials* 20, 4 (Oct. 2018), 3416–3452.
[54] Miguel Correia. 2019. From byzantine consensus to blockchain consensus. *Essentials of Blockchain Technology* (2019), 41.
[55] Ludovic Courcelas, Tom Lyons, and Ken Timsit. 2020. *European Union Blockchain Observatory and Forum 2018-2020 Conclusions and Reflections*. Technical Report. European Union Blockchain Observatory and Forum. Retrieved on 28 September, 2020 from https://www.eublockchainforum.eu/sites/default/files/reports/report_conclusion_book_v1.0.pdf?width=1024&height=800&iframe=true.
[56] Arylyn Culwick and Dan Metcalf. 2018. *Blocknet Design Specification v1.0*. Technical Report. Blocknet. Retrieved on 16 March, 2020 from https://www.blocknet.co/wp-content/uploads/whitepaper/Blocknet_Whitepaper.pdf.
[57] Bingrong Dai, Shengming Jiang, Menglu Zhu, Ming Lu, Dunwei Li, and Chao Li. 2020. Research and implementation of cross-chain transaction model based on improved hash-locking. In *Communications in Computer and Information Science*, Vol. 1267. Springer Science and Business Media Deutschland GmbH, 218–230. https://doi.org/10.1007/978-981-15-9213-3{_}17
[58] Christian Decker and Roger Wattenhofer. 2015. A fast and scalable payment network with bitcoin duplex micropayment channels. In *Lecture Notes in Computer Science*, Vol. 9212. Springer-Verlag, 3–18.
[59] Apoorvaa Deshpande and Maurice Herlihy. 2020. Privacy-preserving cross-chain atomic swaps. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Vol. 12063. Springer, 540–549. https://doi.org/10.1007/978-3-030-54455-3_38
[60] Advait Deshpande, Katherine Stewart, Louise Lepetit, and Salil Gunashekar. 2017. *Distributed Ledger Technologies/Blockchain: Challenges, Opportunities and the Prospects for Standards*. Technical Report. British Standards Institution.
[61] Digital Asset. 2019. DAML SDK 1.1.1 documentation. Retrieved on 24 May, 2020 from https://docs.daml.com/getting-started/installation.html.
[62] Digital Asset. 2020. Canton: A private, scalable, and composable smart contract platform. 1–15. Retrieved on 24 May, 2020 from https://www.canton.io/publications/canton-whitepaper.pdf.
[63] Johnny Dilley, Andrew Poelstra, Jonathan Wilkins, Marta Piekarska, Ben Gorlick, and Mark Friedenbach. 2016. *Strong Federations: An Interoperable Blockchain Solution to Centralized Third-Party Risks*. Technical Report. BlockStream. http://arxiv.org/abs/1612.05491
[64] Thomas Durieux, Joao F. Ferreira, Rui Abreu, and Pedro Cruz. 2020. Empirical review of automated analysis tools on 47,587 ethereum smart contracts. In *Proceedings of the International Conference on Software Engineering*. IEEE Computer Society, 530–541.
[65] Ethereum Foundation. 2019. ETH 2 Phase 2 WIKI. Retrieved on 23 April, 2020 from https://hackmd.io/UzysWse1Th240HELswKqVA?view.
[66] Ethereum Foundation and Consensys. 2015. BTC-relay: Ethereum contract for Bitcoin SPV. Retrieved on 23 April, 2020 from https://github.com/ethereum/btcrelay.
[67] EU Blockchain. 2020. Scalability, interoperability and sustainability of blockchains. Retrieved on 23 April, 2020 from https://www.eublockchainforum.eu/reports.
[68] Ghareeb Falazi, Uwe Breitenbücher, Florian Daniel, Andrea Lamparelli, Frank Leymann, and Vladimir Yussupov. 2020. Smart contract invocation protocol (SCIP): A protocol for the uniform integration of heterogeneous blockchain smart contracts. In *International Conference on Advanced Information Systems Engineering*, Lecture Notes in Computer Science, Vol. 12127. 134–149.
[69] Ghareeb Falazi, Andrea Lamparelli, Uwe Breitenbuecher, Florian Daniel, and Frank Leymann. 2020. Unified integration of smart contracts through service orientation. *IEEE Software* 37, 5 (2020). https://doi.org/10.1109/MS.2020.2994040
[70] Wanchain Foundation. 2019. Wanchain Roadmap. Retrieved on 15 March, 2020 from https://www.wanchain.org/learn/.
[71] Philipp Frauenthaler, Michael Borkowski, and Stefan Schulte. 2019. A framework for blockchain interoperability and runtime selection. *arXiv preprint* (May 2019). https://arxiv.org/abs/1905.07014
[72] Philipp Frauenthaler, Marten Sigwart, Christof Spanring, and Stefan Schulte. 2020. Testimonium: A cost-efficient blockchain relay. arXiv preprint (2020). https://arxiv.org/pdf/2002.12837.pdf.
[73] Philipp Frauenthaler, Marten Sigwart, Christof Spanring, Michael Sober, and Stefan Schulte. 2020. ETH Relay: A cost-efficient relay for ethereum-based blockchains. In *2020 IEEE International Conference on Blockchain (Blockchain'20)*. IEEE, 204–213. https://doi.org/10.1109/Blockchain50366.2020.00032

[74] Fusion Foundation. 2017. *An Inclusive Cryptofinance Platform Based on Blockchain*. Technical Report. Fusion Foundation.
[75] Enrique Fynn, Pedone, Fernando, and Bessani Alysson. 2020. Smart contracts on the move. In *Dependable Systems and Networks*.
[76] Alberto Garoffolo, Dmytro Kaidalov, and Roman Oliynykov. 2020. *Zendoo: a zk-SNARK Verifiable Cross-Chain Transfer Protocol Enabling Decoupled and Decentralized Sidechains*. Technical Report. V. N. Karazin Kharkiv National University.
[77] Peter Gaži, Aggelos Kiayias, and Dionysis Zindros. 2019. Proof-of-stake sidechains. *IEEE Symposium on Security and Privacy*.
[78] Sara Ghaemi, Sara Rouhani, Rafael Belchior, Rui S. Cruz, Hamzeh Khazaei, and Petr Musilek. 2021. A pub-sub architecture to promote blockchain interoperability. (Jan. 2021). http://arxiv.org/abs/2101.12331
[79] Christian Gorenflo. 2020. *Towards a New Generation of Permissioned Blockchain Systems*. Ph.D. Dissertation. University of Waterloo. https://uwspace.uwaterloo.ca/bitstream/handle/10012/15860/Gorenflo_Christian.pdf?sequence=3.
[80] Christian Gorenflo, Stephen Lee, Lukasz Golab, and Srinivasan Keshav. 2019. FastFabric: Scaling hyperledger fabric to 20,000 transactions per second. *ICBC 2019 - IEEE International Conference on Blockchain and Cryptocurrency*, 455–463.
[81] Gideon Greenspan. 2015. MultiChain White Paper. Retrieved on 15 March, 2020 from https://www.multichain.com/white-paper/.
[82] Lewis Gudgeon, Pedro Moreno-Sanchez, Stefanie Roos, Patrick Mccorry, and Arthur Gervais. 2020. SoK: Layer-two blockchain protocols. In *International Conference on Financial Cryptography and Data Security*.
[83] Joel Gugger. 2020. Bitcoin-monero cross-chain atomic swap. *Cryptology ePrint Archive* (2020). https://eprint.iacr.org/2020/1126
[84] Thomas Hardjono. 2021. Blockchain gateways, bridges and delegated hash-locks. arXiv (Feb. 2021). http://arxiv.org/abs/2102.03933
[85] Thomas Hardjono, Martin Hargreaves, and Ned Smith. 2020. An Interoperability Architecture for Blockchain Gateways. Retrieved on 14 Febraury, 2021 https://datatracker.ietf.org/doc/draft-hardjono-blockchain-interop-arch/.
[86] Thomas Hardjono, Alexander Lipton, and Alex Pentland. 2019. Toward an interoperability architecture for blockchain autonomous systems. *IEEE Transactions on Engineering Management* 67, 4 (2019), 1298–1309.
[87] Thomas Hardjono, Alexander Lipton, and Alex Pentland. 2019. Towards a Public Key Management Framework for Virtual Assets and Virtual Asset Service Providers. http://arxiv.org/abs/1909.08607
[88] Martin Hargreaves and Thomas Hardjono. 2020. Open Digital Asset Protocol (draft-hargreaves-odap-01). https://datatracker.ietf.org/doc/draft-hargreaves-odap/.
[89] Timo Hegnauer Zürich, Eder Scheid, Bruno Rodrigues, Timo Hegnauer, Eder Scheid, and Bruno Rodrigues. 2019. *Design and Development of a Blockchain Interoperability API*. Ph.D. Dissertation. University of Zürich. http://www.csg.uzh.ch/.
[90] Maurice Herlihy. 2018. Atomic cross-chain swaps. In *Proceedings of the Annual ACM Symposium on Principles of Distributed Computing*. Association for Computing Machinery, New York, New York, 245–254.
[91] Maurice Herlihy. 2018. Atomic cross-chain swaps. (2018). https://arxiv.org/abs/1801.09515
[92] David Hyland-Wood and Shahan Khatchadourian. 2018. A future history of international blockchain standards. *The Journal of the British Blockchain Association* 1, 1 (2018), 1–10.
[93] Hyperledger. 2015. Docs | Hyperledger Sawtooth. Retrieved on 5 February, 2020 from https://sawtooth.hyperledger.org/docs/.
[94] Hyperledger. 2019. Hyperledger Quilt Documentation. https://wiki.hyperledger.org/display/quilt/Hyperledger+Quilt.
[95] IBC Ecosystem Working Group. 2020. Inter-Nlockchain Communication Protocol (IBC). https://github.com/cosmos/ics/tree/master/ibc.
[96] iiconsortium. 2020. *Distributed Ledgers in IIoT*. Technical Report. Industrial Internet Consortium. https://www.iiconsortium.org/pdf/Distributed_Ledgers_in_IIoT_White_Paper_2020-07-22.pdf.
[97] Interledger. 2020. Interledger Protocol V4 (ILPv4) | Interledger. https://interledger.org/rfcs/0027-interledger-protocol-4/.
[98] Arjun Jain and Patrick Schilz. 2017. *Block Collider Whitepaper*. Technical Report. Retrieved on 5 February, 2020 from https://www.blockcollider.org/whitepaper.
[99] H. Jin and X. Dai. 2018. Towards a novel architecture for enabling interoperability amongst multiple blockchains. In *IEEE 38th International Conference on Distributed Computing Systems*.
[100] Sandra Johnson, Peter Robinson, and John Brainard. 2019. Sidechains and interoperability. arXiv e-prints (March 2019). http://arxiv.org/abs/1903.04077
[101] J. P. Morgan. 2017. Quorum White Paper. Retrieved on 5 February, 2020 from https://github.com/jpmorganchase/quorum/blob/master/docs/QuorumWhitepaperv0.2.pdf.

[102] Luo Kan, Yu Wei, Amjad Hafiz Muhammad, Wang Siyuan, Gao Linchao, and Hu Kai. 2018. A multiple blockchains architecture on inter-blockchain communication. In *Proceedings of the 2018 IEEE 18th International Conference on Software Quality, Reliability, and Security Companion (QRS-C'18)*, 139–145.
[103] Niclas Kannengießer, Michelle Pfister, Malte Greulich, Sebastian Lins, and Ali Sunyaev. 2020. Bridges between islands: Cross-chain technology for distributed ledger technology. *Hawaii International Conference on System Sciences*.
[104] Rami Khalil, Pedro Moreno-Sanchez, Alexei Zamyatin, Arthur Gervais, and Guillaume Felley. [n.d.]. *Commit-Chains: Secure, Scalable Off-Chain Payments*. Technical Report. https://eprint.iacr.org/2018/642.pdf.
[105] Aggelos Kiayias and Dionysis Zindros. 2018. *Proof-of-Work Sidechains*. Technical Report. IOHK. https://eprint.iacr.org/2018/1048.pdf.
[106] T. Koens and E. Poll. 2019. Assessing interoperability solutions for distributed ledgers. *Pervasive and Mobile Computing* 59 (2019), 101079.
[107] Eleftherios Kokoris-Kogias, Philipp Jovanovic, Linus Gasser, Nicolas Gailly, Ewa Syta, and Bryan Ford. 2018. OmniLedger: A secure, scale-out, decentralized ledger via sharding. In *Proceedings of the IEEE Symposium on Security and Privacy*, Vol. 2018-May. IEEE, Inc., 583–598.
[108] Komodo. 2018. *Komodo Whitepaper*. Technical Report. Komodo. Retrieved on 21 March, 2020 from https://komodoplatform.com/wp-content/uploads/2018/06/Komodo-Whitepaper-June-3.pdf.
[109] Jae Kwon and Ethan Buchman. 2016. *Cosmos Whitepaper*. Technical Report. Cosmos Foundation.
[110] Kyber Network. 2018. Peace Relay. Retrieved on 21 March, 2020 from https://github.com/KyberNetwork/peace-relay.
[111] KyberNetwork. 2018. Waterloo Bridge. Retrieved on 21 March, 2020 from https://github.com/KyberNetwork/bridge_eth_smart_contracts.
[112] Pascal Lafourcade and Marius Lombard-Platet. 2020. About blockchain interoperability. *Information Processing Letters* 161 (2020), 105976.
[113] Rongjian Lan, Ganesha Upadhyaya, Stephen Tse, and Mahdi Zamani. 2021. Horizon: A gas-efficient, trustless bridge for cross-chain transactions. (Jan. 2021). http://arxiv.org/abs/2101.06000
[114] Sergio Lerner. 2015. *RSK Whitepaper*. Technical Report. RSK. Retrieved on 21 March, 2020 from https://docs.rsk.co/RSK_White_Paper-Overview.pdf.
[115] Wenting Li, Alessandro Sforzin, Sergey Fedorov, and Ghassan O. Karame. 2017. Towards scalable and private industrial blockchains. In *ACM Workshop on Blockchain, Cryptocurrencies and Contracts*. Association for Computing Machinery, Inc., 9–14.
[116] Xiaoqi Li, Peng Jiang, Ting Chen, Xiapu Luo, and Qiaoyan Wen. 2020. A survey on the security of blockchain systems. *Future Generation Computer Systems* 107 (June 2020), 841–853.
[117] Libra Association. 2019. The libra blockchain. (2019). https://libra.org/en-US/wp-content/uploads/sites/23/2019/06/LibraWhitePaper_en_US.pdf.
[118] Claudio Lima. 2018. Developing open and interoperable DLT/Blockchain Standards [Standards]. *IEEE Computer* 51, 11 (2018). https://doi.org/10.1109/MC.2018.2876184
[119] Zhuotao Liu, Yangxi Xiang, Jian Shi, Peng Gao, Haoyu Wang, Xusheng Xiao, Bihan Wen, and Yih-Chun Hu. 2019. HyperService. Association for Computing Machinery (ACM), 549–566. https://doi.org/10.1145/3319535.3355503
[120] Zhuotao Liu, Yangxi Xiang, Jian Shi, Peng Gao, Haoyu Wang, Xusheng Xiao, Bihan Wen, and Yih-Chun Hu. 2019. Hyperservice: Interoperability and programmability across heterogeneous blockchains. In *ACM SIGSAC Conference on Computer and Communications*.
[121] Loom. 2016. Intro to Loom Network | Loom SDK. Retrieved on 21 March, 2020 from https://loomx.io/developers/en/intro-to-loom.html.
[122] Jack Lu, Boris Yang, Zane Liang, Ying Zhang, Shi Demmon, Eric Swartz, and Lizzie Lu. 2017. Wanchain: Building Super Financial Markets for the New Digital Economy. 34 pages. https://wanchain.org/files/Wanchain-Whitepaper-EN-version.pdf.
[123] C. Manjunath, Daniel Anderson, Thomas Barnes, Srinath Duraisamy, Manoj Gopalakrishnan, Karthika Murthy, Ramakrishna Srinivasamurthy, and Yevgeniy Yarmosh. 2019. Hyperledger Avalon. Retrieved on 21 March, 2020 from https://github.com/hyperledger/avalon/blob/master/docs/avalon-arch.pdf.
[124] Likoebe M. Maruping, Viswanath Venkatesh, and Ritu Agarwal. 2009. A control theory perspective on agile methodology use and changing user requirements. *Information Systems Research* 20, 3 (Sept. 2009), 377–399.
[125] Tom Mayer, Christoph Mai, and Jesse N. 2017. *Tokrex Whitepaper*. Technical Report. Tokrex. www.tokrex.org.
[126] Greg Medcraft. 2021. *Regulatory Approaches to the Tokenisation of Assets*. Technical Report. OECD. Retrieved on 15 May, 2020 from https://www.oecd.org/finance/regulatory-approaches-to-the-tokenisation-of-assets.htm.
[127] Metronome. 2019. Metronome documentation, FAQ, (July 2019). Retrieved on 24 March, 2020 from https://github.com/autonomoussoftware/documentation/blob/master/FAQ.md.
[128] Metronome. 2019. Metronome documentation v0.99. Retrieved on 24 March, 2020 from https://github.com/autonomoussoftware/documentation/blob/master/owners_manual/owners_manual.md.

[129] Hart Montgomery, Hugo Borne-Pons, Jonathan Hamilton, Mic Bowman, Peter Somogyvari, Shingo Fujimoto, Takuma Takeuchi, Tracy Kuhrt, and Rafael Belchior. 2020. Hyperledger Cactus Whitepaper. Retrieved on 24 March, 2020 from https://github.com/hyperledger/cactus/blob/master/docs/whitepaper/whitepaper.md.
[130] Roman Mühlberger, Stefan Bachhofner, Eduardo Castelló Ferrer, Claudio Di Ciccio, Ingo Weber, Maximilian Wöhrer, and Uwe Zdun. 2020. Foundational oracle patterns: Connecting blockchain to the off-chain world. In *Lecture Notes in Business Information Processing*, Lecture Notes in Business Information Processing, Vol. 393. Springer Science and Business Media Deutschland GmbH, 35–51. https://doi.org/10.1007/978-3-030-58779-6_3
[131] Glenford Myers, Tom Badgett, and Corey Sandler. 2012. Test-case design. In *The Art of Software Testing*. John Wiley & Sons, Ltd., Chapter 4, 41–84. https://doi.org/10.1002/9781119202486.ch4
[132] Satoshi Nakamoto. 2008. Bitcoin: A peer-to-peer electronic cash system. Retrieved on 24 March, 2020 from http://bitcoin.org/bitcoin.pdf.
[133] National Interoperability Framework Observatory. 2020. European Interoperability Framework. Retrieved on 20 April, 2020 from https://joinup.ec.europa.eu/collection/nifo-national-interoperability-framework-observatory/3-interoperability-layers#3.6.
[134] Markus Nissl, Emanuel Sallinger, Stefan Schulte, and Michael Borkowski. 2020. Towards cross-blockchain smart contracts. (Oct. 2020). http://arxiv.org/abs/2010.07352
[135] Henry C. Nunes, Roben C. Lunardi, Avelin F. Zorzo, Regio A. Michelin, and Salil S. Kanhere. 2020. Context-based smart contracts for appendable-block blockchains. In *IEEE International Conference on Blockchain and Cryptocurrency*.
[136] Yan Pang. 2020. A new consensus protocol for blockchain interoperability architecture. *IEEE Access* 8 (2020), 153719–153730. https://doi.org/10.1109/ACCESS.2020.3017549
[137] Joe Petrowski. 2020. Polkadot and Ethereum 2.0. Retrieved on 1 April, 2020 from https://wiki.polkadot.network/docs/en/learn-comparisons-ethereum-2.
[138] Babu Pillai and Kamanashis Biswas. 2019. Blockchain interoperable digital objects. In *ICBC2019 International Conference on Blockchain*. https://doi.org/10.1007/978-3-030-23404-1_6
[139] Babu Pillai, Kamanashis Biswas, and Vallipuram Muthukkumarasamy. 2020. Cross-chain interoperability among blockchain-based systems using transactions. *Knowledge Engineering Review* 35 (2020), 1–18. https://doi.org/10.1017/S0269888920000314
[140] Andrew Poelstra, Adam Back, Mark Friedenbach, Gregory Maxwell, and Pieter Wuille Blockstream. 2017. *Blockstream: Confidential Assets*. Technical Report. Retrieved on 1 April, 2020 from https://blockstream.com/bitcoin17-final41.pdf.
[141] Polkadot. 2019. Cross-chain Message Passing (XCMP) · Polkadot Wiki. https://wiki.polkadot.network/docs/en/learn-crosschain.
[142] Polkadot. 2019. Kusama Network. https://wiki.polkadot.network/docs/en/kusama-index.
[143] Joseph Poon and Vitalik Buterin. 2017. *Plasma: Scalable Autonomous Smart Contracts*. Technical Report. Plasma. https://plasma.io/
[144] Joseph Poon and Thaddeus Dryja. 2016. *The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments*. Technical Report. Lightning Network. Retrieved on 1 April, 2020 from https://lightning.network/lightning-network-paper.pdf.
[145] Ilham A. Qasse, Manar Abu Talib, and Qassim Nasir. 2019. Inter blockchain communication: A survey. In *Arab WIC 6th Annual International Conference Research Track*. Association for Computing Machinery.
[146] Minfeng Qi, Ziyuan Wang, Donghai Liu, Yang Xiang, Butian Huang, and Feng Zhou. 2020. ACCTP: Cross chain transaction platform for high-value assets. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Vol. 12404. Springer Science and Business Media Deutschland GmbH, 154–168. https://doi.org/10.1007/978-3-030-59638-5_11
[147] Quant Foundation. 2019. *Overledger Network Whitepaper v0.3*. Technical Report. Quant.
[148] Mayank Raikwar, Danilo Gligoroski, and Katina Kralevska. 2019. SoK of used cryptography in blockchain. *IEEE Access* 7 (2019), 148550–148575.
[149] Drummond Reed, Manu Sporny, Markus Sabadello, Dave Longley, Christopher Allen, and Ryan Grant. 2018. *Decentralized Identifiers*. Technical Report. Retrieved on 20 February, 2020 from https://w3c.github.io/did-core/.
[150] Peter Robinson, David Hyland-Wood, Roberto Saltini, Sandra Johnson, and John Brainard. 2019. *Atomic Crosschain Transactions for Ethereum Private Sidechains*. Technical Report.
[151] Peter Robinson and Raghavendra Ramesh. 2020. General purpose atomic crosschain transactions. arXiv (Nov. 2020). http://arxiv.org/abs/2011.12783
[152] Sara Rouhani, Rafael Belchior, Rui S. Cruz, and Ralph Deters. 2021. Distributed attribute-based access control system using a permissioned blockchain. World Wide Web.

[153] Janick Rueegger and Guilherme Sperb MacHado. 2020. Rational exchange: Incentives in atomic cross chain swaps. In *IEEE International Conference on Blockchain and Cryptocurrency (ICBC'20)*. IEEE. https://doi.org/10.1109/ICBC48266.2020.9169408
[154] Kuheli Sai and David Tipper. 2019. Disincentivizing double spend attacks across interoperable blockchains. In *1st IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications*.
[155] Aetienne Sardon and Thomas Hardjono. 2020. Blockchain Gateways: Use-Cases (draft-sardon-blockchain-gateways-usecases-00). https://datatracker.ietf.org/doc/draft-sardon-blockchain-gateways-usecases/.
[156] Eder Scheid, Bruno Rodrigues, and Burkhard Stiller. 2019. Toward a policy-based blockchain agnostic framework. In *16th IFIP/IEEE International Symposium on Integrated Network Management*.
[157] Eder J. Scheid, Timo Hegnauer, Bruno Rodrigues, and Burkhard Stiller. 2019. Bifröst: A modular blockchain interoperability API. In *IEEE 44th Conference on Local Computer Networks*. Institute of Electrical and Electronics Engineers (IEEE), 332–339.
[158] Stefan Schulte, Marten Sigwart, Philipp Frauenthaler, and Michael Borkowski. 2019. Towards blockchain interoperability. In *International Conference on Business Process Management: BPM 2019: Business Process Management: Blockchain and Central and Eastern Europe Forum*, Vol. 361. Springer-Verlag, 3–10. https://doi.org/10.1007/978-3-030-30429-4_1
[159] Security Token Standard. 2019. Security Token Standard - ERC 1400. Retrieved on 24 March, 2020 from https://thesecuritytokenstandard.org/.
[160] Security Token Standard. 2019. SecurityTokenStandard/EIP-Spec. Retrieved on 24 March, 2020 from https://github.com/SecurityTokenStandard/EIP-Spec.
[161] Narges Shadab, Farzin Hooshmand, and Mohsen Lesani. 2020. Cross-chain transactions. In *IEEE International Conference on Blockchain and Cryptocurrency*.
[162] Omer Shlomovits and Oded Leiba. 2020. JugglingSwap: Scriptless atomic cross-chain swaps. arXiv 2007.14423. http://arxiv.org/abs/2007.14423
[163] Amritraj Singh, Kelly Click, Reza M. Parizi, Qi Zhang, Ali Dehghantanha, and Kim Kwang Raymond Choo. 2020. Sidechain technologies in blockchain networks: An examination and state-of-the-art review. *Journal of Network and Computer Applications* 149 (2020).
[164] Vasilios A. Siris, Pekka Nikander, Spyros Voulgaris, Nikos Fotiou, Dmitrij Lagutin, and George C. Polyzos. 2019. Interledger approaches. *IEEE Access* 7 (2019), 89948–89966.
[165] Matthew Spoke. 2017. *Aion: Enabling the Decentralized Internet*. Technical Report. Retrieved on 23 January, 2020 from https://whitepaper.io/document/31/aion-whitepaper.
[166] He Sun, Hongliang Mao, Xiaomin Bai, Zhidong Chen, Kai Hu, and Wei Yu. 2018. Multi-blockchain model for central bank digital currency. In *Parallel and Distributed Computing, Applications and Technologies (PDCAT)*, Vol. 2017-Dec. IEEE Computer Society, 360–367. https://doi.org/10.1109/PDCAT.2017.00066
[167] Swiss Financial Market Supervisory Authority. 2018. FINMA publishes ICO guidelines. Retrieved on 18 May, 2020 from https://www.finma.ch/en/news/2018/02/20180216-mm-ico-wegleitung/.
[168] Nick Szabo. 1997. Formalizing and securing relationships on public networks. *First Monday* 2, 9 (1997).
[169] Michael Szydlo. 2004. Merkle tree traversal in log space and time. In *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 541–554.
[170] H. Tam Vo, Z. Wang, D. Karunamoorthy, J. Wagner, E. Abebe, and M. Mohania. 2018. Internet of blockchains: Techniques and challenges ahead. In *2018 IEEE International Conference on Internet of Things (iThings), IEEE Green Computing and Communications (GreenCom), IEEE Cyber, Physical and Social Computing (CPSCom), and IEEE Smart Data (SmartData)*. 1574–1581.
[171] Paolo Tasca and Thayabaran Thanabalasingham. 2017. Taxonomy of blockchain technologies. Principles of identification and classification. *SSRN Electronic Journal* (June 2017).
[172] Tendermint. 2016. Tendermint BFT. Retrieved on 15 April, 2020 from https://github.com/tendermint/tendermint.
[173] Stefan Thomas and Evan Schwartz. 2015. A Protocol for Interledger Payments. 25 pages. Retrieved on 15 March, 2020 from https://interledger.org/interledger.pdf.
[174] Hangyu Tian, Kaiping Xue, Shaohua Li, Jie Xu, Jianqing Liu, and Jun Zhao. 2020. Enabling cross-chain transactions: A decentralized cryptocurrency exchange protocol. arXiv (May 2020). http://arxiv.org/abs/2005.03199
[175] Token Taxonomy Consortium. 2019. *Token Specification Summary*. Technical Report. Token Taxonomy Initiative. Retrieved on 3 March, 2020 from https://tokentaxonomy.org/wp-content/uploads/2019/11/TTF-Overview.pdf.
[176] Fakhar Ul Hassan, Anwaar Ali, Siddique Latif, Junaid Qadir, Salil Kanhere, Jatinder Singh, and Jon Crowcroft. 2019. Blockchain and the future of the internet: A comprehensive review. arXiv e-prints (2019). https://arxiv.org/abs/1904.00733
[177] U.S. Securities and Exchange Commission. 2019. *Framework for "Investment Contract" Analysis of Digital Assets 1*. Technical Report. Retrieved on 18 May, 2020 from https://www.sec.gov/files/dlt-framework.pdf.

[178] Gilbert Verdian, Paolo Tasca, Colin Paterson, and Gaetano Mondelli. 2018. *Quant Overledger Whitepaper v0.1*. Technical Report. Quant. 1–48 pages. Retrieved on 12 February, 2020 from http://objects-us-west-1.dream.io/files.quant.network/Quant_Overledger_Whitepaper_v0.1.pdf.
[179] F. B. Vernadat. 2006. Interoperable enterprise systems: Architectures and methods. In *IFAC Proceedings Volumes (IFAC-PapersOnline)*, Vol. 12. Elsevier, 13–20.
[180] Fabian Vogelsteller and Vitalik Buterin. 2015. EIP 20: ERC-20 Token Standard. https://eips.ethereum.org/EIPS/eip-20.
[181] Gang Wang, Zhijie Jerry, and Mark Nixon. 2019. SoK: Sharding on blockchain. In *ACM Conference on Advances in Financial Technologies*.
[182] Hongkai Wang, Dong He, Xiaoyi Wang, Caichao Xu, Weiwei Qiu, Yiyang Yao, and Qiang Wang. 2020. An electricity cross-chain platform based on sidechain relay. In *Journal of Physics: Conference Series*, Vol. 1631. IOP Publishing Ltd, 12189. https://doi.org/10.1088/1742-6596/1631/1/012189
[183] Xinying Wang, Timothy Tawose, Feng Yan, and Dongfang Zhao. 2020. *Distributed Nonblocking Commit Protocols for Many-Party Cross-Blockchain Transactions*. Technical Report. https://arxiv.org/pdf/2001.01174.pdf.
[184] Will Warren and Amir Bandeali. 2017. *0x: An Open Protocol for Decentralized Exchange on the Ethereum Blockchain*. Technical Report. Retrieved on 23 May, 2020 from https://0x.org/pdfs/0x_white_paper.pdf.
[185] WEF. 2020. *Bridging the Governance Gap: Interoperability for Blockchain and Legacy Systems*. Technical Report.
[186] Peter Wegner. 1996. Interoperability. *ACM Computing Surveys* 28, 1 (1996).
[187] Martin Westerkamp. 2019. Verifiable smart contract portability. *ICBC 2019 - IEEE International Conference on Blockchain and Cryptocurrency* (Feb. 2019), 413–421. http://arxiv.org/abs/1902.03868
[188] Gavin Wood. 2016. *Polkadot: Vision for a Heterogeneous Multi-Chain Framework*. Technical Report. 1–21. Retrieved on 29 February, 2020 from https://github.com/w3f/polkadot-white-paper/raw/master/PolkaDotPaper.pdf.
[189] Gavin Wood. 2019. *Ethereum: A Secure Decentralised Generalised Transaction Ledger. Byzantium version 7e819ec*. Technical Report. Retrieved on 29 February, 2020 from https://ethereum.github.io/yellowpaper/paper.pdf.
[190] Xingtang Xiao, Zhuo Yu, Ke Xie, Shaoyong Guo, Ao Xiong, and Yong Yan. 2020. A multi-blockchain architecture supporting cross-blockchain communication. In *Communications in Computer and Information Science*, Vol. 1253 CCIS. Springer Science and Business Media Deutschland GmbH, 592–603. https://doi.org/10.1007/978-981-15-8086-4_56
[191] Xiwei Xu, Ingo Weber, Mark Staples, Liming Zhu, Jan Bosch, Len Bass, Cesare Pautasso, and Paul Rimba. 2017. A Taxonomy of blockchain-based systems for architecture design. In *Proceedings - 2017 IEEE International Conference on Software Architecture (ICSA'17)*. IEEE, 243–252.
[192] Dylan Yaga, Peter Mell, Nik Roby, and Karen Scarfone. 2018. *Blockchain Technology Overview*. Technical Report. NISTIR. https://doi.org/02
[193] Guangsheng Yu, Xu Wang, Kan Yu, Wei Ni, J. Andrew Zhang, and Ren Ping Liu. 2020. Survey: Sharding in blockchains. *IEEE Access* 8 (2020), 14155–14181. https://doi.org/10.1109/ACCESS.2020.2965147
[194] Alexei Zamyatin, Mustafa Al-Bassam, Dionysis Zindros, Eleftherios Kokoris-Kogias, Pedro Moreno-Sanchez, Aggelos Kiayias, and William J. Knottenbelt. 2019. *SoK: Communication Across Distributed Ledgers*. Technical Report. https://eprint.iacr.org/2019/1128.pdf.
[195] Alexei Zamyatin, Dominik Harz, Joshua Lind, Panayiotis Panayiotou, Arthur Gervais, and William J. Knottenbelt. 2019. XCLAIM: A framework for blockchain interoperability. https://arxiv.org/pdf/2002.11771.pdf.
[196] A. Zamyatin, N. Stifter, A. Judmayer, P. Schindler, E. Weippl, and W. J. Knottenbelt. 2019. A wild velvet fork appears! Inclusive blockchain protocol changes in practice. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Vol. 10958. Springer-Verlag, 31–42.
[197] P. Zappalà, M. Belotti, M. Potop-Butucaru, and S. Secci. 2020. *Game Theoretical Framework for Analyzing Blockchains Robustness*. Technical Report. https://eprint.iacr.org/2020/626.pdf.
[198] Dongfang Zhao and Tonglin Li. 2020. Distributed cross-blockchain transactions. arXiv (2020).
[199] Zibin Zheng, Shaoan Xie, Hongning Dai, Xiangping Chen, and Huaimin Wang. 2017. An overview of blockchain technology: Architecture, consensus, and future trends. In *Proceedings - 2017 IEEE 6th International Congress on Big Data, BigData Congress*. IEEE, 557–564.
[200] Qingyi Zhu, Seng W. Loke, Rolando Trujillo-Rasua, Frank Jiang, and Yong Xiang. 2019. Applications of distributed ledger technologies to the internet of things: A survey. *ACM Computing Surveys* 52, 6 (2019), 120:1–120:34.
[201] Aviv Zohar. 2015. Bitcoin: Under the hood. *Communications of the ACM* 58, 9 (Aug. 2015), 104–113.
[202] Guy Zyskind Oz, Nathan Alex 'Sandy' Pentland. 2015. *Enigma: Decentralized Computation Platform with Guaranteed Privacy*. Technical Report.

Received June 2020; revised March 2020; accepted May 2021

D Applications of Blockchain Interoperability

Many applications of blockchain interoperability are still being developed. Network effects and the increasing mass adoption will unlock unexpected paths. However, there are already some use cases for blockchain interoperability. We present the main ones in this section.

D.1 Mutiple Ledger Central Bank Digital Currencies

The first important Internet of Blockchains use case is asset transfers, where users can transfer assets from one blockchain to another. While some approaches implement this use case in an ad-hoc way, the emergence of central bank digital currencies (CBDCs) [249, 45], requires further efforts and standardization [250]. *"A CBDC is a digital payment instrument, denominated in the national unit of account, that is a direct liability of the central bank"* [251], complementing cash and traditional reserve or settlement accounts. Although there are only a few CBDCs already in production, with limited rollout or in minor economies, in recent years, most central banks have performed extensive research and experiments on the topic in preparation for issuing a CBDC in the future. Whether the authoritative ledger of future CBDCs will be decentralized or centralized is still a subject of debate; experiments and prototypes exist for both. By-design tamper resistance, auditability, and fault tolerance are strong supporting arguments for permissioned distributed ledger-based implementations, even despite performance assurance, privacy, and operating consortium diversity challenges.

Blockchain-based applications that implement services for a given business domain (like logistics, retail, insurance, etc.) need a legally recognized vehicle for payment and settlement – and in many cases, the same-chain CBDC will be the best option, when it becomes available [252].

Openly accessible documentation on CBDC experiments and prototypes of central banks, such as work streams in the Digital Euro experiments [253, 254], strongly suggests that the core CBDC ledger will not provide wide-scale support for smart contracts. Instead, the application of interoperability solutions – classic payment initiation triggers, bridging, and payment channels – can be expected. For CBDC-using decentralized application ecosystems,

arguably, bridging is the optimal solution, as it seamlessly enables performing the financial operations encoded in smart contracts. In a wider context, it is also worth noting that the Multi-CBDC project (mCBDC [255]) of the Bank of International Settlements (BIS) demonstrated the use of the bridging mechanism in the context of wholesale (available only to financial institutions) CBDC and cross-border transactions. Some central banks are already experimenting with blockchain, including the Monetary Authority of Singapore and the Bank of Canada [256]. We explore this use case in our work [45].

D.2 Supply Chain

Another major use case is interoperability across supply chains [256, 102]. We explore this use case in BUNGEE [4]. A supply chain is a chain of value transfer between parties, from the raw product (physical or intellectual) to its finalized version. Managing a supply chain is complex because it includes many non-trusting stakeholders (e.g., enterprises, regulators). As many markets are open and fluid, enterprises do not take the time to build trust - and instead rely on a paper trail that logs the state of an object in the supply chain. This paper trail is needed for auditability and typically can be tampered with, leading to blockchain's suitability to address these problems [71]. A key challenge of blockchain-based supply chains is to interoperate with other distributed ledger technology (DLT) systems. Interoperability granted each supply chain participant (e.g., supplier, manufacturer, retailer) can participate at several supply chains (and thus several blockchains) using a single endpoint, simplifying the interaction process while reducing costs. Other use cases comprise connecting Hyperledger Fabric and Ethereum with Singapore Exchange and Monetary Authority of Singapore via node integration and Evrthng, a product connecting multiple chains via API to digitize products [71].

At the IETF Secure Asset Transfer Protocol working group, we are collecting use cases that can leverage our protocol, closely related to supply chain [257]. Those are included in two main groups: International Trade and Supply Chains, and Financial Instruments and Currency Exchanges. For the first group, the use cases are the improvement of trade finance and logistics, tracking food shipments, and supply chain management. For the second group, the use cases are CBDC currency transfers, delivery versus payment of securities, and transferal of digital art and across national borders.

D.3 Cross-Jurisdiction Promissory Notes

Promissory notes are freely transferable financial instruments where issuers denote a promise to pay another party (payee) [258]. Notes are globally standardized by several legal frameworks, providing a low-risk instrument to reclaim liquidity from debt. Notes contain information regarding the debt, such as the amount, interest rate, maturity date, and issuance place. Notes are useful because they allow parties to liquidate debts and conduct financial transactions faster, overcoming market inefficiencies. In practice, promissory notes can be both payment and credit instruments. A promissory note typically contains all the terms about the indebtedness, such as the principal amount, credit rating, interest rate, expiry date, date of issuance, and issuer's signature. Despite their benefits, paper promissory notes are hard to track, require hand signatures and not-forgery proofs, accounting for cumbersome management.

To address these challenges, recent advances in promissory notes' digitalization include FQX's eNote [259]. Blockchain-supported digital promissory notes (eNotes) worth about half a million dollars were used by a "Swiss commodity trader to finance a transatlantic metal shipment" [260]. eNotes are stored in a trusted ledger covered

by the legal framework, belonging to a specific jurisdiction. Consider the following supply chain scenario: a producer (P) produces a certain amount of goods that sells to a wholesaler (W). W accepted the goods, and now P issues an invoice of value V. The wholesaler could pay in, for example, 90 days. Because P does not want to wait up to 90 days for its payment, it requests a promissory note from W, stating that V will be paid in 90 days. This way, P can sell that same promissory note to a third party. The promissory note is abstract from any physical good being exchanged. Depending on the issuer, collateral might not be needed, as the accountability for liquidating the debt is tracked by the blockchain where it is stored.

Blockchain-based promissory notes belonging to a particular jurisdiction are stored in a certified blockchain that exposes a gateway. When a promissory note needs to change jurisdictions (e.g., a promissory note issued in the USA that needs to be redeemed in Europe), the gateways belonging to the source and target blockchains perform an asset transfer the asset is a digital promissory note. Alternatively, the gateway extends to several jurisdictions. Below is an example of an asset profile of a digital promissory note. Such digital promissory notes can be trivially exchanged between blockchains using Hermes and the SATP (previously known as ODAP) protocol, where gateways belonging to different jurisdictions (e.g., representing different blockchains regulated by different entities) perform asset transfers.

D.4 Carbon Emission Tracking

The first example is Hyperledger's Cactus implementation of the Carbon Emission App from the Hyperledger Carbon Accounting and Neutrality Working Group [261]. A detailed explanation of this use case can be found in Hyperledger [262]. The purpose of this use case is to reward carbon emission reduction by orchestrating heterogeneous blockchains: one focused on data collecting, and another on the reward incentives.

A Hyperledger Fabric network collects emission records (activity data), e.g., energy consumption, travel mileage, and widgets produced. The emissions records are not continuous because both the emissions factors and the data for calculating emissions are based on long time windows (e.g., utility bills are produced each month). Periodically, the activity is aggregated to be later converted to an emission token (ERC-721). Emission tokens are created on Ethereum's public network from the collected data on Fabric to be traded against allowances that reward emission reduction.

This use case contemplates a private, permissioned ledger used for performance and privacy reasons, but where the final output (carbon emission tokens) are stored in public blockchains as a reward. In particular, the performance of Hyperledger Fabric in terms of throughput and end-to-end latency is superior to most public blockchains due to its consensus and low number of peers. Privacy can be assured because only the peers involved can read the global state or if needed, only a subset of peers could read part of the global state (i.e., by utilizing channels or private data).

D.5 Providing Off-chain information to Blockchains

Smart contracts consume information stored outside their blockchain via oracles. Oracle interoperability solutions allow a DLT system to make use of external data from another system [160, 159, 158, 263], increasing the connectivity of DLT-based applications. There is a lot of on-going research on the security and fairness of interoperability via oracles. In particular, oracles could be selecting certain transactions to be included in the target blockchain for its own benefit, similarly to the miner extractable value problem [264].

Code deployed into a distributed ledger cannot access external resources or data without the help of an intermediary. These intermediaries (known as oracles) gather external data, placing it into transactions that are subsequently added to the distributed ledger, therefore allowing this retrieved data to be read by deployed smart contracts. Note that oracles can fetch data from a non-DLT system or another DLT-system. Oracles are classified into two types [160], pull-based, or push-based.

Examples of real-world oracle networks are Chainlink [265], Band Protocol [266], and API3 [267]. We refer the interested reader to [100, 158] for further information.

D.6 Cross-Chain Identity

Identity and data portability can be provided by a multi ledger approach. Identity paradigms like self-sovereign identity [37] can increase identity portability by providing users control of their identities. Typically, this is achieved by rooting user credentials in a blockchain. Hence, if blockchains can communicate with blockchains' identity providers, one can use the same identity in different blockchains. Data portability complies with blockchains, allowing blockchain users to use their data outside of a blockchain without requiring significant effort.

zkEEs: Zero-Knowledge Easter Eggs

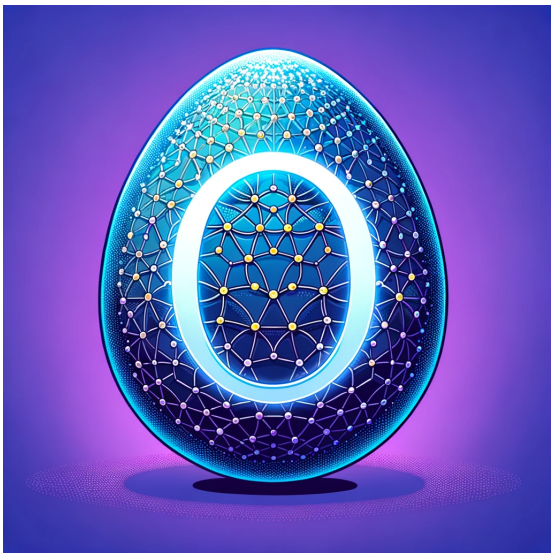


Figure E.1: Zero-knowledge Easter Eggs - zkEEs

All work and no play make Jack dull, so we introduce something different for this thesis. Meet **zkEEs**, *Zero-Knowledge Easter Eggs*! Across this thesis, you can collect ten easter eggs for your amusement. The bounty on each easter egg found is a very special on-chain collectible[1]. Those easter eggs contain what I believe to be cool references to Blockchain, this thesis, and my own interests.

To redeem the NFTs, you need to interact with a smart contract. While I do not want to bother you with technical details[2], a quick introduction might prove useful. Our system relies on a hashlock, that protects the

[1] in the form of an NFT. An NFT is an entry in a smart contract containing a table (ownership, address) with your blockchain address and NFT address. It binds the ownership of this token to your wallet. For all intents and purposes, it is a collectible you can exchange.

[2] OK, maybe just a few technical details: zkEEs leverage zero-knowledge proofs (see a gentle introduction here [151]) via the Groth16 proof system. We leverage Circom, a domain-specific language to compile programs into arithmetic circuits, and an on-chain component based on a zero-knowledge proof verifier and an ERC-1155 minter. Zero-knowledge SNARKs make sense in this case because 1) the prover and verifier cannot run as one trusted entity; 2) part of the input should be confidential, namely the secret component of the easter egg; 3) we need a constant-size proof; and 4) we do not need a sublinear prover (would be better, but it seems there none is yet available) [198].

smart contract functionality that mints NFTs (one of the traditional mechanisms for interoperability [100]). You can redeem your NFT when you provide a secret that, when hashed, corresponds to the hashlock guarding it. See the following example:

"Interesting" (wink wink) Example

You find your last easter egg on this page. It says bounty:arrakis. *Then, it means that the secret to being input (the last of this thesis) is "arrakis". Use the script in the provided code to input the learned secret and redeem your NFT. Instructions: https://github.com/RafaelAPB/phd-thesis-on-chain*

Of course, creating transactions and transacting against the required contract can be a tedious process. I have automated everything for you. The code for this smart contract, supporting code for issuing transactions, and reading from the blockchain is available in Github[3]. Please follow the instructions on the README.md.

Good Easter Egg hunting! 

That's all folks!

[3]https://github.com/RafaelAPB/phd-thesis-on-chain