

CC2C: Confidential Channel-to-Channel Data Exchange in a Permissioned Blockchain

Emanuel Nunes^{*‡}, Samih Eisa[†], Miguel L. Pardal[†], and Mário Calha[‡]

^{*} Sensefinito, Lisbon, Portugal

emanuel.nunes@sensefinito.com

[†] INESC-ID, Instituto Superior Técnico, Universidade de Lisboa, Portugal

samih.eisa@inesc-id.pt miguel.pardal@tecnico.ulisboa.pt

[‡] Departamento de Informática, Faculdade de Ciências, Universidade de Lisboa, Portugal
mcalha@ciencias.ulisboa.pt

Abstract—Blockchain technology is increasingly studied as a means to replace trusted intermediaries in consortium settings. While permissioned blockchains offer better control and security, they face the challenge of balancing decentralization with local governance. Each organization must define and enforce its own access rules, which complicates collaboration. In particular, ensuring data confidentiality is challenging, as data owners are often very strict about what to share and with whom. This paper presents the design of mechanism for secure and confidential data sharing between separate blockchain organization groups. It leverages channels in Hyperledger Fabric to achieve data isolation but also includes the ability to filter and transform the data to be shared. A case study in fruit traceability demonstrates how the proposal meets important requirements of real-world supply chains.

Index Terms—Confidentiality, Access Control, Smart Contracts, Blockchain, Hyperledger Fabric

I. INTRODUCTION

Supply chains involve multiple organizations collaborating to produce, transport, and deliver goods. These business-to-business (B2B) interactions require trust and efficient data exchange. However, participants often hesitate to share sensitive business information, such as inventory levels. Current practices rely on centralized systems or trusted intermediaries, such as ERP systems¹ or shared databases. While these mechanisms provide partial solutions, they introduce single points of failure, and require significant trust as they expose sensitive data to potential misuse. A more effective solution for supply chains must therefore combine *data ownership* by each organization and the ability to *selectively share* data with relevant partners.

Permissioned blockchains, such as Hyperledger Fabric (HLF) [1], provide a decentralized solution that can foster more trust between partners. Each partner provides computers to the system, called *peers*, that execute and record transactions together.

Work supported by national funds through Fundação para a Ciência e a Tecnologia, I.P. (FCT) under projects UID/50021/2025 and UID/PRR/50021/2025 (INESC-ID) and Project Blockchain.PT – Decentralize Portugal with Blockchain Agenda, (Project no 51), WP 1: Agriculture and Agri-food, Call no 02/C05-i01.01/2022, funded by the Portuguese Recovery and Resilience Program (PRR), The Portuguese Republic and The European Union (EU) under the framework of Next Generation EU Program.

¹Enterprise Resource Planning (ERP) is an software system that manages a company's internal processes such as finance, inventory, and operations.

An HLF transaction follows four phases to preserve integrity and confidentiality: (1) a client submits a *proposal* to endorsing peers; (2) endorsers *simulate* execution and return signed responses; (3) the ordering service *sequences* valid proposals into blocks; and (4) peers *validate and commit* blocks against endorsement and Access Control List (ACL) policies before updating the world state. HLF is well-suited for multi-party applications requiring secure data exchanges, such as supply chain traceability, healthcare, and financial services [2], [3].

Two HLF mechanisms are particularly relevant for data ownership: *channels*, which isolate ledgers per group of partners, and *private data collections* (PDCs), which restrict record visibility to a subset of members in the same channel [4]–[6]. However, neither channels nor PDCs address the secure and controlled sharing of information *across* different channels. It is not just a matter of data visibility, but the data contents themselves may need to be modified, for example, to remove details that are not to be shared. This limitation is particularly relevant in collaborative scenarios, such as supply chains, where selective data exchange is crucial [7]. It is precisely this challenge that motivates our proposal: the *Confidential Channel-to-Channel* (CC2C) framework, which enables confidentiality-preserving data sharing across blockchain channels while maintaining the principles of control, transparency, and trust.

A. Motivating Example

Consider a traceability system involving several organizations in a food supply chain. It starts on the farm, where agricultural products are cultivated and registered. These raw products are then sent to factories for transformation, processing, and packaging. The goods are next delivered to depots for intermediate storage, then distributed to supermarkets for public sale. Finally, consumers can access traceability data via a package identifier, encoded as a QR code, for example, to verify the origin and journey of the product. Data is sent to the blockchain, providing a secure and transparent method for verifying provenance across all actors. Figure 1 illustrates data capture and recording at each step.

We implemented our framework in a specific use case: the cherry supply chain in the Fundão region of Portugal. These cherries are a geographic certified product “*De Origem*

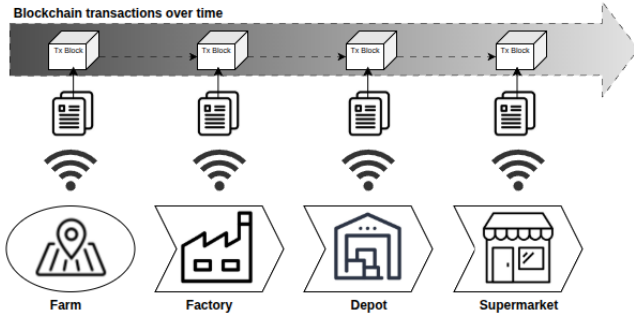


Fig. 1: Information flow along the supply chain

Protegida” (DOP) that is recognized by consumers for its distinguished quality.

In this use case, the factory producer imposes these requirements: data is shared only for cherries purchased by the supermarket chain, excluding shipments to other retailers; timestamps are resampled to daily granularity (date only); and original source fields are aggregated and reported at the civil parish level in the region.

Field and production data is recorded in the factory’s HLF channel. Sharing this information with the supermarket involves more than just granting visibility; it also requires specific data transformations. This is precisely where the structured CC2C mechanism is introduced.

The specific requirements of the use case include:

- R1 - Data Integrity & Security: must ensure immutability of records and protect sensitive business data;
- R2 - Permissioned Access: different stakeholders should have controlled access to relevant data;
- R3 - Scalability: should handle the necessary volumes of transactions efficiently;
- R4 - Interoperability: integration with IoT, ERP, and external databases should be supported;
- R5 - Regulatory Compliance: compliance with food safety and data protection laws is mandatory.

II. PROPOSAL OF CC2C

The Confidential Channel-to-Channel (CC2C) is a mechanism designed to facilitate secure and confidential data exchange between blockchain channels. An asset is a digital representation of value or data (e.g., records) that is owned and managed by authorized participants. It is stored on the ledger.

A. Clients

This system also includes two types of clients that interact with the mechanism, one designed for the *owner* of the information, Owner Client (OC), and the second client is designed for *subscribers* that query the information, called Subscriber Client (SC). In our example, the owner is the factory, and the subscriber is the supermarket chain.

The owner when using the CC2C together with the OC will have several options to work with:

- **Select:** choose private assets to share based on specific criteria or identifiers;
- **Transform:** perform minimal modifications to selected assets;
- **Aggregate:** combine multiple assets into a single consolidated output;
- **Split:** divide a single asset into multiple distinct assets;
- **Authorize:** grant or revoke permissions for other clients to query assets;
- **Emit Event:** send events for other clients listening on the channel.

The subscriber with the SC will have different options:

- **Listen:** monitor events and decide to read or write data;
- **Query:** manually or automatically query events with proper permissions to read or write data;
- **Write:** mark down information to be read from the source channel and written to the destination channel.

These methods let clients listen to subscribed events and, upon receiving a specific event, automatically query the corresponding asset to retrieve information. This enables auditing or forwarding data to another channel. Once copied and written to the shared channel, it cannot be undone, and all participants can view the resulting asset and its contents.

B. Design Principles

The identified design principles and implementation considerations are presented as follows:

- Data Confidentiality across Channels: ensure a secure and privacy-preserving data exchange between channels using techniques such as those provided by Certificate Authorities (CA);
- Granular Access Control: implement fine-grained, certificate-based access policies to regulate data visibility and sharing among stakeholders;
- Auditability: implementation of an audit trail mechanism to log data transfers and ensure transparent, verifiable records for regulatory compliance and trust;
- Data Manipulation: support configurable operations (e.g., Select, Transform, Aggregate, Split, Copy) to allow tailored data processing while maintaining confidentiality and integrity;
- Resilience to Threats: design the framework to mitigate risks such as unauthorized access and data tampering, leveraging SSL/TLS for secure communication;
- Scalability: the solution should be scalable to accommodate a growing number of participants and data exchanges;
- User Experience: allow participants to intuitively and effectively manage data sharing and access permissions, without compromising security.

C. Threat Model

The design must first define a clear *threat model* specifying attacker types and their possible actions. In the CC2C framework, attackers are classified by key possession: they may lack a certificate or hold a valid Subscriber or Owner certificate.

We then consider possible attacks such as replay, unauthorized read/write, and request manipulation. This classification guides the design of suitable defenses. For example, we can consider an adversary with a valid Owner certificate that replays transactions, disrupting data flow between channels. To counter this, robust audit logging can be put in place, to ensure accountability so that the problem can be discovered and addressed later by the partners.

D. Architecture

The CC2C framework is designed for two types of users: those who control and own the data, and those who wish to read, audit, or copy data. The system comprises several components: a user interface, a database that stores non-business data such as credentials and configurations, Node.js clients with specific functions and endpoints, and a central chaincode that validates transactions and client accesses. The workflow involves owners accessing data via REST API, with the chaincode validating transactions, while stakeholders use REST API calls to retrieve shared data. The system also integrates external systems and leverages HLF for support.

HLF uses the Raft consensus protocol for deterministic transaction ordering with crash fault tolerance. Raft is suited to a permissioned blockchain because it can perform efficiently among a limited set of authenticated, semi-trusted partners.

The CC2C Chaincode is the core component of the system, deployed on each peer. It encapsulates the logic required to manage the lifecycle of blockchain assets and enforces the rules for creating, querying, transforming, and sharing data. When new information is submitted, the chaincode registers the asset on the blockchain and within an internal structure defined as the `CC2C_struct`. This structure stores essential metadata, that include:

- `id`: the asset identifier;
- `date`: the creation timestamp;
- `hash`: a SHA-256 hash summarizing the asset;
- `who_authorize`: authorization metadata;
- `source`: origin type (original or transformed);
- `allow_anyone_read` and `allow_read`: access control flags;
- `event_emit_type`: event types the asset may trigger.

Every request for the CC2C Chaincode undergoes an authentication and authorization process. The identity of the requesting client is verified through an *X.509 certificate*, which is validated both at the peer level and internally within the chaincode. Following this process, the chaincode provides a set of functions for validating and handling the main requests.

E. Functions

1) *CreateAsset*: Enables the recording of new assets on the blockchain. At the same time, it creates an entry in the CC2C structure to support fast indexing and permission checks. This dual registration ensures that access control policies can be enforced efficiently during future queries.

2) *Transform*: Creates a new version of an asset by modifying specific details, enabling partial disclosure of information. For example, replacing exact GPS coordinates with general locations or, for simplicity, rounding quantities.

3) *Aggregate*: Combines several assets into one. For instance, multiple deliveries (e.g., 100 kg, 59 kg, 300 kg, 41 kg) can be aggregated into a single asset of 500 kg, abstracting detailed transactional data.

4) *Split*: Divides a single asset into two or more. For example, a 500 kg asset may be split into two assets of 300 kg and 200 kg respectively, enabling selective sharing of assets.

The original assets are never modified; each operation creates a new asset with preserved traceability, ensuring auditability.

5) *Authorize*: Allows owner clients to grant or revoke permission for other clients to access asset data. These permissions are stored in the CC2C structure, which links client certificates to access the asset.

6) *emitEvent*: Trigger events are based on the asset's defined `event_emit_type` field. Each listed event is emitted accordingly. Subscriber Clients listen for these events and automatically run if properly configured.

III. DATA TRANSFER IN ACTION

This section details the process of data transfer within the system, illustrating in detail how information is written to the blockchain and subsequently transferred between channels. We describe step by step how data moves from the data owner to the blockchain and how it can later be accessed or replicated through the framework.

First, the data owner writes an asset into its own channel. Figure 2 represents how the information is persisted.

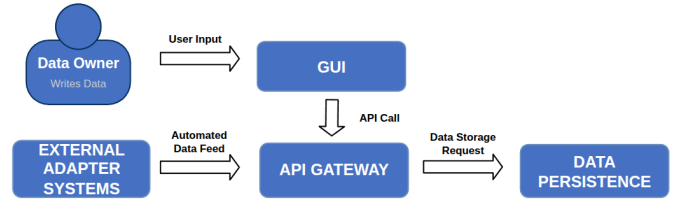


Fig. 2: Data Flow: Write Data

Data can be persisted in two ways; the first one is triggered by a user that wants to store the data and uses a Graphical User Interface (GUI) to send the data to the API Gateway; the second way is automated by external adapters that gather and send information to the API. In both ways, the API Gateway forwards the data to the persistence zone.

When the owner wants to transfer data to a partner, a request is sent to the CC2C Chaincode. It first undergoes an authentication and authorization process to determine which client certificate is being used and whether it has the appropriate permissions to invoke the requested function.

Once all processing steps are completed, as defined by the data owner, the actual data copying occurs. Figure 3 represents how the information is copied using CC2C. Certain events or users, using the identifier of the asset can activate the copy

service. The copy service will request the stored data and save it as new persistent data on the target channel.

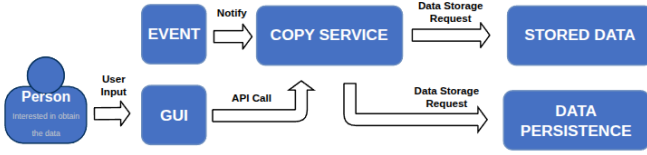


Fig. 3: Data Flow: Copy Data

In summary, the data transfer mechanism describes the channel-to-channel data flow, where data goes from a source channel to a target channel. For example: the *source* is a private channel belonging to Organization A; the *target* is a channel shared by Organizations A, B, and C.

IV. RELATED WORK

Hyperledger Fabric (HLF) has been extensively studied as a permissioned blockchain framework for secure and efficient data sharing. Androulaki et al. [8] introduced the concept of channels in HLF to facilitate private communication between consortium members, addressing scalability and confidentiality concerns. However, this work does not specifically address cross-channel data confidentiality. Liu et al. [9] proposed a multi-channel structure for data exchange in the Internet of Vehicles (IoV) for HLF. Although this structure improves data exchange efficiency, it lacks mechanisms that ensure confidentiality when data crosses channel boundaries. Kang et al. [10] introduced FabZK, a framework integrating Zero-Knowledge Proofs (ZKPs) with HLF for privacy-preserving smart contracts. Although it enhances privacy at the contract level, it does not specifically address confidentiality during inter-channel data exchanges. Hoyer et al. [6] explored cross-organizational data exchanges based on HLF. However, their study does not sufficiently tackle the challenge of maintaining confidentiality when data moves between channels. Chen et al. [11] identified privacy leakage vulnerabilities in HLF chaincodes, emphasizing the need for robust confidentiality-preserving mechanisms. The focus was on individual chaincode vulnerabilities rather than multi-channel confidentiality. Thakkar and Shah [12] proposed a privacy-preserving sharing framework for Electronic Health Records (EHR) using HLF. Despite ensuring patient data protection, their solution does not consider cross-channel data confidentiality, especially in multi-party environments.

State-of-the-art approaches do not address cross-channel confidentiality, particularly mechanisms that allow data owners to control both the visibility and the content of their data, according to their needs. CC2C addresses this gap by enabling secure data exchange across blockchain channels.

V. CONCLUSION AND FUTURE WORK

For participants to join a permissioned blockchain network, a key requirement was the ability to retain ownership and control of their data. Information could not be universally visible or shared without explicit consent. This principle of separation

and controlled sharing proved essential in the Fundão Cherry supply chain case study.

HLF channel structure allows for data isolation, where each channel operates as an independent ledger. A Channel Access Policy (CAP) defines which peers, having dual-channel visibility, are allowed to read from the source and write to the target channel.

Future work will assess the performance of CC2C under higher transaction volumes and extend its application to other domains, while systematically evaluating its confidentiality, integrity, and access control guarantees. To regulate visibility after data is transferred, Multi-Organization Access Rules (MOAR) will need to be defined, to ensure that access rights are explicitly set when assets enter a target channel. These rules will operate in conjunction with fine-grained role-based access control (RBAC), which restricts data visibility to authorized stakeholders only. Together, they will establish a controlled and secure sharing environment across organizations. In addition, the framework will integrate several supporting mechanisms: an Audit Trail Mechanism (ATM) to log transfers with immutable and verifiable records; an Integrity Check Module (ICM) to guarantee data correctness via cryptographic validation; and a Data Processing Module (DPM) to apply confidentiality-preserving operations such as select, transform, and aggregate before transfer.

REFERENCES

- [1] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. D. Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, et al., "Hyperledger fabric: A distributed operating system for permissioned blockchains," *Proceedings of the Thirteenth EuroSys Conference*, 2018.
- [2] N. P. S. A. M. I. K. A. P. A. P. C. S. Joy, B. C. R. and M. K. Lal, "Hyperledger fabric as a secure blockchain solution for healthcare 4.0 framework," in *International Conference on Advances in Computing and Communication Technology*, 2023.
- [3] R. Alhabib and P. Yadav, "Hyperledger fabric platform for secure and efficient data sharing in autonomous vehicles," in *International Conference on Wireless Networks and Communications*, 2024.
- [4] X. Fu, W. Shi, Q. Wen, and Y. Chen, "On private data collection of hyperledger fabric," *IEEE Transactions on Blockchain*, 2021.
- [5] Y. Xu, Y. Shen, X. Liu, and R. H. Deng, "The privacy protection mechanism of hyperledger fabric and its application in supply chain finance," *Cybersecurity*, 2019.
- [6] F. D. T. L. V. Hoyer, T. Wauters and B. Volckaert, "Trustful ad hoc cross-organizational data exchanges based on the hyperledger fabric framework," *International Journal of Network Management*, 2020.
- [7] S. Brotsis, N. Kolokotronis, K. Limniotis, G. Bendiab, and S. Shiaeles, "On the security and privacy of hyperledger fabric: Challenges and open issues," *arXiv*, 2021.
- [8] A. D. C. E. Androulaki, C. Cachin and E. Kokoris-Kogias, "Channels: Horizontal scaling and confidentiality on permissioned blockchains," in *International Conference on Financial Cryptography and Data Security*, 2018.
- [9] Y. F. Y. Liu and K. Sakurai, "Hyperledger fabric-based multi-channel structure for data exchange in internet of vehicles," 2024.
- [10] N. J.-L. S. T. H. Kang, T. Dai and X. Gu, "Fabzk: Supporting privacy-preserving, auditable smart contracts in hyperledger fabric," in *Dependable Systems and Networks*, 2019.
- [11] J. G.-J. Z. K. W. J. H. Z. G. Z. Chen, Y. Li and Z. Chen, "Understanding and detecting privacy leakage vulnerabilities in hyperledger fabric chaincodes," *International Symposium on Software Reliability Engineering*, 2024.
- [12] V. A. Thakkar and V. Shah, "A privacy-preserving framework using hyperledger fabric for ehr sharing applications," *International Journal of Electrical and Computer Engineering Systems*, 2023.