UNIVERSIDADE DE LISBOA
INSTITUTO SUPERIOR TÉCNICO

Low-Latency Privacy-Preserving Access to Edge Storage

Cláudio José Pereira Correia

Supervisor: Doctor Luís Eduardo Teixeira Rodrigues
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

2024

UNIVERSIDADE DE LISBOA
INSTITUTO SUPERIOR TÉCNICO

Low-Latency Privacy-Preserving Access to Edge Storage

Cláudio José Pereira Correia

Supervisor: Doctor Luís Eduardo Teixeira Rodrigues
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

Jury

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

Members of the Committee:

Doctor Luís Eduardo Teixeira Rodrigues, Instituto Superior Técnico, Universidade de Lisboa

Doctor António Manuel Raminhos Cordeiro Grilo, Instituto Superior Técnico, Universidade de Lisboa

Doctor Rolando da Silva Martins, Faculdade de Ciências, Universidade do Porto

Doctor Anjo Lucas Vahldiek-Oberwagner, Intel Labs

Funding Institutions

Universidade de Lisboa
Fundação para a Ciência e Tecnologia

2024

Acknowledgements

Firstly, I want to extend my heartfelt gratitude to everyone who has stood by me, motivated, and guided me throughout this journey. Their collective influence, from those I've known closely to those whose paths briefly crossed with mine, has been immeasurable. I'm also deeply thankful for the equitable foundation provided by the Portuguese educational system, which empowered me to dream beyond my small limits.

From the inception of this project to its culmination, my sincere thanks go to my supervisor, Prof. Luís Rodrigues. He has been a grounding force during our highs and an unwavering believer during our lows. His steadfast presence over the past five years, regardless of the challenges and outcomes we faced, combined with his ambition and tireless drive for excellence, has been instrumental in steering this work to its successful completion. Similarly, I owe a profound debt of gratitude to my co-supervisor, Prof. Miguel Correia. His consistent support and camaraderie during the ups and downs of this project have been a great source of strength. My deepest appreciation goes out to both for their unparalleled guidance every step of the way.

I'd like to express my deep appreciation to the faculty of the Distributed Systems Group (GSD) at INESC-ID. Their kindness and support have been invaluable in numerous ways. A heartfelt thank you to Paula Barrancos, Prof. Nuno Santos, Prof. Fernando Ramos, Prof. Rodrigo Bruno, Prof. Paolo Romano, Prof. Luís Pedrosa, and Prof. Miguel Pardal. A special thanks to Prof. Carlos Ribeiro for his assistance and patience in understanding the cryptographic nuances of my work. I'm also grateful to those who collaborated with me during this journey: Eng. Rita Prates for her dedication, and Eng. Rodrigo Silva for his camaraderie and enduring friendship.

I extend my thanks to all my friends and colleagues both within and outside of the GSD for their friendship and companionship. Thank you, João Rafael Soares, for your time spent on deep and intriguing discussions about topics most would find eccentric. My gratitude goes to Diogo Barradas and Maria Casimiro for their unwavering support over the last five years. Thank you, Tiago Brito, for your daily warmth and friendship. A special nod to Daniel Porto for his unfailing readiness to assist everyone in their time of need. Daniela Lopes, your friendship will always remain close to my heart. Thank you, Mafalda Ferreira, for being the first to open the door to room 501. Thank you, Bernardo Bacalhau and Raquel Prata, for the unity throughout our years at IST. João Teixeira, I'm grateful for motivating me to step out every night. Bruno Guerra, your belief in my capabilities has been a source of strength. To my little princess, João Bernardo, thank you for always brightening my day with your smile. Guilherme Bonifácio, your presence in my moments of need has been invaluable. Andre Guerra and Tomas Guerra, I'm blessed to be part of your family. In no particular order, my thanks extend to: João Gonçalves, Daniel Castro, Francisco Chamiça, João Loff, Daniel Andrade, Antonio Ruano, Pedro Miguel, Miguel Coimbra, Ângelo Tomás, Ana, Ana Marques, Taras Lykhenko, Anibal Mateus, Edi Pereira, Francisco Simão, João Aveiro, João Pinto, Rodrigo Lorvão, Duarte Henriques, and João Veríssimo.

Most importantly, I want to dedicate the most heartfelt thank you to those closest to me, whose daily impact on my life is beyond measure. To my parents, Nazaré Correia and Orlando Correia, your unwavering strength, affection, and love have been my foundation. I owe immense gratitude to my entire family, always greeting me with a smile and open arms. Sónia Santos, though our time was short, you brought immense light into my life. Beatriz Feliciano, with the kindest heart I've ever known, your radiant smile and endless patience have been a beacon, reinforcing the belief that happiness is attainable even in the demanding world of scientific research. You're also the best sushi companion one could ask for.

In loving memory of my father, Orlando Correia, whose guiding light continues to inspire and shape my journey even in his absence. This thesis is a testament to the values and strength he instilled in me.

To each and every one of you – Thank you.

This work was supported by the Fundação para a Ciência e a Tecnologia (FCT) scholarship 2020.05270.BD, by national funds through FCT via the INESC-ID grant UIDB/50021/2020 and project DACOMICO (financed by the OE with ref. PTDC/CCI-COM/2156/2021).

Delving into the enigmatic labyrinth of cryptography
to protect our past, present, and future.

Abstract

Edge computing is a paradigm that extends cloud computing with storage and processing capacity close to the user, providing bandwidth savings and lower latencies. This paradigm assumes the availability of microdatacenters, also known as fog nodes, that are located close to the edge. These nodes are installed and managed by various local providers, whose privileged access to the infrastructure represents a significant security risk for applications and clients. Unethical edge providers my engage in malicious behaviors for financial gains, particularly if their actions remain undetected. Given the high risk associated with dishonest providers, it is crucial to secure the functions fog nodes provide.

This thesis is devoted to the design of security mechanisms for data storage in edge computing environments. Given that accessing data with low latency is a primary motivation for adopting edge computing, it is crucial to ensure that data is effectively replicated at the edge and can be accessed in a timely and privacy preserving manner. This thesis address these two relevant problems that emerge in edge computing, namely how to ensure that edge providers use local storage as specified in their service level agreements and how to preserve the privacy of edge clients. In this context, the thesis:

- Proposes an audit technique that verifies whether a storage node at the edge can retrieve a data object within a specified latency threshold. The technique is based on a cryptographic time-bounded challenge that needs to be executed by the audited node. Leveraging the capabilities of secure hardware, we ensure that the proof of data retrieval is generated by the audited fog node itself.
- Proposes a novel authentication technique for access control at the edge to protect stored data from unauthorized entities. This technique aims to preserve client anonymity during authentication processes, despite their physical proximity to fog nodes. The proposed scheme preserves the privacy of clients even after they have been revoked from the system, achieving this more efficiently than all the related work.

A promising approach to enhance security in edge storage systems is to resort to the usage of secure hardware, such as Intel SGX enclaves. This thesis explores the use of hardware enclaves to design these two mechanisms, that together, will help edge clients in accessing data with low latency while respecting their privacy.

Resumo

A computação na periferia da rede é um modelo emergente que estende a arquitetura tradicional da computação na nuvem para próximo dos clientes, com computação e armazenamento. Esta proximidade é viabilizada pela proliferação de vários microdatacenters, também conhecidos por *fog nodes*, localizados estrategicamente na periferia da rede, permitindo assim reduzir a latência dos serviços e o consumo de largura de banda. Estes microdatacenters são instalados e mantidos por uma variedade de fornecedores locais, que, em alguns casos, podem revelar-se não confiáveis, adotando comportamentos maliciosos com o objetivo de obter vantagens financeiras. Devido ao elevado risco associado à presença de fornecedores desonestos, é crucial assegurar a proteção dos serviços prestados na periferia da rede.

Esta tese visa desenvolver mecanismos de segurança para o armazenamento de dados na periferia da rede. Tendo em conta que aceder a dados com baixa latencia é uma das motivações principais para a adoção de computação na periferia da rede, é crucial garantir que os dados são efetivamente replicados nos microdatacenters e podem ser acedidos de forma expedita e com garantias de privacidade. Esta dissertação responde a estes dois problemas relevantes que se levantam na utilização de armazenamento na periferia da rede, nomeadamente, como assegurar que os fornecedores de armazenamento na periferia respeitam os níveis de qualidade de serviço acordados e como preservar a privacidade dos clientes que acedem aos dados. Neste contexto, esta dissertação propõe:

- Uma técnica de auditoria que verifica se um nó de armazenamento na periferia da rede consegue recuperar um objeto armazenado dentro de um limite de latência especificado. Esta técnica é baseada num desafio criptográfico que deve ser resolvido de forma atempada. Recorrendo às funcionalidades de hardware seguro, asseguramos que a prova criptográfica dos dados é gerada pelo próprio nó auditado.
- Uma nova técnica de autenticação para proteger os dados armazenados de entidades não autorizadas. Este técnica tem como objetivo preservar o anonimato dos clientes durante os processos de autenticação, mesmo estando estes fisicamente próximos dos microdatacenters. O esquema proposto garante a privacidade dos clientes, inclusive após a sua revogação do sistema, sendo mais eficiente que as soluções propostas anteriormente.

Uma técnica promissora para reforçar a segurança nos sistemas de armazenamento consiste na utilização de hardware seguro, como os enclaves da Intel SGX. Esta dissertação explora a utilização de enclaves no desenho destas duas técnicas que, quando combinadas, iram assegurar que os clientes de serviços de armazenamento na períferia da rede conseguem aceder aos dados com baixa latência e com garantias de privacidade.

Keywords
Palavras Chave

Keywords

Distributed Systems Security

Trusted Computing

Edge Storage

Proofs of Storage

Pseudonymity

Palavras Chave

Segurança em Sistemas Distribuídos

Hardware Seguro

Armazenamento na Periferia da Rede

Provas de Armazenamento

Pseudonimidade

Table of Contents

1 Introduction

1

Overview

The cloud computing model has become integral to everyday life, permeating through a plethora of applications and devices [14]. The expansion of these applications, increasingly reliant on cloud services, is propelled by the affordability of the cloud. This affordability is attributed to economies of scale from large shared infrastructures, which provide vast storage, fault tolerance, computational power, and high availability at low costs. It is projected that over 41.6 billion devices will be connected by 2025 [198], known as the Internet of Things (IoT), many equipped with sensors that generate substantial volumes of data needing collection and processing [89]. However, cloud resources are hosted in remote data centers, imposing long latencies and stressing network infrastructure due to escalating bandwidth demands.

Edge computing [99] has emerged as a solution to the inherent constraints of cloud computing, aiming to process data closer to the devices, significantly reducing network bandwidth usage and offering services with lower latency [159]. Edge computing will complement the services provided by remote data centers with the service of smaller data centers, or even individual servers, located closer to the edge. This concept is often named *fog computing* [187, 53, 38]. It assumes the availability of fog nodes that are located close to the edge. The number of fog nodes is expected to be several orders of magnitude larger than the number of data centers in the cloud. Cloud nodes are physically located in secure premises, administered by a single provider. Fog nodes, instead, are most likely managed by several different local providers and installed in physical locations that are more exposed to tampering. Therefore, fog nodes are substantially more vulnerable to being compromised [199, 137], and developers of applications and middleware for edge computing need to take security as a primary concern in the design.

To unleash their full potential, fog nodes should not only provide processing capacity but also cache data that may be frequently used [3]; otherwise, the advantages of processing on the edge may be impaired by frequent remote data accesses [135]. By using cached data, requests rarely need to be served by data centers. Consequently, a key ingredient of edge-assisted cloud computing is a storage service that extends the one offered by the cloud in a way that relevant data is replicated closer to the edge. However, these services must prioritize security and reliability, particularly because they operate on potentially vulnerable fog nodes.

The work in this thesis is focused on the development of security mechanisms for data storage at the network edge. It tackles two distinct challenges individually, with solutions that can be integrated to facilitate the implementation of edge storage systems. The first challenge is to develop a cryptographic proof of data replication to audit an edge provider's local storage and

identify any potential overselling of storage capacity. The second challenge involves data access control at the edge; the close proximity between clients and fog nodes during authentication may put user privacy at risk. We create an anonymous authentication scheme based on pseudonyms that surpasses current state-of-the-art methods in terms of storage efficiency and fault tolerance.

The use of Trusted Execution Environments (TEEs) has been identified as one of the most promising technologies for securing computation and sensitive data in fog nodes [146, 56]. One of the most popular TEE technologies is Intel Software Guard Extensions (SGX), due to its widespread availability in standard Intel CPUs. SGX offers a form of TEE known as *enclaves* [10, 129]. The potential advantages of this technology for fog computing have been acknowledged by Intel [104] and have seen practical applications [35, 4, 58]. In recent years, various storage systems have been developed with the support of Intel SGX to achieve different security properties. However, many of these systems [49, 40, 114, 58] still face limitations and challenges, particularly when deployed in edge computing environments.

1.1 Objectives

One of the primary motivations for adopting edge computing is its capability to provide computing and storage in close proximity to users, ensuring low latency — a property not achievable with cloud computing due to the potentially distant location of data centers. However, this close proximity also brings the risk of compromising users' privacy by local edge providers. The potential for unethical behavior among edge providers presents a significant challenge in maintaining both low latencies and the necessary privacy for edge clients. In light of these considerations, the main objective of this thesis is:

♢ To ensure access to edge storage with privacy and low latency.

Unfortunately, as further detailed below, realizing this objective becomes challenging when faced with the possibility of edge providers engaging in hidden, financially motivated malicious activities. Providers may resort to cheap remote storage, instead of the contractual obligation of the edge storage, imposing long latency's to clients when accessing this data. Moreover, enforcing privacy is an mandatory requirement in edge environments. Existing authentication methods that safeguard privacy rely on complex and cumbersome schemes, that can also impose long latencies on clients.

In this work, we aim to design schemes that assure low latency access to edge storage by enhancing security at the edge. This approach is intended to stimulate the development and adoption of storage services specifically designed for edge computing environments. Next we define and discuss these two major research problems that can compromise the realization of the above objective.

1.2 Problem Statement

In our comprehensive study of the state-of-the-art in edge storage systems, as detailed in Section 2.2.2, we have observed a clear trend towards the utilization of TEEs to address security vulnerabilities at the edge. A number of storage systems are already leveraging TEEs to

securely store data on fog nodes, ensuring privacy, integrity, and authentication. Despite these advancements, there remain several critical, unresolved security challenges that can compromise this thesis goal.

One significant challenge is auditing edge storage in fog nodes, where untrusted service providers may oversell their storage capacity. This leads to an essential question that guides our research:

- Is it possible to design an auditing scheme to verify edge storage?

As storage systems expand to the network edge to provide lower latencies, it becomes crucial to verify that edge providers are effectively replicating data at the edge, complying with their Service-Level Agreement (SLA). Applications that pay providers to store data close to their users necessitate robust auditing tools to ensure these storage services are fulfilling their obligations. A viable method of auditing involves the use of storage proofs, which offer a variety of cryptographic proofs with distinct characteristics, as detailed in Section 2.3.1. The most reliable storage proofs are underpinned by TEEs at the audited node, effectively preventing the delegation of proof to a remote node. Nonetheless, these proofs are dependent on the local clocks at the audited node, making them vulnerable to potential security breaches [11, 128].

Consequently, there is a need to develop an auditing scheme that can confirm whether fog nodes are storing data in a manner that aligns with the low latency requirements of edge clients. This scheme should facilitate a *Proof of Timely-Retrievability*, one that does not rely on the local clocks of the audited node but rather leverages the enclave to support and validate the proof response.

The second focus of this thesis is providing anonymous authentication. This issue is particularly acute in edge computing environments, where the close proximity of entities can potentially be exploited to reveal sensitive information. This prompts the following open question:

- Is it possible to design an anonymous authentication scheme for the edge?

Access control based on authentication is vital for any storage service, especially at the edge, due to the vulnerable and exposed nature of fog nodes, which leads to a higher risk of being compromised. Additionally, the process of authenticating at the edge could inadvertently reveal sensitive client information, such as identity, location, and usage patterns. Therefore, anonymous authentication becomes a requirement in edge environments to preserve client privacy. Our investigation into the latest solutions for anonymous authentication, as detailed in Section 2.3.2, revealed a range of approaches, from group signatures [106] to more complex methods like zero-knowledge proofs [19]. These solutions rely on heavy cryptographic schemes to protect client privacy but also result in a significant latency penalty for client authentication. Ideally, the optimal solution for the edge would involve the use of public key encryption-based pseudonyms, chosen for their efficiency and broad adoption.

Despite the extensive array of solutions for anonymous authentication, most face significant challenges in delivering two crucial properties [96]: *Backward Unlinkability* and *Revocation Auditability*. Both aim to maintain client anonymity even after the client is revocated from the system. Revoking a client's access can lead to the disclosure of sensitive information, risking their anonymity. Section 2.3.2 delves deeper into these properties. Briefly, backward unlinkability

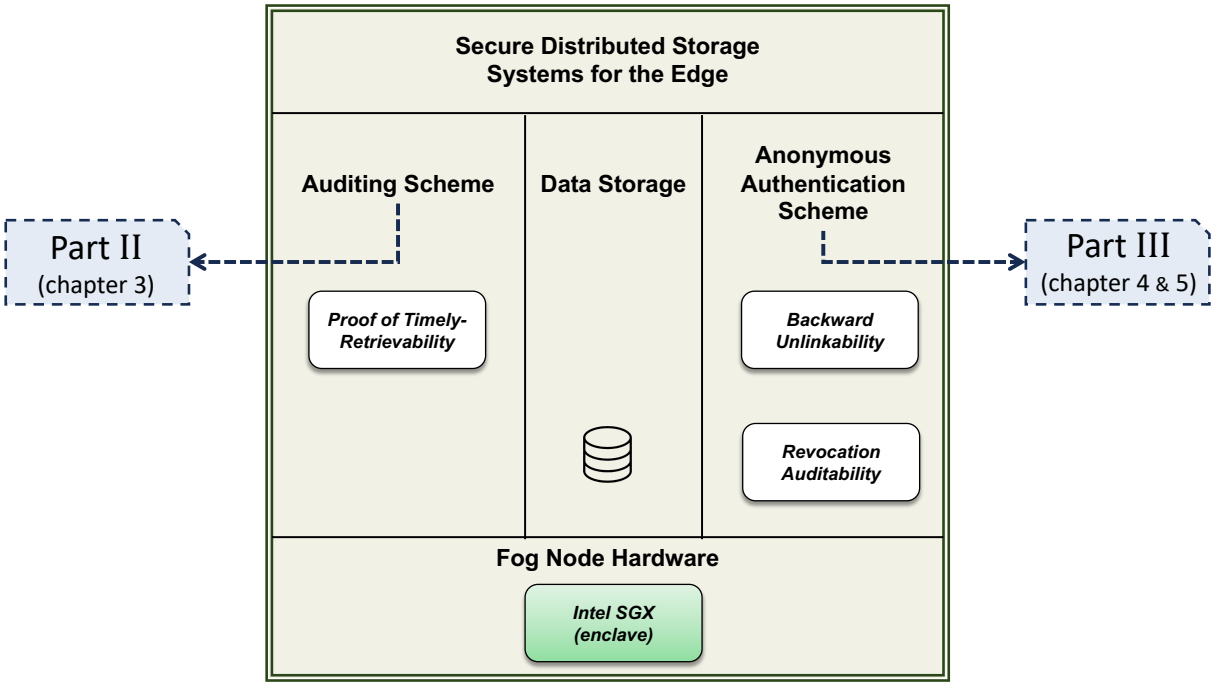


Figure 1.1: Architecture overview of a secure storage system for the edge and mapping to this thesis structure.

maintains the anonymity of a client's past authentications following their revocation, whereas revocation auditability ensures that the client is informed of their revocation. The implementation of these properties within public key encryption remains a significant, unresolved challenge.

In Figure 1.1, we provide a comprehensive overview of how the two central problems discussed in this thesis integrate within the architecture of an edge storage system. Each problem encapsulates specific properties that are the focal points of our research. The challenge of auditing edge storage systems is comprehensively addressed in Part II, which involves developing a proof of timely-retrievability. The foundation of our auditing tool lies in the utilization of TEEs, which is elaborated upon in more detail in Chapter 3. The second problem, concerning anonymous authentication, is tackled in Part III of this thesis. Here, Chapter 4 focuses on achieving backward unlinkability, while Chapter 5 delves into attaining revocation auditability. The next section outlines our three key contributions, each aligning with the three distinct properties we endeavor to realize within the framework of these two pivotal problems.

1.3 Contributions Summary

This thesis delves into the two aforementioned problems, materialized in two distinct components: an auditing scheme and an anonymous authentication scheme. While various security risks exist for storage systems at the edge, our research is particularly focused on these two critical areas, which are vital for the successful deployment of secure storage systems at the edge. Driven by these considerations, our work is divided into three significant scientific contributions:

- **C1:** *A cryptographic secure and accurate auditing tool for edge replicated storage systems.*
- **C2:** *A distributed anonymous authentication scheme that provides efficient backward unlinkability.*
- **C3:** *A distributed anonymous authentication scheme that provides revocation auditability.*

Contribution C1 focuses on the component that audits whether data replication at the edge is being effectively provided. The contributions C2 and C3 pertain to the anonymous authentication component, with C3 offering different tradeoffs with regard to C2. In this refinement, both backward unlinkability and revocation auditability are accomplished within a unified scheme.

In the following sections, we offer a comprehensive overview of the key contributions of this thesis. This exposition is structured around the results from the three pivotal contributions outlined above.

1.3.1 (C1) Auditing Tool for Proof of Timely-Retrievability

Our analysis on the challenges of current storage proofs, led to the development of our novel Proof of Timely-Retrievability (PoTR). PoTR assesses whether a storage provider can retrieve data objects within a latency lower than a specific SLA threshold δ. By setting δ to a value that can only be met if data is stored locally, PoTR can distinguish whether an edge node stores data locally or remotely. We avoid the use of vulnerable clocks at the audited node, measuring time from the auditor's side. This approach is possible by meticulously designing PoTR to mitigate network noise, we experimentally validated PoTR using a challenging edge computing environment. The results demonstrate PoTR's effectiveness in differentiating a fog node adhering to the latency SLA from one that does not, even when the dishonest node uses a nearby fog node, resulting in a latency discrepancy of less than 1.5ms.

PoTR offers several advantages for edge storage systems. Firstly, it enables applications to confidently trust that their data is effectively replicated at the edge, ensuring low latency access for their clients. Secondly, PoTR is a cost-effective, easy-to-deploy solution, seamlessly integrating with the edge computing model. Thirdly, it is beneficial for providers, building trust in what is often considered a zero-trust edge environment. These advantages significantly contribute to the design and deployment of future edge storage systems. Detailed information about this contribution can be found in Chapter 3.

The results of this contributions were published at the IEEE Pacific Rim International Symposium on Dependable Computing (PRDC) 2023 [60]. This publication can be found at: `https://web.ist.utl.pt/claudio.correia/papers/PoTR_PRDC.pdf`.

1.3.2 (C2) Anonymous Authentication with Backward Unlinkability

Given the limitations of current state-of-the-art anonymous authentication schemes, we developed a novel pseudonym-based anonymous scheme, termed Range-Revocable Pseudonyms (RRP). This system achieves near-perfect backward unlinkability, overcoming the shortcomings of prior solutions. It introduces a storage-efficient scheme where clients are required to store only the pseudonyms necessary, independent of any system variable. Additionally, our innovative

revocation approach incurs only logarithmic costs due to the granularity of backward unlinkability, a significant improvement over the linear costs associated with revocation list sizes in previous systems. Additionally, thanks to the strategic selection of public key encryption, our system's authentication latency stays below 3.5ms. This is well within the optimal 5-30ms range for edge computing mentioned in [159].

Our study validates the practicality of deploying an efficient and anonymous authentication scheme in edge computing environments. This fully distributed approach, anchored by Intel SGX technology, establishes a robust trust foundation for edge clients. As mobile devices increasingly depend on edge infrastructure, our system presents a low-latency and privacy-preserving solution for both service providers and clients at the edge. Full details of this contribution are given in Chapter 4.

The results of this contribution were published at the ACM Conference on Computer and Communications Security (CCS) 2023 [59]. This publication can be found at: `https://dl.acm.org/doi/pdf/10.1145/3576915.3623111` *and the extended version can be found at:* `https://arxiv.org/pdf/2308.03402.pdf`

1.3.3 (C3) Anonymous Authentication with Revocation Auditability

The previously presented contribution, C2, efficiently provides backward unlinkability, enabling authentication with the low latencies required in edge computing environments. However, a limitation of C2 is its lack of revocation auditability, a crucial step for ensuring complete authentication anonymity in pseudonym-based schemes. Therefore, we have utilized the primitives introduced by C2 to develop this final contribution, C3, aimed at achieving revocation auditability. This was essential because the only existing offline solutions capable of offering this feature are based on zero-knowledge proofs, which are too computationally intensive and expensive for edge computing.

We demonstrate that this final contribution can achieve both revocation auditability and backward unlinkability. This new scheme continues to allow clients the freedom to choose the number of pseudonyms they wish to store and further enhances the storage efficiency of revocation lists. However, there is a trade-off: clients in this scheme must download the revocation list at the authentication instant. This requirement introduces an additional latency overhead in authentication compared to the previous contribution. Comprehensive details of this contribution are outlined in Chapter 5.

1.4 Ramifications

This section summarizes a ramification of our research. This work, was performed in collaboration with Rodrigo Silva, a MSC student of our research group, focused on the trade-offs between deduplication and privacy in cloud storage.

In cloud storage services, data privacy often leads users to encrypt files, yet this encryption blocks server-side deduplication, as identical files appear different when encrypted. There is considerable work on merging file encryption with data deduplication, known as encrypted deduplication. This integration usually necessitates some form of coordination among clients, potentially creating privacy vulnerabilities like frequency analysis attacks. These attacks deduce

the contents of an encrypted file based on its storage or access frequency. Our research sought to balance the need for data privacy with the advantages of deduplication, focusing on methods to avoid frequency analysis attacks. We proposed a new protocol for assigning encryption keys to files using secure hardware to obscure chunk frequencies from adversaries.

The protocol offers a tunable encrypted deduplication system that leverages TEEs to perform sensitive cryptographic operations and keep track of the frequency of operations performed on every chunk. This data is stored in a frequency table (cache) internal to the TEE and a larger encrypted table is stored in an untrusted environment. The resulted protocol is the first to offer full privacy protection while achieving exact deduplication. It is capable of providing secure deduplication through frequency hiding. We study and evaluate different privacy policies to protect this table in insecure memory, and show that is possible to enforce privacy over this table, while others can not. Additionally, the proposed protocol allows clients to read their files without requiring interaction with the TEE, offering increased performance on reading operations, more than 5x faster over the state-of-the-art [133] that require read operations to be performed inside the enclave.

A manuscript describing this collaborative work was accepted for publication at the ACM/SIGAPP Symposium on Applied Computing (SAC) 2023 [167]. This publication can be found at: `https://doi.org/10.1145/3555776.3577711`

1.5 List of Publications

Considering the above contributions and collaborations, the publications that resulted from this thesis, in chronological order, are the following:

- Cláudio Correia, Miguel Correia, Luís Rodrigues. Using Range-Revocable Pseudonyms to Provide Backward Unlinkability in the Edge. Proceedings of the 30th ACM Conference on Computer and Communications Security (CCS), Copenhagen, Denmark, November 2023.
- Cláudio Correia, Rita Prates, Miguel Correia, Luís Rodrigues. PoTR: Accurate and Efficient Proof of Timely Retrievability for Storage Systems. Proceedings of the 28th IEEE Pacific Rim International Symposium on Dependable Computing (PRDC), Singapore, October 2023.
- Rodrigo Silva, Cláudio Correia, Miguel Correia, Luís Rodrigues. Deduplication vs. Privacy Tradeoffs in Cloud Storage. Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing (SAC), Tallinn, Estonia, March 2023.
- Cláudio Correia. Safeguarding Data Consistency at the Edge. Proceedings of the 50th IEEE/IFIP International Conference on Dependable Systems and Networks Supplemental Volume (DSN-S) PhD Forum, Virtual Event, Spain, June 2020.

1.6 Thesis Structure

The remaining of the thesis has the following structure:

Chapter 2: introduces the fundamental concepts relevant for the work on this thesis.

Chapter 3: presents and evaluates PoTR, a new cryptographic proof of storage, tailored to meet the challenges of edge computing.

Chapter 4: presents and evaluates RRPs, a novel cryptographic scheme for anonymous authentication with backward unlinkability.

Chapter 5: presents and evaluates Privacy Keeper, an enhanced cryptographic scheme providing anonymous authentication with both backward unlinkability and revocation auditability.

Chapter 6: concludes the thesis, summarizing the results achieved and suggesting directions for future research.

2

Background

This section provides the necessary background of the main topics that are relevant to this work. First, Section 2.1 covers the background on edge computing, different examples of edge applications, and their current security vulnerabilities. Then, Section 2.2 addresses the benefits and limitations of current implementations of TEEs and existing storage systems supported by TEEs. Lastly, Section 2.3 identifies different existing challenges in the design of a secure storage service at the edge and discusses this challenges in light of current state-of-the-art solutions.

2.1 Edge Computing

The emergence of IoT and the stress it places on services that operate in the cloud motivates the use of computing resources close to the edge. Edge computing is a model of computation that aims at leveraging the capacity of edge nodes to save network bandwidth and provide results with low latency [99]. However, many edge devices are resource constrained (in particular, those that run on batteries) and may benefit from the availability of small servers placed in the edge vicinity, a concept known as fog computing [187, 53, 38]. Fog nodes provide computing and storage services to edge nodes with low latency, setting the ground deploying resource-eager latency-constrained applications, such as augmented reality. These fog nodes are computers or clusters of computers with heterogeneous capacity in terms of storage, computation and internet connection, also known as cloudlets [162]. Services that leverage fog nodes are able to guarantee availability even if a central data center becomes intermittent due to a cyberattack [149], an outage [178, 28], or a natural disaster [22, 72].

Figure 2.1 shows a general architecture of fog-edge computing, which consists of multiple infrastructures layers. As we move down the stack, from the cloud to the edge, the number of nodes in each layer increases but the capacity of each of these nodes decreases. In the cloud layer, we may find data centers while in the fog layer we may find ISP servers, private data centers, and 5G towers. Finally, in the edge, we may find all kinds of devices such as desktops, laptops, tablets, smartphones, sensors, and actuators.

The architecture suggests that the devices located in the edge layer communicate mainly with those located in the fog layer. In addition, fog nodes also have to communicate with the cloud where critical data will be stored in a secure and persistent way.

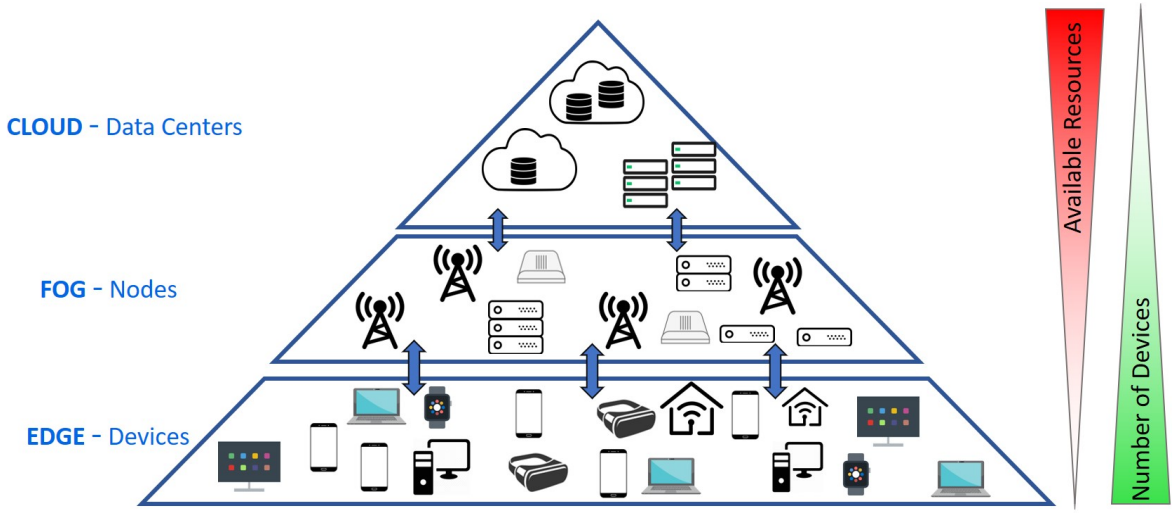


Figure 2.1: Three-layer fog computing architecture.

2.1.1 Applications

One of the major motivations to deploy fog nodes close to the network edge is the ability to offer resources with low latency. Edge applications can use these services at a distance of a hop, many existing applications and future ones can leverage the edge storage services, where latency may be a requirement, example of such edge applications are:

- Vehicular communications (e.g., VANETs [158, 43, 153], V2V [87] and V2I [165, 8, 32]) that require access to local traffic information or real-time maps.
- Location-based services (e.g., Google Maps [92], Waze [191], Foursquare [84], MobiShare [192], Yelp [197, 21], Pokemon GO [145], Yahoo! Weather [195]) that can store all movements and places clients visit over time, that we consider sensitive information regarding the user's privacy.
- Various smart appliances (e.g., traffic lights [196, 12], smart locks [98], video cameras [107, 164], industrial machines [97, 91]) that may upload large quantities of redundant information, such as video frames or sensors data.

Many of these applications require latencies below 30ms, leaving as a requirement for the fog computing model latencies between 5ms-30ms range [159]. Besides, these applications may disclose sensitive information regarding both the users and stored data. Leaving user privacy as a real concern at the network edge.

2.1.2 Data Storage

Storage services are widely used in cloud computing today, and a large number of designs have been implemented [70, 116, 157]. Most of these systems support geo-replication, where data copies are kept in multiple data centers. Geo-replication is relevant to ensure data availability in case of network partitions and catastrophic faults, but it is also instrumental to serve clients with

lower latency than what would be possible with a non-replicated system. However, as discussed previously, cloud-based geo-replication may not suffice to achieve the small latencies required by novel latency-critical applications. Therefore, extending storage services to operate on fog-nodes is a relevant research challenge.

Many geo-replicated storage services, such as COPS [123], Saturn [39], or Occult [130], support geo-replication across a large number of nodes, and have been carefully designed to operate their replicas in several well-connected data centers. Fog nodes connections may become unavailable, and is important to design systems that can operate even if disconnected from the network. To answer this challenge new storage services have been designed specifically for the edge, such as GESTO [2], ElfStore [134], and A-DECS [190], that aim to manage multiple replicas while striving to offer data availability under possible replicas malfunction or failures. There is a clear motivation to deploy storage services at the edge, however, all these systems make a strong assumption that replicas will not fall under attack. Such assumption cannot be made in the edge environment, where fog nodes are considerably more vulnerable and expose, therefore, edge storage services must take security as a requirement to ensure service correctness.

2.1.3 Security Vulnerabilities

In the edge computing model, fog nodes bring resources closer to the edge. However, the security guarantees offered by the cloud cannot be moved so easily. In the cloud model, data centers are isolated and well protected. On the other hand, fog nodes will be more exposed, and as a result, it is not possible to use the same techniques as in the previous model. The fact that fog nodes are dispersed among multiple geographic locations, close to the edge, increases the risk of being attacked and becoming malicious as discussed in the Zhang *et al.* [199], Mukherjee *et al.* [137] and Zhou *et al.* [201] surveys. A compromised fog node may delete, copy, or alter operations requested by edge devices, causing information to be lost, leaked, or changed in such a way that it can lead the application to a faulty state. Therefore one of the major challenges in the fog computing model is how to deal with malicious fog nodes.

Given the exposed location and significant heterogeneity of fog nodes, the behavior of a malicious fog node can differ based on the attacker's objectives. To clarify, we categorize two distinct behaviors for a malicious fog node, each correlating with the attacker's specific intent.

1. Rational behavior: An edge provider may adopt a malicious behavior if it can gain some benefit and pass unnoticed. Examples of such behavior are: inferring personal information from its clients, scrutinizing the stored data, or overselling its storage capacity.

2. Destructive behavior: This is the more traditional attack model, where an attacker can gain either physical or remote control of the system with the sole intention to leave the system or clients in a failed state. Such an attacker may attempt to corrupt the data stored in the fog node, discard stored information, replay or block requests, and serve faulty or stale data to the client.

Our intention in separating the attacker behavior is only to offer a better explanation for the reader, throughout our document, we assume that a malicious fog node can have both behaviors simultaneously, and we strive to design mechanisms that protect our storage system from both. A fundamental objective is to provide an appropriate level of security without

compromising the performance of the system, given that one of the main goals of supporting edge computing is to provide services with low latency to clients. Therefore in this document, we study security techniques to face the challenge of malicious fog nodes while maintaining the low latency requirements, one such technique is the use of TEE.

2.2 Trusted Execution Environments

Edge and fog devices are a very compelling opportunity to use processors with Trusted Execution Environment (TEE) [74, 146] technology, a secured execution environment with guarantees provided by the processor hardware. The code that executes inside a TEE is logically isolated from the Operating System (OS) and other processes, providing integrity and confidentiality, even if the OS is compromised. Therefore, the use of a TEE is a natural choice to secure computation and sensitive data in fog nodes. With it, fog nodes can securely store private keys and offer clients a base of trust. Should edge providers opt to adopt such technology and hardware, it would significantly enhance the appeal and adoption of edge computing among applications and users.

A processor with this technology can operate in one of two modes, *normal mode* or *secure mode*. *Normal mode* is where the operating system and applications run without any special security guarantees, while in *secure mode*, application code has isolation, integrity, and confidentiality. This division between two different modes requires developers to build applications in two parts, one for each mode. One important goal of these architectures is enabling any application to take advantage of this secure environment. Therefore, device hardware configuration provides a TEE API, where applications in *normal mode* can request secure operations in *secure mode*. These requests require the processor to switch context between the two modes. When the processor switches from *normal mode* to *secure mode*, it fetches the last state of the application that will run in *secure mode*, decrypts it and verifies its integrity. When it goes from *secure* to *normal mode*, it encrypts this state and stores it in memory. Between context switch, this state is encrypted and decrypted using a key that is secure in a chip element.

TEE also offers a remote call to verify that *secure mode* inside the processor has not been tampered with, this operation is known as *attestation*.

Examples of TEE technologies are: *AMD Secure Encrypted Virtualization (SEV)*, which allows to protect system memory using encryption, but encrypting all the memory of an application has a significant latency cost; *ARM TrustZone* that exists in processors focused on the mobile devices, offering low-power consumption; RISC-V architecture is gaining popularity and also provide a secure environment [62]; *Intel SGX* which is an architecture available in many common Intel processors [1]. In the following paragraph we introduce in more detail Intel SGX. We chose Intel SGX technology as it has led to significant new work in the field [4, 114, 24, 115, 57] in the recent years. This increases its potential to be used in the real world by the industry, besides the fact that it is considered a practical solution to the fog (Intel Fog Reference Design [104]).

[1] Currently Intel is only incorporating SGX technology for server side CPUs, such as Intel Xeons.

2.2.1 Intel SGX

Intel Software Guard Extensions (SGX) is a set of functionalities introduced in sixth generation Intel Core microprocessors that implement a form of TEEs named *enclaves* [10, 129]. The potential benefits of this technology for the fog have already been recognized by Intel [104] and it has already been used in practice [35, 4, 57].

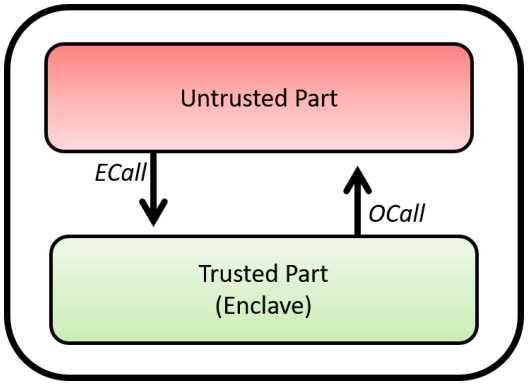


Figure 2.2: An application in Intel SGX.

Applications designed to use SGX have two parts: an untrusted part (normal mode) and a trusted part (secure mode). Figure 2.2 shows a simplification of the Intel SGX design. The trusted part runs inside the enclave, where the code and data have integrity and confidentiality; the untrusted part runs as a normal application. The untrusted part can make an Enclave Call (ECALL) to switch into the enclave and start the trusted execution. The opposite is also possible using an Outside Call (OCALL). The SGX architecture implements a number of mechanisms to ensure the integrity of the code, including an *attestation* procedure that allows a client to get a proof that it is communicating with the specific code in a real SGX enclave, and not an impostor [26]. Such technology allows to store cryptographic keys in a secure way inside untrusted machines such as fog nodes and have guarantees about the code that is executed for example to generate metadata [57]. A limitation of the first SGX implementations is that the protected memory region, named Enclave Page Cache (EPC), is typically limited to 128 MB [102]. A significant increase in EPC capacity was introduced with SGXv2, an increased EPC capacity from the previous 128 MB to up to 512 GB per socket [75]. Despite this increase, it still falls short of the expectations for a standard storage system memory requirements. Therefore, it is essential to optimize the memory usage inside the enclave. Additionally, enclaves cannot persist their state, being critical to store data outside the enclave for any long term service, such as storage system.

Since an enclave depends on the untrusted part for launching and execute and store persistent state, the Intel SGX threat model does not protect a single enclave from suffering Denial of Service (DoS), we follow the same assumption in this thesis. However, there are workarounds to mitigate this issue. One solution is to replicate the system, as discussed in Section 4.3.2. By replicating a system across different nodes, one can estimate the number of f faults the system can tolerate. Such mechanisms can also withstand eclipse attacks [170].

Aside from blocking the enclave, the untrusted part can also perform various attacks that have proven successful in recent years under the Intel SGX threat model. The most notable

attacks against SGX are Foreshadow and LVI [186]. Intel is actively investigating ways to mitigate these issues [148]. An alternative solution to mitigate these attacks could be leveraging system replication, where nodes use different TEEs, such as some resorting to SGX while others use RISC-V or ARM TrustZone. This approach is orthogonal to the work in this thesis and could be a direction for potential future research. Implementing such a solution would require a multi-party computation framework to tolerate any leaks or vulnerabilities in a single TEE. This vision has motivated the design of our solutions to be easily adaptable to other TEEs.

In this thesis, we leverage Intel SGX due to its high availability and ease of access, making it a suitable solution for implementation inside fog nodes and providing a trust base for edge devices. In our work, such guarantees help secure cryptographic material or execute secure functions in fog nodes.

2.2.2 SGX Storage Systems

The use of SGX has been a hot research topic in recent years, as the security properties offered by the SGX enclaves are desired by many distributed applications. In particular, this has inspired many storage systems based on Intel SGX to improve the security of current cloud storage services. However, as previously mentioned, SGX enclaves have limited memory, therefore in many of these systems, we see novel mechanisms and schemes to mitigate this memory limitation. In the next paragraphs, we present the state-of-the-art storage systems based on Intel SGX that strive to overcome the enclave limited memory.

Enclavecache [49] is a storage system designed to operate at the cloud layer as many others, in particular, it leverages the Intel SGX enclaves to process clients' requests, enforcing confidentiality and integrity of the data. To overcome the limited memory of the enclave, the enclave is only responsible to encrypt/decrypt the requested data, the data is then stored outside the enclave in the untrusted part. Before sending the data to be persistently stored in the untrusted part, the enclave encrypts the data to enforce confidentiality. By resorting to the untrusted part to store the encrypted data, Enclavecache is able to mitigate the SGX memory limitation.

SecureKeeper [40] is another storage service based on SGX, however, is more concerned with the distributed aspect of the data replication. SecureKeeper is design on top of ZooKeeper and leverages the enclaves to enforce data consistency between the multiple replicas, it performs consensus among replicas that can result in long latencies.

ShieldStore [114], similar to Enclavecache, stores the data in the untrusted part of the system. However, it offers a novel technique to verify the integrity of this vulnerable data. Note that, despite being encrypted, data stored outside the enclave can be corrupted by a malicious OS, assumed to be compromised in the SGX threat model. To answer this challenge, ShieldStore implements a flat Merkle tree over the encrypted data, and only requires the Merkle tree root to be stored in the protected memory of the enclave. Such mechanism also overcomes the limited memory of the enclave and offers a latency efficient solution for the data access and integrity check. Following the same path, Speicher [24] and Concerto [13], present alternative techniques to protect the data outside the enclave. Speicher implements a Log-Structured Merge Tree (LSM)-based storage, with 2 different levels, where the first is the enclave memory and the second in SSLTables in disk. While Concerto [13] verifies data integrity in a deferred manner and at client request, a solution that is not practical for the edge. Interestingly Concerto also implements a Merkle tree outside the enclave but needs to pass as input the entire path from the leaf to the root through the Ecall.

These storage systems based on Intel SGX were designed having in mind an untrusted cloud provided, that may try to read or modify the stored data without authorization. We argue that an edge provider will be more susceptible to engage in such behavior, which can be amplified due to the characteristic of the edge environment. The concern for edge security has already been noted, Harpocrates [4] and Omega [58] are two storage systems that have already been carefully designed for the network edge, and also leverage the use of enclaves.

Harpocrates [4] implements SGX enclaves at Content Distribution Networks (CDNs) that are a geographically distributed network of proxy servers, which is located close to the users at the edge, and host static content. CDNs are capable of mitigating DoS and offer low latency for stored data, in Harpocrates the enclaves are responsible for the mutual authentication between the clients and the servers while protecting the required cryptographic material.

Finally, Omega [57, 58] offers a storage service carefully designed for the edge computing model. In Omega, similar to previous systems, the enclave delegates the storage responsibility for the untrusted part and also implements a Merkle tree for integrity protection. Additionally, Omega publishes a public log generated from the enclave, this log leverages blockchain technology for integrity protection, allowing clients to perform read operations over this data without requiring the enclave intervention. This particular technique enables to reduce the latency overhead of the read operations, which no longer requires the enclave interaction.

2.2.3 Limitations

All the previous storage systems based on SGX are centred on a single critical limitation of the enclaves, their limited memory. Many of their proposed mechanisms offer desirable solutions for this challenge, with tradeoffs relative to the latency and scalability. Another important concern is the latency overhead required to perform the context switch between the enclave and the untrusted part. These limitations reveal that SGX based solutions should resort to the use of the enclave only for critical operations, and output protected data that can later be reused. Despite the use of the SGX enclaves, a malicious provider may attempt to perform other attacks, as discussed in Section 2.1.3, infer information from the client's authentication or data access patterns, or oversell the edge storage, non of these systems discuss these issues.

2.3 Protecting an Edge Storage Service

The security vulnerability of fog nodes at the edge is a clear concern, and research efforts have been redirected to this challenge, where the use of SGX enclaves is one promising approach to mitigate such concerns. However, the majority of the related work on storage services based on enclaves has mainly centered its efforts on mitigating the memory limitations of the SGX enclaves. In this section, we identify two different vulnerabilities that we will strive to solve, and that we believe to be paramount for the viability of a secure storage service for the edge. For both identified vulnerabilities, we discuss the state of the art, which is again, mainly designed for cloud computing. We search for mechanisms and techniques that can either be adapted or moved to the edge to answer our challenges.

2.3.1 Auditing Storage Replication

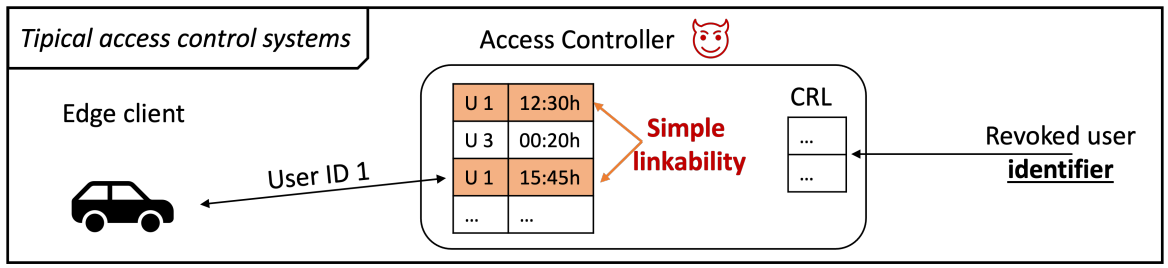
Today, there are numerous scenarios where an end-user or an organization stores data on machines operated by third parties. This is often done to ensure data durability and availability or to provide low latency access for others. One such example is edge storage [176], where fog nodes are utilized for file storage, enabling clients to access data with low latency. Due to the limited capacity of fog nodes, edge storage providers might be tempted to oversell their capacity and compensate for this by fetching data on-demand from the cloud, instead of delivering it with the requisite low latency. It is important to note that even when leveraging a storage system based on SGX, the data is stored in a untrusted zone and can be easily relocated by the provider. While the literature abounds with auditing techniques, i.e., evidence that the third party is adhering to (or deviating from) the agreed-upon quality of service, most of these schemes were designed for cloud environments. Next, we will present the state-of-the-art schemes in proofs of storage.

The literature is rich in techniques that allow to obtain different proofs of storage, that assess different properties of the provided service. The most relevant ones are *Proofs of Data Possession* (PDP) [16] and *Proofs of Retrievability* (PoRet) [15] that aim to check if the storage provider keeps at least some copies of the stored data; *Proofs of Replication* [30, 120] that assess if the storage provider has n copies of a data item (even though they might all be placed in the same machine); and *Proofs of Geographic Replication* (PoGR) [31, 90] that test if the storage provider keeps n data copies in distinct machines, in distinct geographic locations. Each one of these proofs is issued as a response to a *challenge* [120], that is sent to the storage provider by one, or more auditing entities. Typically, a challenge requires the storage provider to execute a set of reading operations over a subset of the stored data, and return on-time a value that proofs the access of the correct data items. This return value, may be an cryptographic hash of the retreived data items.

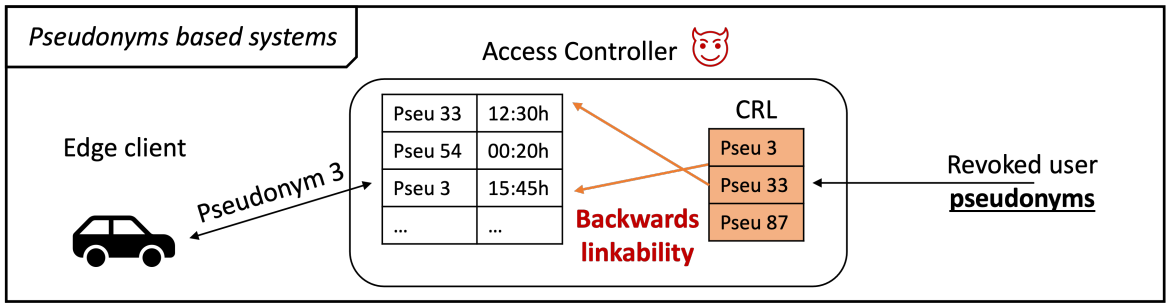
To save bandwidth most challenges require the storage provider to read a subset of the stored data items, and compute a cryptographic hash of them. Typically, the response has to be issued within a pre-defined time deadline, determine by the auditor [30, 31, 120, 66]. Furthermore, when sending the challenge, the list of data items to be access should not be revealed entirely. Instead, the list should be interactively revealed, i.e., the next data item to be accessed, it is just known after accessing the previous one [120]. This interactivity prevents the storage provider to download the missing data items on-demand. If the storage provider cannot guess in advance the data items to be accessed, it has to keep all data items within the constraint access time, to build a correct and timely proof.

There are two main mechanisms to effectively guarantee that a proof is generated in a given geographic location. The first one consists on using the response delay to predict the audited node location, since it may be difficult for the storage provider to issue a timely proof if data is kept in a distant geographic location, due to the extra network latency to access the data. The usage of a set of auditors, in different locations, may increase the accuracy of this mechanism by resorting to triangulation mechanisms [31, 90]. The second approach consists on resorting to a TEE to ensure that operations essential to the proof construction are executed in a given machine [66]. At the same time, this secure environments may be used to interactively reveal the challenge.

Recently, several schemes have been developed offering different storage proofs that leverage Intel SGX to enhance their security or efficiency [200, 108, 66]. By utilizing the enclave, it is



(a) Simple linkability



(b) Backwards linkability

Figure 2.3: Linkability and anonymity discloser.

possible to ensure that these proofs are genuinely generated on the audited machine, by requiring a continuous iteration between the enclave and the untrusted part, where data is stored, through Ecalls and Ocalls. However, as discussed in more detail in Section 3.2.3, these schemes use the enclave's trusted clock to monitor the duration of the challenges. Such clocks, whether TPM or provided by the SGX library, incur substantial overhead; and the untrusted part may maliciously delay trusted timer messages to fetch remote data blocks on-demand, thereby evading detection [11, 128]. Given these factors, using the enclave for time measurements becomes impractical. Therefore, it is necessary to design a new storage proof that does not rely on clocks in the audited machine, but instead utilizes the enclave solely to ensure that the proof is effectively constructed at the audited node and not at a neighboring one [128].

2.3.2 Imposing Authentication Anonymity

In any distributed storage service, client authentication before granting data access is vital to ensure only authorized clients access the data. However, it has been previously noted that authentication between a fog node and an edge client can disclose sensitive information due to the strong locality of edge computing. This concern is even more aggravated in other edge applications such as V2V [87], traffic lights [196, 12], or smart locks [98], where authentication may not be mediated by a TEE [2]. Therefore, we address this concern from a broader perspective, ensuring authentication protects the client's identity when accessing any type of edge resource, including our proposed edge storage system operating in fog nodes.

The diversity of edge resources, often managed by different entities, presents a risk where curious owners might track access control requests to gather private information about clients,

[2]Despite the use of a TEE in our storage system, is paramount that the untrusted part participates in the client authentication process to avoid possible DoS.

such as daily routines and habits. This possibility arises because access control is usually based on the identity of the authenticated client requesting resource access. Figure 2.3a shows a typical access control system where users authenticate using their true identifiers, allowing the controller to easily track their activities. To counter this, anonymous authentication schemes must be implemented, one suggested solution is the use of pseudonyms for distinct resource accesses, as required by the GDPR [132]. This approach, however, increases the data transmitted for client renewal or revocation, potentially leading to large Certificate Revocation Lists (CRL). Furthermore, revocation of a client using different pseudonyms allows an adversary to link these pseudonyms, breaking anonymity [95]. Figure 2.3b illustrates a pseudonym-based access control system, demonstrating how revocation information can contain all pseudonyms of a user, enabling a malicious controller to link previously used pseudonyms.

To enhance privacy in such systems, *backward unlinkability* is a key requirement, where actions performed under one pseudonym must not be traceable to past pseudonyms used by the same client, especially after a pseudonym is revoked [95, 113]. We can define backward unlinkability as follows: *when a revocation occurs, the pseudonyms used by the client before the revocation remain anonymous.* This ensures that revocation information does not compromise the client's past anonymity, thus providing a safeguard against profiling and tracking. Enforcing client anonymity during authentication has been addressed by several authors through innovative techniques and cryptographic protocols. However, authentication based on symmetric keys [32] is easily compromised if one of the parties is compromised. Identity Based Encryption schemes [36] impose considerable overhead due to their expensive cryptographic operations. Public key encryption remains the preferred choice for authentication and managing revocation information, due to its computational efficiency and widespread adoption.

Since typical pseudonym systems compromise client anonymity upon revocation, there has been a push for schemes that preserve pseudonym unlinkability after revocation. Haas *et al.* [95], followed by Khodaei *et al.* [113], propose a solution involving the association of pseudonyms with time intervals and revoking only those for current and future intervals, thereby not disclosing past pseudonyms. However, within their scheme, all pseudonyms used in the current interval remain linkable after revocation. Unfortunately, reducing the granularity of these time intervals in their design leads to a linear increase in storage costs for revocation material. Designing space-efficient solutions is crucial due to the heterogeneity of fog nodes and the limitations of enclaves.

Another property that must be achieved to fully respect anonymous authentication is *revocation auditability* [96]. Revocation auditability ensures that the system can verify the revocation status of any given credential or pseudonym, upholding user privacy and anonymity. This implies that users should be able to check their revocation status at a service provider prior to attempting authentication [96, 182]. If a user is revoked, they can opt to disconnect from the service without revealing any sensitive information. This capability is vital to avoid scenarios where a malicious service provider might process authentication requests from revoked users, thereby narrowing the user's anonymity set without their knowledge. Such subversion can lead to significant privacy violations by enabling the service provider to link all of the user's activities, which is particularly problematic in schemes that are challenged to provide robust anonymity. This property has not yet been achieved in pseudonym-based schemes, only in schemes supported by zero-knowledge proofs.

A final concern is the correct generation of pseudonyms that must be offered by a decentralized trusted party, which should also strive to offer availability and correctness. However, many solutions manage the pseudonyms generation and revocation in a centralized manner [165, 8, 112,

174], which can compromise the system availability or the system corrections (if a user evades revocation due to information loss).

2.4 Discussion

This chapter presented the background and motivation to design an edge storage system and highlighted two major challenges that such design will face at the edge. Recent storage systems have shown a clear interest in pursuing solutions based on SGX, that offer confidentiality and integrity despite the untrusted storage provider. We envision such a solution in the distributed edge environment, leveraging an enclave in each fog node, offering strong security guarantees for edge users and applications.

Despite the use of Intel SGX at the edge, different security challenges remain, we have highlighted two challenges that we aim to solve, thus contributing to the deployment of a more secure storage service at the network edge. One of these challenges is the design of an auditing tool to enforce edge storage providers to effectively store data at the edge, to detect and penalize if they resort to cheap remote cloud storage. The other challenge is the implementation of an authentication service capable of enforcing user anonymity while reducing the required storage at the fog node. Previous solutions either require a central entity or impose a linear storage cost for the system operation. Note that, even with the implementation of TEEs in fog nodes, this alone is not sufficient to ensure privacy. Edge storage systems, such as Omega [57], necessitate dual authentication: within the enclave for private data storage and in the untrusted part to avert Distributed Denial of Service (DDoS) attacks, thus motivating the need for anonymous authentication schemes. In the next sections, we present the contribution that we achieve under the context of these two security challenges.

II

Auditing Replicated Data in the Edge

3 Auditing Edge Storage: PoTR

This chapter introduces and evaluates an auditing mechanism that offers a novel Proof of Timely-Retrievability (PoTR). Our new cryptographic proof of storage, PoTR, aims at assessing whether a storage provider is able to retrieve data objects with a latency lower than some SLA-specific threshold δ. PoTR is capable of detecting a misbehaving storage provider in the challenging edge environment.

The remaining of this chapter is organized as follows: we begin by establishing the motivation for this contribution and defining its objectives in Section 3.1. Section 3.2 delves into related work. Our assumptions are laid out in Section 3.3, followed by the presentation of our solution in Section 3.4. Finally, Section 3.5 offers experimental results concerning the computational impact and the accuracy of our proposed cryptographic proof.

3.1 Motivation and Goals

Today, there are many scenarios in which end users, or organizations, store data in machines run by third parties, either to ensure durability and availability, or to ensure that customers can access data with low latency. Relevant examples include cloud storage (e.g., Dropbox, iCloud, and Google Drive), peer-to-peer storage (e.g., Filecoin [82], IPFS [29], and Swarm [175]), content distribution networks (e.g., Akamai [6] and Cloudflare [54]), and, more recently, edge storage [176, 6]. As highlighted in the preceding chapter, latency poses a significant challenge and concern when accessing data in edge environments. The delay in data retrieval is a critical issue impacting the efficacy of data placement algorithms within the extensively distributed and zero-trust edge environment [20].

Despite significant advances in storage systems, trust in providers has remained unchanged [94]. Customers require mechanisms to verify that QoS (Quality of Service) is being respected. Relevant QoS aspects include the guarantee that the third party will not discard or corrupt the stored data, that the data is stored on multiple distinct machines, in specific geographic locations, and that users are served with some bounded delay. Unfortunately, a misbehaving provider may opt to avoid complying with the agreement if it can gain some benefits and pass unnoticed. For example, the provider may keep the data in fewer locations than agreed with the customer, assuming that it may be impossible for the customer to audit how many

Systems	Data Locality	Efficient/Cheap Execution	Challenge Delegation Protection	Timer Delay Protection
Single Remote Auditor				
PoTR	✓	✓	✓	✓
PDP (2007) [16]	✗	✓	✗	✓
PoRet (2016) [15]	✗	✓	✗	✓
Benet et al. (2017) [30]	✗	✓	✗	✓
Li et al. (2020) [120]	✗	✓	✗	✓
Filecoin [82]	✗	✓	✗	✓
Multiple Auditors				
Benson et al. (2011) [31]	✓	✗	✗	✓
Gondree et al. (2013) [90]	✓	✗	✗	✓
Local Auditor Relying on TEE Clocks				
Dang et al. (2017) [66]*	✓	✓	✓	✗
EnclavePoSt (2022) [200]	✓	✓	✓	✗
Multiple Auditors and Reliance on TEE Clocks				
ReliableBox (2021) [108]*	✗	✗	✗	✗

* deprecated since Intel excluded trusted time from SGX Linux PSW [7].

Table 3.1: PoTR properties compared with the related work

replicas are used or where these replicas are placed. This threat has motivated the development of auditing techniques that are capable of extracting storage proofs, that is, evidence that the third party is complying with (or violating) the defined quality of service [31, 90, 66, 30, 120].

In this section, we present our new scheme, Proof of Timely-Retrievability (PoTR), which has the following objective:

Goal: Propose an auditing tool that aims at assessing whether a given server node is able to retrieve data objects with a latency lower than some SLA-specific threshold δ. By estimating an upper bound on the data access latency, we can also determine if the data is placed where expected. Namely, when the SLA threshold δ is small, it is possible to verify if data is being stored at the audited node or elsewhere. Our new proof mechanism, PoTR, takes advantage of the existence Intel SGX enclaves to ensure that the challenge is executed by the node being audited [128], not by some other remote node. By using SGX, we can also avoid prematurely revealing the data to be accessed during an audit, while keeping the communication between the auditor and the audited node to a single request-reply exchange. PoTR has been carefully designed to mitigate the noise introduced by this single message exchange. Our approach minimizes the network impact on our challenge accuracy, eliminating the need to rely on vulnerable and discontinued TEE clocks [7, 11], in contrast with recent related work, listed in Table 3.1.

Compared with previous work [108, 200, 66], we take a step forward and evaluate our proof in the highly challenging edge computing environment. When auditing edge services, the auditor may be located far away from the audited node and the communication network may exhibit large delays and jitter that can affect the accuracy of the proof. Edge computing relies on placing resources physically close to end users, such that applications can offer a latency lower than some SLA-specific threshold δ. By setting the SLA threshold δ to a small value that can only be satisfied by the provider if data is kept locally, we use PoTR to distinguish the case where the edge node stores data locally (Figure 3.1a) from the case where it keeps the data in some remote node or in the cloud (Figure 3.1b). We show that, even in the case where the dishonest node stores the file in a nearby fog node (and the observed latency differs by less than $1.5ms$) our PoTR can accurately pinpoint the misbehaviour.

(a) Honest Edge Provider. (b) Dishonest Edge Provider.

Figure 3.1: PoTR auditing scenarios.

3.2 Related Work

In this section, we delve into the critical related work and pertinent studies that underpin our research. Here, we explore the the importance to audit edge storage, scrutinizing its advantages and inherent challenges. We also examine auditing mechanisms for third-party storage services and discuss their current limitations.

3.2.1 Overselling Edge Storage

Many applications strive to reduce their loading times (e.g., social media, entertainment, e-commerce, advertising, and gaming), knowing that even a small difference in milliseconds can significantly impact their revenue [5]. In recent years, a growing number of applications have relied on content delivery networks (CDN) [93], such as Akamai, Amazon, or Verizon, to store static content closer to end users, achieving lower access latencies. Edge computing has emerged to offer storage and computation even closer to end users, supporting applications for face or object recognition [172], real-time databases [6], and just-in-time video indexing [163], which demand response times below 5-30 milliseconds [160], something that cannot be guaranteed with cloud storage alone or CDNs.

Fog nodes are managed by many local providers, and due to their limited resources, they cannot store copies of all objects stored in the cloud [57]. Instead, they typically keep copies of items that are required by local applications. Moreover, due to their limited storage capacity, edge storage providers may be tempted to oversell their storage and hide this behaviour by fetching, on-demand, data from the cloud or from other servers, instead of serving them from local (edge), resulting in increased latency for data users. One effective approach to address and detect such misbehaviour is through auditing mechanisms. These mechanisms can enhance the intelligent edge environment by providing real measurements to build trust autonomously between the local entities (e.g., through rating schemes) or to penalize non-compliant providers.

Filecoin storage [82] is a concrete example of an auditing mechanism, where a node that fails to prove its ability to store data as required will no longer receive financial rewards and may be expelled from the network. In cloud or edge storage, providers can be compelled to compensate their customers for violations of the defined contract, known as the SLA. Is essecnial to audit that the SLA is being delivered in edge applications such as: 1) Web, EdgeKV [6] (similar to CDNs) charge their clients to store data near clients and reduce latency; 2) Augmented reality apps, they need low-latency access to data to be shown to the user; 3) Autonomous vehicles, they need low-latency access to maps, directions, etc.

3.2.2 Auditing Third-Party Storage Services

When a storage provider is subject to an audit, it must provide a proof that it is applying the storage policies specified in the SLA. This proof is generically called a *proof of storage* [30]. Moreover, since an SLA may cover different aspects of storage implementation, such as the target number of replicas, the location of those replicas, or the latency observed by users when accessing data, it is possible to define different proofs.

3.2.2.1 Proofs of Storage

In the realm of data storage auditing mechanism, particularly in cloud storage as outlined in Section 2.3.1, substantial research has been undertaken. Fundamental techniques include *Proofs of Data Possession* (PDP) [16] and *Proofs of Retrievability* (PoRet) [15], which mainly verify the existence of some data copies. While, more complex methods like Proofs of *Proofs of Geographic Replication* [31, 90] attempt to audit whether data is replicated in distinct physical locations. These auditing methods required the audited node to respond to challenges initiated by auditing entities, typically involving the storage provider demonstrating access to the correct data items through specific reading operations and providing timely evidence, like a cryptographic hash of the audited data.

3.2.2.2 Structure of a Challenge

A simple way to verify if a storage provider keeps a given data item would be to request that item and then check its integrity. Although this method could, in fact, offer a PoRet, it has several limitations.

First, this approach is inefficient, as it requires the auditor to consume a large amount of bandwidth to obtain the proof. Therefore, to save bandwidth, most challenges require the storage provider to read a subset of the stored data items and compute a cryptographic hash of them. Typically, the response has to be issued within a predefined deadline, determined by the auditor [30, 31, 120, 66]. Furthermore, when sending the challenge, the data items should be revealed interactively to prevent the storage provider from downloading missing items on demand [120]. If the provider cannot guess in advance the data items to be accessed, it has to keep all items within the constraint access time to build a correct and timely proof.

Second, a single request for data items cannot verify some of the service requirements. For example, it cannot assess whether the storage provider keeps just one or several replicas of the data items. A typical solution is to encode each file with a distinct secret key [30]. Unfortunately, this does not prevent a provider from keeping all replicas in the same machine, which may compromise availability. An option is to design the challenge so that it is impossible to respond on time if all replicas are kept on the same machine, although it can be feasible if the proof is built in parallel[120].

These mechanisms are not enough to guarantee that the proof was generated by the target node. For this purpose, it is possible to leverage a TEE to interactively reveal the challenge and ensure that the operations essential to the proof construction are executed in a given machine [66].

3.2.3 Discussion

We now discuss the limitations of the related work in auditing storage systems, summarized in Table 3.1.

Efficient Execution. Both PDP [16] and PoRet [15] are designed to offer efficient cryptographic mechanisms to verify the integrity of cloud-stored data. Benet et al. [30] present a mechanism capable of verifying if there are a certain number of copies of a file, while Li et al. [120] audit if such copies are stored on different physical machines. These proofs depend on a single remote auditor, requiring low cost for deployment and execution. Filecoin [82] offers blockchain-based cooperative digital storage, which requires expensive cryptographic operations. These proofs cannot estimate data location or access latency, making them unsuitable for testing if a provider stores the required data in a specific node.

Data Locality. Triangulation mechanisms are one way to estimate data location. Gondree et al. [90] and Benson et al. [31] follow this approach by relying on multiple auditors/landmarks for the estimation. Unfortunately, triangulation can be expensive since it requires the intervention of multiple landmarks for each proof execution, and the accuracy of this mechanism depends on the proximity of the landmarks to the audited node. In this work, we take a different approach, where we decouple geolocation from data locality. An auditor can first geolocate a node (using techniques as in [90]) and then run our PoTR to check if the files are stored locally at that node. This approach has the advantage that the accuracy of the locality proof no longer depends on the landmarks, and exclusively relies on a single parameter that affects the proof duration, making it practical, particularly in edge environments.

Delegation Attack. Despite the use of triangulation, none of these systems is capable of enforcing the proof to be executed on a given machine; this is a critical obstacle when attempting to audit specific nodes, particularly in an edge storage environment. A storage provider, which controls the infrastructure, may capture the challenge request and delegate the production of the response to the remote storage: such behavior may be difficult to detect. Dang et al. [66], and EnclavePoS [200] resort to a TEE to execute their challenge on the correct machine to prevent this attack but, unfortunately, are still vulnerable to a clock delay attack.

Delay Attack. Using a TEE to trivially implement the auditor alone is not enough to ensure the generation of correct and honest proofs. These systems assume that the SGX clock can be trusted to measure the duration of the challenge at the audited node [66, 108, 200], but recent research has shown that the enclave cannot read an accurate and reliable system clock, due to the following issues [11]: (1) the storage node system clock is vulnerable to manipulations by the untrusted part; (2) the operation to read trusted timers, such as those provided by *SGX* or, TPM (*Trusted Platform Module*), has a large overhead, penalizing the proof accuracy; and (3) the untrusted part can maliciously delay trusted timer messages, to fetch remote data blocks on demand, and thus escape detection. Additionally, since 2020, Intel has excluded SGX trusted time from Linux PSW, leaving Dang et al. [66] and ReliableBox [108] deprecated. As a result, it is not practical to use the enclave for time measurements. Furthermore, as discussed below, ReliableBox focus on the triangulation task and is unable to verify if the audited node keeps the files locally or remotely.

Geolocation of the Node Being Audited. Some applications may require the geolocation of the machine being audited. The geolocation problem is orthogonal to the problem of timely retrievability and we advocate that these problems must be addressed by different, complementary, mechanisms. Geolocation typically requires the use of multiple auditors (namely, to perform

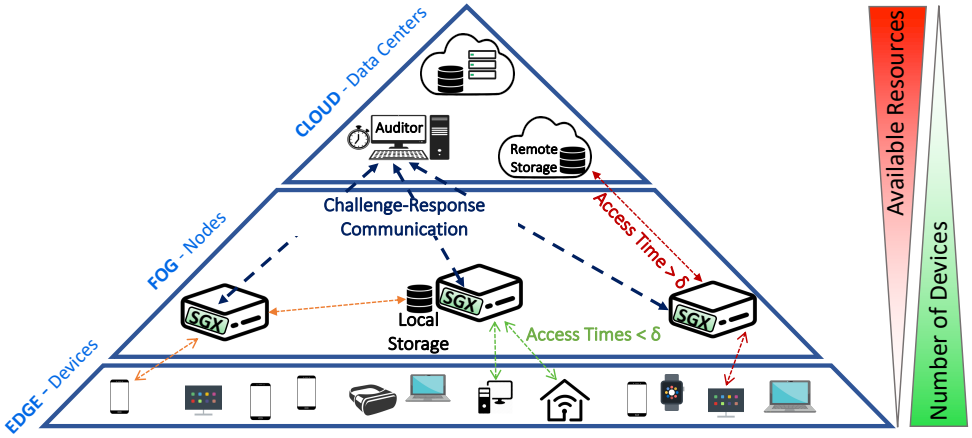


Figure 3.2: System architecture.

triangulation) while, as we show in this chapter, timely retrievability can be achieved using a single auditor. This separation allows performing the proof of geolocation only sporadically (for instance, when the machine boots) and then run a proof of locality more often.

One way to perform triangulation is by leveraging One-Way Delay (OWD) estimation. ReliableBox [108] measures the duration of the challenge both at a set of remote auditors and inside the enclave. It then computes the time difference between these two measurements to estimate the Round-trip time (RTT) between the auditors and the server, performing geolocation via triangulation. Unfortunately, ReliableBox only captures the network delays and is unable to assess accurately the duration of the proof construction at the audited node. This means that the response to the challenge can take an arbitrarily long duration (e.g. when data is stored in a remote node) without affecting the operation of ReliableBox: triangulation would still be accurate but the result of the challenge cannot guarantee data locality or any other properties. To provide both geolocation and timely retrievability, schemes such as ReliableBox need to be combined with schemes such as the one we proposed here.

3.3 System Architecture

We now present our system architecture in the context of edge computing that includes an auditor, a set of fog nodes, and a set of remote storage systems connected to the fog nodes (see Figure 3.2). Our proof, presented in the next section, is computed by a fog node to prove that it can access data with a latency lower than a given agreed threshold δ.

3.3.1 Assumptions

The protocol for obtaining a proof is executed between two nodes: the auditor and the audited fog node. A fog node is said to be *correct* if it can retrieve the files assigned to it with a latency lower than some given threshold δ. Any fog node that cannot satisfy this requirement is denoted *faulty*. The value of δ is application specific, but can be small and, in some cases, can only be satisfied if data is stored in a storage device directly connected to the fog node. Providers

should offer SLAs that define δ based on their capacity to maintain consistent performance under varying workloads. If a provider cannot meet δ latency under under high loads, they should consider offering an SLA with higher latency.

Our proof requires the audited node to read a configurable number N of data blocks. N can be conservatively selected to mask worst-case errors that may result from the variability of access to local storage and the variability in round-trip times. We also show that knowledge of the distribution of network delays and the distribution of storage access delays can be used to optimize the value of N when auditing honest nodes. Interestingly, these optimizations cannot be exploited by a rational provider to evade auditing. The configuration of PoTR is discussed in Section 3.5.3.

We assume that each fog node has a processor with *Intel SGX*, as we rely on the guarantees provided by a TEE. We assume that the auditor has the guarantee that it communicates with the expected enclave, due to the attestation process [57, 128], and also that the integrity and confidentiality of the data and code inside the enclave are guaranteed [57]. As explained in Section 3.2.3, the enclave cannot read a reliable system clock [11, 7]. However, the enclave can provide us with the guarantee that the proof is effectively built at the audited node, which is the property we leverage in the solution [128].

Finally, we assume that it is not feasible for the edge storage provider to reallocate enough objects in less than δ at the beginning of the audit, from a remote storage location. As it will be seen, our audit executes quickly (under 500ms), limiting the number of files that can be downloaded in time, even if the edge node has a high bandwidth link.

3.3.2 Fog Node Storage Organization

The edge storage provider is responsible for storing the files in the fog layer and ensuring that the files are stored in such a way that they can be retrieved with a latency lower than δ, the SLA defined threshold.

In each fog node with *Intel SGX*, local documents are kept in the untrusted part, as the storage capacity of the enclave is limited [57]. Thus, if the processor is running inside the enclave, there must be an exchange between execution environments to read a data item (from the enclave to the untrusted part). Even if the data item is remote, there is also an exchange of execution environments, as the untrusted part is responsible for communicating with remote machines [57].

The set of auditable files is known to both the auditor and audited node and sorted in a deterministic manner. Thus, both parties can use the index of a file in the sorted list as a mutually agreed short unique identifier for that file. We denote this index by *set index*. However, agreeing on the set of files and supporting file modifications is beyond the scope of this work. For simplicity, files assigned to a fog node, including their content, do not change. The modification of files can be trivially supported using versioning.

3.3.3 Enclave Geolocation

The goal of PoTR is to check whether a given node stores data locally. When an enclave is attested, it is possible to uniquely identify the specific enclave, but attestation does not reveal its location. Therefore, PoTR alone cannot ensure that data is stored at a given target location; for

that, one also needs to geolocate the node. Several geolocation techniques have been proposed in the literature [50, 185, 1] and any of them can be combined with PoTR to ensure data locality *and* geolocation. Although geolocation proofs are orthogonal to our work, we briefly sketch two methods to discover the location of the enclave: 1) *Proximity Attestation* – the auditor physically launches the enclave in the local machine at the correct location and exchanges a certificate with that enclave for later authenticating the same specific machine/enclave; 2) *Triangulation* – any technique of triangulation can be applied without requiring heavy cryptographic operations (the goal is to geolocate the machine and not data), so the accuracy is not compromised in any way by the storage proof.

3.4 Proof of Timely-Retrievability

We now introduce the *Proof of Timely-Retrievability* (PoTR) mechanism: a storage proof that aims to assess whether a storage node can access stored data with a latency smaller than some specific threshold δ. The auditor can select the value of the δ parameter based on the specific requirements of a given application, but it is typically small and, in some cases, can only be satisfied if the audited (edge) node keeps the data locally. With the goal of applying the proof in edge computing scenarios, we assume small values of δ, in the order of the time required to read a data block from the local disk. By supporting such strict values of δ, PoTR is capable of distinguishing the case in which the fog node stores the data locally (Figure 3.1a) from the case it keeps it in some remote node or cloud (Figure 3.1b).

The PoTR is obtained by the storage node in response to an audit, so our solution requires a machine with auditing capabilities. We do not restrict the placement of these auditing machines, as there may be many geographically distributed fog nodes [138]. Therefore, our proof was designed to be obtainable from an audit machine anywhere on the Internet. This property offers a lot of flexibility regarding the deployment of the auditor and allows a single auditor to perform audits on a large number of fog nodes.

Alternatively, it could be possible to use multiple auditors and combine our proofs with related work to achieve better accuracy. However, we strive to design our proof to rely on a single auditor for easy, efficient, and cost-effective deployment and execution of our PoTR. Another alternative could be to ask clients to report the latency they observe and use this information to perform the audit. However, this approach raises many privacy challenges; an attacker could infer the client's location based on its latency to a fog node. If a PoTR can be extracted independently of the auditor's location, we avoid these limitations.

3.4.1 Challenges

In the design of our PoTR, we face some obstacles, namely: i) the timing information provided by the audited node cannot be trusted [11], thus the time to produce the proof must be measured by the auditor; ii) the network between the auditor and the audited node is subject to variance that introduces errors when estimating the time the audited node took to produce the proof; iii) storage/fog nodes are heterogeneous [138], and the time they require to perform computations and read data (even if the data is local) is not constant, so the proof should be based on average values from multiple readings; and iv) the audited node may attempt to delegate the generation of the proof to another node that has faster access to the data than the

audited node itself, so it is required to ensure the proof is produced by the audited node, and not delegated.

3.4.2 Design of the Challenge

The challenge requires the untrusted part of the fog node to access a given number of data objects, in a certain sequence, and return, at the end, a value related to these data objects. *The delay the fog node takes to read these data blocks, and to compute the final value, is used by the auditor to estimate the reading delay observed at the audited node and to check if it matches the target threshold δ.*

Each challenge (implicitly) specifies a sequence of files that must be accessed by the audited node, and each file is uniquely identified by a *set index*. For efficiency reasons, the fog node for each file reads a data block of size s_b, instead of the entire file. In practice, the block size should be a multiple of the block size used by the fog node file system.

In each challenge c, the fog node has to read a pseudorandom and unpredictable sequence of N data blocks (each of size s_b), and return a cryptographic hash of the concatenation of all data blocks accessed. The number N of data blocks is a configuration parameter that influences the accuracy and efficiency of the challenge: the higher the value N, the more accurate but less efficient the proof. The approach to configure this parameter is explained in Section 3.4.4, and if the user is unable to configure s_b, it is still possible to execute our challenge defining only N, as discussed in Section 3.5.3.

The pseudorandom sequence of the data blocks is determined by a nonce η^c (unique per challenge, and generated by the auditor) and the content of the data blocks. For each challenge, the auditor sends the nonce (encrypted with a symmetric key), the number N of data blocks and the size s_b of a block to the enclave. To ensure that the nonce η^c is not disclosed to the untrusted part, the nonce never leaves the enclave. The untrusted part only has access to a cryptographic hash of the nonce, we denote this hash as *#index*. With the *#index*, the untrusted part is able to determine the set index of the first file to be accessed. The set index is determined by applying the modulo function (*mod*) to the hash with the total size of the set of files, i.e., the set index is the remainder of the division of the hash by the number of files.

Since the fog node has to read a data block of size s_b, and not the entire file, the auditor sends to the enclave a second nonce η_b^c that will determine a data block inside the file, with the computed set index. The enclave, in turn, computes the cryptographic hash of this second nonce and forwards the result to the untrusted part. Both nonces are hashed by the enclave, to ensure that the untrusted part does not know them in plain text. Otherwise, the untrusted part could maliciously predetermine the N data blocks. Now, with the hashed η_b^c, and the total size of the file to be accessed, the untrusted part can determine the data block, with size s_b, inside the file to be retrieved, by applying the modulo function.

When the untrusted part receives the first *#index* from the enclave, it proceeds to determine and read the corresponding initial data block. It then calculates a cryptographic hash of the data block's content, combined with the *#index*, resulting in *result_hash*. Subsequently, the untrusted part returns the value of *result_hash* to the enclave. The enclave, in turn, computes a new *#index* by hashing the response *result_hash* together with the nonce η^c. This newly generated *#index* is then forwarded to the untrusted part, which uses it to identify the next file.

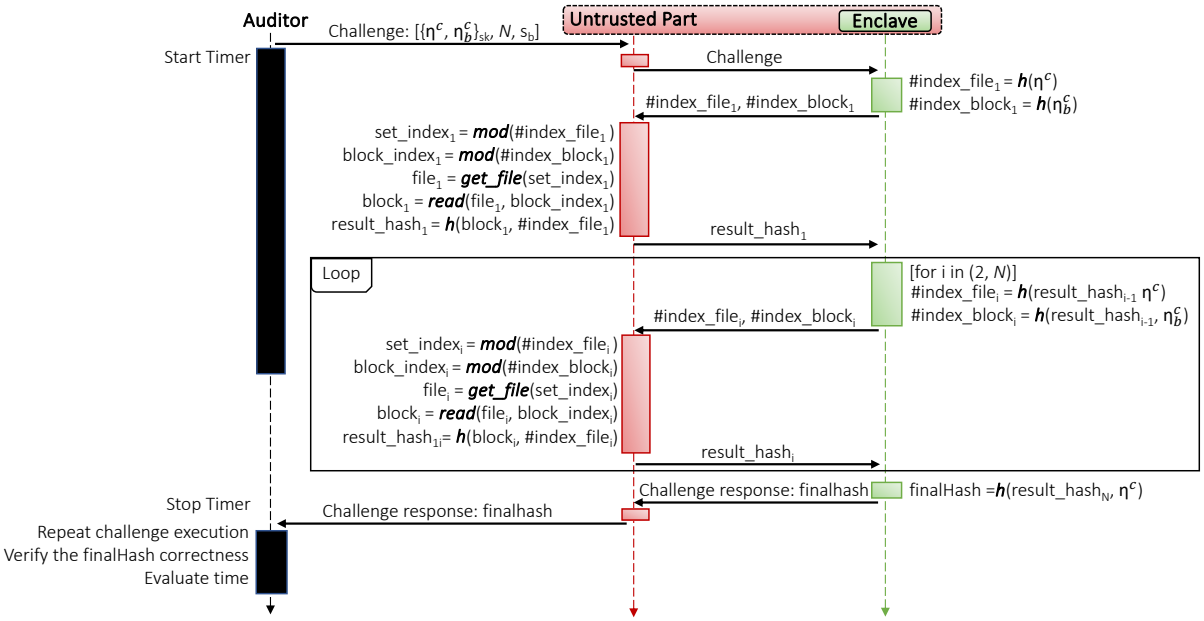


Figure 3.3: PoTR challenge execution. set_index_i is the index of a file, and $block\_index_i$ is the index of a data block with size s_b inside $file_i$. $block_i$ is the read data block. sk is a secret key between the auditor and the enclave. $\#index$ is a hash.

For each subsequent hash, the untrusted part repeats the execution of the modulo function to obtain the next set index and the index of the data block within the identified file. It reads the corresponding data block and calculates its cryptographic hash together with the file $\#index$, yielding *result_hash* that is returned to the enclave. The enclave computes two new hashes (block and file index), and this process continues until all N data blocks are read. Note that each new $\#index$ is computed using the hash of the previously retrieved data blocks. Finally, the enclave computes a final hash using η^c and sends the result back to the auditor as the conclusive proof. The auditor, upon receiving the final hash, repeats all the aforementioned computations to construct a verifiable version and verify the correctness of the proof. If both computations match, the issued proof is deemed correct. In Figure 3.3, we present in detail all the steps of our communication protocol in the execution of our PoTR challenge. We separate the steps on the auditor's side between the untrusted part and the enclave, where the sensitive parts of our challenge, such as the nonce, are always securely stored inside the enclave.

Security Guarantees: Each challenge requires two unique nonces that will determine in a sequential and pseudo-randomly faction which N data blocks the untrusted part must read. This interaction with the enclave ensures that the fog node is unable to retrieve the N data blocks in parallel from neighboring nodes. Instead, it must retrieve one block at a time to determine the next block it needs. Additionally, the dependency between data blocks, since the next block index depends on the content of the previous one, ensures that the fog node cannot reuse outputs from previous challenges, thus maintaining challenge freshness. The constant exchange between execution environments (enclave to untrusted part and vice versa) guarantees that the fog node cannot rely on a remote machine for the entire proof computation. The proof is computed solely on the expected machine [128]. It is important to note that through the verification of proof correctness, the auditor can assess the integrity of the stored documents. Any modification to a file will render the proof invalid.

3.4.3 Reading Delay δ at the Audited Node

In addition to checking *proof correctness*, the auditor has to check the *proof timeliness*. When the challenge is sent to the enclave of the audited node, the auditor starts a timer that is stopped when the proof is received. A proof is valid if it is correct and the reading delay estimation, for a single data block, is acceptable to the auditor.

Having T_i as the time elapsed between the auditor sending the challenge i and receiving the response from the fog node, T_i can be decomposed into different factors, namely:

$$T_i = rtt_i + \alpha_i^1 + \ldots + \alpha_i^N + \delta_i^1 + \ldots + \delta_i^N \tag{3.1}$$

where rtt_i is the network round trip time for challenge-response messages, N the number of data blocks to be accessed, α_i^j the delay observed at the fog node to compute the cryptographic hash of a given data block j and compute the index of the next block, and δ_i^j the delay observed to read a data block j. Note that for sufficiently large values of N, we will have the following:

$$\frac{\alpha_i^1 + \ldots + \alpha_i^N}{N} = \overline{\alpha}_i \approx \overline{\alpha} \qquad \frac{\delta_i^1 + \ldots + \delta_i^N}{N} = \overline{\delta}_i \approx \overline{\delta} \tag{3.2}$$

However, the auditor is unable to measure accurate values for rtt_i, $\overline{\alpha}_i$ and $\overline{\delta}_i$, as they are different and variable in each challenge i and the auditor is only able to measure the total delay T_i. Therefore, to estimate $\overline{\delta}$, i.e., the actual mean delay a fog node takes to read a data object, in our work we resort to mean values:

$$T_i = \overline{rtt} + N\overline{\alpha} + N\overline{\delta} \tag{3.3}$$

$$\overline{\delta}_{\text{estimate}} = \frac{T_i - \overline{rtt} - N\overline{\alpha}}{N} \tag{3.4}$$

Therefore, the estimate error for a given challenge i will depend on how far the observed values are from the mean values. The $\overline{rtt}$ is calculated independently of our challenge, being obtained by measuring several network samples and dividing them by the number of samples. Experimentally, we verified that the α values are subject to a negligibly small variance, whereby we assumed $\overline{\alpha}_i = \overline{\alpha}$, i.e., we ignore the error introduced by α sampling. Therefore, the error of the estimate $\overline{\delta}_{\text{estimate}}$ depends on two factors: i) the reading error ε^{δ} when estimating $\overline{\delta}$ (which depends on the sample size, i.e., the number of data blocks accessed) and ii) the variance between the observed network round-trip time (rtt_i) and the expected mean value ($\overline{rtt}$), divided by the number N of data blocks, as Equation 3.5 describes:

$$\varepsilon^{\text{rtt}} = \frac{\left| rtt_i - \overline{rtt} \right|}{N} = \frac{\Delta^{rtt}}{N} \tag{3.5}$$

3.4.4 Configuring the Challenge

The challenge can be configured by providing two parameters: the maximum error $\varepsilon_{\max}^{\overline{\delta}}$ that he is willing to tolerate, when estimating $\overline{\delta}$, and the reliability of the challenge ϕ, a parameter

specified by the auditor that captures the probability of estimating $\overline{\delta}$ with an error lower than $\varepsilon^{\overline{\delta}}_{\max}$ (in practice we use the 99.99 percentile). As noted before, the error of the estimate $\varepsilon^{\overline{\delta}}_{\max}$ results from the variance of local reads experienced by the audited node ($\varepsilon^{\delta_i}_{\max}$) and by the variance of the network delays ($\varepsilon^{\text{rtt}}_{\max}$), i.e., $\varepsilon^{\overline{\delta}}_{\max} = \varepsilon^{\delta_i}_{\max} + \varepsilon^{\text{rtt}}_{\max}$. For both sources, the error diminishes with the number of samples N, thus, N should be chosen so that $\varepsilon^{\text{rtt}}_{\max} + \varepsilon^{\delta_i}_{\max} < \varepsilon^{\overline{\delta}}_{\max}$.

Let $\varepsilon^{\delta^j_i}$ be the difference between the average access latency $\overline{\delta}$ and the latency observed when reading block j during the execution of challenge i. The error caused by the variance of storage access in a given challenge i is given by:

$$\varepsilon^{\delta_i} = \frac{\sum_{j=1}^{N} \varepsilon^{\delta^j_i}}{N}$$

Given ϕ, the expected value of $\varepsilon^{\overline{\delta}}_{\max}$ can be derived from the distribution of the access delays. Similarly, with the inverse cumulative distribution function (ICDF) of the network distribution, we can also calculate the expected maximum difference between the observed value rtt_i and the expected mean $\overline{rtt}$ that matches ϕ.

3.4.5 Network Delay Distribution

Our proof is capable of finding the correct $\overline{\delta}_{\text{estimate}}$ even if delay distributions are unknown, by choosing worst-case default values that can be used conservatively (see Section 3.5.3). However, it is possible to leverage information regarding the network delay distribution to select a more efficient N value such that the sum of both errors is kept below a target value of $\varepsilon^{\overline{\delta}}_{\max}$. We show this in the concrete edge node scenario in Section 3.5.4.2. This optimization, based on measuring network behavior, is safe and cannot be exploited by a malicious provider. The provider can introduce artificial delays when replying to the auditor, but this will not reduce the variance that is introduced by the network itself; on the contrary, it may only increase the variance. In turn, this may force the auditor to use larger values of N than strictly required, increasing the accuracy of the proof at an extra cost to the malicious provider. Therefore, it is always beneficial for the adversary to be honest when the network behavior is measured.

3.5 Evaluation

We now present our proof evaluation. We begin by describing our experimental setup and the different benchmark scenarios. Then we show that our proof works even without knowledge of network distribution. Afterward, we discuss how the challenge should be configured, for the block size, and N samples. After finding the correct configuration, we evaluate the performance of the challenge in the different scenarios and show experimentally that our proof can accurately distinguish a misbehaving server from a correct server, in the context of edge computing. Finally, we evaluated different alternatives to reduce the impact of our proof on the audited node.

3.5.1 Experimental Setup

We resorted to two different types of machines: 1) Intel NUCs to represent fog nodes, and 2) virtual machines in the Windows Azure cloud to represent remote entities. For NUCs, we have

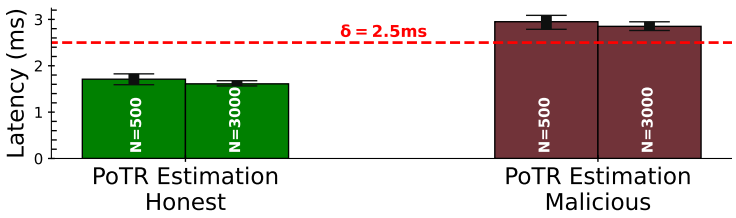


Figure 3.4: Overview of PoTR δ estimative without network knowledge, picking a large N value of 3000, 64KB block size.

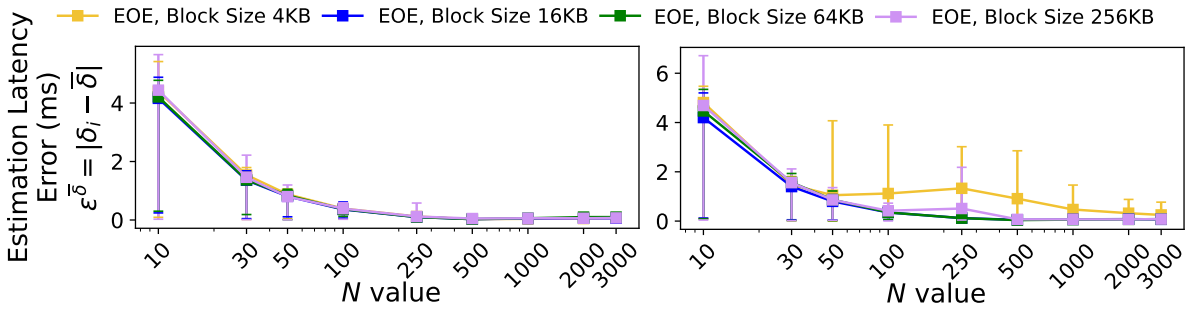
resorted to the Intel NUC10i7FNB, it has an Intel i7-10710U CPU that supports Intel SGX, 16GB RAM, 240GB SSD M.2, and Ubuntu 20.04 LTS. We run the Intel SGX SDK Linux 2.13 Release and OpenSSL 1.1.1k. An Intel NUC is an example of what a fog node might be, as it possesses modest computational resources but is relatively inexpensive for large-scale deployments. For the cloud deployment in Azure, we resort to Standard D2ads v5 with two vcpus, and 8 GB RAM in west Europe (Netherlands) the closest data center to our laboratory; our laboratory is located in a European capital. We leverage these two types of deployment to create different scenarios and evaluate our storage proof; we explain our scenarios in more detail in the next section.

For the stored data that our proof is responsible for auditing, we generated 150GB with random data divided into files of 1GB, our proof leverages a hash function to choose the next block to be accessed; this offers uniform distribution access over the 150GB of data. In our code implementation, we used the SHA-256 function as a cryptographic hash function and AES in GCM mode with 128 bit keys to cipher the nonces, both currently considered secure [27, 111].

3.5.2 Evaluation Scenarios

In our experiments, we leverage three different entities to evaluate our PoTR storage proof: an auditor; the audited node, which represents a fog node that stores data; and the remaining one is the remote storage node, which is used by the audited node to access data in configurations where it does not store the files locally. In our experiments, the fog node stores all files at the same location, i.e., all files are stored locally, or all files are stored in the remote node.

We deployed four different scenarios to evaluate PoTR; the difference between the scenarios is the location of the remote storage node, since the closer the remote storage is to the audited node, the more difficult it is to detect malicious behavior. The first scenario is represented in Figure 3.1a: the fog node has honest behavior and stores all data locally. Figure 3.1b presents the malicious behavior that the edge provider can adopt by resorting to remote storage. Figure 3.1b represents the three remaining scenarios in which we vary the location of the remote storage from: a) a virtual machine on the Azure cloud; b) a NUC on a different campus of our university; c) a NUC located in our laboratory and connected through a switch to the audited node, with a mean network delay of $0.1ms$. Regardless of the scenario, both the auditor and the fog node/audited node do not change their location, the auditor is running in the Azure cloud, and the audited node in an NUC at our laboratory. By deploying different machines at different geographic locations, we capture the different variations and delays of wide-area links.



(a) Honest Fog Node. (b) Nearby Fog Storage.

Figure 3.5: PoTR experimental error for different scenarios and block size, EOE is Experimentally Observed Error.

3.5.3 PoTR Without Network Knowledge

As stated in Section 3.3.1, the efficiency of PoTR can be improved given knowledge on the distribution of network delays and access delays. When executing PoTR without this knowledge, a large default value for N should be used, enough to compensate worst-case network variance; we show that a default value $N = 3000$ can be safely used in most scenarios. We have selected this default value based on latency measurements from our laboratory to the farthest Azure data center[1], the Australia Central 2. We have observed an average RTT of $\approx 286ms$ and a maximum latency of $\approx 1143ms$. When considering a network variance with this order of magnitude in Equation 3.5, by setting $N = 3000$ we can reduce ε^{rtt} to $\approx 0.28ms$. In this scenario, the variance in storage access delay is negligible compared to the network variance.

Figure 3.4 shows the results of running PoTR using the default value of $N = 3000$ in different scenarios. We can observe that it is possible to set the SLA threshold to a small value such as $\delta = 2.5ms$, to ensure that the audited node has the data stored locally (the local delay to read a data block is $\approx 1.7ms$), and that our $\overline{\delta}_{\text{estimate}}$ achieves high accuracy in both scenarios.

However, with $N = 3000$ our challenge takes approximately $5.2s$ to execute. When the challenge is executed using more stable networks, it is possible to fine-tune the value of N to achieve similar accuracy with a lower cost for the audited node. In the next section, we demonstrate results for various values of N. For instance, with $N = 500$, we achieve accurate results with a challenge executed in just $0.9s$.

3.5.4 PoTR Configuration

We now derive, based on experimental results, the best configuration for our storage proof, by setting the block size and the N value for PoTR. Note that in a situation where the user is unable to configure the block size, they can simply use a high value for N, as demonstrated in the previous Section 3.5.3.

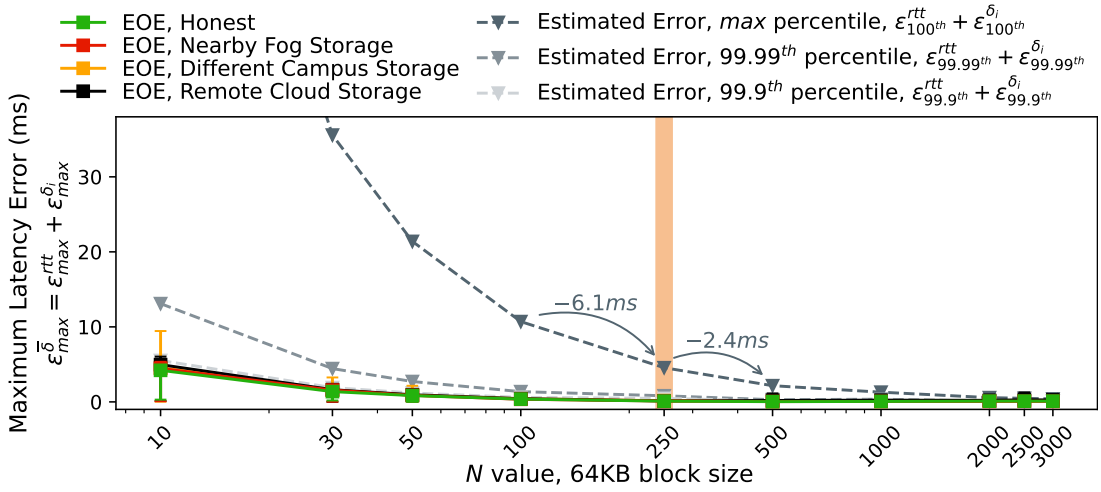


Figure 3.6: Theoretical and experimental error for the PoTR challenge (EOE means Experimentally Observed Error).

3.5.4.1 Selecting the appropriate Block Size

To find the appropriate value for the size of the block to use in our challenge, we evaluated the performance of the challenge with different block sizes and different N values. The results are presented in Figure 3.5. We executed our challenge to estimate the δ value and calculated the error relative to the real reading delay in the fog node, we present the results for the honest fog node scenario in Figure 3.5(a) and for the nearby fog storage scenario in Figure 3.5(b). We observe that for small values of N the results obtained exhibit a large variance and only by increasing N do we obtain more accurate results. In both scenarios, we observe that the size of the block has a relatively smaller impact on the accuracy of the result than the value of N. Still, it is possible observe that, when using larger block sizes one can obtain more accurate results. This is clearly noticeable in the scenario where files are stored in a nearby fog node, where the configuration using blocks of 4KB takes longer to converge than configurations with larger block sizes. Furthermore, in Figure 3.5(b), it is possible to observe that when the N value goes above 100, blocks of 64KB offers a slightly smaller error than blocks of 16KB. For these reasons, we opted to set our challenge with blocks of 64KB.

3.5.4.2 Finding the appropriate value for N

As described in Section 3.4.4, the value N can be determined by providing $\varepsilon^{\overline{\delta}}_{\max}$, which depends both on the network error and the reading error, this network knowledge allows us to find an optimal value for N. Figure 3.6 presents the estimated error for our challenge under different percentile values, of the reliability ϕ, for each error ε^{rtt} and ε^{δ_i} (these two were measured experimentally). Note that $\varepsilon^{\overline{\delta}}_{\max}$ decreases as we increase the value N, this results from the impact that N has on ε^{rtt} reducing its effect on the challenge, on the other hand, ε^{δ_i} has a negligible impact on $\varepsilon^{\overline{\delta}}_{\max}$, ε^{δ_i} is mostly below $1ms$ reaching at most $5ms$. Figure 3.6 also presents the measured error of our challenge in real experiments in the four different scenarios presented

[1]We leveraged this website: `https://cloudpingtest.com/azure`

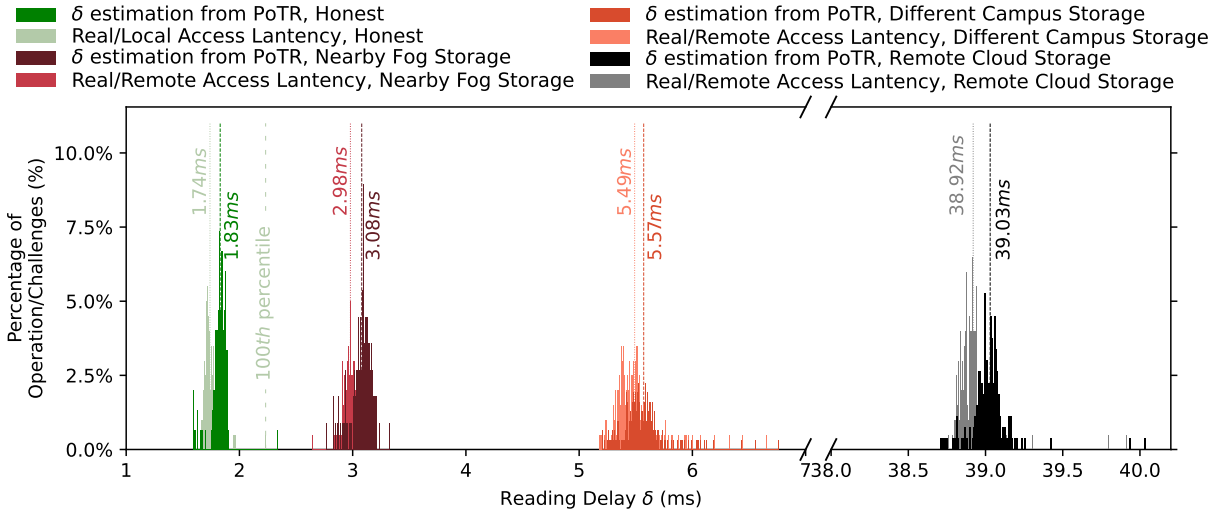


Figure 3.7: Distribution and correspondent averages for reading delay δ, in milliseconds, for different scenarios where PoTR is configured with $N = 250$ and 64KB block size.

earlier. We observe that the error increases when we also increase the distance between the data and the audited node.

Both ϕ values of 100^{th} and 99.99^{th} percentiles are able to predominantly capture the real observed error of our challenge. We use the estimated error to choose a desirable value for N; in this case, we chose $N = 250$ since this is where we observe the last most significant reduction in the estimated error of $-6.1ms$, versus the increase in the value of N, of 150. As expected, increasing the value of N also improves the accuracy of our challenge, for $N = 10$ we obtain an error of $\approx 5ms$, which can be undesirable for edge environments, while for $N = 250$ the observed error is only $\approx 0.1ms$, such a small error can provide a $\overline{\delta}_{\text{estimate}}$ with high reliability.

3.5.5 PoTR Accuracy

Using the configuration parameters derived from the previous analysis, i.e., using a block size of 64KB and by setting the number of samples N to 250, we have executed our challenge in multiple scenarios and captured the distribution of the data access latency estimated by our challenge. We compare the estimated values with the real values, observed at the audited node itself. The results are depicted in Figure 3.7. Each point in the figure represents either the real latency to access a data block or the $\overline{\delta}_{\text{estimate}}$ to access a block resulting from the execution of our challenge.

It is possible to observe that our challenge estimate closely approximates the real value with a small error, as discussed in Section 3.5.4. The error between the real value and $\overline{\delta}_{\text{estimate}}$ increases as the audited data move further from the audited node. This happens because the farther away the audited node is, the more likely the network delay exhibits a larger variance. Nonetheless, our challenge can still distinguish whether the data is local to the fog node or at a remote site. The accuracy of the PoTR is sufficient to differentiate between scenarios where files are stored locally or at a nearby fog node. Even with a latency difference of less than $1.5ms$, our PoTR accurately identifies the configuration used by the audited node.

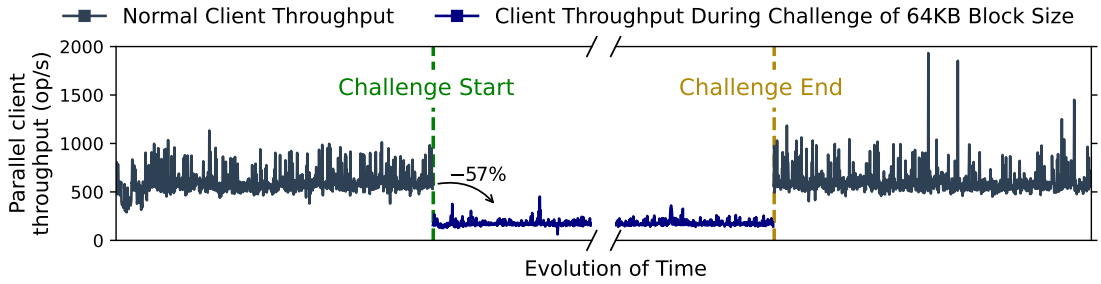


Figure 3.8: Challenge impact on concurrent reads.

Looking at the results in Figure 3.7, our PoTR enables the establishment of a threshold to accurately differentiate between a correct node (storing all files locally) and a faulty node that stores files elsewhere (even if they are kept in nearby nodes). This threshold is determined by setting a very strict value for δ in the SLA, such as the 100^{th} percentile for reading delays measured locally at the fog node. The 100^{th} percentile line, located at $2.2ms$, corresponds to 0.005% false positives and 0% false negatives (we define as positive the detection of malicious behavior of the audited node).

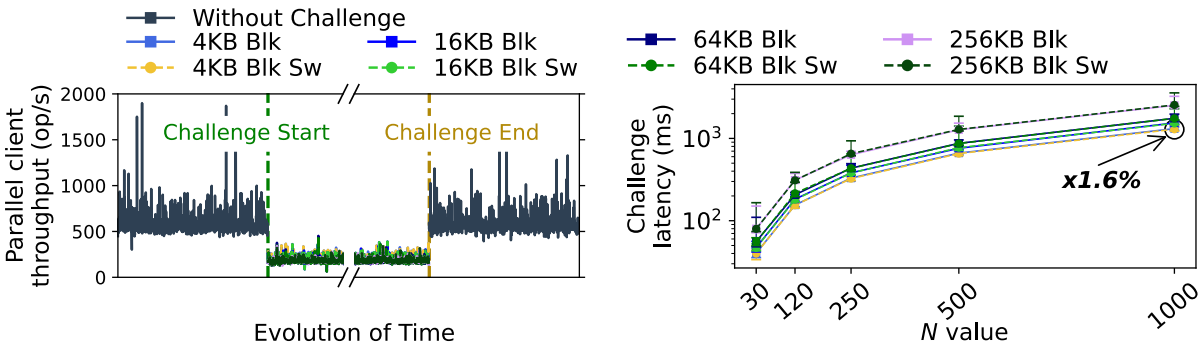
3.5.6 Overhead Imposed by a PoTR Challenge

We now assess the overhead imposed on the audited node while it responds to a challenge. For this, we run a local client that performs read/write access to the data stored on the edge node, and then we measure the loss in throughput of that client during challenge execution. The results are shown in Figure 3.8.

The throughput of the concurrent client decreases by approximately 57% during the processing of the challenge by the audited node. This is expected as our resource-constrained edge nodes need to access multiple files and compute their digest to respond to the challenge. Despite this non-negligible overhead, we argue that its impact on the overall operation of an edge node is not significant in practice. The challenge typically takes less than $500ms$ to complete, and auditing occurs sporadically in most scenarios (e.g., once per day). Nonetheless, in the next section, we discuss and evaluate strategies to further reduce this overhead.

3.5.7 Strategies to Reduce PoTR Overhead

We considered three complementary avenues to attempt to reduce the overhead of running a challenge in an audited node. The first was to1modify the implementation, taking advantage of switchless calls [179] that reduce the ECall/OCall overhead. The second was to use blocks size smaller than 64K. As discussed earlier, smaller sizes can reduce the accuracy of the test but can have an impact on the overhead. Finally, one can simply reduce the number of samples N. By reducing the number of samples, one would reduce the challenge duration and therefore, the period during which clients would be affected by the concurrent execution of a challenge. However, as we show in this section, using switchless calls or reducing the size of the blocks has a minor impact on the overhead.



(a) Throughput Impact.

(b) Challenge Duration.

Figure 3.9: PoTR duration and throughput impact on concurrent reads, Blk is block, and Sw is switchless.

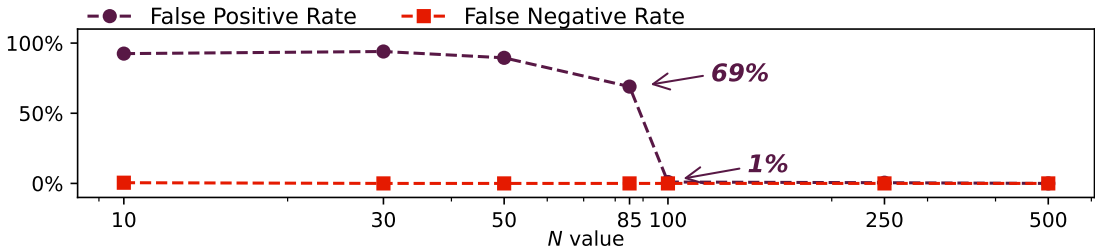


Figure 3.10: False positive and negative rates of PoTR under different N values for honest and nearby fog node scenarios.

Figure 3.9 shows how the use of switchless calls, different block sizes, and different values of N, affect the drop in throughput observed by clients during the execution of the challenge and also the duration of the challenge.

It is clear from Figure 3.9a that the drop in throughput during the execution of the challenge is roughly the same regardless of the block size used or the use of switchless calls. Switchless calls offered a small performance improvement, just a latency reduction of $\approx 1.6\%$ in the best case. When considering block size, the largest difference is observable from 256KB block size, with a throughput reduction of -55%, versus 4KB block size alternative, with a reduction of -38%, relative to the parallel client. This is not a large improvement considering that the 256KB block size is 64 times the 4KB block size. This is because, during the execution of our challenge, our disk access is continuous, independently of the block size, and therefore the impact on other clients will be similar; however, our challenge duration may vary. Additionally, as discussed in Section 3.5.4.1, having a very small block size can still affect our storage proof error.

These results suggest that the only effective way to reduce the overhead of the challenge is to reduce its duration. Figure 3.9b shows that, although block size has some impact on the duration of challenge executions, its impact is relatively minor compared to the effect of reducing the number of samples N. This indicates that the most effective strategy to reduce the overhead of the challenge is to reduce the number of samples, as this reduces the duration of the period in which clients are affected. Of course, as discussed before, reducing the value of N will degrade the accuracy of the test, increasing the chances of producing false positives and false negatives.

Figure 3.10 shows the effect of N on both false positive and false negative rates when using the threshold $\delta = 2.2ms$, as discussed in Section 3.5.5. In this analysis, we consider the scenario where a faulty node stores data at the nearby fog storage, as this is the scenario where it is most challenging to accurately detect a misbehavior, given the small difference between the latency offered by correct and faulty nodes. It is interesting to observe that our PoTR offers a false negative rate close to 0% for most values of N. Therefore, even where the values of N are smaller than 250 (the required to obtain the best accuracy), our PoTR will detect a misbehaving client. Unfortunately, the true positive rate can increase significantly when using low values of N, risking identifying a correct node as faulty. More precisely, the false positive rate increases slowly as we decrease the number of samples from 250 to 100 (at this point, our PoTR offers a false positive rate of approximately 1%), but then increases sharply. For values of N smaller than 100 the challenge no longer provides acceptable values for the false positive rate.

In summary, by tolerating a small fraction of false positives, the auditor can reduce the challenge overhead by decreasing the number of samples from 250 to 100. This roughly halves the period during which clients are affected by our challenge.

3.5.8 Combining Multiple PoTR Configurations

As seen in the previous section, using a reduced number of samples, 100 instead of 250, our challenge imposes significantly less overhead while still obtaining a negligible false negative rate and a very small positive rate ($\approx 1\%$). Faced with these observations, we conclude that in some deployments the auditor should combine different configurations of the PoTR: one that is cheaper and works 99% of the time; another that is more expensive but provides extreme accurate results.

For example, the auditor might use $N = 100$ to challenge the audited node. If the node was flagged as malicious (a positive), the auditor could then launch a second challenge with $N = 250$, to filter out false positives. Note that in such a case, the combined challenge would read on average 102.5 ($0.01 \times 100 + 0.01 \times 250 + 0.99 \times 100$) blocks instead of the 250 in the normal auditor, a reduction of 41% on average for the number of blocks required to execute our challenge. Notice that, as hinted before, for $N = 85$ this strategy would no longer be worth it since it would require reading 257.5 ($0.69 \times 85 + 0.69 \times 250 + 0.31 \times 85$) blocks on average, which is larger than the 250 required by the normal auditor solution.

Summary

In this chapter we presented our novel auditing mechanism that is capable of extracting a proof of timely-retrievability, that is, a proof that a given storage node is able to serve requests without violating some given data access latency constraint δ. The proof is designed in a way that, if the storage node does not store locally a significant fraction of the objects, it will be unable to respond in time. We enforce our proof to be executed in the audited node by leveraging SGX enclaves, for iteratively revealing the next data block to be read to the untrusted part.

Our evaluation shows that our proof can accurately detect a node that cannot satisfy the target latency constraint δ under different edge computing scenarios, resulting in the detection of a misbehaving storage provider.

The significance of auditing tools for cloud storage has been well-established, enabling clients to utilize cloud storage providers without complete trust. With the advent of new edge storage solutions managed by numerous local providers, clients have even more reasons to be cautious with these new providers. Therefore, the development of auditing mechanisms for edge storage, such as our PoTR, is increasingly vital. We are confident that this contribution will enable the implementation of diverse edge storage systems more reliably and transparently, thus promoting and easing their adoption.

In the next chapter, we introduce the RRPs, an anonymous authentication scheme that offers backward unlinkability. RRPs present a novel cryptographic scheme enabling edge clients to perform authentication while preserving their privacy. The RRPs scheme is constructed solely based on public key encryption, achieving the low latencies required for the authentication process in edge environments.

III

Anonymous Authentication in the Edge

4

Providing Backward Unlinkability: RRP

This chapter introduces and evaluates a novel abstraction named Range-Revocable Pseudonyms (RRPs), an anonymous authentication scheme that provides effective backward unlinkability based solely on public key encryption.

The remainder of this chapter is structured as follows: Section 4.1 provides the motivation and outlines the goals for this contribution, while Section 4.2 reviews related literature, and Section 4.3 describes our assumptions. Section 4.4 presents the RRP algorithm, and Section 4.5 offers an extensive proof of our scheme's unlinkability. Section 4.6 proposes an implementation of our scheme in an access control system tailored for the edge environment. Finally, Section 4.7 presents experimental results to demonstrate the practicality of our proposed scheme.

4.1 Motivation and Goals

Anonymous authentication offers both accountability and privacy, protecting clients from curious application providers while ensuring that only authorized participants are able to use the application [156, 106, 113, 174]. The number of edge applications requiring anonymous authentication is on the rise, and many of these applications will depend on edge storage. Examples include crowdsensing [88, 151, 144, 173] and Vehicular Networks (VANETs) [158, 87, 32], which may need to access local maps or publish information about their environment for the common good. To ensure the reliability of edge storage systems and subsequently these edge applications [155, 88], the client authentication is a crucial mechanism to provide accountability for malicious and erroneous activity. Unfortunately, as previously discussed in Section 2.3.2, authentication can compromise *user privacy*, as it may be associated with sensitive information, such as location [151]. This is exacerbated by the fact that, in most of these applications, clients are mobile and may need to *authenticate frequently*, e.g., when they move to the range of a different base station or cell. Multiple authentications may be linked to extract additional information such as daily routines [88] or health status [121] for financial gain [132, 51, 119, 18, 184, 34]. Anonymous authentication can be achieved using Group Signatures (GS) schemes [48, 37, 173] or pseudonym certificates [126, 47].

A challenging task in this context is to support *revocation* without violating privacy. Revocation aims to prevent some clients from further authenticating in the system. Client revocation may be required in the event of credential misuse, sensor malfunctioning, change in client privileges, stolen secret keys, or when a client leaves voluntarily. Client revocation can be implemented in

different ways. We distinguish two main classes of revocation strategies, namely, *global client revocation* and *verifier local revocation*.

Strategies based on global client revocation require all clients to obtain new credentials (or update their credentials) every time a single client is revoked. Examples of this strategy include Ateniese *et al.* [17] (where the group public key is renewed at each revocation) and Ohara *et al.* [147] (where a small public membership message is broadcast at each revocation). These approaches make revocation very onerous in scenarios with many clients (e.g., consider vehicle numbers in VANETs) and impractical in mobile settings, where clients may become temporarily disconnected from the network.

Strategies based on Verifier Local Revocation (VLR) [37, 42] do not require that all clients are contacted when a given client is revoked. Instead, only the nodes that perform authentication (often called the signature *verifiers*) have to be informed about the revoked clients [165, 8, 112, 174, 106, 173]. In systems that use pseudonyms, this involves sending to the verifiers a Certificate Revocation List (CRL) with the pseudonyms of the revoked client. In systems based on group signatures, this involves sending a cryptographic token that can be used to trace the digital signatures of the revoked client.

As describe in Section 2.3.2, a problem with with these approaches is that, if one or more credentials have been used before revocation, an attacker can cross-check the information used for revocation with the information collected when those pseudonyms were used to break the privacy of the client. In order to respect *backward unlinkability* [95, 113, 106, 139], client revocation should not allow linking credentials that have been used prior to the revocation. Previous strategies to provide backward unlinkability assign credentials that are valid only during a given time slot of a certain duration [95, 113, 156, 173]. Then, when a client is revoked, only the credentials for future time slots are revoked and no information is disclosed regarding credentials used prior to revocation. One can divide these recent strategies as GS with time-bounded keys [52, 76] or pseudonyms with time slots [95, 113]. However, these schemes require the use of revocation lists whose size grows linearly with the number of time slots which, in practice, preclude the use of fine-grain time slots.

Revoking only the credentials for future time avoids backward linkability but, unfortunately, if time slots are large, it may be unacceptable to let revoked clients continue accessing resources until the current slot expires. For this reason, many systems immediately revoke the credentials for the current time slot, at the expense of exposing the client's privacy during that period [95, 113, 106, 139]. Our new class of pseudonyms supports efficient revocation even when fine-grain time slots are used, avoiding this dilemma.

In this section, we present our new anonymous authentication scheme, Range-Revocable Pseudonyms (RRPs), which has the following objectives:

Goal 1: Propose a new class of pseudonyms, RRPs, that can be revoked for any time-range within their original validity period while respecting backward unlinkability. Clients hold a number of RRPs that is proportional to the number of authentication actions they need to perform, regardless of the granularity of the linkability window. Each RRP should be used at most once during its lifetime. In runtime, the RRP can be used to generate a capability that is only valid for the specific time slot where the RRP is being used. The key feature of RRPs is that the information provided to revoke a pseudonym for a given time-range cannot be linked with the information provided when using the pseudonym outside the revoked range. In particular, if a pseudonym is revoked at some point in time, it is impossible for an attacker to find out if that

pseudonym has been used before that time. We provide an algorithm to implement RRPs where the space complexity of the pseudonym is constant, regardless of the granularity of the revocation range, and the space complexity of the revocation information only grows logarithmically with the granularity; this supports the use of fine-grain slots and makes the use of RRPs far more efficient than the use of many short-lived pseudonyms. We show that RRPs can be used to solve efficiently the backward unlinkability problem for anonymous authentication.

Goal 2: Demonstrate the feasibility of RRPs in an edge use case. We implemented an access control system, named EDGAR, that uses RRPs to offer backward unlinkability. EDGAR illustrates how edge applications, such as VANETs, can leverage RRPs to enforce access control to local edge storage or resources. In EDGAR, we deploy Pseudonym Manager (PM) servers that run on the edge of the network, serving clients with new RRPs. Since a PM server holds sensitive information, and the edge infrastructure is known to be exposed to attacks [199, 137], we have designed the PM server to be executed with the support of Intel SGX enclaves [129, 57, 143]. This allows the server to provide new RRPs to clients without disclosing their identities, even if the untrusted environment is compromised.

EDGAR paves the way for arbitrarily small time slots with minimal overhead. Consider a client who during the day goes to the hospital and before that to a nearby shop. When using EDGAR, clients only need a number of pseudonyms proportional to the number of resources they need to access (in this example, 2 resources), and not proportional to the granularity of the time slots. With previous work, if the two events above could occur within 20 minutes of each other, a client would require 72 pseudonyms; if the events could occur within 5 minutes of each other, previous works could require 288 pseudonyms. Also, with RRPs the cost of revocation is logarithmic with granularity: only 12 credentials would need to be revoked with a 20 minute granularity and only 16 credentials would need to be revoked with a 5 minute granularity.

4.2 Related Work

Backward Unlinkability has been defined in previous work [95, 113, 106, 139] as follows: *when a revocation occurs, the signatures produced by the client before the revocation interval remain anonymous.* The notion of unlinkability captures the inability of an adversarial server to link a revocation phase of the protocol to any individual signing phase. We are interested in non-blocking approaches such as VLR [37, 42], where the credential of non-revoked clients remain valid, and only the verifiers need to be informed about the credentials of revoked clients. As discussed next, the most popular anonymous authentication schemes that offer backward unlinkability are based on GS or pseudonyms. In both cases, solutions typically consist of assigning different credentials to different time intervals and then revoking only the credentials for future intervals. Unfortunately, in these previous works, the cost of revocation grows linearly with the granularity of the intervals. Note that if intervals are large, it may be unacceptable to wait for the next interval to revoke the credentials: in this case, it may also be necessary to revoke the credentials for the current interval. Unfortunately, this makes all the credentials used in the current interval vulnerable to being linked.

Anonymous Blacklisting is a term used to describe techniques that are able to safeguard the privacy of revoked clients. Techniques to ensure this goal include the use of pseudonyms [182], group signatures [171], accumulators [19], and zero-knowledge proofs (ZKPs) [180]. Most systems

that aim to offer anonymous blacklisting also aim at offering backward unlinkability [96], but either use computationally expensive cryptographic operations or also incur a cost that is linear with the granularity of the linkability window. For example, BLAC [180] relies on inherently computationally expensive ZKPs [96]. We avoid the use of ZKPs to implement RRPs due to their high cost; instead, we explore more efficient approaches.

Accumulators, Symmetric Keys, and IBE: Cryptographic accumulators [45] may be vulnerable to linkability [171], i.e., the previously performed operations become linkable when a user is revoked. These solutions also lack VLR since clients need to update their witness at each revocation [19, 44]. Credentials based on symmetric keys [32] require a high level of trust in the verifier and are susceptible to identity theft if the verifier is compromised [100]. Nymble [182] achieves backward unlinkability but requires a central manager to share a symmetric key with every verifier. Credentials based on Identity-Based Encryption (IBE) [36] do not provide anonymity (as they consider the user's identity as the public key) and incur considerable overhead due to expensive cryptographic operations.

Group Signatures with Time-Bound Keys: GS [48] allow different signatures produced by different group members to be verified using a common group public key, achieving anonymity in the set formed by the group members. GS schemes have been augmented with mechanisms to support VLR, as suggested by Brickell [41] and formalized by Boneh and Shacham [37]. Unfortunately, in this scheme, the revocation is performed by publishing a cryptographic token that links all the signatures produced from a revoked member, compromising the anonymity of signatures produced before the revocation. Nakanishi and Funabiki [139] extend [37] to offer backward unlinkability while preserving VLR. Their approach divides the time into slots and locks a different secret key for each slot, revoking only the keys for current and future slots. Chu *et al.* [52] introduce the notion of Time-Bound Keys (TBK) by setting a configurable expiration date in each key, improving the revocation performance in VLR-GS schemes. In recent years, different solutions have been proposed, following a similar path while aiming to reduce the revocation cost and complexity. LBR [171] requires a trusted online manager to check revocation. Rahaman *et al.* [156] embed pseudoIDs in private key parameters and ties the pseudoID to an epoch, improving the revocation check complexity to $\log(R)$, where R is the size of the revocation list. Emura *et al.* [76] propose an efficient solution with a constant signing cost, but clients are required to download expiration information at each time slot. In Sucasas *et al.* [173], the authors also achieve backward unlinkability, yet, their solution prevents clients from participating in the same task several times. Ishida *et al.* [106] leverage a mixture of IBE with GS, generating IBE private keys locked to time slots. However, their revocation is still based on the GS private key, also having $\mathcal{O}(R\ T)$ (where T is the number of time slots). Despite the interesting properties of GS schemes, GS solutions are usually complex and some may require heavy cryptography operations (such as ZKPs), resulting in few implementations and deployments in real-world scenarios.

Pseudonyms with Bounded Time Slots: Client anonymity can also be achieved by using pseudonyms. These can be implemented using a Public Key Infrastructure (PKI), where clients maintain multiple keys to represent pseudonyms [32, 165, 161]. Pseudonym-based solutions also struggle to offer backward unlinkability. Some solutions invalidate global information, forcing clients to renew credentials at each revocation [45, 87], failing to preserve VLR. V-token [165], IFAL [188] and PRESERVE [78] follow the C2C-CC standard [78], revoking only the long-term vehicles certificates and letting the pseudonyms expire, also failing the VLR. ACPC [169]

is inspired by IFAL, resorting to activation codes to enable valid certificates, requiring the publication of a typical revocation list only for currently activated pseudonyms. PrivacyPass [69] is an anonymous authentication scheme implemented in Cloudflare CDNs, unfortunately, no revocation technique is presented. PUCA [83] requires the owner of the pseudonym to trigger revocation, letting a misbehaving entity evade revocation. The most common solution is to publish all pseudonyms of the revoked client in a CRL [95, 113, 194, 174], respecting VRL but failing backward unlinkability. The challenge of maintaining the unlinkability of pseudonyms after revocation was first addressed by Haas *et al.* [95], followed by Khodaei *et al.* [113], and implemented in SCMS POC [194] and CAMP [46] pilots, supported by Volkswagen, Mazda, and Nissan. These solutions associate pseudonyms with time intervals and revoke only pseudonyms of the current and future intervals. However, all interactions in the current slot can still be linked and an adversary can use the revocation information to break anonymity [95].

Pseudonyms at the Edge: VANETs serve as an exemplary use case to underscore the imperative of privacy in edge storage systems. Vehicles within VANETs produce and disseminate vast amounts of data, which necessitates storage and processing at the network's edge. Consequently, safeguarding privacy is of paramount importance for edge storage systems that support VANET operations. Presently, private entities are investigating ways to capitalize on user data [184, 34, 189], frequently at the cost of user privacy. Projections indicate that the value of this market could ascend to a staggering $750 billion globally by 2030 [184, 34]. For example, unethical edge providers [177, 18] may sell user data to insurance companies, that can subsequently tailor insurance plans based on individual driving habits [141]. Car-sharing and rental agencies can exploit user data with the same purpose [85]. An attacker could also gain access to user data in the edge infrastructure [67, 25, 65, 154, 101, 73], inferring if the certain individual is out of the household or has been attending the hospital [127, 184, 183, 34, 150, 142, 140].

To mitigate these problem, pseudonyms are recommended in the GDPR [68] and by ETSI [79], and are a standard practice in various connected vehicle pilot programs of major car manufacturers (ETSI [77], IEEE [136], NHTSA [141]) such as CAMP [63], New York City [105], and Canada [9] pilots. According to a study by ETSI on the use of pseudonyms [78], frequent pseudonym changes enhance privacy: "*the more often an ITS-S*[1] *changes its pseudonym, the higher its privacy*". However, revoking access rights for a client using different pseudonyms can compromise anonymity when an adversary leverages the revocation information to link the pseudonyms [95]. Approaches that mitigate backward linkability by associating pseudonyms with time slots result in increased storage requirements for pseudonyms at the client side and, consequently, in the CRL [95].

Comparison: Table 4.1 summarizes the differences between the related work. We highlight that, with our RRPs implementation, the size of the revocation information grows only logarithmically with the number of time slots. Cryptographic accumulators do not offer VLR. Symmetric encryption cannot protect user privacy from the verifier, and IBE schemes revoke clients by timeout [36] (failing VLR) or issuing the user identifier in a CRL [72] (breaking anonymity). When using GS, clients can generate multiple unlikable signatures with the same secret key, achieving $\mathcal{O}(1)$ for the client storage. Some of these GS schemes revoke clients without providing backward unlinkability, by simply publishing a token for all possible signatures [42, 52]. Other solutions cannot provide the VLR property [171, 147]. Although GS schemes that offer both

[1] *Intelligent Transport Systems (ITS) refer to network components, including the On-Board Equipment (OBE) of a vehicle.*

Systems	VLR	BU	RDS	Revocation data size for an epoch	PM storage (excluding revocation data)	Client storage for an epoch	Signature size or verification	Revocation communication cost
Cryptographic Accumulators								
PEREA [19]	✗	✗	S-b	$R\ T$	N	p	p	R
Camenisch *et al.*[45]	✗	✗	–	R	N	1	1	R
Symmetric Encryption								
Octopus [100]	✓	✗	–	R	N	1	1	R
Mix Zones [86]	✓	✗	–	R	N	1	1	R
Nymble [182]	✓	✓	S-b	$R\ p\ T$	V	$p\ T$	1	$R\ p$
Identity Based Encryption								
Boneh *et al.* [36]	✗	✓	–	–	N	1	1	–
Echeverría *et al.* [72]	✓	✗	S-b	R	N	1	1	R
Group Signatures								
Bringer *et al.* [42]	✓	✗	S-b	$R\ T$	N	1	1	R
Chu *et al.* [52]	✓	✗	E-b	$R\ \log(T)$	$N\ \log(T)$	$\log(T)$	$\log(T)$	$R\ \log(T)$
LBR [171]	✗	✓	–	R	R	1	1	–
Ohara *et al.* [147]	✗	✓	S-b	$R\ \log(N/R)$	$N\ \log(N)$	$\log(N)$	1	R
Emura *et al.* [76]	✓	✓	S-b	$R\ T$	N	1	1	R
Sucasas *et al.* [173]	✓	✓	S-b	$R\ p\ T$	$(p\ +\ 1)\ N$	1	1	$R\ p$
Rahaman *et al.* [156]	✓	✓	S-b	$R\ p\ T$	N	1	1	$R\ p$
Nakanishi *et al.* [139]	✓	✓	S-b	$R\ T$	N	1	1	R
Ishida *et al.* [106] .	✓	✓	S-b	$R\ T$	N	1	1	R
Public-key Encryption								
Rigazzi *et al.* [161]	✓	✗	S-b	$R\ p\ T$	$N\ p$	$p\ T$	1	$R\ p$
ACPC [169]	✓	✗	S-b	–	N	$p\ T$	1	$R\ p$
PRESERVE [78]	✗	✓	–	–	N	$p\ T$	1	–
IFAL [188]	✗	✓	–	–	N	$p\ T$	1	–
V-token [165]	✗	✓	S-b	$R\ p\ T$	1	$p\ T$	1	–
PASS [174]	✓	✓	S-b	$R\ p\ T$	N	$p\ T$	1	R
SCMS POC [194]	✓	✓	S-b	$R\ p\ T$	N	$p\ T$	1	R
Khodaei *et al.* [113]	✓	✓	S-b	$R\ p\ T$	N	$p\ T$	1	R
Haas *et al.* [95]	✓	✓	S-b	$R\ p\ T$	N	$p\ T$	1	R
RRPs/EDGAR (Ours)	✓	✓	E-b	$R\ p\ \log(T)$	1	p	$\log(T)$	$R\ p\ \log(T)$

N: Number of clients. R: Number of revoked clients. p: Required pseudonyms.
RDS: Revocation Data Structure. S-b: Slot-based. E-b: Epoch-based.
T: The number of time slots in one epoch. PM: Pseudonym Manager. V: Number of verifiers.
BU: Backward Unlinkability

Table 4.1: Properties and complexity offered by different systems, we omit $\mathcal{O}()$ notation for simplicity.

backward unlinkability and VLR simultaneously require the revocation procedure to manage a number of credentials that is proportional to the number of time slots $\mathcal{O}(R\ p\ T)$ [173, 156] and $\mathcal{O}(R\ T)$ [76, 139, 106]. Another limitation of GS schemes is that they rely on complex and heavy cryptographic operations, in particular, to support revocation; this can induce large latencies when performing authentication. PKI based schemes are appealing due to their cryptographic efficiency and wide adoption. Some PKI schemes delay the revocation until all pseudonyms expire [69, 78, 188, 165], breaking VLR. Previous schemes that provide both VLR and backward unlinkability suffer from the same issue as GS scheme [174, 194, 113, 95], by locking pseudonyms to time slots, they require revocation information that is linear with the time slots, times each pseudonym, $\mathcal{O}(R\ p\ T)$. In addition, these solutions require the clients to carry pseudonyms for all time slots, imposing a storage burden of $\mathcal{O}(p\ T)$.

4.3 System Model

This section presents preliminaries on RRPs and EDGAR, which is an edge authentication system based on RRPs. In EDGAR, to perform authentication, a client first uses an RRP to

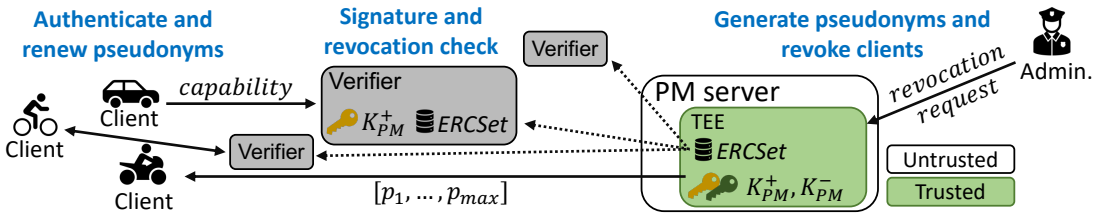


Figure 4.1: System Overview.

obtain a *capability*. This capability, which is only valid for a given target time slot, is then presented to a verifier. To revoke the use of a RRP during a range of time slots, the corresponding capabilities are revoked. There is a level of indirection between the RRPs assigned to clients and the capabilities used for authentication and revocation that is the enabler to achieve backward unlinkability.

4.3.1 Entities

EDGAR is composed of three main types of entities: clients, verifiers, and a (distributed) pseudonym manager service. We follow a nomenclature similar to previous work [156, 95].

Clients: the application client that generates signatures to perform authentication against any verifier. Clients are the holders of RRPs that they use to generate capabilities to ensure anonymity. Clients are responsible for renewing their RRPs when needed.

Verifiers: the component that performs client authentication before granting access to a resource, such as edge storage. Verifiers are responsible for checking the capabilities provided by clients before granting access. They are also responsible for updating their state by fetching the list of revoked capabilities from the manager servers.

Pseudonym Manager (PM): this component is responsible for providing new RRPs to clients and, when necessary, revoking capabilities generated from these pseudonyms. PM servers are the only entity capable of accessing the true identity of a client so, in our implementation, they run partially inside a TEE, ensuring users' anonymity even if the device is compromised.

Administrator: a trusted entity responsible for adding clients to the system and request PMs to revoke clients.

Figure 4.1 shows the interactions between these entities, where the PM uses a TEE. The figure represents a typical collective perception scenario, where mobile devices are used to extend human perception. In this example, mobile devices (the clients) authenticate towards the verifiers, to update or download information. When required, clients can contact a nearby PM to renew the set of RRPs used to generate capabilities. Periodically, verifiers will pull from the PM updated revocation information.

In our system, only the PM depends on TEE support. If both clients and verifiers possessed TEEs, implementing an anonymous authentication system would be trivial. Unfortunately, this scenario is not realistic. Edge devices, including both clients and verifiers, are heterogeneous, and it would be an unrealistic assumption to expect all of them to support TEEs. For these reasons, we aim to minimize the trust base required only at the PM.

Notation	Definition	Notation	Definition
t/epoch	Large time interval	s	Time slot, part of an epoch
δ	Duration of a time slot s	c	Capability
K^-, K^+	Private and public key	p	Pseudonym
e^s	Node label for the time slot s	ERCSet/erc	Encoded revoked capability sets
l_x	Latchkeys for e^s	i	Identifier of pseudonym from a user in an epoch
h	Latchkey tree height	M	Extra pseudonyms to circumvent false positives
f	Number of faulty nodes	cid	Client identifier
m	Bloom filter size (bits)	n	Number of items inserted in a Bloom filter
k	Number of hash or index functions	c_s	Number of clients in EDGAR
f_r	Fraction of pseudonyms to be revoked	d	Branching factor of the latchkey tree
x	False positive rate	I	Maximum number of pseudonyms
N	Number of PM replicas		a client can possess

Table 4.2: Table of notations.

4.3.2 Fault Model

For RRPs, we assume a partial synchrony model [71]. In this model there are unstable periods when messages may be arbitrarily delayed, and stable periods when messages between correct entities arrive within at most Δ units of time. Additionally, we assume that correct processes have access to loosely synchronized clocks, which can differ at most by ϵ. We assume that at most f server nodes can be faulty. We do not place constraints on the number of faulty clients.

Furthermore, verifiers and clients are insecure and prone to Byzantine faults [117]. PM servers are executed (partially) inside TEEs and are only subject to crash and omission faults [64]. Thus, faulty clients may use expired or invalid credentials when contacting servers, faulty verifiers may arbitrarily deny or grant access to resources, but faulty PMs will never provide faulty information, and will never renew pseudonyms for revoked clients. EDGAR ensures liveness during stable periods and offers graceful degradation during unstable periods: when the network is unstable and nodes are unable to receive up-to-date information in a timely manner, they may stop providing service, but never compromise safety.

4.3.3 Threat Model

We trust only administrators and PMs. Following related work [95, 165], an administrator is responsible for adding and revoking users in the system by contacting the PM server. We assume that each PM has a processor with TEE (e.g., Intel SGX), as shown in Figure 4.1. All other entities within the system are considered untrusted and susceptible to the control of attackers, potentially engaging in malicious activities. Table 4.2 provides the notations.

Malicious Client: may attempt to generate pseudonyms or capabilities to impersonate a valid client and access resources to which it is not authorized. It can also try to use old capabilities and pseudonyms after being revoked to authenticate towards verifiers.

Malicious Verifier: if a verifier is compromised, the resource it is tasked with protecting is left vulnerable, but this is not the problem we consider in this paper. For example, under a DoS attack, a verifier may be unable to refresh revocation information and should enter a "safe-mode" (the safe-mode behaviour is application specific but may be as simple as halting).

The problem we consider is that a malicious verifier may try to perform linking attacks [165, 95]. Such attacks involve associating (linking) various pseudonyms with a single client, thereby compromising user anonymity. This type of attack becomes trivial when revocation lists, which enumerate all of a client's pseudonyms, are disclosed. A malicious verifier could potentially compile all observed data with the aim of deducing user identities.

Malicious Pseudonym Manager: PM code is split in two parts, one that runs inside the TEE and one that runs outside the TEE. The latter can be compromised and engage in malicious behavior, supporting many of the previously introduced attacks. The untrusted part of PM may attempt to modify, delay, block, or read all messages on the system. This behavior may be done in collusion with other entities to facilitate Linking Attacks or allow a user to evade revocation. Additionally, in our system, we do not protect a single enclave from suffering DoS, as this is also not covered in the Intel SGX threat model. However, we assume that a node suffering from DoS is one of the f faulty nodes and that at most f servers can be faulty.

Trust Assumptions: Entities use asymmetric key pairs to establish secure channels. Clients employ RRPs for authentication, integrity, and non-repudiation. Both the PM and the administrator hold unique key pairs, (K^-_{PM}, K^+_{PM}) and $(K^-_{admin}, K^+_{admin})$, respectively, being both public keys known to all entities. Specifically, the administrator's public key K^+_{admin} is hard-coded in the enclave's source code. We assume that the PM correctly executes our protocol within the TEE, where K^-_{PM} remains securely within the enclave. The PM will only revoke users if instructed by the trusted and authenticated administrator, and will generate fresh pseudonyms for non-revoked and authenticated clients. We assume that there is no collusion between the trusted PM and the verifiers.

The communication between the administrator and the enclave supported by TLS using their keys. We assume a trusted administrator who only revokes pseudonyms after informing the corresponding clients. Revocation auditability is beyond the scope of RRPs and EDGAR. Furthermore, both capabilities and revocation information are accompanied by a digital signature created using K^-_{PM}, confirming the origin from the PM TEE.

We make the usual assumptions about the security of TEEs/enclaves [57] (code/data executed/stored inside the TEE have integrity and confidentiality guaranteed), about the cryptographic schemes (they satisfy their security properties) and cryptographic keys (secret and private keys are never disclosed). In the EDGAR prototype, we use Ed25519 to generate digital signatures [33]. As a collision-resistant hash function, we use SHA-256. We use Intel SGX as our TEE, although our scheme can be easily adapted to other TEEs. We leverage the Intel SGX SDK inside the enclave and OpenSSL outside (all in C/C++).

Although side-channel attacks such as Foreshadow and LVI [186] exist, we consider the defense from these attacks to be orthogonal to our contribution; possible mitigations are discussed in Bagher*et al.* [23]. Correctly synchronizing concurrent data structures can mitigate exploits against synchronization bugs [193], with the help of debugging checkers[2] [122].

[2]In EDGAR implementation only a Bloom filter and the current epoch value are accessed concurrently inside the enclave.

4.4 Range-Revocable Pseudonyms

RRPs are a novel abstraction that provides authentication based on pseudonyms whose validity can be revoked for any time-range within their original validity period. Clients hold a number of RRPs that is proportional to the number of authentication actions they need to perform. A validity of an RRP is bounded to an *epoch*. An epoch is divided into time *slots* of length δ. The parameter δ is application-specific but can be small, e.g., 1 minute or less. An epoch is assumed to be much larger than the slot, e.g., 1 day. Each RRP should be used for authentication at most once. To perform authentication, a client instantiates a *capability* that is specific to target slot. If a client is revoked for a time period, pseudonyms are not revoked directly; instead, only the capabilities associated with the time-slots of that period are revoked. We store these capabilities in an *Encoded Revoked Capability Sets* (ERCSet). An RRP can be revoked for a short period, by revoking only the capabilities associated with an interval of time-slots, or permanently, by revoking the capabilities associated with all future time-slots. Since the revocation information is connected, indirectly, by capabilities, to pseudonyms, when using RRP, a client is required to carry a different RRP for each access it needs to perform. However, any given RRP can be used at any slot of the epoch. Thus, the number of RRPs a client needs to keep is independent of the granularity of the time-slots. This contrasts with previous pseudonym-based solutions, where clients need to carry a number of pseudonyms that grows linearly with the *epoch granularity* (i.e., the number of slots in an epoch).

4.4.1 Operations Overview

Authentication based on RRPs uses 3 different related objects, namely (range-revocable) *pseudonyms*, (time-bound) *capabilities*, and ERCSet. At an abstract level, the operations supported by these objects are the following:

- $p^{epoch} \leftarrow$createRRP(*cid, epoch,* K^-_{PM}): used to create a new RRPs, that can be used by client *cid* during a target *epoch*. Only PMs, using their private key K^-_{PM}, can create RRPs.

- $c^s \leftarrow$getCapability(p^{epoch}, s, K^-_p): used to create a capability associated with an RRP p^{epoch} for time slot s (s must belong to the epoch for which the pseudonym was created). Only PMs and the client that owns the pseudonym, and the correspondent private key K^-_p, can create capabilities.

- *boolean* $\leftarrow$verifyCapability(c^s, K^+_{PM}): To verify if a capability was generated from a valid RRPs, used by verifiers during authentication, requires the PM public key K^+_{PM}.

- *ERCSet* $\leftarrow$createERCSet(*capabilities*): used only by PMs to create an ERCSet that encodes one or more given capabilities, using some one-way function, such that it is unfeasible to extract a capability from the ERCSet. These capabilities are filtered to ensure that they do not compromise unlinkability (Section 4.4.3).

- *ERCSet* $\leftarrow$mergeERCSet(erc_1, erc_2): used to merge two ERCSets so that a single ERCSet can be used to capture the revocation of multiple capabilities. PMs and verifiers can merge ERCSets.

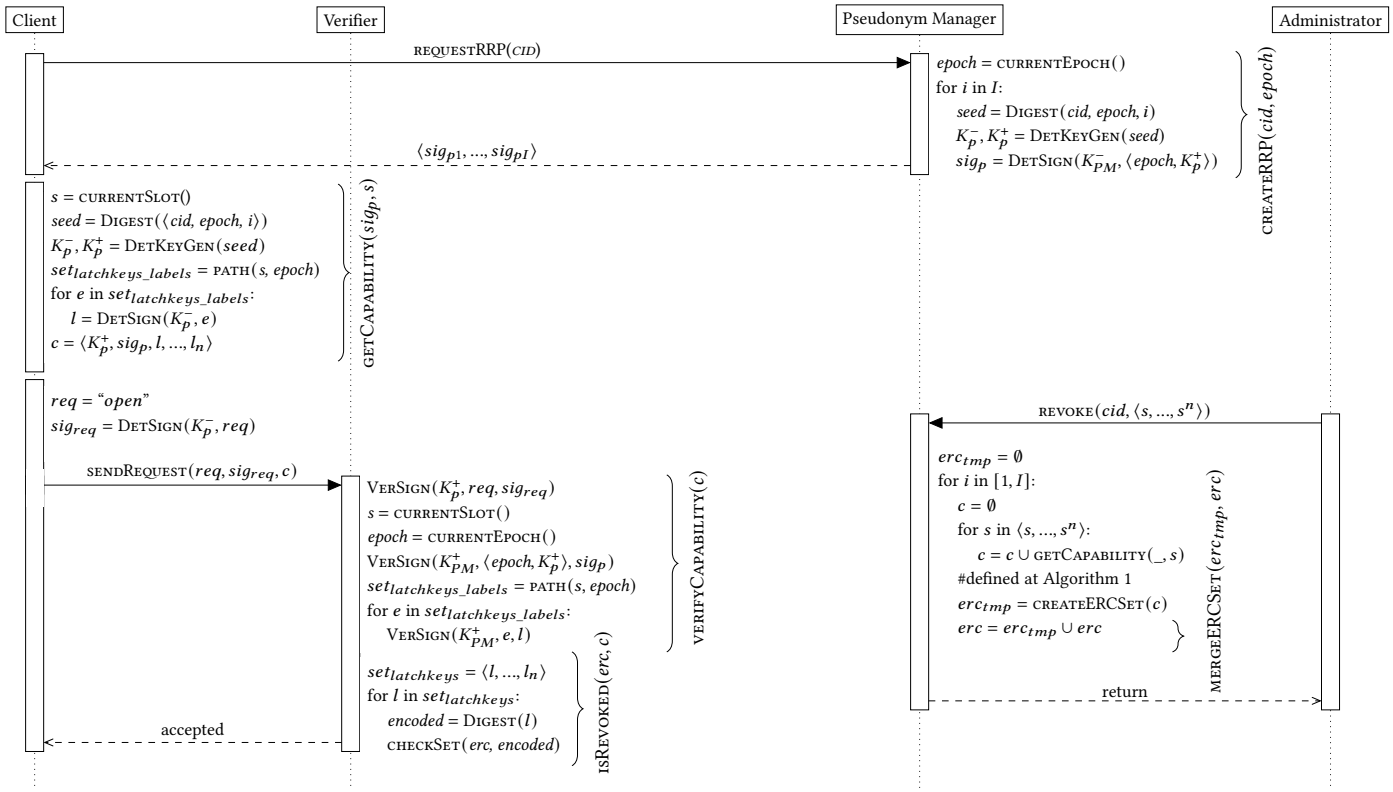


Figure 4.2: Protocol workflow for RRPs and pseudocode for each operations.

– *boolean* ←isRevoked(*erc*, *capability*): used to verify if a capability is part of an ERCSet. This operation is used by verifiers to check if a capability has been revoked.

The manager creates RRPs on request from authorized clients. If later an RRP needs to be revoked for a given range of time slots, the PM generates the corresponding capabilities and encodes them in an ERCSet that is sent to the verifiers. Figure 4.2 provides in detail the operations workflow for each operation, and how each entity interacts in the system.

Clients hold a small number of RRPs (e.g., corresponding to the number of distinct events), and instantiated a short-lived capability (for the current slot) to authenticate. Then, it presents the capability to the verifier. The verifier checks if the capability is correctly constructed, is *genuine* (i.e., if it was generated from a valid RRPs) and subsequently check if the capability has not been revoked; only in this case, the client is granted access to the resource.

To ensure unlinkability, a client must never present two capabilities generated from the same RRP, as capabilities generated from the same RRP can be linked (cf. Section 4.4.3). Therefore, clients have to carry a number of RRPs proportional to the number of resources they need to access. However, contrary to previous systems, the revocation of an RRP for a time-slot does not expose capabilities that may have been used in non-revoked time slots: this is guaranteed by the use of a one-way function to encode revoked capabilities.

4.4.2 Making Range-Revocation Efficient

A problem with the use of time-bound pseudonyms is that the number of pseudonyms that need to be revoked grows with the granularity of the time slots. RRPs are not immune to this problem, because to revoke the use of an RRP in a range of time slots, all capabilities associated with those time slots need to be encoded in the ERCSet. However, our implementation of RRPs uses a mechanism that allows the revocation cost to grow only logarithmically with the granularity, rather than linearly, as previous approaches.

To achieve this goal, a capability is represented by a sequence of *latchkeys*, extracted from a set of latchkeys that are associated with a given RRP. The latchkeys are organized in a tree of fanout d, such that there is a leaf latchkey for each individual time slot on an epoch (in this work, we use $d = 2$, i.e., binary latchkey trees). Figure 4.3 provides a simple example where a binary tree of latchkeys is associated with an epoch of 1 hour divided in 4 time-slots of 15 minutes. Note that the latchkey tree structure resembles but is not a Merkle tree [131]: the tree nodes are generated independently (the value of a parent node does not depend on the value of its children).

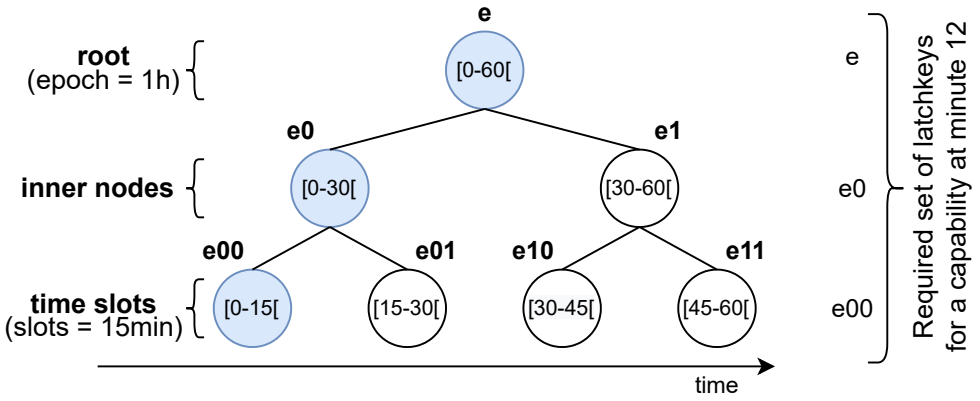


Figure 4.3: Latchkeys for time slot $[0, 15[$ of epoch $[0, 60[$.

A capability for a given time slot is represented by the set of latchkeys in the path from the root of the tree to the corresponding leaf node in the tree. Using the example of Figure 4.3, the capability for the first slot would be represented by the following set of latchkeys: $\{e, e0, e00\}$. Note that each capability, for each time slot, is always different, because it contains one unique leaf latchkey. However, different capabilities may have some latchkeys in common; in particular, all capabilities include the root latchkey.

A capability is only considered *valid* if all latchkeys used to represent it are valid. Therefore, the capabilities can be revoked by invalidating any of its latchkeys. In particular, a capability for a given time slot can be revoked by invalidating the leaf latchkey associated with that slot. However, it is also possible to revoke multiple latchkeys by invalidating latchkeys that are inner nodes of the tree: by invalidating an inner node, all the capabilities that are part of the sub-tree rooted at that inner node are invalidated. This can also be illustrated using our example. Consider that the client is revoked at the beginning of the second slot ($e01$) until the end of the epoch. At this point, the client may already have used its pseudonym p to generate a capability to access the resource during the first slot. To prevent linkability, the latchkeys used in the first slot cannot be revoked, i.e, the latchkeys $e, e0$ and $e00$ cannot be revealed. To revoke all future capabilities that may be generated with pseudonym p, it suffices to revoke latchkeys $e01$ and

$e1$. Note that the capabilities generated for the second slot must use $e01$, and the capabilities generated for the third/fourth slots must use $e1$.

We use this construction to perform revocation efficiently. ERCSets do not explicitly contain capabilities, but only latchkeys that belong to those capabilities and a single latchkey can be used to revoke multiple capabilities. It is easy to show that the number of latchkeys that need to be revoked is at most $\log_d$ with the number of slots. In fact, for each pseudonym valid in the epoch, the number of latchkeys will be given by $\log_d(granularity)$.

4.4.3 RRP Implementation

We now describe the construction of RRPs.

Scheme assumptions: We assume that the epoch and time slot size are publicly known to all parties in the system. This means that any party can independently and consistently calculate all the labels from any leaf to the root (i.e., e, e^0, etc.), as illustrated in Figure 4.3. There is also a maximum number of pseudonyms I that any client can use in any given epoch.

Cryptographic primitives: We assume there are sources of entropy and a function that allow generating random asymmetric key pairs (K^-, K^+). We assume that there is a function, named DetKeyGen(*seed*), to generate asymmetric key pairs deterministically from a *seed* value. There is also a deterministic signature scheme that, given some private key K^- and a *text* as input, will output a deterministic signature $sig = \text{DetSign}(K^-, text)$. The output *sig* can be verified by $true/false = \text{VerSign}(K^+, text, sig)$. Lastly, there is a secure one-way function Digest(*text*) that computes a digest on the *text* input and is not possible to invert given the output.

PM Keys: There is an asymmetric key pair (K^-_{PM}, K^+_{PM}) associated with every PM. The private key K^-_{PM} is only known by the PMs and is kept in the implementation inside the TEE enclave. The public key K^+_{PM} is known to all participants, including clients and verifiers.

RRPs: An *RRP* is a tuple $\langle cid, epoch, i, K^-_p, K^+_p, sig_p \rangle$ where *cid* is the client identifier (only known by the client and the PM), *epoch* is the time windows for which the pseudonym is valid, i is a label that can be used to distinguish each pseudonym instance generated for the same epoch, where $i \in [1, I]$. The (K^-_p, K^+_p) is a unique asymmetric key pair associated with the pseudonym, and sig_p is the signature performed with the private key of the PM over the concatenation of the epoch, and public key of the pseudonym, $sig_p = \text{DetSign}(K^-_{PM}, epoch \parallel K^+_p)$. Note that some fields of an RRP are secrets known only to the client and PM and never revealed to a verifier. In particular, only the client and the PM know the secret key K^-_p associated with a given pseudonym. To obtain an RRP, a client establishes a secure channel with a PM, presents its client identifier *cid*, and obtains one or more RRPs for some given target *epoch*. When describing EDGAR, we will discuss for which epochs clients are allowed to obtain RRPs from a PM.

Generating (K^-_p, K^+_p)***:*** The asymmetric key pair associated with a pseudonym is generated using the DetKeyGen(*seed*) primitive. We use as seed the tuple $\langle cid, epoch, i \rangle$, avoiding the need for the PM to memorize the information associated with all the pseudonyms it created, as it can always re-create them (as explained below, the key pair is also needed to perform revocation). Recall that *cid* is known only by the client and the PM. This identifier is securely stored by the PM inside the enclave. Also, DetKeyGen is non-reversible, thus two different public keys

created for different epochs and/or instances for the same client cannot be linked with the secret *cid*.

Latchkeys: Latchkeys are unique for each pseudonym and are obtained by deterministically signing the label of the corresponding node with the private key K_p^- of the pseudonym. Therefore, the latchkey l_0 associated with the label node e^0 of an *RRP*, is generated as $l_0 = \textsc{DetSign}(K_p^-, e^0)$, and can be verified by using the public key of the pseudonym by performing $\textsc{VerSign}(K_p^+, e^0, l_0)$.

Capabilities: A capability c for a given time slot s is a tuple:

$$c = \langle K_p^+, sig_p, l_{leaf}, \ \ldots, \ l_{00}, \ l_0, \ l_{root} \rangle$$

where K_p^+ is the public key of the pseudonym and the latchkeys correspond to the nodes on the path from the root of the latchkey tree to the leaf latchkey node associated with the time-slot s. Note that a capability has a number of latchkeys that is logarithmic with the granularity of the time-slots in the epoch. The latchkeys that are part of a capability can be generated on demand, when the capability is created, and are not required to be stored explicitly by the client. It should also be noted that any two capabilities generated from the same RRP reveal the same K_p^+ and can be linked; therefore, a client that wants to prevent authorization request to be linked should always use different RRPs.

To verify a capability, a verifier performs the following steps. First, it uses the public key of the PM to verify sig_p, calculating $\textsc{VerSign}(K_{PM}^+, epoch \| K_p^+, sig_p)$. Then, it uses K_p^+ to verify if the latchkeys presented with the capability are in fact associated with that RRP, by performing $\textsc{VerSign}(K_p^+, e^x, l_x)$. If *all* latchkeys can be verified using K_p^+ and follow a correct path from the current slot to the root, the capability is genuine. Note that a capability can be genuine but may have been revoked, as explained next.

ERCSet: an ERCset in an encoding of a set of latchkeys that represents a set of revoked capabilities. The set of latchkeys encoded in an ERCset has the following properties: *inclusion-of-revoked* – if a capability has been revoked, at least one of its latchkeys is encoded in the ERCSet; *exclusion-of-non-revoked* – if a capability has not been revoked, none of its latchkeys are encoded in the ERSet. Below we explain how latchkeys are selected to be encoded in the ERSet to satisfy these properties. Latchkeys are encoded in the ERCSet using a one-way function, $\textsc{Digest}(l_x)$. Thus, verifiers can check if a given latchkey belongs to an ERCSet but cannot extract latchkeys from the ERCSet. Different data structures that rely on one-way functions could be used to implement ERCSet, including SHA256, or compact data structures such as Cuckoo filters [81], Cascade filters [118] or Count-min sketch [55]. We use Bloom filters to implement the ERCSet. Bloom filters are efficient and, as discussed later, a good fit for the EDGAR architecture. A disadvantage of Bloom filters is that they can present false positives, but we will explain later how EDGAR circumvents this limitation.

Revoking a single capability: To revoke a capability c_p of a pseudonym p, the PM encodes in the ERCSet the leaf latchkey l_x associated with c_p. This trivially satisfies the *inclusion-of-revoked* and *exclusion-of-non-revoked* properties: the encoded latchkey belongs to the revoked capability but does not belong to any other capability (each capability is associated with a distinct, unique, leaf latchkey).

Revoking a range of capabilities: Revoking a set of capabilities of a pseudonym could be trivially achieved by encoding the corresponding set of leaf latchkeys, but this would have a linear cost. The latchkey hierarchy is used to reduce this cost as follows. Let d be the fanout of the latchkey tree. Any d latchkeys that have the same parent in the latchkey tree can be replaced by their parent. This also satisfies the *inclusion-of-revoked* and *exclusion-of-non-revoked* properties of ERCSets: 1) the parent of any latchkey is part of the capability that includes that latchkey and 2) a parent latchkey is not included in capabilities other than the capabilities that include its children. The susbtitution of all d sibling latchkeys by their parent latchkey can be applied recursively in the tree. Note that the root latchkey can only be included in an ERCSet when a pseudonym is revoked for the entire duration of the epoch, because the root latchkey belongs to all capabilities for that epoch. Next Section 4.4.4 provides a precise description of this algorithm with pseudo-code.

Merging ERCSet: An advantage of using Bloom filters is that ERCSet can be easily merged by performing bitwise OR operations. This makes it easy to disseminate revocation lists for many different RRPs in a single data structure.

Checking if capability revoked: Verifiers receive ERCSets from PMs, store them, and use them to check if the capabilities presented by clients have not been revoked. After checking if a capability is genuine, verifiers test if any of the latchkeys are included in the most recent ERCSet. If even a single latchkey is in the ERCSet, the capability is considered revoked.

4.4.4 ERCSet Creation Algorithm

We now present in Algorithm 1 the pseudo-code for the algorithm used to implement the createERCSet function, previously described using natural language in Section 4.4.1. The algorithm receives multiple capabilities to be revoked (from a given pseudonym). It first merges all the latchkeys from the set of provided capabilities (function mergeLatchkeys). Then, the function removeUnsafe will search and remove any latchkeys that may cover time slots that have not been revoked, to ensure that any authentication information that may be used outside the revocation interval, represented by the provided capabilities, is not present in the ERCSet. Afterward, the function removeRedundant will remove any redundant latchkeys, i.e. any latchkey children, that are already covered by their parent. This step is only to achieve efficient storage since it takes advance of our latchkey tree and only requires a logarithmic number of latchkeys to cover the revoked time slots. Finally, the function latchkeysEncoding takes the remaining latchkeys and inserts each one in a Bloom filter that implements the ERCSet. Our prototype implements an optimized version of this algorithm.

4.4.5 RRPs Linkability Analysis

A key problem with previous approaches for performing pseudonym revocation is that the information used for revocation could be linked with the information used for access control (in particular, this is obvious when the pseudonym identifier is used both for authentication and revocation). This allows an adversary to collect information about the resources that have been accessed by revoked pseudonyms. When a client has several pseudonyms that are revoked together, an attacker can link the past usage of these pseudonyms to break the privacy of the user. RRPs avoid this problem because the information used for revocation cannot be linked

with the information used for access control. Thus, if a client used one or more pseudonyms prior to revocation, the use of these pseudonyms cannot be linked based on the revocation data.

Here we present an argument that RRPs offer unlinkability, this is a simpler analyse for a better understating. In the next Section 4.5, we provide a more complex and complete proof that RRPs offer unlinkability.

Observation 1. *Verifiers cannot generate latchkeys associated with a pseudonym.*

Basis: Latchkeys are generated using the private key K_p^- of the pseudonym. The private key is generated using the secret *cid* that is shared between the PM and the client and never revealed to other entities. Therefore, verifiers cannot generate latchkeys. □

Observation 2. *ERCSets do not include latchkeys used outside the revocation interval.*

Basis: This property is achieved by construction, that ensures the exclusion-of-non-revoked. As described above, when a PM assembles an ERCSet, it never includes in the ERCSet latchkeys that are part of capabilities for time-slots outside the revocation interval. □

Argument 1. *Revocation information cannot be linked with authorization information used outside of the revocation interval.*

Basis: The revoked latchkeys are encoded in an ERCSet using $\textsc{Digest}(l_x)$, so verifiers cannot extract latchkeys from an ERCSet. Verifiers can only test if a given latchkey has been revoked. However, by Obs. 1, verifiers cannot generate latchkeys, so they can only test latchkeys that are provided by the client when presenting a capability. By Obs. 2, latchkeys for capabilities associated with non-revoked time-slots are not included in an ERCSet. □

Argument 2. *Capabilities generated from different pseudonyms cannot be linked.*

Basis: All the information in a capability depends on the asymmetric key pair associated with the pseudonym. Asymmetric key pairs for different pseudonyms are different because they are generated using different seeds (the unique instance number i is part of the seed $\langle cid, epoch, i \rangle$). Additionally, asymmetric key pairs cannot be linked to the seed used for generation (this derives directly from the standard properties of DetKeyGen). □

4.5 Unlinkability Proof

We now provide a proof that our implementation of RRPs ensures that the information used to revoke clients cannot be linked to the information used by clients when authenticating on non-revoked time-slots. For clarity of exposition and without loss of generality, the proof considers a single epoch, a single client *cid* with a single pseudonym p, and a single PM.

For convenience we use three sets, *MayHaveBeenUsed*, *Safe* and ERCSet, that represent different sets of information that are relevant for the correctness of the algorithm. The set *MayHaveBeenUsed* contains all latchkeys that the client may have provided when authenticating in non-revoked time-slots. The set *Safe* contains all latchkeys that are used to perform revocation of a range of time-slots (that corresponds to the output of function removeUnsafe in Algorithm 1). The set ERCSet includes an encoded version of all latchkeys in the set *Safe*, which results from

applying a one-way function to each latchkey in *Safe* (that corresponds to the output of function LATCHKEYSENCODING in Algorithm 1). Note that the function REMOVEREDUNDANT is relevant to reduce the size of the *Safe* but is not relevant to the proof (which remains valid, even if this layer of filtering is not applied).

MayHaveBeenUsed captures the information that the client may provide to the verifier when performing authentication. The set ERCSet captures the information included in *Safe*. We consider that *MayHaveBeenUsed* and ERCSet are public in the sense that an adversary can be aware of their content. Only the PM has access to the set *Safe*.

We begin by demonstrating that the sets *MayHaveBeenUsed* and *Safe* are disjoint, i.e., that $MayHaveBeenUsed \cap Safe = \emptyset$. Then, we show that encoding the latchkeys of *Safe* in the set ERCSet achieves the desirable unlinkability.

Assumption 1. *There is a secure deterministic digital signature scheme.*

We assume the availability of a secure deterministic digital signature scheme, such as Ed25519 [33], which ensures the usual authentication, integrity, and non-repudiation properties. This scheme supports DETKEYGEN(*seed*) that creates a key pair (K^-, K^+) from any given seed. Signatures are also deterministic, generated by DETSIGN() using K^-, and verified by VERSIGN() with K^+. Finally, it is not possible to forge a signature without K^-.

Assumption 2. *There is a secure one-way function.*

We assume the availability of a secure collision-resistant hash function DIGEST(), such as SHA256, that is easy to compute on every input, but not possible to invert given the output.

Assumption 3. *Only the client and the PM can access cid.*

We assume that both the client and the PM will securely store the secret *cid* and never disclose it.

Definition 1. *A time-slot tree is a n-ary tree data structure that represents the time period of an epoch divided in time slots.*

A time-slot tree (see Figure 4.3) is a tree data structure where the time period of an epoch is divided into smaller slots of size δ; each time slot is represented as a leaf node in the tree, identified by a unique node identifier e^x. An inner node of the tree has n children and represents the entire time interval of its children nodes; the tree root node e captures the entire epoch time interval.

Definition 2. *A valid latchkey is a digital signature of a tree node.*

A latchkey l_x represents the unique bond of a pseudonym p to a node e^x in the tree. This bond is implemented by a digital signature performed over the node identifier (cf. Definition 1), where $l_x = \text{DETSIGN}(K_p^-, e^x)$. By using the private key of the pseudonym (K_p^-), we enforce authentication, integrity, and non-repudiation of the latchkey (see Assumption 1).

Definition 3. *A valid capability C_p is defined as:*

$$c = \langle K_p^+, sig_p,\ l_{leaf},\ \ldots\ l_{root} \rangle$$

where K_p^+ is the public key of the pseudonym p, and the signature sig_p is used to prove the pseudonym authenticity (cf. Section 4.4.3), being generated by the PM as $sig_p = \text{DETSIGN}(K_{PM}^-, epoch \| K_p^+)$, $\|$ represents the concatenation of both values. Finally, $l_{leaf}, \ldots l_{root}$ is a sequence of valid latchkeys (from Definition 2) associated with the nodes in a path from a leaf node to the root node of the time-slot tree. The path is required to perform pseudonym revocation efficiently. Each time-slot s is associated with a different capability, which is uniquely defined by the path from the root node of the time-slot tree to the leaf node associated to time-slot s.

Definition 4. *The Non-Revoked Capability Set is the set of capabilities for all non-revoked time-slots.*

Note that the Non-Revoked Capability Set is never used in the algorithm (in fact, a client uses each pseudonym a single time, and never for multiple time-slots). The Non-Revoked Capability Set is an abstract artifact that is useful for the proof; it represents all capabilities that may be used by a client in non-revoked time lots.

Definition 5. *We define MayHaveBeenUsed as the union of all the latchkeys in all capabilities from the Non-Revoked Capability Set.*

Like the Non-Revoked Capability Set, *MayHaveBeenUsed* is never explicitly used in the algorithm and can never be collected by verifiers because, as noted before, a correct client only uses each pseudonym once (for a single time-slot). The set *MayHaveBeenUsed* captures all latchkeys that may be exposed to verifiers by the client when authenticating in non-revoked time lots.

Definition 6. *The Revoked Capability Set is the set of all capabilities for all revoked time-slots.*

The Revoked Capability Set is provided as input for Algorithm 1, that constructs an ERCSet that is subsequently used for revocation.

Definition 7. *The Unfiltered Revoked Latchkey Set, or simply Unfiltered, is the union of the latchkeys in all capabilities in the Revoked Capability Set.*

The *Unfiltered* set is obtained by applying the MERGEMATCHKEYS function to the Revoked Capability Set in Algorithm 1.

Definition 8. *The Safe set is constructed by applying the* REMOVEUNSAFE *function from Algorithm 1 to the Unfiltered set.*

Definition 9. *The ERCSet is constructed by encoding all the latchkeys in the Safe set with one or more one-way hash functions.*

In our implementation we use Bloom filters to implement the ERCSet. The one-way hash functions are implemented using *Digest(l_x)* from Assumption 2.

Lemma 1. *MayHaveBeenUsed and Safe are disjoint.*

Proof. We now show that $MayHaveBeenUsed \cap Safe = \emptyset$ by contradiction: The same latchkey being in both sets can be written as: $\exists x : x \in MayHaveBeenUsed \cap Safe$.

Let us assume that the latchkey x belongs to *MayHaveBeenUsed* then, from Definition 4, there is a capability c_s, of a non-revoked time slot s, that contains x. Let l_s be the latchkey associated with the leaf node for time slot s. Note also that if x is associated with an inner node of the time-slot tree, the leaf node for time slot s is a descendant of that inner node, because a capability includes all nodes in the path from the root to the leaf node, and nodes in the path have a parent-child relation.

Let us assume that the latchkey x belongs to *Safe*. Then l_s must belong to *Unfiltered*, otherwise, because l_s descends from x, function REMOVEUNSAFE in Algorithm 1 would remove x from *Safe* if $l_s \notin$ *Unfiltered*.

Thus, if $\exists x : x \in$ *MayHaveBeenUsed* $\cap$ *Safe* then l_s must belong both to *MayHaveBeenUsed* and to *Unfiltered*. Because l_s is associated with a leaft node, this means that there is a time slot that is both revoked and non-revoked, a contradiction. □

Lemma 2. *Only the client and the PM are capable of generating a valid capability C_p and the respective latchkeys.*

Proof. From Definition 3, capability C_p includes a set of latchkeys. From Definition 2, to construct each latchkey, access to the pseudonym private key K_p^- is required. From Assumption 1, the key pair (K_p^-, K_p^+) for the pseudonym p that the client holds is generated using the *cid* as seed in the asymmetric scheme. From Assumption 3 only the client and PM can access the *cid* and, therefore, only the client and the PM can generate K_p^-. Therefore, only the client and the PM can generate valid latchkeys for the capability C_p. □

Lemma 3. *An adversary can only access latchkeys from MayHaveBeenUsed.*

Proof. *MayHaveBeenUsed* contains all the latchkeys that appear in non-revoked capabilities. An user, to authenticate in a non-revoked time-slot, must generate and present the corresponding capability to a verifier (that is non-trusted). Therefore, the attacker may have access to latchkeys in *MayHaveBeenUsed*. On the other hand, by Definition 9, ERCSet is constructed by encoding each latchkey l_x using DIGEST(l_x), and by Assumption 2 the adversary is not capable of inverting any entry in ERCSet to extract a latchkey. □

Theorem 1. *The information used to revoke clients cannot be linked to the information used by clients when authenticating on non-revoked time-slots.*

Proof. Revocation is performed using exclusively latchkeys from the *Safe* set that are encoded using DIGEST(). To link the information provided during authentication with the information used for revocation, the adversary would need to have access to at least a latchkey $u \in$ *MayHaveBeenUsed* and to a latchkey $r \in$ *Safe*. From Lemma 3, the adversary can only access the latchkeys in *MayHaveBeenUsed*. From Lemma 1, *MayHaveBeenUsed* and *Safe* are disjoint. Therefore, the information used to revoke clients (latchkeys from the *Safe* set) cannot be linked to the information used by clients when authenticating (the latchkeys in *MayHaveBeenUsed*). □

From Theorem 1, a client using a single pseudonym obtains unlinkability guarantees, since no revocation information can be linked with authorization information used outside the revocation interval. Next we show that RRPs also offer unlinkability when multiple pseudonyms are used.

Theorem 2. *Capabilities generated from different pseudonyms cannot be linked.*

Proof. From Definition 3, all the information in a capability depends on the asymmetric key pair associated with the pseudonym. From Assumption 1, asymmetric key pairs for different pseudonyms are different because they are generated using different seeds (the unique instance number i is part of the seed $\langle cid, epoch, i \rangle$). Additionally, asymmetric key pairs cannot be linked to the seed used for generation: this is guaranteed by the security of the key generation scheme. □

4.6 EDGAR

We now present the design of an anonymous authentication system for the edge that leverages RRPs to offer backward unlinkability. We have named our system EDGAR: EDGe distributed Access contRol, targeting the VANET scenario [158, 87, 32]. The goal of EDGAR is to reduce the linkability window, improving client privacy at the edge. EDGAR demonstrates how to use our RRPs abstraction and how to address implementation challenges in a distributed setting.

4.6.1 EDGAR in VANETs

In the VANET scenario, vehicles continuously broadcast CAM messages [80] containing various information such as their geolocation, sensor readings, direction, and speed. This information is crucial for various edge applications, including enhanced navigation, traffic congestion estimation, remote vehicle diagnostics, autonomous cars, and others [85]. However, as explained in Section 4.2, edge providers can collect and monetize this data, at the expense of users privacy, highlighting the importance for clients to use anonymous authentication methods such as EDGAR. We now contextualize the RRPs entities to the corresponding entities in EDGAR:

Clients: These are vehicles that constantly propagate CAM messages with location and sensor readings, with the purpose of enhancing their safety and that of others.

Verifiers: Mainly compose by Roadside Units (RSUs) [61] that listen to all CAM messages, aggregate them, store them, and broadcast them in the network. These devices can be deployed by various local entities (e.g., municipal authorities) or edge providers to improve traffic flow, pedestrian safety, and provide services to vehicles such as infotainment or software updates.

PM servers: These are fog nodes with the same services as verifiers but with higher computational capacity and storage, and may be physically more distant than verifiers. Any fog node with TEEs can serve as a PM. We assume the same trust level as mentioned in Section 4.3.3, where the PM does not share its private key or the client pseudonyms. However, the PM is controlled by edge providers, who can access the non-trusted zone outside the TEE.

Administrator: In the context of the edge, the administrator should be a trusted entity independent of all applications and providers within the edge. It should work similarly to the current Certificate Authorities (CAs) in PKI.

Figure 4.4 illustrates the interactions in the edge environment. In this example, a vehicle (a client) presents a capability to the RSU (the verifier). If the capability is valid, the RSU accepts the message from the client and alerts the vehicles about pedestrians behind the corner. Authentication is critical to avoid false information that may cause other drivers to break without justification. The RSU works a storage node that maintains a local map of the street. The RSU

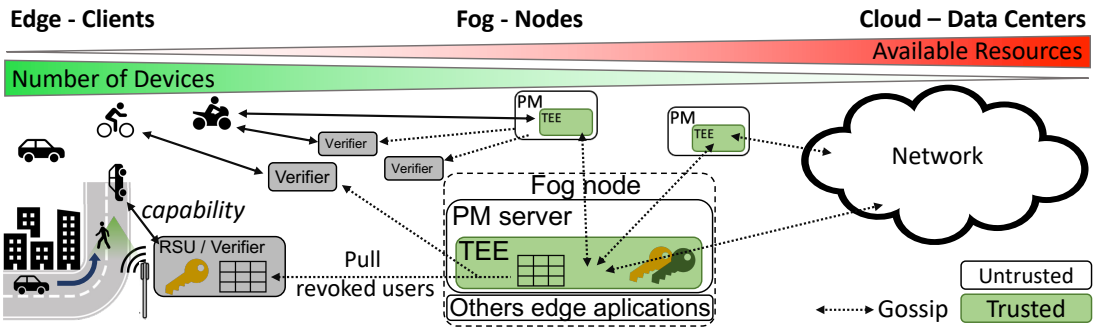


Figure 4.4: EDGAR entities and VANETs interaction at the edge.

also relies on multiple nearby fog nodes, running edge replicas of the PM, to update its state. Vehicles can contact a nearby PM replica to renew the pseudonyms used to generate capabilities, if necessary.

EDGAR prevents the de-anonymization of clients based on the tracking the RRP usage (i.e., even if edge providers like Verizon, Akamai, and Amazon aggregate data from verifiers and the untrusted zone of the PM). Our pseudonyms also protect users' privacy in case of data leaks from verifiers.

4.6.2 Revocation in EDGAR

Although RRPs supports the revocation of a pseudonym in any range of time slots, in EDGAR we assume that clients can be revoked at a target *revocation time slot* (RTS), selected by the administrator, and that all capabilities of that client are revoked for all time slots after RTS (i.e., the revocation range spans from RTS to the end of the epoch). Also, after being revoked, clients become unable to obtain new pseudonyms from edge PMs after some time. In particular, as we will show later, EDGAR is able to provide the following guarantee: if a client is revoked in a given epoch t, that client may still attempt to use pseudonyms it has obtained for epoch $t + 1$ but will not be able to obtain pseudonyms for epoch $t + 2$.

4.6.3 Epochs, RRPs, and ERCSet

Time is divided in epochs and epochs are divided in time slots. The length of an epoch and the granularity δ of the time slot are application specific. As we show in the evaluation, the efficient revocation mechanism of RRPs, based on the latchkey hierarchy, supports the use of relatively large epochs and fine-grain granularity, for instance, epochs of one day and time slots of 1 minute.

There is a limit I of the number of pseudonyms that a client can request for a given epoch. When using RRPs this is not a limitation because clients only need to have a pseudonym for each access regardless of the time slot where the pseudonym is used (and not a different pseudonym for each time slot, as in previous work). Also, clients are only allowed to obtain pseudonyms for the current epoch and for the next epoch (we allow clients to obtain in advance pseudonyms for the next epoch to avoid having PMs to be overload with a rush of requests whenever an epoch begins). This allows us to limit the number of pseudonyms that need to be added to ERCSet

when a client is revoked. Also, verifiers only accept requests that use pseudonyms from the current epoch. This allows to garbage collect revocation information from previous epochs safely.

Due to the constraints described above, EDGAR is only required to maintain two ERCSets: one associated with the current epoch and another associated with the next epoch. When an epoch t terminates, the ERCSet associated with epoch t can be discarded and a fresh ERCSet is created for the next future epoch $(t + 2)$.

4.6.4 ERCSet dissemination

The revocation of a client is initiated in the central PM. The PM first generates all possible pseudonyms that the client may have obtained for the current epoch (i.e., by creating the pseudonyms for all instances $1 \ldots I$) and creates an ERCSet that revokes all the capabilities that may be generated for these pseudonyms in the range starting from the *revocation time slot* (RTS) to the end of the epoch. For this, it uses the algorithm described in Section 4.4.2. It then merges this ERCSet into the global $ERCSet_t$ for the current epoch. The PM then generates all possible pseudonyms that the client may have obtained for the next epoch and creates an ERCSet that revokes these pseudonyms for the entire epoch (this is very efficient, because it suffices to include the root latchkey of each pseudonym in ERCSet); it then merges this ERCSet in the global $ERCSet_{t+1}$.

Disseminating client revocation among the PMs. The updated values of $ERCSet_t$ and $ERCSet_{t+1}$ are then disseminated in the system using a two-step procedure. First, they are disseminated from the central PM to all edge replicas of the PM. Then, verifiers pull these values from their nearest PM replicas. EDGAR implements the propagation of ERCSets among PM replicas using a gossip-based broadcast protocol. The central PM first selects $f + 1$ edge PMs at random and sends them the updated ERCSets. When receiving an ERCSet from another replica, a PM checks if the ERCSet is different from the local version. If the ERCSet is the same, it discards the redundant update. If the ERCSet is different, it assumes that it may contain new information and merges it with its own ERCSet, picks other $f + 1$ edge PMs at random, and sends them the updated ERCSets. This eager push strategy allows revocation information to be propagated quickly on the network. Additionally, a PM that does not receive any updates for more than a predefined gossip timeout engages in pull-gossip with another random PM. Pull gossip is used to recover from temporary crashes or disconnections. A PM that is down when a revocation is eagerly propagated will later obtain the information using pull gossip. Note that the ERCSet for a given epoch always accumulates new information. Thus, any single gossip exchange with an up-to-date server will convey all the information that a node may have missed while disconnected.

Disseminating latchkey revocations to the verifiers. The edge will consist of many verifiers placed at different locations. It is not efficient to have all these PM servers sending the same information to all verifiers. Therefore, we only use pull-gossip to propagate ERCSets to each verifier. Each controller periodically picks a PM at random, pulls $ERCSet_t$ from that server, and merges its content with a local copy of $ERCSet_t$. If a verifier fails to execute the pull-gossip procedure (possibly due to an adversary jamming the PM), it enters "safe-mode" (the specific behavior varies depending on the application, but could involve stopping the service to protect the resource).

4.6.5 Obtaining New Pseudonyms

Clients can obtain pseudonyms from edge PMs. EDGAR does not require edge PMs to keep an explicit list of all clients that have been revoked and of their corresponding *revocation time slot* as this is already encoded in the ERCSet. When requesting new pseudonyms, a client establishes a secure channel with any edge PM and provides its own *cid* and a valid capability. If the client has not been revoked, the PM can provide the requested pseudonyms for the current of for the next epoch. If the client has been revoked, it will be denied access to additional pseudonyms.

To check if a client has not been revoked, a PM performs the following checks: first it verifies if the capability presented by the client is in fact associated to a pseudonym of that client. This procedure leverages that fact that any PM can create all pseudonyms of a client, and therefore, can check if the public key included in the capability corresponds to a public key of one of the valid pseudonyms for that client. Then, it checks if the capability has not been revoked. If the request passes these tests, the PM generates and sends the requested pseudonyms to the client.

4.6.6 Size of ERCSets

EDGAR uses Bloom filters to implement ERCSets. Bloom filters have $O(1)$ insertion and query time [125], are space-efficient, and can be merged easily. However, Bloom filters suffer from false positives and should be used with care. In fact, there is evidence that, if not used properly, the false positives generated by Bloom filters can jeopardize the operation of large-scale systems[118]. We first discuss how the size of the Bloom filters used to implement ERCSets is chosen in EDGAR. Later, we discuss how we deal with the fact that false positives cannot be entirely avoided.

The false positive rate of a Bloom filter depends on the filter size m (bits), the number of items to be inserted n, and the number k of hash or index functions used for insertion and search. The false positive rate can be approximated as described in [95]:

$$P(\textit{false positive}) = \left(1 - \left(1 - \frac{1}{m}\right)^{kn}\right)^{k} \tag{4.1}$$

In the case of ERCSets, the average number of items in the Bloom filter is given by:

$$n = c_s \times I \times f_r \times \log_d\left(\frac{\textit{epoch}}{\delta}\right) \tag{4.2}$$

where c_s is the number of clients, I is the average number of pseudonyms that each client uses, f_r is the fraction of pseudonyms that may need to be revoked, d is the branching factor of the latchkey tree, *epoch* is the length of an epoch, and δ is the length of the time slot. By using Eq. 4.2 to compute the number of latchkeys that are expected inserted in an epoch in a Bloom filter, we can use Eq. 4.1 to select the size of the Bloom filter that limits the probability of having a false positive to some pre-defined threshold.

Let us assume a scenario with $c_s = 250,000,000$ clients (the estimated number of vehicles in the USA), and assume a fraction of pseudonyms that need to be revoked of 10^{-4} per year (from [95]). If we set the length of an epoch $24h$, this provides an average of revocations per epoch of $f_r = 10^{-4}/365$. Then, if we set the granularity of the time slots to $\delta = 10$ minutes.

This yields 144 time slots per epoch. If we use a binary tree of latch keys, the average number of latchkeys used per revocation is $\log_2(144)$. If we assume that clients need at most 10 pseudonyms per day, the expected resulting number of items to be added to the Bloom filter is:

$$n = 250,000,000 \times 10 \times (10^{-4}/365) \times \log_2(144) \approx 4911$$

In this scenario, a Bloom filter of $9KB$ provides a false positive rate of 0.1% (from Eq. 4.1). We can determine the false positive rate of a capability by: $P_{FP}(C) = 1 - (1 - x)^h$, where x is the false positive rate of the Bloom filter, from Eq. 4.1, and h is the tree height. In the same scenario, that corresponds to a false positive of 6 in every 1000 capabilities. Additionally, if one wants to increase the granularity of the time slot to $\delta = 1$ minute, the number of time slots per epoch increases 10 times, but the number of latchkeys increases only logarithmically, thus the size of the ERCSet must increase only by a factor of $\log_2(1,440)/\log_2(144) = 1.46$, i.e, an ERCset of $13KB$ will be enough to maintain the same false positive rate

4.6.7 Circumventing False Positives

Even if the size of Bloom filters is set appropriately, there is always some probability of the occurrence of false positives. In EDGAR we bypass this problem by having clients request M extra pseudonyms, in addition to those that are strictly needed to access the resources. If a false positive occurs, the system automatically picks another unused pseudonym and resubmits the authorization request to the verifier. The only perceived effect by the client is an additional latency in serving the request. We show below that the number of additional pseudonyms that a client needs to carry to circumvent the occurrence of false positives is small. Equation 4.3 describes the probability that a client will execute all authentication successfully with the help of the M extra pseudonyms.

$$P(\textit{full access}) = 1 - \sum_{j=M+1}^{p+M} {}^{p+M}C_{M+1} \times (P_{FP}(C))^j \times (1 - P_{FP}(C))^{p+M-j} \tag{4.3}$$

When applying Equation 4.3 to the previous scenario, where $\delta = 10$ minutes, and setting $M = 0$, it is possible to derive that 1% of the clients may fail some authentication; this number can be reduced to $1.9 \cdot 10^{-12}$ by setting $M = 4$. These extra 4 pseudonyms will require increasing the filter size from $9KB$ to just $13KB$ (to achieve the same probability with $M = 0$ would require increasing the size of the Bloom filter by $34KB$). If each client would require 1000 pseudonyms instead, the probability of a client successfully executing all authentication with $M = 0$ is just 63% but when using $M = 15$ it increases to $1 - (1.9 \cdot 10^{-14})$, with a storage increase from $883KB$ to $896KB$ (to achieve the same probability with $M = 0$ would require a filter of $3.89MB$).

Leveraging M extra pseudonymous is a space-efficient solution to make the effect of false positives negligible, even for large-scale systems such as EDGAR. We could consider alternative encoding techniques that completely eliminate false positives, such as cascade filters [118]. However, this would require anticipating all possible false positives in order to create the multiple filter levels; in a scenario of millions of vehicles with multiple pseudonyms, this operation would be very expensive and might become infeasible.

4.6.8 Handling Epoch Changes and Quarantine

When a client is revoked, all future capabilities that can be generated from the pseudonyms it may have obtained are revoked. As discussed above, if a client is revoked in epoch t, this requires revoking capabilities for future time slots in epoch t and all the capabilities for epoch $t+1$. Capabilities for epoch $t+2$ and other future epochs do not need to be revoked because EDGAR ensures that a client that is revoked in epoch t cannot obtain pseudonyms for epoch $t+2$. This property is guaranteed by a coordination phase that is executed by any PM when it transitions from epoch t to epoch $t+1$. The purpose of the coordination phase is to ensure that any PM that enters in epoch $t+1$ is aware of all revocations performed in epoch t and, therefore, will refuse to issue pseudonyms for epoch $t+2$ to clients that have been revoked in epoch t. During coordination, a PM enters in a quarantine mode, where it cannot serve pseudonym requests for epoch $t+2$.

The coordination protocol is implemented by forcing each PM to send to every other PM its version of ERCsets_t and ERCsets_{t+1} at the beginning of the quarantine. Furthermore, a PM waits to receive revocation information from at least $N-f$ PMs before ending the quarantine. Because revocation is performed by updating $f+1$ PMs, and a PM waits for the input of other $N-f$ PMs, for each revoked client, a PM is guaranteed to receive at least one up-to-date ERCset that includes the corresponding revoked capabilities. At the end of the quarantine, a PM is guaranteed to be fully aware of all revocations that have occurred in epoch t and can start serving requests for pseudonyms in epoch $t+2$.

4.6.9 Handling a PM failure

The temporary failure or disconnection of a PM is treated as follows. When the PM server recovers, it will immediately start the pull-gossip procedure. Eventually, it will be able to get up-to-date information on the revoked clients. The same applies to temporarily disconnected PMs. A PM that is offline for a short period of time can operate normally, even if it is slightly outdated. If it is contacted by a verifier, it will not be able to provide the most recent revocation information, but the verifier will be able to fetch that information from another PM in the next gossip interaction. If it is contacted by a revoked client, it may issue new pseudonyms to that client for the current or the next epoch. However, the corresponding latchkeys for those pseudonyms have already been revoked by other PMs, and the client will be revoked in a bounded time.

4.6.10 Discussion

In this section, we discuss the key features of EDGAR.

Epoch Based Pseudonyms: Pseudonyms in EDGAR are bound to epochs instead of slots. Capabilities are bound to slots, but can be generated at any moment by the client. This decoupling allows clients to store only the desired number of pseudonyms based on application logic, instead of the δ granularity of slots.

Space Efficiency: Both edge computing and TEEs have memory constraints, making space efficiency a crucial aspect [57]. With EDGAR, revoking a client only requires a logarithmic

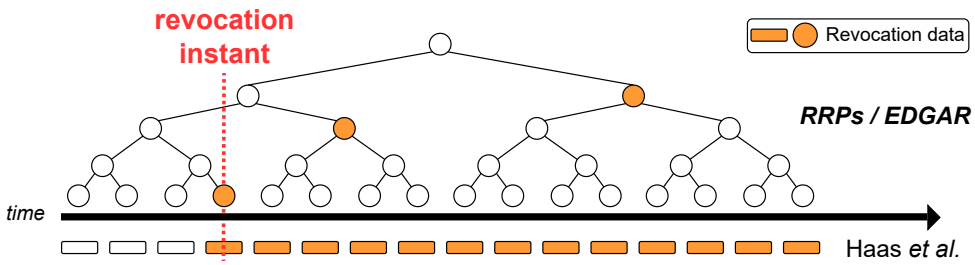


Figure 4.5: Amount of revocation data: EDGAR vs [95].

number of latchkeys, while previous solutions require a linear amount of revocation information relative to the number of time slots (see Figure 4.5).

Backward Unlinkability: EDGAR revokes future capabilities while ensuring that these capabilities cannot be linked to capabilities used in the past. Unlike previous work, the δ granularity of the time slots can be arbitrarily reduced without imposing a burden on the system: the number of pseudonyms used by a client does not depend on δ and the cost of revocation is $\log_2(1/\delta)$.

Support for Distributed Fault-tolerance: EDGAR distributes and replicates the PM functionality. This increases both availability and resilience. It increases availability because clients can obtain pseudonyms from any correct PM. It increases resiliency, because the coordination required to change epoch effectively prevents PMs that have been isolated or whose clock has been attacked from providing new pseudonyms to revoked clients in future epochs (if the clock is moved backward in time and the server generates invalid pseudonyms for old epochs; if the clock is moved forward in time, the server cannot progress through quarantine).

Traceability and Accountability: If required, EDGAR can be extended with mechanisms in which verifiers share (limited) information with the PM to provide traceability and accountability. Specifically, a verifier may present one or more used capabilities to the administrator and ask to revoke or trace the anonymous client that is responsible for such capabilities. Depending on the application and the facts to justify the request, the administrator may agree and forward the capabilities to the trusted central PM. Since the PM can generate all the public keys associated with the pseudonyms it has provided, it can subsequently match the used capabilities with the clients identifiers (note that only the trusted PM can perform this operation; this does not conflict with ensuring unlikability, which aims at preventing non-trusted entities, such as verifiers, from achieving the same goal). However, we have not implemented or evaluated such extensions as part of this work.

4.7 Evaluation

We evaluate the power of RRPs, using a prototype of EDGAR. We compare the space efficiency of EDGAR against a state-of-the-art scheme for backward unlinkability in the PKI setting. We also show that our scheme offers a latency suitable for edge applications. Finally, we evaluate EDGAR's throughput when serving pseudonyms. The source code is available at `https://github.com/claudio-correia/RRP-EDGAR`.

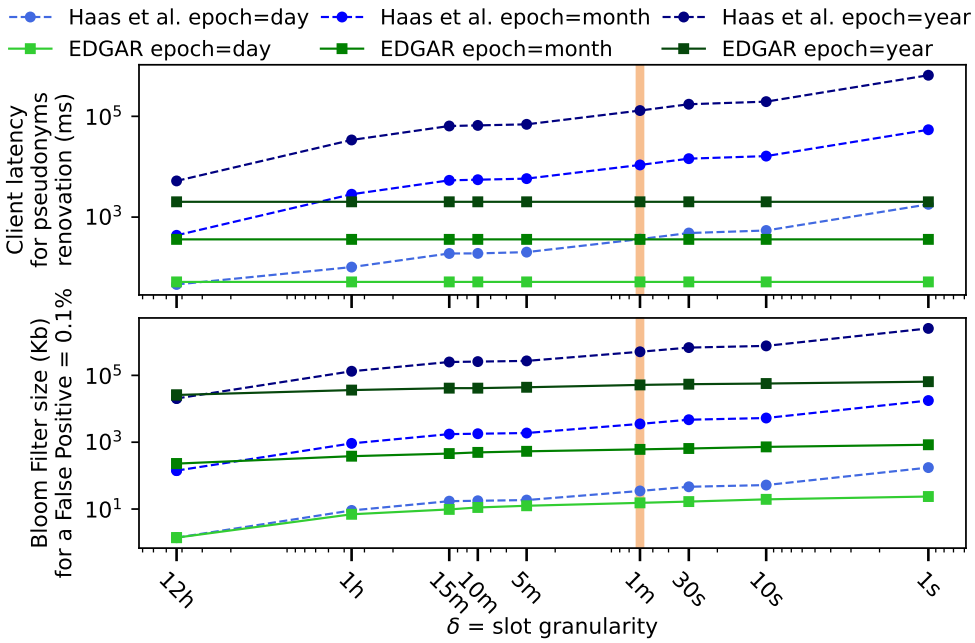


Figure 4.6: Storage and latency in Haas *et al.* Vs EDGAR.

We have implemented both a verifier and a PM server on an Intel NUC10i7FNB. An Intel NUC is an example of what a fog node might be, as it possesses modest computational resources but is relatively inexpensive for large-scale deployments. It has an Intel i7-10710U CPU with Intel SGX, 16GB RAM, and Ubuntu 20.04 LTS. We run the Intel SGX SDK Linux 2.13 Release, Intel SSL-SGX [103] version Linux 2.14_1.1.1k and OpenSSL 1.1.1k. We used a real-world data set composed of multiple vehicle trajectories in the city of Porto [110].

4.7.1 Space Efficiency

We have experimentally compared EDGAR with Haas *et al.*, as both support backward unlinkability by dividing the epoch in time intervals in the PKI setting. Haas *et al.* scheme was more recently implemented in the SCMS POC pilot [46] under the name of linkage values technique. The comparison is not trivial since the pseudonyms in Haas *et al.* are locked to a time slot, being invalid if used in any other, while in RRP the pseudonyms are free to be used at any moment of an epoch.

For a clear comparison, we test both mechanisms in a real-world use case of a taxi company operating in the city of Porto, using a dataset of taxi trajectories [110]. We choose the mix zones strategy [32, 86, 153] for pseudonym changes, i.e., taxis change pseudonyms at crossroads. This use case requires a large number of pseudonyms due to constant vehicle movement, favoring the Haas *et al.* design. In scenarios with fewer pseudonyms needed over the same period, EDGAR will outperform Haas *et al.* by even larger margins.

Figure 4.6 shows the results obtained in our experiment. The top part of the figure presents the user latency experienced when acquiring all pseudonyms for a specific epoch, and the bottom part presents the required Bloom filter size for each solution.

In the dataset, we observed that some taxis drive long distances, requiring 692 pseudonyms per day (which entails 5,677 per moth and 32,142 per year). Since EDGAR clients can use pseudonyms freely, these values were used as I, the number of pseudonyms instances in an epoch. On the other hand, Haas *et al.* must fix the number of pseudonyms in each δ interval, e.g., with $\delta = 1$ min there was a taxi driver who crossed 12 intersections, forcing to fix 12 different pseudonyms for each δ interval, which has an explosive effect on the number of pseudonyms generated. Furthermore, as we tighten the δ interval, the difference between Haas *et al.* and EDGAR is more pronounced, for $\delta = 1$ sec and epoch = 1 year we observe an improvement of $\approx 2.5 * 10^6$KB to $\approx 6.4 * 10^4$KB, two orders of magnitude lower in the required storage. This results from the logarithmic effect provided by RRPs, while Haas *et al.* suffers from a linear effect. For large values of δ, when an epoch is divided into a few time slots, Haas *et al.* slightly outperforms EDGAR, since the overhead imposed by the latchkey hierarchy is no longer compensated by a significant reduction in pseudonyms.

Finally, we also observed that a large number of taxi trips take around 10 min., so we believe that $\delta = 1$ min would be a reasonable configuration for this use case, representing a storage saving between 35KB and 15KB (for epoch = 1 day) and 508MB to 51MB (for epoch = 1 year) by implementing EDGAR instead of Haas *et al.*, highlighted with the vertical orange line.

4.7.2 δ Granularity vs Latency Trade-off

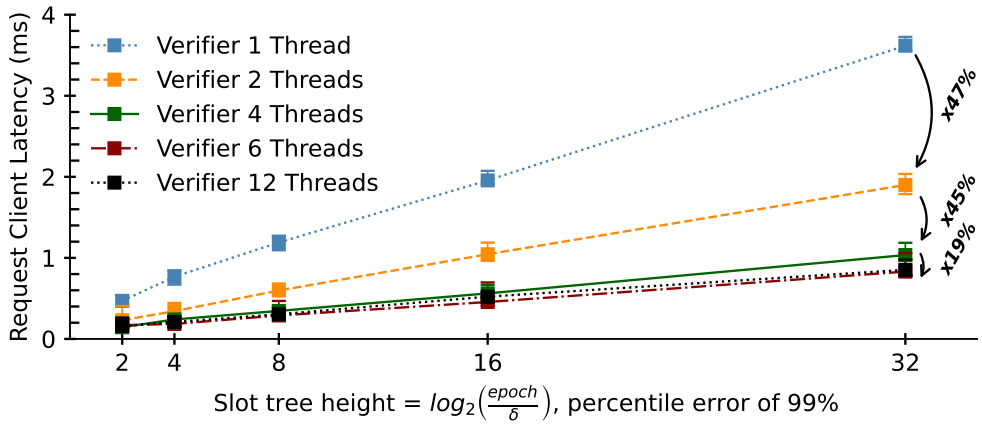


Figure 4.7: Client latency for capability verification, for different levels of granularity and number of threads.

The use of the latchkey hierarchy makes revocation of a range of time slots efficient, because a single latchkey can be used to revoke many capabilities. The efficiency of revocation comes at the cost of penalty in the authentication procedure, because the verifier must check a number of latchkeys equal to the tree height (instead of verifying only the leaf latchkey). Fortunately, the height of the latchkey tree grows only logarithmically, and latchkeys can be verified in parallel. Therefore, the penalty of RRPs on latency is small. Figure 4.7 shows the latency of the verification procedure as the granularity of the time slots increases. For instance, a system that uses $\delta = 1$ min and an epoch of one day requires a binary latchkey tree of depth 11; if the epoch is increased to a month, the depth of the binary tree increases to 16. In such a case, clients will incur a latency below $2ms$ in a single-thread verifier, still an acceptable latency for edge applications. Note that a tree with 32 levels can support an extreme large number of time

slots, such as the ones resulting from using $\delta = 1$ sec and epoch = 70 years; even in this case, clients would experience only 3.5 ms of latency. We have used multithreading to parallelize the verification of latchkeys, reducing the impact on latency. We observe a latency reduction that is linear with the number of cores of 47% and 45% from 1 to 2 threads and from 2 to 4 threads, respectively. Hyperthreading, from 6 to 12 threads, offers no improvement since most of the time is spent in cryptographic operations, and each core has a single ALU.

4.7.3 PM Server Throughput

We use the Ed25519 scheme [33] to obtain deterministic digital signatures, but is not yet available in the SGX SDK. Since the PM is responsible for generating the pseudonyms and runs inside the enclave, we implemented two different versions of the Ed25519 inside the enclave: a portable one [152] and one based on OpenSSL [103].

The portable version is straightforward to implement in any type of TEE, but the lack of optimization affects its performance. In the second implementation, we use the Intel SSL-SGX library to import the OpenSSL library into the enclave. This library was designed for the SGX enclaves, being the most efficient implementation of the scheme. We also evaluated the system with and without SGX, using the OpenSSL library outside the enclave as well.

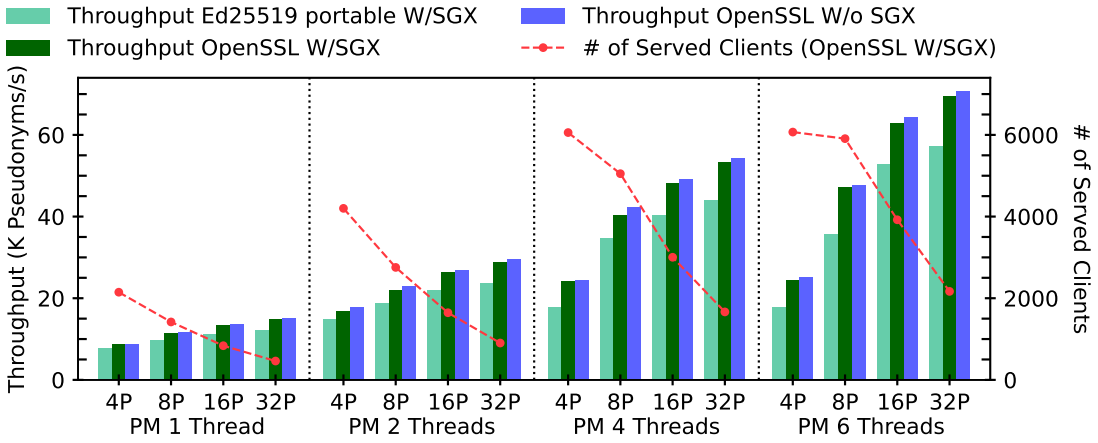


Figure 4.8: Pseudonym throughput and number of served clients by the PM w/ and w/o SGX. P is the number of pseudonyms generated in each request.

Figure 4.8 presents the pseudonym throughput for each of these implementations. We vary the number of threads on the PM node side, improving throughput as expected. Additionally, we vary the number of pseudonyms that each client requests from the PM, from 4, 8, 16, and 32 new pseudonyms. We observe that increasing the requested pseudonyms also increases throughput by reducing other system overheads, such as signing and encrypting. On the contrary, the number of served clients decreases since the PM server requires more time for each request. Following the previous discussion and fixing $\delta = 1$ min, a taxi could request 32 new pseudonyms in just 133 ms, useful for at least the next 32 min.

Summary

In this chapter we presented Range-Revocable Pseudonyms, an abstraction that supports an anonymous authentication scheme based on pseudonyms that is able to enforce backward unlinkability with storage costs that are multiple orders of magnitude lower than those of the related work. The gains derive from our novel technique to decouple pseudonyms from time slots, and perform authentication/revocation based on the use of latchkeys that can be generated from a given pseudonym for any desired time slot. This technique prevents clients from having to store a large number of unnecessary pseudonyms. As a proof of concept, we have designed and implemented a prototype of EDGAR, an authentication system for the edge based on the use of RRPs. We have used this prototype to perform an experimental evaluation using a real dataset of vehicle traces. The results show that EDGAR is capable of offering low latency ($0.5 - 3.5$ ms) and storage savings (even when using time slots as small as 1 second) when serving clients.

We motivated our work using a VANET scenario, as one of the most prominent use cases for anonymous authentication at the edge computing environment. Both VANETs and other applications, such as crowdsensing, supply chain tracking, and augmented reality, will be integrated with or complemented by edge storage services. Our RRPs can be implemented at any level of the application stack, whether at the application level or the storage level. Anonymous authentication is a crucial property for the adoption and deployment of edge applications, including the storage systems that are the focus of this thesis.

In the next chapter, we present the Privacy Keeper, an alternative anonymous authentication scheme to RRPs. The Privacy Keeper is capable of providing backward unlinkability and revocation auditability simultaneously. However, the enhanced security offered by the Privacy Keeper comes at a higher cost in the authentication process, which may not be tolerable for all edge applications as it requires downloading the revocation list at the time of authentication.

Algorithm 1: ERCSet creation, trusted PM side

$\mathcal{C} = \langle C^a, \ldots, C^b \rangle$: capabilities to be revoked for a pseudonym

Function MERGELATCHKEYS($\mathcal{C}$)**:**
  latchkeySet $\leftarrow \emptyset$ // *latchkeySet* is a genuine set (not a multiset)
  foreach $c_i \in \mathcal{C}$ **do**
    latchkeySet $\leftarrow$ *latchkeySet* $\cup$ EXTRACTLATCHKEYS(c_i)
  end
  return *latchkeySet*

Function REMOVEUNSAFE($\mathcal{L}$)**:**
  Safe $= \mathcal{L}$
  `/* Remove latchkeys from non-revoked time slots */`
  foreach $l^i \in \mathcal{L}$ **do**
    `/* Returns all children nodes/latchkeys of` l_i `*/`
    $descendants_i \leftarrow$ GETLATCHKEYSSUBTREE(l_i)
    if $\exists x \in descendants_i \wedge x \notin \mathcal{L}$ **then**
      Safe $\leftarrow$ *Safe* $\setminus\, l_i$
    end
  end
  return *Safe*

Function REMOVEREDUNDANT($\mathcal{L}$)**:**
  NonRedundantSet $= \mathcal{L}$
  `/* Remove latchkeys that are covered by parent */`
  foreach $l_i \in \mathcal{L}$ **do**
    $parent_i \leftarrow$ GETPARENTLATCHKEY(l_i)
    if $parent_i \in \mathcal{L}$ **then**
      NonRedundantSet $\leftarrow$ *NonRedundantSet* $\setminus\, l_i$
    end
  end
  return *NonRedundantSet*

Function LATCHKEYSENCODING($\mathcal{L}$)**:**
  ERCSet $\leftarrow$ CREATENEWBF()
  foreach $l_i \in \mathcal{L}$ **do**
    ERCSet.BFADD(l_i)
  end
  return *ERCSet*

Function CREATEERCSET($\mathcal{C}$)**:**
  Enclave Zone Start
  Unfiltered $\leftarrow$ MERGELATCHKEYS($\mathcal{C}$)
  Safe $\leftarrow$ REMOVEUNSAFE(*Unfiltered*)
  SafeNonRedundant $\leftarrow$ REMOVEREDUNDANT(*Safe*)
  ERCSet $\leftarrow$ LATCHKEYSENCODING(*SafeNonRedundant*)
  Enclave Zone End
  return *ERCSet*

5 Providing Revocation Auditability: Privacy Keeper

This chapter introduces and evaluates a more secure anonymous authentication scheme than RRPs, named Privacy Keeper. It not only provides backward unlinkability but goes a step further by also providing revocation auditability, a missing property in RRPs to achieve complete authentication anonymity. Similarly to RRPs, Privacy Keeper is entirely based on public key encryption.

This chapter is structured as follows: Section 5.1 provides the motivation and defines the objectives of Privacy Keeper, while Section 5.2 reviews related work. The solution is detailed in Section 5.3. Finally, Section 5.5 presents experimental results evaluating the performance of Privacy Keeper.

5.1 Motivation and Goals

Privacy is an increasingly vital requirement for modern-day customers. However, large companies are actively exploring novel approaches to monetize the abundance of available data. Unfortunately, as previously discussed, the process of client authentication on applications or services often exposes sensitive information that can be exploited for monetary gain. To address this issue, many anonymous authentication schemes offer solutions that safeguard client privacy during multiple authentication instances, even after revocation. Yet, revoking a client without breaking anonymity presents a significant challenge since the revocation data can be used to link past and future anonymous authentications from a client [96].

Existing solutions that try to provide anonymous authentication leverage various mechanisms, including efficient approaches like public key encryption [182] and Group Signatures (GS) [171], as well as more robust techniques like ZKP [180]. A crucial aspect of these solutions is their compliance to Verifier Local Revocation (VLR) [37, 42], which involves publishing Revocation Lists (RLs) to verifiers, maintaining the availability of large distributed applications.

PKI-based solutions are more appealing due to their efficiency and simplicity, resulting in widespread adoption in real-world applications [165, 188, 78, 69, 83]. In the RRPs presented in Chapter 4 we have achieved near perfect backward unlinkability [95, 113] while relying exclusively on public key encryption, whereas prior solutions required Non-Interactive Zero Knowledge

Proofs (NZKPs) [19, 181]. We achieved backward unlinkability in RRPs by employing short time intervals for proof generation, which introduces trade-offs between storage and privacy.

Despite the fact that RRPs solve the backward unlinkability problem, RRPs still suffer from a limitation; they are unable to offer *revocation auditability*. This feature is crucial to prevent situations where a malicious service provider accepts authentication requests from revoked users, thereby reducing the size of the user's anonymity set without his awareness.

Following the same rational as in RRPs, we want a scheme that provides revocation auditability using only public key encryption, a challenge not yet achieved. A direct benefit of this choice is that clients no longer suffer the burden of cryptographically intensive proofs, as is required in solutions that rely on NZKPs. In this chapter, we present a new scheme, thatt wa have named *Privacy Keeper*, which has the following features:

Goal: Propose an efficient authentication scheme, Privacy Keeper, designed to provide revocation auditability. Our approach shares similarities with the mechanisms used by NZKPs for providing revocation auditability [96]. However, our scheme relies solely on public key encryption. Besides revocation auditability, our solution provides backward unlinkability without depending on short time intervals, as many previous solutions did [95, 113, 156, 173] (including our own [59]). This feature allows Privacy Keeper to achieve immediate revocation for clients by publishing RLs without any time interval delay, while also requiring smaller RLs.

By avoiding computational costs on the client side, Privacy Keeper introduces some complexity in RL construction on the server side. To mitigate the effects of this complexity, we introduce an optimization for Privacy Keeper that allows for the pre-computation of RLs. In our evaluation, we demonstrate that despite the cost associated with creating our RLs, Privacy Keeper offers acceptable latencies in RL construction.

5.2 Related Work

Despite the existence of various schemes for anonymous authentication [59, 96], they face a significant challenge in maintaining client anonymity after their revocation [96]. This issue arises in authentication systems that respect VLR [37, 42], as it requires to publish revocation lists. As described in Section 2.3.2, the existence of these lists allows an attacker to flag/identify the authentication instances performed by a revoked client, which can be exploited to link multiple authentications and, consequently, compromise the client's anonymity. An adversary can exploit these lists to link past and future authentications. Backward unlinkability is the property that safeguards past authentication anonymity, achieved by RRPs in Chapter 4. Instead, revocation auditability, our main focus in this chapter, aims to provide unlinkability in future authentications performed by a revoked user. In the next sections, we provide details of how the related work provides both of these properties.

5.2.1 Revocation Auditability

Informally, revocation auditability means that a user has the ability to verify his revocation status at a service provider before attempting to authenticate [96, 182]. If the user is indeed

revoked, he can then safely disconnect from the service without disclosing any potentially sensitive information. Otherwise, clients could be revoked without their knowledge, and a malicious service provider might still accept authentication requests from these revoked users, thereby compromising their privacy. Such an attack can lead to severe privacy breaches, enabling the service provider to link all of the user's actions, which is especially concerning when schemes rely on uncircumventable forward linkability for achieving revocability. In the literature on anonymous authentication we have identified four main techniques to achieve revocation auditablity:

Central authority: The simplest solution is to have clients access a central and remote node that issues revocation lists. Before each authentication, clients pull the most recent list and verify their status in the system (i.e., whether they have been revoked). However, this solution is not desirable for large-scale and dynamic systems like distributed edge storage systems or VANETs, where clients have intermittent connections, and reliance on a central node can lead to multiple availability failures. Additionally, it does not respect VLR.

Contract-based revocation: Another approach is to use contract-based revocation [96, 166, 124], where the contract semantics are agreed upon by both the user and the provider. This enables the user to determine whether a certain action will constitute misbehavior before deciding whether to engage in it. Thus, the client is aware that it may be revoked after such an action. Unfortunately, due to the large variety of applications nowadays, it is very difficult to define all the possible behaviors a client may exhibit, making these approaches inflexible and impractical.

Revocation list freshness: A more desirable approach is to ensure that fresh revocation information reaches the client. This is achieved by having RLs published at regular Δt time intervals, containing a signature with the corresponding timestamp [182]. When a client performs authentication, it can first request the local revocation list, which must have a fresh signature for the current Δt, and then check if it has not been revoked; otherwise, it should halt the authentication process. This is a practical and easily deployable solution, but the downside is that Δt imposes a tradeoff between system availability (clients do not authenticate if revocation information is not fresh) and effective revocation (the larger the Δt, the longer it takes for a revocation to take effect).

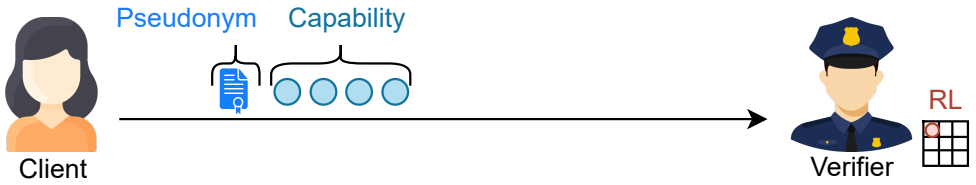
Non-Revocation proof based on the RL: The more secure approach is to have clients locally generate a non-revocation proof unique to the presented RL (before authentication, the client downloads the list from the local provider). This guarantees that the generated proof cannot be tested against another RL (that may contain the client), and it is only valid for the locally presented RL. This solution ensures system availability and is flexible. However, this solution has only been applied with NZNPs [19, 181], where clients prove that their pseudonym is not in the presented RL. Unfortunately, the use of NZNPs requires clients to construct complex proofs, in the authentication instant, suffering a high latency and computation cost.

In this chapter, we present a novel solution that shares the same security properties as a non-revocation proof based on NZNPs, but our solution is solely based on public key encryption, avoiding heavy cryptographic operations.

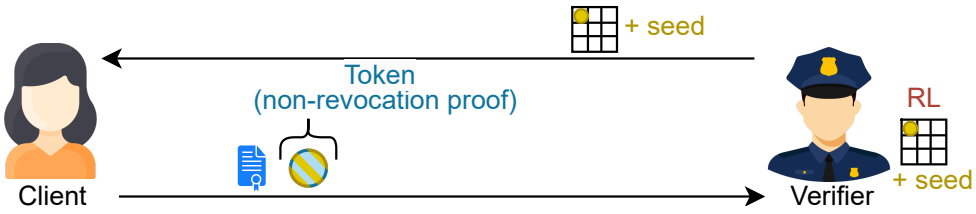
5.2.2 Backward Unlinkability

When providing revocation auditability, we want to continue to ensure that all authentications performed by a user in the past remain anonymous after revocation, i.e., we want to continue to

provide backward unlinkability [96]. Both our RRPs and the related scheme of Haas et al. [95] aim to achieve backward unlinkability in pseudonym schemes. Both solutions follow a similar approach in which time is divided into large epochs and then further subdivided into smaller time intervals known as slots. This time division allows either locking pseudonyms (Haas et al.) or cryptographic proofs (RRPs) to specific time slots. During revocation, only information related to future time slots is used in RLs, and no information relative to past time slots is used, thus achieving backward unlinkability.



(a) Authentication in RRP, client is not aware of its revocation status.



(b) Authentication in Privacy Keeper, proof is generated based on the RL.

Figure 5.1: Difference overview between authentication from Privacy Keeper and RRPs. RL mean revocation list.

Both of these schemes provided contributions in achieving backward unlinkability in the pseudonyms domain. Privacy Keeper takes a step forward and provides backward unlinkability without resorting to time slots. This simple difference is crucial, as in both previous schemes, the level of privacy achieved depends on the size of the time slots. In other words, the smaller the slots, the higher the level of privacy. However, this has consequences for the amount of information required for the system's operation. For instance, in RRPs, the smaller the time slots, the larger the size of the authentication proofs/capabilities. In Figure 5.1, we present a simplified illustration of how authentication is carried out in RRPs, where the client sends the pseudonym and a capability generated from the pseudonym at the moment of authentication. The size of this capability is directly related to the slot dimensions. In contrast to RRPs, Privacy Keeper does not use time slots, resulting in constant size proofs and instant revocation. The only disadvantage is the need to download the RL, and only in the case of the client having an outdated RL. Finally, since our proofs are generated uniquely for each RL, we achieve perfect backward unlinkability, because, previously used authentication data will never be found in future RLs.

5.3 Privacy Keeper

We now introduce our scheme for anonymous authentication, Privacy Keeper, the first system to offer perfect revocation auditability and backward unlinkability simultaneously in a pseudonymous domain. We use the term "perfect" because both of these properties are

fully achieved, without trade-offs, unlike RRPs and Haas et al., where the size of time slots compromises the quality of backward unlinkability provided. Furthermore, our solution enables immediate revocation, letting the new RL be published without delay, while systems based on time slots must wait for the slot to end to publish the RL.

In Privacy Keeper, the authentication is performed against a Revocation List (RL). A RL includes a unique seed and an encoded set of revoked access tokens. Revoked access tokens are encoded in an RL using a secure one-way function: it is possible to check if a token is in the set but impossible to extract tokens from the set. The authentication requires the client to transmit a pseudonym and an access token that is not included in the set of revoked access tokens. This token proofs that the given pseudonym was not revoked; this is conceptually similar to the non-revocation proofs used in NZNPs [19, 181], as illustrated in Fig. 5.1b. By using access tokens, our scheme enables client revocation without disclosing pseudonyms. Tokens have a similar purpose to the capabilities found in RRPs; we use a distinct name to prevent any confusion. To achieve auditability, we require tokens to be uniquely linked to a givenRL, similarly to the non-revocation proofs used in NZNPs. To accomplish this, clients request a RL from the verifier at the time of authentication and generate their token exclusively for that RL. Our scheme achieves this by letting access tokens for a given RL be a function of the unique seed of that RL. Clients then generate the token using this seed, establishing a unique connection between the token and the presented RL, as depicted in Fig. 5.1b. The verifier is then responsible for cryptographically checking that all the presented information has been correctly constructed, for instance, verifying that the presented token corresponds to the RL seed. In this work, we follow the same system and threat model as RRPs, presented in Section 4.3.

5.3.1 Implementation

We will now describe in more detail how the tokens used for authentication and revocation are constructed.

Entities: In Privacy Keeper we have the same three entities as those introduced in RRPs in Section 4.3.1, where clients store pseudonyms and authenticate towards verifiers. We also assume the presence of a trusted entity, such as a Certificate Authority (CA) that operates within a Trust Execution Environment (TEE) to securely store and perform computations on sensitive data.

Nomenclature: We assume $\langle K^+, K^- \rangle$ is a pair of public and private asymmetric keys, respectively. A digital signature is defined by $\{digest(a)\}^{K^-}$, where the private key K^- is used to sign the digest of the content of a.

Time: In this scheme, time is divided into epoch periods with a duration of e. These epochs are relatively long time intervals, such as one year or one month. Each pseudonym is associated with a specific epoch period. This is crucial to restrict the number of pseudonyms that exist at any given moment. It's important to emphasize that epochs are lengthy time intervals, in contrast to the time slots in RRPs, which, for example, can have a duration of just one minute.

Client Storage: We assume that a client has performed some initial setup and stores locally multiple pseudonyms valid for the current epoch. Each pseudonym p has a public and private key pair, $\langle K_p^+, K_p^- \rangle$, and the pseudonym is defined by its public key and a digital signature from

a CA, $p = \langle K_p^+, \{K_p^+\}^{K_{CA}^-} \rangle$. The client holds a number I of valid pseudonyms for the current epoch.

Access Token: An access token $\sigma_x^{RL_z}$ in Privacy Keeper is always cryptographically bounded to a pseudonym p_x and a unique $seed_z$. The token is used to proof that some selected pseudonym, $p_x = \langle K_{p_x}^+, \{K_{p_x}^+\}^{K_{CA}^-} \rangle, x \in [1, I]$, has not been revoked in the given revocation list RL_z. The next paragraph explains how RL_z is uniquely bounded to $seed_z$. The construction of our token is quite straightforward. It involves generating a digital signature using the private key associated with p_x over the digest of the provided $seed_z$. Therefore, $\sigma_x^{RL_z}$ is defined as $\{digest(seed_z)\}^{K_{p_x}^-}$. For authentication, the client forwards both the selected pseudonym and the corresponding token, denoted as $\langle p_x, \sigma_x^{RL_z} \rangle$, as depicted in Fig. 5.1b.

Revocation Lists: In Privacy Keeper, the RLs include an unique seed and an encoded set of revoked access tokens. Each RL is constructed by a trusted entity, the CA running inside a TEE, similarly to RRPs. This CA has the capability to generate all the access tokens that can be ties to any given RL. The generation of RLs is the only aspect of our scheme that has some computational impact because each RL is dependent on a distinct seed. Therefore, when the CA needs to revoke a client at a specific time t, it first generates a random value, which becomes the $seed_z$ associated with a new revocation list, RL_z. It then retrieves each pseudonym of the recently revoked client, along with all other pseudonyms that were revoked in the current epoch, from other previously revoked clients. Subsequently, it calculates the digest of the seed and generates each token using all the revoked pseudonyms, resulting in $\langle \{digest(seed_z)\}^{K_{p_i}^-} \rangle, \forall i \in R_{set}$, where R_{set} is the set of all revoked pseudonyms in the current epoch. Each of these tokens is encoded in the RL's set of revoked access tokens using a secure one-way function. We denote the resulting encoded set of revoked access tokens $esrat_z$. Finally, the CA creates a final digital signature covering both the RL and the seed together, denoted as $\{digest(esrat_z \parallel seed_z)\}^{K_{CA}^-}$. The CA can then *immediately publish* these three elements: the seed, the encoded set of revoked access tokens, and the signature, which define the new RL.

In our scheme we ensure the verifier cannot provide a fake RL_z by leveraging digital signatures for authenticity. However, similar to related work, a malicious verifier might provide an old RL, misleading a revoked client into a successful authentication attempt. Such scenarios undermine revocation auditability in schemes like RRPs, where verifiers could exploit authentication information against newer RLs, potentially violating privacy of a revoked client. Unlike these schemes, Privacy Keeper does not suffer from this vulnerability and preserves revocation auditability. Our scheme uniquely binds the authentication information with the provided RL (through the seed), rendering it useless to test across different or newer RL versions, regardless of the user's revocation status. Next, we explain how our unique token can still provide proof of non-revocation.

Non-Revocation proof: For authentication, the client must provide a pseudonym and a non-revocation proof that this pseudonym is valid. The proof is provided by presenting a valid access token, associated with the given pseudonym *and* to the unique seed of the RL that is not included in the RL's set of revoked access tokens. In detail, the process to demonstrate that the given pseudonym p_x is not revoked is as follows.

In the first step, the client must obtain a revocation list RL_z and its associated $seed_z$ from the verifier. Subsequently, the client validates the signature, generated by the CA, on the revocation list. This validation confirms the integrity and authenticity of both the revocation list and the

seed. After this verification, the client generates a token $\sigma_x^{RL_z}$ for a selected pseudonym p_x, as described earlier. It is important to note that this access token is only applicable for testing against RL_z and it is not valid to be tested against any other RL. Additionally, as previously mentioned, the verifier cannot provide a fake RL_z.

Consequently, it is the verifier's responsibility to validate $\sigma_x^{RL_z}$ by ensuring that it has been correctly constructed, that it corresponds to the presented pseudonym p_x, and that it has not yet been revoked. To achieve this, the verifier first checks if the pseudonym has a valid signature from the CA. Subsequently, it uses the provided pseudonym's public key, $K_{p_x}^+$, to verify the digital signature within $\sigma_x^{RL_z}$. This signature must be correctly constructed using $K_{p_x}^-$ and must correspond to the correct $seed_z$ from RL_z. If this is confirmed, it indicates that the proof has been correctly constructed and corresponds to the presented pseudonym p_x. The next step for the verifier is to ascertain whether the access token $\sigma_x^{RL_z}$ can be found in the encoded set of revoked access tokens of RL_z. If this access token belongs to the encoded set of revoked access tokens, the pseudonym has been revoked; otherwise, the access token provides proof that this pseudonym has not been revoked and is valid.

Authentication: At a high level, authentication begins with the client downloading the RL from the local verifier. Then, the client selects a pseudonym and generates am access token for the seed associated with RL, using the private key of this pseudonym. Next, the client checks whether it has been revoked by testing if the token is found in the RL's encoded set of revoked access tokens. If not, both the access token and pseudonym are sent to the verifier, which authenticates the token and tests it against the same RL. If the token is found in the RL's encoded set of revoked access tokens, the client is considered revoked; otherwise, they it is deemed valid, and the authentication is accepted. The step of downloading the list is necessary to ensure auditability, enabling the client to verify his status. It is important to note that even if the verifier presents an older RL, the proof generated by the client is only valid for that specific RL (unlike RRPs), thus guaranteeing revocation auditability.

5.3.2 Optimization

We now introduce two enhancements designed to optimize the performance and facilitate the adoption of our Privacy Keeper

RL based on Bloom Filter: Our scheme requires encoding all revoked access tokens using a one-way hash function to prevent direct access to the inserted data. Only when the client provides a token, can the verifier ascertain if it belongs to encoded set of revoked access tokens. To construct the encoded set of revoked access tokens we use Bloom Filters [59, 95]. Bloom Filters enable data compression, occupying less space, and offering constant-time check.

Precomputed RLs: Since the most resource-intensive part of our system is the construction of the RL from scratch every time a new client is revoked, we have implemented an optimization to accelerate this process. This optimization hinges on the CA consistently maintaining a precomputed RL. In other words, the CA consistently prepares in advance a new version of the RL, selecting a random seed and generating all the revoked access tokens associated with that RL for all previously revoked users. When there is a need to revoke a new client, it is merely a matter of generating the revoked access tokens for the new client and inserting them into the encoded set of revoked access tokens. This precomputation significantly reduces the time required to generate a new RL and to publish it, making revocation more effective.

5.4 Security Proof

In this section, we first present a proof that Privacy Keeper preserves unlinkability, and afterward we prove that Privacy Keeper is also capable of offering auditability.

5.4.1 Unlikability

We now provide a proof that Privacy Keeper offers full unlinkability, meaning that revocation information cannot be linked to the information used by clients when authenticating before and after the revocation. This proof is simpler than the one presented in Section 4.5 for the RRPs, since the scheme is also relatively simpler. Our scheme provides unlinkability because access tokens are only valid for a specific RL and, therefore, the revoked access tokens that are encoded in different RLs are necessarily distinct.

Assumption 4. *There is a secure one-way function.*

We assume the availability of a secure collision-resistant hash function $\text{H}()$, such as SHA256, that is easy to compute on every input, but not possible to invert given the output.

Assumption 5. *Only the client and the CA can access pseudonyms.*

We assume that both the client and the CA will securely store the secret private key of each pseudonym and never disclose it. The CA is responsible for generating different pseudonyms for a client. The cryptographic keys $\langle K^-, K^+ \rangle$ of a pseudonym are randomly generated by the CA using a secure cryptographic scheme.

Assumption 6. *There is a secure deterministic digital signature scheme.*

We assume the availability of a secure deterministic digital signature scheme, such as Ed25519 [33], which ensures the usual authentication, integrity, and non-repudiation properties. A deterministic signature is generated by $\{\text{H}(a)\}^{K^-}$ using K^- to sign the digest of the content of a, and can be verified with K^+.

Assumption 7. *The client picks an unused pseudonym to authenticate.*

For each authentication, the client chooses a pseudonym that has never been used. If it runs out of pseudonyms to use, it requests more from the CA.

Assumption 8. *Given a revocation list RL_z, the client only proceeds with authentication with a pseudonym p_x if p_x has not been added to the RL_z's encoded set of revoked access tokens.*

When performing authentication, the client requests the current RL_z from the verifier, and checks if pseudonym p_x has been revoked RL_z. We recall that p_x has been revoked RL_z if $\sigma_x^{RL_z}$ belongs to the RL's encoded set of revoked access tokens. If the client finds that p has been revoked in RL, it stops from using the service and does not provide any more information; otherwise, it continues with the authentication.

Definition 10. *A valid pseudonym p_x is defined as:*

$$p_x = \langle K_x^-, K_x^+, \{\text{H}(K_x^+)\}^{K_{CA}^-} \rangle$$

Where (K_x^-, K_x^+) represent the private and public key of the pseudonyms respectively, and $\{\text{H}(K_x^+)\}^{K_{CA}^-}$ is a digital signature of the CA over the public key to provide authenticity for the pseudonym. From Assumption 5, the CA provides the pseudonyms to clients with the respective signature.

Remark 1. *A valid token σ for revocation list RL is a unique digital signature over the RL's seed.*

A token $\sigma_x^{RL_z}$ represents the unique bond of a pseudonym p_x to a given revocation list RL_z. This bond is implemented by a digital signature performed over the unique $seed_z$ that accompanies the list, where $\sigma_x^{RL_z} = \{\text{H}(seed_z)\}^{K_x^-}$. By using the private key K_x^- of the pseudonym (from Definition 10), we enforce authentication, integrity, and non-repudiation of the token (from Assumption 6).

Remark 2. *A RL is immutable, unique, and cannot be tempered with.*

All the revocation lists are generated and published by the CA. For a new RL_z, the CA generates a unique and random $seed_z$, then computes a revoked access token for each revoked pseudonym as $\sigma_i^{RL_z} = \{\text{H}(seed_z)\}^{K_{p_i}^-}$ (from Remark 1) and then encodes each revoked access token using a one-way hash function as $\text{H}(\sigma_i^{RL_z})$ (from Assumption 4). The encoded set of revoked access tokens $esrat_z$ is then bound to the $seed_z$ through a digital signature that proves its authenticity, resulting in $\{\text{H}(esrat_z \parallel seed_z)\}^{K_{CA}^-}$. Any atempt to tamper the RL will invalidate its signature. The CA will never reuse the same seed, making each encoded set of revoked access tokens unique and immutable.

Remark 3. *A valid authentication request is defined as:*

$$\langle x, \sigma_x^{RL_z} \rangle$$

When autenticating the client must provide the authentication request where x is defined as the public information of some pseudonym p_x, specifically the public key and the CA signature $\langle K_{p_x}^+, \{\text{H}(K_{p_x}^+)\}^{K_{CA}^-} \rangle$. Before presenting this information towards the verifier, the client requests the current revocation list RL_z and generates $\sigma_x^{RL_z}$ from Remark 1. Following Assumption 8, if the client does not find $\sigma_x^{RL_z}$ in the encoded set of revoked access tokens for RL_z, it proceeds with the authentication by sending the authentication request $\langle x, \sigma_x^{RL_z} \rangle$ to the verifier.

Theorem 3. *Two different authentications cannot be linked to the same client.*

Proof. Consider an authentication from client c that provides an authentication request $\langle x, \sigma_x^{RL_i} \rangle$ for pseudonym $x \in p_c$ in face of some revocation list RL_i. Consider another authentication from client c' that provides the request $\langle y, \sigma_y^{RL_j} \rangle$ for pseudonym $y \in p_{c'}$ in face of some revocation list RL_j. For an attacker to successfully link the two authentications, it needs to infer that $c = c'$.

There are two ways for an attacker to achieve this goal.

One is to assert that x and y belong to the same client, i.e.:

$$\text{ASSERT}(\exists_{c,\mathcal{P}_c},\ x \in p_c \ \wedge \ y \in p_c) \tag{5.1}$$

This condition holds true when x and y belong to the same pseudonym p_c, and according to Remark 3, this would require x and y to share the same public key $K^+_{p_c}$. However, by Assumption 7, the client never uses the same pseudonym twice. Additionally, according to Assumption 5, the CA generates all the public and private keys of each pseudonym using a secure, random, and invertible cryptographic scheme, this will result in unlinkable and random pseudonyms by construction. Therefore, an attacker will always observe $x \neq y$ and never be able to assert Equation 5.1 as true.

The other way is to use the information in some other revocation list RL_k to link pseudonyms x and y. Let us use $RL.esrat$ to denote the encoded set of revoked access tokens of revocation list RL.

$$\text{ASSERT}(\exists_{RL_k},\ \text{H}(\sigma_x^{RL_k}) \in RL_k.esrat\ \wedge\ \text{H}(\sigma_y^{RL_k}) \in RL_k.esrat) \tag{5.2}$$

This condition holds true if the attacker can verify that exist a list RL_k that contains encoded tokens for both pseudonyms x and y. Because tokens are encoded by secure one-way functions, the attacker cannot extract $\sigma_x^{RL_k}$ and $\sigma_y^{RL_k}$ from RL_k or $RL_k.esrat$ and therefore these must be provided by the client. By construction (Assumption 8), the client will not provide $\sigma_x^{RL_k}$ if $\text{H}(\sigma_x^{RL_k}) \in RL_k.esrat$. Thus, the attacker can only have access to $\sigma_x^{RL_k}$ if there is some other $RL_{k'}$ such that $\sigma_x^{RL_k} = \sigma_x^{RL_{k'}}$, however, by Remark 2, this is impossible, because tokens are unique, given that they depend cryptographically on different seeds. Therefore, the attacker is not able use some public revocation list to assert Equation 5.2 as true.

Therefore, we can conclude that the attacker is incapable of leveraging the available information to assert either Condition 5.1 or 5.2 as true. This implies that the attacker cannot infer whether $c = c'$. Consequently, an attacker is unable to link two different authentication attempts to a single client.

□

5.4.2 Auditability

We now provide a proof that Privacy Keeper offers full auditability.

Lemma 4. *After a pseudonym p_x is revoked, any RL presented to clients must encode $\sigma_x^{RL_z}$ in the RL's encoded set of revoked tokens.*

Proof. A client authenticates against a given RL by selecting a pseudonym p_x and presenting $\langle x, \sigma_x^{RL_z} \rangle$. Authentication is granted if $\sigma_x^{RL_z}$ has not been added to the RL's encoded set of revoked tokens. Thus, for revocation of pseudonym p_x to succeed, $\sigma_x^{RL_z}$ must be added to the RL's encoded set of revoked tokens of all RL presented to a client after revocation. □

Lemma 5. *A client authenticates using pseudonym p_x only if, when presented with a revocation RL, it cannot find $\sigma_x^{RL_z}$ in the RL's encoded set of revoked tokens.*

Proof. A client authenticates against a given RL by selecting a pseudonym p_x and presenting $\langle x, \sigma_x^{RL_z} \rangle$. By construction (see Assumption 8) a client will only authenticate using p_x if it cannot find $\sigma_x^{RL_z}$ in the RL's encoded set of revoked tokens. □

Theorem 4. *Privacy Keeper ensures auditability*

Proof. The proof, is by contradition. A client is not guaranteed auditability if it attempts to authenticate with a pseudonym p_x that has been revoked. Let p_x be a pseudonym used by some client to perform authentication against some revocation list RL. By Lemma 4, if pseudonym p_x has been revoked $\sigma_x^{RL_z}$ must be encoded in the RL's encoded set of revoked tokens. By Remark 2 RLs are immutable and cannot be tempered with by the verifier. By Lemma 5, if the client authenticates, it cannot find $\sigma_x^{RL_z}$ in the RL's encoded set of revoked tokens. A contradiction. □

5.5 Evaluation

We evaluated our system against the two main solutions that offer backward unlinkability in public key encryption, namely Haas et al. and RRPs, despite the fact that these systems do not provide revocation auditability, a distinctive feature that Privacy Keeper uniquely introduces. For our evaluation, we followed a similar assessment as in RRPs, taking advantage of an Intel NUC10i7FNB, with an Intel i7-10710U CPU with Intel SGX, 16GB RAM, and Ubuntu 20.04 LTS. Additionally, to ensure a fair comparison, we selected the ideal parameters for RRPs, an epoch of one month and a time slot of one minute. This means that for Haas et al. it requires at least one different pseudonym for each minute of the month.

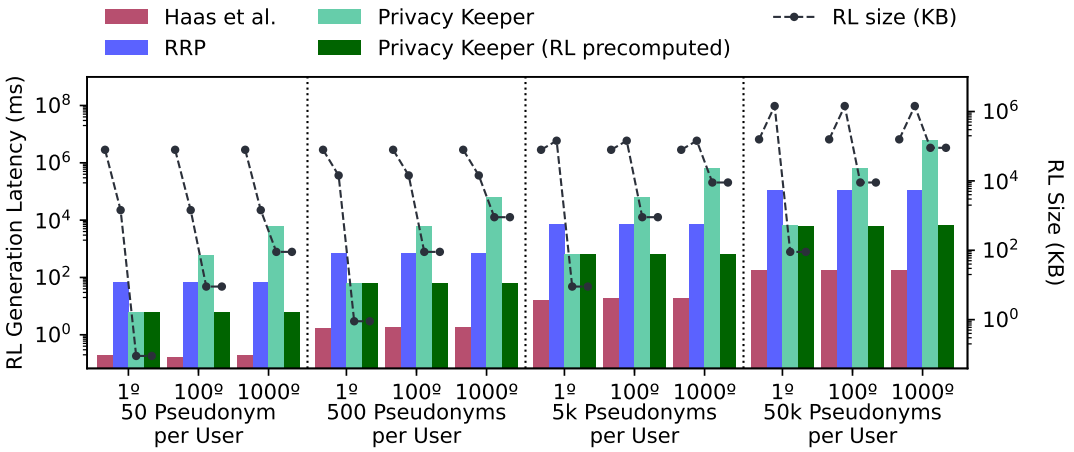


Figure 5.2: Comparison of the revocation lists size and creation latency, between blacklisting systems based on pseudonyms. RL is revocation list.

In the evaluation, we aimed to compare the critical aspects of our solution: the time required to generate a new version of the RL and the size this list can reach. Figure 5.2 illustrates both of these experiments, demonstrating the latency required to create a new version of the RL for the first revoked user (1^{o}), the one-hundredth (100^{o}), and the one-thousandth (1000^{o}) while varying the number of pseudonyms each client possesses. It's important to note that our basic Privacy Keeper solution implies that with each new revocation, we need to recreate the RL from scratch and generate the necessary information for all previously revoked clients. Therefore, the more clients have been revoked, the longer the latency to create an RL will be.

However, our optimization to precompute the RL has latency similar to Haas et al., with a worst-case time of just under 10 seconds to generate. Another visible advantage of our system is that the size of the RL is much more efficient, maintaining the same flexible properties as RRPs. In our scheme, clients store only the desired number of pseudonyms, and the revocation only requires a single token per pseudonym, while RRPs need $\log(\frac{epoch}{slot})$ of revocation data in the RL per pseudonym. It's worth noting that although Haas et al. is more efficient, it offers fewer security properties and flexibility, as it forces clients to carry many pseudonyms, even if they don't need them.

Summary

We have introduced Privacy Keeper, a novel scheme for anonymous authentication based on pseudonyms, offering perfect revocation auditability. While our previous scheme, RRPs, could only provide backward unlinkability, Privacy Keeper deliverers both of these properties, alongside immediate revocation, and in a significantly more efficient manner compared to previous systems. The use of access tokens that act as non-revocation proofs, constructed using public key encryption, represent the first scheme to offer both revocation auditability and backward unlinkability, a feat previously achieved solely through resource-intensive ZKPs. One drawback of our Privacy Keeper solution is that at the moment of authentication, the client needs to download the revocation lists, something that can impact the crucial authentication latency.

This was the last of the three main contributions of the thesis. The next chapter discusses additional contributions that emerged from the work previously presented in the thesis, and that were achieved through collaborations and concludes the thesis.

IV Final Remarks

6

Conclusions and Future Work

This chapter closes the thesis. Section 6.1 summarizes the main results and answers the questions posed in Chapter 1. The final section, Section 6.2, concludes by outlining future research directions.

6.1 Conclusions

In this thesis, we have explored the benefits and security vulnerabilities arising from the novel edge computing model. A key finding in our research is the critical need for robust security features in services operating on the edge, with a particular focus on storage systems and how to respect the low latency requirements. Having edge storage replicated on exposed fog nodes poses a significant risk to storage services, potentially hindering the adoption and deployment of edge storage solutions. We have conducted a survey of cutting-edge techniques in distributed storage system design, highlighting the benefits of leveraging TEEs and cryptographic protocols. While TEEs offer a reliable foundation for trust at the edge, relying solely on TEEs is insufficient to safeguard storage services against the full spectrum of vulnerabilities present in edge computing.

Building upon our analysis of vulnerabilities in edge storage systems, this thesis progresses in two complementary directions: (i) the design and implementation of a cryptographically secure and accurate auditing tool for edge replicated storage systems (Chapter 3), and (ii) the design and implementation of a distributed anonymous authentication scheme for edge environments (Chapters 4 and 5). We now draw conclusions from these three major outcomes of this thesis.

Firstly, we created a cryptographically secure and precise auditing tool for edge replicated storage systems. This tool is designed to pinpoint the location of data on edge machines, ensuring that the data is replicated and accessed with minimal latency, thus benefiting edge services and clients. Our solution leverages local TEEs on the machine to guarantee that the proof is physically executed on the audited machine, while configuring the challenge duration's for optimal performance and precision. As outlined in our experimental results, we achieved an accuracy level sufficient to ascertain whether data is local to a machine, thereby achieving locality auditing capability. This initial contribution addresses the first question posed in Section 1.2 by introducing an auditing scheme capable of verifying data locality in edge storage.

Secondly, we introduce an anonymous authentication scheme that effectively ensures backward unlinkability. Respecting user privacy at the edge is imperative, especially since authentication,

crucial for all services including storage systems, can compromise client privacy due to its geographic proximity to the infrastructure, potentially exposing sensitive information. To mitigate this, we have developed a distributed anonymous authentication scheme for the edge. This scheme provides authentication whit low latency while providing backwards unlinkability, surpassing existing solutions, and also reduces storage costs for both client and verifier. By utilizing TEE to generate client credentials at the edge, we establish a trustworthy foundation. This contribution has taught us that complete anonymity in authentication requires more than just backward unlinkability; revocation auditability is equally vital, presenting a complex challenge when relying exclusively on public key encryption.

Thirdly, we have developed an anonymous authentication scheme that ensures both backward unlinkability and revocation auditability. The insights from our previous contribution highlighted the importance of revocation auditability, as lack thereof could lead to privacy vulnerabilities, especially if an attacker has control over the infrastructure. Consequently, we devised a new scheme that successfully provides backward unlinkability and revocation auditability. While this scheme offers complete anonymity in authentication, it encounters challenges, particularly in downloading the revocation list at each authentication instant, which can markedly increase latency. This two last contributions address the second question posed in Section 1.2 by introducing an anonymous authentication scheme specifically designed for the challenges of edge computing.

The insights gained through this thesis may pave the way for future development in edge storage systems, suggesting that such systems should be designed with TEEs as fundamental components to establish varying trust levels at the edge. It is imperative for all edge services, including storage systems, to secure their data, especially when operating on vulnerable fog nodes. Our contributions may lay the groundwork for creating a secure edge environment.

6.2 Future Work

Throughout the work presented in this thesis, numerous decisions were made, leading to certain paths and questions that remained unexplored and unanswered. In this section, we discuss potential research directions, where some are currently under active development. These directions are directly related to the two main focuses of this thesis. We now provide motivation for these directions and offer a brief introduction to them.

Auditing partial remote storage

In the first contribution of this thesis, we introduced PoTR, our auditing tool designed to detect if an edge provider is using remote storage. Our experimental tests primarily focused on scenarios with data being completely remote or local. A key area for future research is to evaluate PoTR's effectiveness when the proportion of remote data varies. Our initial findings indicate a struggle in identifying scenarios with a small percentage of remote data, especially when only 5% is remote and 95% is local. PoTR's method of measuring an average δ value limits its ability to detect blocks with higher latency. An intriguing research direction would be to refine the proof, possibly through multiple, smaller executions, to assess the variance in latency for accessing data blocks. Shifting the focus from average latency to latency variance could enable detection of more complex and potentially malicious behaviors by edge providers.

Selective download for RLs

As previously noted, this thesis's third contribution, the Privacy Keeper, requires clients to download the revocation list at each authentication instance, potentially increasing latency for edge authentication. Addressing this latency challenge is crucial for edge applications. A direction for further research involves utilizing redactable signatures [109] in conjunction with Bloom filters. Redactable signatures enables the clients to request only a portion of the Bloom filters while preserving the capability to verify the filter integrity. In the case of the Privacy Keeper, clients would only need to download the entries that the client needs to check, requiring only to be transfer a few bytes in the network, drastically reducing the latency for authentication.

Another exploration avenue involves leveraging multiparty computation for the Pseudonym Manager/CA implementation. Implementing the revocation-capable entity across different TEEs simultaneously ensures trust isn't confined to one type of TEE. By using multiparty in this context, a client revocation requires consensus among all TEEs, protecting the system if a TEE is compromised. It would also be interesting to explore the use of Butterfly keys [168] for pseudonym generation and complementing this with cryptographic accumulators to efficiently transfer all this pseudonyms in the network.

Bibliography

[1] Abdou, A., Matrawy, A., and van Oorschot, P. Accurate one-way delay estimation with reduced client-trustworthiness. *IEEE Communications Letters 19*, 5 (2015), 735–738.

[2] Afonso, N., Bravo, M., and Rodrigues, L. Combining high throughput and low migration latency for consistent data storage on the edge. In *Proceedings of International Conference on Computer Communications and Networks* (Honolulu, HI, USA, Aug. 2020).

[3] Ahmed, E., and Rehmani, M. Mobile edge computing: Opportunities, solutions, and challenges. *Pervasive Computing 70* (2017).

[4] Ahmed, R., Zaheer, Z., Li, R., and Ricci, R. Harpocrates: Giving out your secrets and keeping them too. In *Proceedings of the ACM/IEEE Symposium on Edge Computing* (Bellevue, WA, USA, Oct. 2018).

[5] Akamai. Online retail performance report: Milliseconds are critical. `https://www.akamai.com/newsroom/press-release/akamai-releases-spring-2017-state-of-online-retail-performance-report`, 2017. Accessed: 2023-12-12.

[6] Akamai. Real time key value store for edge computing. `https://www.akamai.com/products/edgekv`, 2021. Accessed: 2023-12-12.

[7] Alder, F., Scopelliti, G., Van, J., and Mühlberg, J. About time: On the challenges of temporal guarantees in untrusted environments. In *Proceedings of the Workshop on System Software for Trusted Execution* (Rome, Italy, May 2023).

[8] Alexiou, N., Laganà, M., Gisdakis, S., Khodaei, M., and Papadimitratos, P. Vespa: Vehicular security and privacy-preserving architecture. In *Proceedings of the ACM workshop on Hot topics on wireless network security and privacy* (Budapest, Hungary, Apr. 2013).

[9] American Association of State Highway and Transportation Officials. National Connected Vehicle Field Infrastructure Footprint Analysis, Final Re-

port. https://ntlrepository.blob.core.windows.net/lib/52000/52600/52602/Cnct_Veh_Footprint_20181017.pdf, 2014. Accessed: 2023-12-12.

[10] Anati, I., Gueron, S., Johnson, S., and Scarlata, V. Innovative technology for CPU based attestation and sealing. In *Proceedings of the International Workshop on Hardware and Architectural Support for Security and Privacy* (Tel-Aviv, Israel, June 2013).

[11] Anwar, F., Garcia, L., Han, X., and Srivastava, M. Securing time in untrusted operating systems with TimeSeal. In *Proceedings of the IEEE Real-Time Systems Symposium* (York, UK, Dec. 2019).

[12] Applied Information. Emergency Vehicle Preemption System. https://appinfoinc.com/solutions/preemption-priority/, 2016. Accessed: 2023-12-12.

[13] Arasu, A., Eguro, K., Kaushik, R., Kossmann, D., Meng, P., Pandey, V., and Ramamurthy, R. Concerto: A high concurrency key-value store with integrity. In *Proceedings of the ACM International Conference on Management of Data* (Chicago, IL USA, May 2017).

[14] Armbrust, M., Fox, A., Griffith, R., Joseph, A., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., and Zaharia, M. A view of cloud computing. *Communications of the ACM 53*, 4 (2010).

[15] Armknecht, F., Barman, L., Bohli, J., and Karame, G. Mirror: Enabling proofs of data replication and retrievability in the cloud. In *Proceedings of the USENIX Security Symposium* (Austin, TX, USA, Aug. 2016).

[16] Ateniese, G., Burns, R., Curtmola, R., Herring, J., Kissner, L., Peterson, Z., and Song, D. Provable data possession at untrusted stores. In *Proceedings of the ACM Conference on Computer and Communications Security* (Alexandria, Virginia, USA, 2007).

[17] Ateniese, G., Song, D., and Tsudik, G. Quasi-efficient revocation of group signatures. In *Proceedings of the International Conference on Financial Cryptography* (Southampton, Bermuda, Mar. 2002).

[18] AT&T. Cybersecurity Insights Report: Securing the Edge. https://cdn-cybersecurity.att.com/docs/industry-reports/cybersecurity-insights-report-eleventh-edition.pdf, 2022. Accessed: 2023-12-12.

[19] Au, M., Tsang, P., and Kapadia, A. PEREA: Practical TTP-free revocation of repeatedly misbehaving anonymous users. *ACM Transactions on Information and System Security* (2008), 1–34.

[20] Autopoietic Cognitive Edge-cloud Services (ACES). Cloud-edge service for data management. `https://www.aces-edge.eu/`, 2023. 2023-07-01.

[21] Backes, M., Berrang, P., Goga, O., Gummadi, K., and Manoharan, P. On profile linkability despite anonymity in social media systems. In *Proceedings of the ACM on Workshop on Privacy in the Electronic Society* (Vienna, Austria, Oct. 2016).

[22] Bagchi, S., Siddiqui, M., Wood, P., and Zhang, H. Dependability in edge computing. *Communications of the ACM* (2019), 58–66.

[23] Bagher, K., and Lai, S. SGX-Stream: A secure stream analytics framework in SGX-enabled edge cloud. *Journal of Information Security and Applications 72* (2023).

[24] Bailleu, M., Thalheim, J., Bhatotia, P., Fetzer, C., Honda, M., and Vaswani, K. Speicher: Securing LSM-based key-value stores using shielded execution. In *Proceedings of the USENIX Conference on File and Storage Technologies* (Boston, MA, USA, Feb. 2019).

[25] BankInfo Security. Verizon breach: 6 million customer accounts exposed. `https://www.bankinfosecurity.com/verizon-breach-6-million-customer-accounts-exposed-a-10107`, 2017. Accessed: 2023-12-12.

[26] Barbosa, M., Portela, B., Scerri, G., and Warinschi, B. Foundations of hardware-based attested computation and application to SGX. In *Proceedings of the IEEE European Symposium on Security and Privacy* (Saarbrücken, Germany, Mar. 2016).

[27] Barker, E., and Roginsky, A. Transitioning the use of cryptographic algorithms and key lengths. NIST Special Publication 800-131A Revision 2, National Institute of Standards and Technology, Mar. 2019.

[28] BBC News. AWS: Amazon web outage breaks vacuums and doorbells. `https://www.bbc.com/news/technology-55087054`, 2020. Accessed: 2023-12-12.

[29] Benet, J. IPFS: content addressed, versioned, P2P file system. *arXiv preprint arXiv:1407.3561* (2014).

[30] Benet, J., Dalrymple, D., and Greco, N. Proof of replication. *Protocol Labs 27* (2017).

[31] Benson, K., Dowsley, R., and Shacham, H. Do you know where your cloud files are? In *Proceedings of the ACM Workshop on Cloud Computing Security Workshop* (Chicago, Illinois, USA, Oct. 2011).

[32] Beresford, A., and Stajano, F. Mix zones: User privacy in location-aware services. In *Proceedings of the IEEE Conference on Pervasive Computing and Communications Workshops* (Orlando, FL, USA, Mar. 2004).

[33] Bernstein, D., Duif, N., Lange, T., Schwabe, P., and Yang, B. High-speed high-security signatures. *Journal of cryptographic engineering* (2012), 77–89.

[34] Bertoncello, M. Monetising car data: New service business opportunities to create new customer benefits. `https://www.thinkwithgoogle.com/intl/en-gb/future-of-marketing/digital-transformation/monetizing-car-data-new-service-business-opportunities-create-new-customer-benefits/`, 2017. Accessed: 2023-12-12.

[35] Bhardwaj, K., Shih, M., Agarwal, P., Gavrilovska, A., Kim, T., and Schwan, K. Fast, scalable and secure onloading of edge functions using airbox. In *Proceedings of the IEEE/ACM Symposium on Edge Computing* (Washington DC, USA, Oct. 2016).

[36] Boneh, D., and Franklin, M. Identity-based encryption from the weil pairing. In *Proceedings of the Annual International Cryptology Conference* (London, UK, Aug. 2001).

[37] Boneh, D., and Shacham, H. Group signatures with verifier-local revocation. In *Proceedings of the ACM Conference on Computer and Communications Security* (Washington, DC, USA, 2004).

[38] Bonomi, F., Milito, R., Zhu, J., and Addepalli, S. Fog computing and its role in the internet of things. In *Proceedings of the Workshop on Mobile Cloud Computing* (Helsinki, Finland, Aug. 2012).

[39] Bravo, M., Rodrigues, L., and Van Roy, P. Saturn: A distributed metadata service for causal consistency. In *Proceedings of the ACM European Conference on Computer Systems* (Belgrade, Serbia, Apr. 2017).

[40] Brenner, S., Wulf, C., Goltzsche, D., Weichbrodt, N., Lorenz, M., Fetzer, C., Pietzuch, P., and Kapitza, R. Securekeeper: Confidential ZooKeeper using Intel SGX. In *Proceedings of the Middleware Conference* (Trento, Italy, Dec. 2016).

[41] Brickell, E. An efficient protocol for anonymously providing assurance of the container of a private key. *Submitted to the Trusted Computing Group* (2003).

[42] Bringer, J., and Patey, A. Backward unlinkability for a VLR group signature scheme with efficient revocation check. *Cryptology ePrint Archive* (2011).

[43] Caballero, C., Molina, J., Hernández, J., León, O., and Soriano, M. Providing k-anonymity and revocation in ubiquitous VANETs. In *Proceedings of the International Conference on Ad Hoc Networks* (Ottawa, Canada, Sept. 2016).

[44] Camacho, P., and Hevia, A. On the impossibility of batch update for cryptographic accumulators. In *Proceedings of the International Conference on Cryptology and Information Security in Latin America* (Puebla, Mexico, Aug. 2010).

[45] Camenisch, J., and Lysyanskaya, A. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *Proceedings of the International Cryptology Conference* (Santa Barbara, California, USA, Aug. 2002).

[46] Camp, LLC. Security credential management system proof-of-concept implementation—EE requirements and specifications supporting SCMS software release 1.1. Tech. rep., Vehicle Safety Communications Consortium, 2016.

[47] Chaum, D. Security without identification: Transaction systems to make big brother obsolete. *Communications of the ACM* (1985), 1030–1044.

[48] Chaum, D., and Heyst, E. Group signatures. In *Workshop on the Theory and Application of Cryptographic Techniques* (1991), pp. 257–265.

[49] Chen, L., Li, J., Ma, R., Guan, H., and Jacobsen, H. EnclaveCache: A secure and scalable key-value cache in multi-tenant clouds using Intel SGX. In *Proceedings of the Middleware Conference* (Davis, CA, USA, Dec. 2019).

[50] Choi, J., and Yoo, C. One-way delay estimation and its application. *Computer Communications 28*, 7 (2005), 819–828.

[51] Christin, D. Privacy in mobile participatory sensing: Current trends and future challenges. *Journal of Systems and Software* (2016), 57–68.

[52] Chu, C., Liu, J., Huang, X., and Zhou, J. Verifier-local revocation group signatures with time-bound keys. In *Proceedings of the ACM Symposium on Information, Computer and Communications Security* (Seoul, Korea, May 2012).

[53] Cisco. Cisco delivers vision of fog computing to accelerate value from billions of connected devices, press release. `https://newsroom.cisco.com/press-release-content?type=webcontent&articleId=1334100`, 2014. Accessed: 2023-12-12.

[54] Cloudflare. CDN global network. `https://www.cloudflare.com/cdn/`, 2009. Accessed: 2023-12-12.

[55] Cormode, G., and Muthukrishnan, S. An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms* (2005), 58–75.

[56] Correia, C. Safeguarding data consistency at the edge. In *Proceedings of the IEEE-IFIP International Conference on Dependable Systems and Networks-Supplemental Volume (DSN-S)* (València, Spain, June 2020), pp. 65–66.

[57] Correia, C., Correia, M., and Rodrigues, L. Omega: a secure event ordering service for the edge. In *Proceedings of the IEEE/IFIP International Conference on Dependable Systems and Networks* (València, Spain, June 2020).

[58] Correia, C., Correia, M., and Rodrigues, L. Omega: a secure event ordering service for for the edge. *IEEE Transactions on Dependable and Secure Computing* (2021).

[59] Correia, C., Correia, M., and Rodrigues, L. Using range-revocable pseudonyms to provide backward unlinkability in the edge. In *Proceedings of the ACM Conference on computer and Communications Security* (Copenhagen, Denmark, Nov. 2023).

[60] Correia, C., Prates, R., Correia, M., and Rodrigues, L. PoTR: Accurate and efficient proof of timely-retrievability for storage systems. In *Proceedings of the IEEE Pacific Rim International Symposium on Dependable Computing* (Singapore, Singapore, Oct. 2023).

[61] Correia, M., Almeida, J., Bartolomeu, P., Fonseca, J., and Ferreira, J. Performance assessment of collective perception service supported by the roadside infrastructure. *Electronics* (2022), 347.

[62] Costan, V., Lebedev, I., and Devadas, S. Sanctum: Minimal hardware extensions for strong software isolation. In *Proceedings of the USENIX Security Symposium* (Vancouver, BC, Canada, 2016).

[63] Crash Avoidance Metrics Partners LLC (CAMP). Technical publications. https://www.campllc.org/, 2021. Accessed: 2023-12-12.

[64] Cristian, F. Understanding fault-tolerant distributed systems. *Communications of the ACM* (1993), 56–78.

[65] Cybersecurity and Infrastructure Security Agency. People's Republic of China State-Sponsored Cyber Actors Exploit Network Providers and Devices. https://www.cisa.gov/uscert/ncas/alerts/aa22-158a, 2022. Accessed: 2023-12-12.

[66] Dang, H., Purwanto, E., and Chang, E. Proofs of data residency: Checking whether your cloud files have been relocated. In *Proceedings of the ACM on Asia Conference on Computer and Communications Security* (Abu Dhabi, United Arab Emirates, 2017).

[67] Dark Reading. Cybercriminals Take Aim at Connected Car Infrastructure. https://www.darkreading.com/attacks-breaches/cybercriminals-take-aim-at-connected-car-infrastructure, 2021. Accessed: 2023-12-12.

[68] Data Privacy Manager platform. Pseudonymization according to the GDPR. https://dataprivacymanager.net/pseudonymization-according-to-the-gdpr/, 2021. Accessed: 2023-12-12.

[69] Davidson, A., Goldberg, I., Sullivan, N., Tankersley, G., and Valsorda, F. Privacy Pass: Bypassing internet challenges anonymously. In *Proceedings of the Privacy Enhancing Technologies* (Barcelona, Spain, July 2018).

[70] DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., Sivasubramanian, S., Vosshall, P., and Vogels, W. Dynamo: Amazon's highly available key-value store. In *Proceedings of the ACM Symposium on Operating Systems Principles* (Stevenson, WA, USA, Oct. 2007).

[71] Dwork, C., Lynch, N., and Stockmeyer, L. Consensus in the presence of partial synchrony. *Journal of the ACM* (1988), 288–323.

[72] Echeverría, S., Klinedinst, D., Williams, K., and Lewis, G. Establishing trusted identities in disconnected edge environments. In *Proceedings of the IEEE/ACM Symposium on Edge Computing* (Washington, DC, USA, Oct. 2016).

[73] Edge Computing News. Critical edge security issues across industries: Cyberattacks on the rise. https://www.edgecomputing-news.com/2022/05/31/critical-edge-security-issues-across-industries-cyberattacks-on-the-rise/, 2022. Accessed: 2023-12-12.

[74] Ekberg, J., Kostiainen, K., and Asokan, N. The untapped potential of trusted execution environments on mobile devices. In *Proceedings of the IEEE Security & Privacy* (San Jose, CA, USA, May 2014), no. 4.

[75] El-Hindi, M., Ziegler, T., Heinrich, M., Lutsch, A., Zhao, Z., and Binnig, C. Benchmarking the second generation of Intel SGX hardware. In *Proceedings of the International Workshop on Data Management on New Hardware* (Philadelphia, PA, USA, June 2022).

[76] Emura, K., Hayashi, T., and Ishida, A. Group signatures with time-bound keys revisited: A new model, an efficient construction, and its implementation. *IEEE Transactions on Dependable and Secure Computing* (2017), 292–305.

[77] European Telecommunications Standards Institute. ETSI TS 103 097 V1.2.1:Intelligent Transport Systems (ITS); Security; Security header and certificate formats. https://www.etsi.org/deliver/etsi_ts/103000_103099/103097/01.02.01_60/ts_103097v010201p.pdf, 2015. Accessed: 2023-12-12.

[78] European Telecommunications Standards Institute. ETSI TR 103 415 V1.1.1: Intelligent Transport Systems (ITS); Security; Pre-standardization study on pseudonym change management. https://www.etsi.org/deliver/etsi_tr/103400_103499/103415/01.01.01_60/tr_103415v010101p.pdf, 2018. Accessed: 2023-12-12.

[79] European Telecommunications Standards Institute. ETSI TS 102 941 V1.4.1: Intelligent Transport Systems (ITS); Security; Trust and Privacy Management. https://www.etsi.org/deliver/etsi_ts/102900_102999/102941/01.04.01_60/ts_102941v010401p.pdf, 2021. Accessed: 2023-12-12.

[80] European Telecommunications Standards Institute (ETSI), ETSI TR 102 863 V1.1.1 (2011-06). Intelligent Transportation Systems (ITS); Vehicular Communications; Basic Set

of Applications; Local Dynamic Map (LDM); Rationale for and guidance on standardization. Tech. rep., 2011.

[81] Fan, B., Andersen, D., Kaminsky, M., and Mitzenmacher, M. Cuckoo filter: Practically better than bloom. In *Proceedings of the ACM International on Conference on emerging Networking Experiments and Technologies* (Sydney, Australia, Dec. 2014).

[82] Filecoin. A blockchain based storage network. `https://spec.filecoin.io/`, 2014. Accessed: 2023-12-12.

[83] Förster, D., Kargl, F., and Löhr, H. PUCA: A pseudonym scheme with user-controlled anonymity for vehicular Ad-Hoc networks (VANET). In *Proceedings of the IEEE Vehicular Networking Conference* (Paderborn, Germany, Sept. 2014).

[84] Foursquare Labs, Inc. Foursquare Website. `https://www.foursquare.com/`, 2009. Accessed: 2022-02-08.

[85] Freedom of Information and Privacy Association (FIPA). The Connected Car: who is in the driver's seat? `https://fipa.bc.ca/wp-content/uploads/2018/01/CC_report_lite.pdf`, 2019. Accessed: 2023-12-12.

[86] Freudiger, J., Raya, M., Félegyházi, M., Papadimitratos, P., and Hubaux, J. Mix-zones for location privacy in vehicular networks. In *Proceedings of the ACM Workshop on Wireless Networking for Intelligent Transportation Systems* (Vancouver, Canada, Aug. 2007).

[87] Gañán, C., Munoz, J., Esparza, O., Mata-Díaz, J., and Alins, J. EPA: An efficient and privacy-aware revocation mechanism for vehicular ad hoc networks. *Pervasive and Mobile Computing* (2015), 75–91.

[88] Ganti, R., Ye, F., and Lei, H. Mobile crowdsensing: Current state and future challenges. *IEEE Communications Magazine* (2011), 32–39.

[89] GC Idex. Cisco global cloud index: Forecast and methodology, 2016–2021. Tech. rep., White Paper, 2016.

[90] Gondree, M., and Peterson, Z. N. Geolocation of data in the cloud. In *Proceedings of the ACM Conference on Data and Application Security and Privacy* (San Antonio, Texas, USA, 2013).

[91] Gong, Y., Cai, Y., Guo, Y., and Fang, Y. A privacy-preserving scheme for incentive-based demand response in the smart grid. *IEEE Transactions on Smart Grid* (2015), 1304–1313.

[92] Google. Google maps website. `https://www.google.com/maps`, 2005. Accessed: 2023-12-12.

[93] Grand View Research. CDN market report, 2022. `https://www.grandviewresearch.com/industry-analysis/content-delivery-networks-cnd-market`.

[94] Gunawi, H., Hao, M., Suminto, R., Laksono, A., Satria, A., Adityatama, J., and Eliazar, K. Why does the cloud stop computing? lessons from hundreds of service outages. In *Proceeding of the ACM Symposium on Cloud Computing* (Santa Clara, California, Oct. 2016).

[95] Haas, J., Hu, Y.-C., and Laberteaux, K. Efficient certificate revocation list organization and distribution. *IEEE Journal on Selected Areas in Communications* (2011), 595–604.

[96] Henry, R., and Goldberg, I. Formalizing anonymous blacklisting systems. In *Proceedings of the IEEE Symposium on Security and Privacy* (Oakland, California, USA, May 2011), pp. 81–95.

[97] Hieb, J., Schreiver, J., and Graham, J. Using bloom filters to ensure access control and authentication requirements for scada field devices. In *Proceeding of the International Conference on Critical Infrastructure Protection* (Washington, DC, USA, Mar. 2012).

[98] Ho, G., Leung, D., Mishra, P., Hosseini, A., Song, D., and Wagner, D. Smart locks: Lessons for securing commodity internet of things devices. In *Proceedings of the ACM on Asia Conference on Computer and Communications Security* (Xi'an, China, May 2016).

[99] Hu, Y., Patel, M., Sabella, D., Sprecher, N., and Young, V. Mobile edge computing—a key technology towards 5G. *ETSI white paper 11*, 11 (2015).

[100] Ibrahim, M. Octopus: an edge-fog mutual authentication scheme. *International Journal of Network Security* (2016), 1089–1101.

[101] Info Security. Mitsubishi electric discloses information leak. `https://www.infosecurity-magazine.com/news/mitsubishi-electric-discloses/`, 2020. Accessed: 2023-12-12.

[102] Intel Corporation. Intel(R) Software Guard Extensions Developer Reference for Linux* OS. https://download.01.org/intel-sgx/linux-2.3/docs/Intel_SGX_Developer_Reference_Linux_2.3_Open_Source.pdf. Accessed: 2022-02-08.

[103] Intel Corporation. Intel Software Guard Extensions SSL. https://github.com/intel/intel-sgx-ssl, 2017. Accessed: 2023-12-12.

[104] Intel Corporation. Intel's Fog Reference Design Overview. https://www.reflexces.com/wp-content/uploads/2018/11/fog-reference-design-overview-guide.pdf, 2017. Accessed: 2023-12-12.

[105] Intelligent Transportation Systems Joint Program Office. Connected Vehicle Pilot Deployment Program. https://www.its.dot.gov/pilots/overview.htm, 2021. Accessed: 2023-12-12.

[106] Ishida, A., Sakai, Y., Emura, K., Hanaoka, G., and Tanaka, K. Fully anonymous group signature with verifier-local revocation. In *Proceedings of the International Conference on Security and Cryptography for Networks* (Amalfi, Italy, Sept. 2018).

[107] Jain, S., Zhang, X., Zhou, Y., Ananthanarayanan, G., Jiang, J., Shu, Y., Bahl, V., and Gonzalez, J. Spatula: Efficient cross-camera video analytics on large camera networks. In *Proceedings of the ACM/IEEE Symposium on Edge Computing* (Virtual, Nov. 2020).

[108] Jiang, T., Meng, W., Yuan, X., Wang, L., Ge, J., and Ma, J. Reliablebox: Secure and verifiable cloud storage with location-aware backup. *IEEE Transactions on Parallel and Distributed Systems 32*, 12 (2021).

[109] Johnson, R., Molnar, D., Song, D., and Wagner, D. Homomorphic signature schemes. In *Cryptographers' track at the RSA conference* (2002), Springer, pp. 244–262.

[110] KAGGLE. Data Set ECML/PKDD 15: Taxi Trajectory Prediction. https://www.kaggle.com/c/pkdd-15-predict-taxi-service-trajectory-i/, 2015. Accessed: 2022-02-08.

[111] Kampanakis, P., Campagna, M., Crocket, E., Petcher, A., and Gueron, S. Practical challenges with AES-GCM and the need for a new cipher. *NIST* (2024).

[112] Khodaei, M., Jin, H., and Papadimitratos, P. SECMACE: Scalable and robust identity and credential management infrastructure in vehicular communication systems. *IEEE Transactions on Intelligent Transportation Systems* (2018), 1430–1444.

[113] Khodaei, M., and Papadimitratos, P. Efficient, scalable, and resilient vehicle-centric certificate revocation list distribution in VANETs. In *Proceedings of the ACM conference on security & privacy in wireless and mobile networks* (Stockholm, Sweden, June 2018).

[114] Kim, T., Park, J., Woo, J., Jeon, S., and Huh, J. Shieldstore: Shielded in-memory key-value storage with SGX. In *Proceedings of the ACM European Conference on Computer Systems* (Dresden, Germany, Mar. 2019).

[115] Krahn, R., Trach, B., Vahldiek-Oberwagner, A., Knauth, T., Bhatotia, P., and Fetzer, C. Pesos: policy enhanced secure object store. In *Proceedings of the ACM European Conference on Computer Systems* (Porto, Portugal, Apr. 2018).

[116] Lakshman, A., and Malik, P. Cassandra: a decentralized structured storage system. *ACM SIGOPS Operating Systems Review 44*, 2 (2010).

[117] Lamport, L., Shostak, R., and Pease, M. The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems* (1982), 203–226.

[118] Larisch, J., Choffnes, D., Levin, D., Maggs, B., Mislove, A., and Wilson, C. CRLite: A scalable system for pushing all TLS revocations to all browsers. In *Proceedings of the IEEE Symposium on Security and Privacy* (San Jose, CA, USA, May 2017).

[119] Lauinger, T., Laoutaris, N., Rodriguez, P., Strufe, T., Biersack, E., and Kirda, E. Privacy implications of ubiquitous caching in named data networking architectures. In *ACM Special Interest Group on Data Communication* (Helsinki, Finland, Aug. 2012).

[120] Li, L., and Lazos, L. Proofs of physical reliability for cloud storage systems. *IEEE Transactions on Parallel and Distributed Systems 31*, 5 (2020).

[121] Lin, M., Lane, N., Mohammod, M., Yang, X., Lu, H., Cardone, G., Ali, S., Doryab, A., Berke, E., Campbell, A., and Choudhury, T. BeWell+ multi-dimensional wellbeing monitoring with community-guided user feedback and energy optimization. In *Proceedings of the conference on Wireless Health* (San Diego, CA, USA, Oct. 2012).

[122] Liu, B., and Huang, J. D4: Fast concurrency debugging with parallel differential analysis. *ACM SIGPLAN Notices* (2018), 359–373.

[123] Lloyd, W., Freedman, M., Kaminsky, M., and Andersen, D. Don't settle for eventual: Scalable causal consistency for wide-area storage with cops. In *Proceedings of the ACM Symposium on Operating Systems Principles* (Cascais, Portugal, Oct. 2011).

[124] Loesing, K. Measuring the tor network: Evaluation of client requests to the directories. Tech. rep., Tor Project, 2009.

[125] Luo, L., Guo, D., Ma, R., Rottenstreich, O., and Luo, X. Optimizing bloom filter: Challenges, solutions, and comparisons. *IEEE Communications Surveys & Tutorials* (2018), 1912–1949.

[126] Lysyanskaya, A., Rivest, R., Sahai, A., and Wolf, S. Pseudonym systems. In *Proceedings of the International Workshop on Selected Areas in Cryptography* (Ontario, Canada, Aug. 1999).

[127] Markey, E. Tracking & hacking: Security & privacy gaps put american drivers at risk. *Congressional Report* (2015).

[128] Matetic, S., Ahmed, M., Kostiainen, K., Dhar, A., Sommer, D., Gervais, A., Juels, A., and Capkun, S. ROTE: Rollback protection for trusted execution. In *Proceedings of the USENIX Security Symposium* (Vancouver, BC, Canada, 2017).

[129] McKeen, F., Alexandrovich, I., Berenzon, A., Rozas, C., Shafi, H., Shanbhogue, V., and Savagaonkar, U. Innovative instructions and software model for isolated execution. In *Proceedings of the International Workshop on Hardware and Architectural Support for Security and Privacy* (Tel-Aviv, Israel, June 2013).

[130] Mehdi, S., Littley, C., Crooks, N., Alvisi, L., Bronson, N., and Lloyd, W. I can't believe it's not causal! scalable causal consistency with no slowdown cascades. In *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation* (Boston, MA, USA, Mar. 2017).

[131] Merkle, R. A digital signature based on a conventional encryption function. In *Proceedings of the Conference on the Theory and Application of Cryptographic Techniques* (Amsterdam, The Netherlands, Apr. 1987).

[132] Meyer, D. What the GDPR will mean for companies tracking location. https://iapp.org/news/a/what-the-gdpr-will-mean-for-companies-tracking-location/, 2018. Accessed: 2023-12-12.

[133] Miranda, M., Esteves, T., Portela, B., and Paulo, J. S2Dedup: SGX-enabled secure deduplication. In *Proceedings of the ACM International Conference on Systems and Storage* (Virtual, June 2021).

[134] Monga, S., Ramachandra, S., and Simmhan, Y. ElfStore: A resilient data storage service for federated edge and fog resources. In *Proceedings of IEEE International Conference on Web Services* (Milan, Italy, July 2019).

[135] Mortazavi, S., Salehe, M., Gomes, C., Phillips, C., and Lara, E. Cloudpath: A multi-tier cloud computing framework. In *Proceedings of the ACM/IEEE Symposium on Edge Computing* (San Jose, CA, USA, Oct. 2017).

[136] Msahli, M., Cam-Winget, N., Whyte, W., Serhrouchni, A., and Labiod, H. RFC 8902 TLS Authentication Using Intelligent Transport System (ITS) Certificates. https://www.hjp.at/doc/rfc/rfc8902.pdf, 2020. Accessed: 2023-12-12.

[137] Mukherjee, M., Matam, R., Shu, L., Maglaras, L., Ferrag, M., Choudhury, N., and Kumar, V. Security and privacy in fog computing: Challenges. *IEEE Access* (2017).

[138] Mukherjee, M., Matam, R., Shu, L., Maglaras, L., Ferrag, M., Choudhury, N., and Kumar, V. Security and privacy in fog computing: Challenges. *IEEE Access 5* (2017).

[139] Nakanishi, T., and Funabiki, N. Verifier-local revocation group signature schemes with backward unlinkability from bilinear maps. In *Proceedings of the International conference on the theory and application of cryptology and information security* (Chennai, India, Dec. 2005).

[140] National Automobile Dealers Association and the Future of Privacy Forum. Personal Data in Your Car. https://fpf.org/wp-content/uploads/2017/01/consumerguide.pdf, 2017. Accessed: 2023-12-12.

[141] National Highway Traffic Safety Administration. Department of Transportation (DOT), Federal Motor Vehicle Safety Standards; V2V Communica-

tions. https://www.federalregister.gov/documents/2017/01/12/2016-31059/federal-motor-vehicle-safety-standards-v2v-communications, 2017. Accessed: 2023-12-12.

[142] National Highway Traffic Safety Administration. Vehicle Data Privacy. https://www.nhtsa.gov/technology-innovation/vehicle-data-privacy, 2021. Accessed: 2023-12-12.

[143] Nguyen, P., Silence, A., Darais, D., and Near, J. Duetsgx: Differential privacy with secure hardware. In *Proceedings of the Workshop on Theory and Practice of Differential Privacy* (Virtual, Nov. 2020).

[144] Ni, J., Zhang, A., Lin, X., and Shen, X. Security, privacy, and fairness in fog-based vehicular crowdsensing. *IEEE Communications Magazine* (2017), 146–152.

[145] Niantic, Inc. Pokemon GO Website. http://pokemongo.nianticlabs.com, 2016. Accessed: 2023-12-12.

[146] Ning, Z., Liao, J., Zhang, F., and Shi, W. Preliminary study of trusted execution environments on heterogeneous edge platforms. In *Proceedings of the ACM/IEEE Workshop on Security and Privacy in Edge Computing* (Bellevue, WA, USA, Oct. 2018).

[147] Ohara, K., Emura, K., Hanaoka, G., Ishida, A., Ohta, K., and Sakai, Y. Shortening the Libert–Peters–Yung revocable group signature scheme by using the random oracle methodology. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* (2019), 1101–1117.

[148] Oleksenko, O., Trach, B., Krahn, R., Silberstein, M., and Fetzer, C. Varys: Protecting SGX enclaves from practical side-channel attacks. In *Proceedings of the USENIX Annual Technical Conference* (Boston, MA, USA, July 2018).

[149] Osanaiye, O., Choo, K., and Dlodlo, M. Distributed denial of service (DDoS) resilience in cloud: review and conceptual cloud DDoS mitigation framework. *Journal of Network and Computer Applications* (2016), 147–165.

[150] Otonomo. A Privacy Playbook for Connected Car Data. https://fpf.org/wp-content/uploads/2020/01/OtonomoPrivacyPaper.pdf, 2019. Accessed: 2023-12-12.

[151] Pan, B., Zheng, Y., Wilkie, D., and Shahabi, C. Crowd sensing of traffic anomalies based on human mobility and social media. In *Proceedings of the ACM international conference on advances in geographic information systems* (Orlando, Florida, Nov. 2013).

[152] Peters, O., and Szakats, V. Portable C implementation of Ed25519. `https://github.com/orlp/ed25519`, 2013. Accessed: 2023-12-12.

[153] Petit, J., Schaub, F., Feiri, M., and Kargl, F. Pseudonym schemes in vehicular networks: A survey. *IEEE Communications Surveys & Tutorials* (2014), 228–255.

[154] Phil Muncaster. Info Security, Connected Rental Cars Leak Personal Driver Data. `https://www.infosecurity-magazine.com/news/connected-rental-cars-leak/`, 2017. Accessed: 2023-12-12.

[155] Popa, R., Blumberg, A., Balakrishnan, H., and Li, F. Privacy and accountability for location-based aggregate statistics. In *Proceedings of the ACM conference on computer and communications security* (Chicago, IL, USA, Oct. 2011).

[156] Rahaman, S., Cheng, L., Yao, D., Li, H., and Park, J. Provably secure anonymous-yet-accountable crowdsensing with scalable sublinear revocation. In *Proceedings of the Privacy Enhancing Technologies* (Minneapolis, USA, July 2017).

[157] Redis. Key-Value store. `http://redis.io`. Accessed: 2023-12-12.

[158] Remeli, M., Lestyán, S., Acs, G., and Biczók, G. Automatic driver identification from in-vehicle network logs. In *Proceedings of the IEEE Intelligent Transportation Systems Conference* (Auckland, New Zealand, Oct. 2019).

[159] Ricart, G. A city edge cloud with its economic and technical considerations. In *Proceedings of the International Workshop on Smart Edge Computing and Networking* (Kona, HI, USA, June 2017).

[160] Ricart, G. A city edge cloud with its economic and technical considerations. In *IEEE International Conference on Pervasive Computing and Communications Workshops* (Kona, HI, USA, 2017).

[161] Rigazzi, G., Tassi, A., Piechocki, R., Tryfonas, T., and Nix, A. Optimized certificate revocation list distribution for secure V2X communications. In *Proceeding of the IEEE Vehicular Technology Conference* (Sydney, Australia, June 2017).

[162] Satyanarayanan, M., Bahl, V., Caceres, R., and Davies, N. The case for VM-based cloudlets in mobile computing. *IEEE Pervasive Computing 8*, 4 (2009).

[163] Satyanarayanan, M., Gibbons, P., Mummert, L., Pillai, P., Simoens, P., and Sukthankar, R. Cloudlet-based just-in-time indexing of IoT video. In *Global Internet of Things Summit* (Geneva, Switzerland, June 2017).

[164] Satyanarayanan, M., Simoens, P., Xiao, Y., Pillai, P., Chen, Z., Ha, K., Hu, W., and Amos, B. Edge analytics in the internet of things. *IEEE Pervasive Computing* (2015), 24–31.

[165] Schaub, F., Kargl, F., Ma, Z., and Weber, M. V-tokens for conditional pseudonymity in VANETs. In *Proceedings of the IEEE Wireless Communication and Networking Conference* (Sydney, Australia, Apr. 2010).

[166] Schwartz, E., Brumley, D., and McCune, J. *Contractual anonymity.* PhD thesis, Carnegie Mellon University. Information Networking Institute, 2009.

[167] Silva, R., Correia, C., Correia, M., and Rodrigues, L. Deduplication vs privacy tradeoffs in cloud storage. In *Proceedings of the ACM/SIGAPP Symposium on Applied Computing* (Tallinn, Estonia, Mar. 2023).

[168] Simplicio, M., Cominetti, E., Patil, H., Ricardini, J., and Silva, M. The unified butterfly effect: Efficient security credential management system for vehicular communications. In *Proceedings of the IEEE Vehicular Networking Conference* (Taipei, Taiwan, Dec. 2018).

[169] Simplicio, M., Cominetti, L., Patil, K., Ricardini, J., and Silva, M. Acpc: Efficient revocation of pseudonym certificates using activation codes. *Ad Hoc Networks* (2019).

[170] Singh, A., Castro, M., Druschel, P., and Rowstron, A. Defending against eclipse attacks on overlay networks. In *Proceedings of the workshop on ACM SIGOPS European workshop* (Leuven, Belgium, 2004).

[171] Slamanig, D., Spreitzer, R., and Unterluggauer, T. Linking-based revocation for group signatures: A pragmatic approach for efficient revocation checks. In *Proceedings of the International Conference on Cryptology in Malaysia* (Kuala Lumpur, Malaysia, Dec. 2016).

[172] Streiffer, C., Srivastava, A., Orlikowski, V., Velasco, Y., Martin, V., Raval, N., Machanavajjhala, A., and Cox, L. ePrivateEye: To the edge and beyond! In *Proceedings of the ACM/IEEE Symposium on Edge Computing* (San Jose, CA, USA, 2017).

[173] Sucasas, V., Mantas, G., Bastos, J., Damião, F., and Rodriguez, J. A signature scheme with unlinkable-yet-acountable pseudonymity for privacy-preserving crowdsensing. *IEEE Transactions on Mobile Computing* (2020), 752–768.

[174] Sun, Y., Lu, R., Lin, X., Shen, X., and Su, J. An efficient pseudonymous authentication scheme with strong privacy preservation for vehicular communications. *IEEE Transactions on Vehicular Technology* (2010), 3589–3603.

[175] Swarm. Distributed storage platform, 2021. `https://github.com/ethersphere/`.

[176] Taleb, T., Samdanis, K., Mada, B., Flinck, H., Dutta, S., and Sabella, D. On multi-access edge computing: A survey of the emerging 5G network edge cloud architecture and orchestration. *IEEE Communications Surveys Tutorials 19*, 3 (2017).

[177] Tech Monitor. Data from your connected car could be sold to the highest bidder. `https://techmonitor.ai/policy/privacy-and-data-protection/connected-vehicle-data-apply-carplay`, 2021. Accessed: 2023-12-12.

[178] The Wall Street Journal. Is Google Down? Gmail, YouTube Suffer Outages. `https://www.wsj.com/articles/google-suffers-widespread-outage-11607950400`, 2020. Accessed: 2023-12-12.

[179] Tian, H., Zhang, Q., Yan, S., Rudnitsky, A., Shacham, L., Yariv, R., and Milshten, N. Switchless calls made practical in Intel SGX. In *Proceedings of the orkshop on System Software for Trusted Execution* (Toronto, Canada, Oct. 2018).

[180] Tsang, P., Au, M., Kapadia, A., and Smith, S. Blacklistable anonymous credentials: blocking misbehaving users without ttps. In *Proceedings of the ACM conference on Computer and communications security* (Alexandria, Virginia, USA, Oct. 2007).

[181] Tsang, P., Au, M., Kapadia, A., and Smith, S. BLAC: Revoking Repeatedly Misbehaving Anonymous Users without Relying on TTPs. Tech. rep., Dartmouth College, Computer Science Department, TR2008-635, 2010.

[182] Tsang, P., Kapadia, A., Cornelius, C., and Smith, S. Nymble: Blocking misbehaving users in anonymizing networks. *IEEE Transactions on Dependable and Secure Computing 8*, 2 (2009).

[183] United States Government Accountability Office. Vehicle-to-Infrastructure Technologies Expected to Offer Benefits. https://www.gao.gov/assets/gao-15-775.pdf, 2015. Accessed: 2023-12-12.

[184] United States Government Accountability Office. Vehicle Data Privacy. https://www.gao.gov/assets/gao-17-656.pdf, 2017. Accessed: 2023-12-12.

[185] Vakili, A., and Gregoire, J. Accurate one-way delay estimation: Limitations and improvements. *ieee transactions on instrumentation and measurement 61*, 9 (2012), 2428–2435.

[186] Van-Bulck, J., Moghimi, D., Schwarz, M., Lipp, M., Minkin, M., Genkin, D., Yuval, Y., Sunar, B., Gruss, D., and Piessens, F. LVI: Hijacking Transient Execution through Microarchitectural Load Value Injection. In *Proceedings of the IEEE Symposium on Security and Privacy* (San Francisco, California, May 2020).

[187] Vaquero, L., and Rodero-Merino, L. Finding your way in the fog: Towards a comprehensive definition of fog computing. *ACM SIGCOMM Computer Communication Review* (2014), 27–32.

[188] Verheul, E., Hicks, C., and Garcia, F. IFAL: Issue first activate later certificates for V2X. In *Proceedings of the IEEE European Symposium on Security and Privacy* (Stockholm, Sweden, June 2019).

[189] Verizon. Hum, Verizon Mobile application that connects with car. https://www.hum.com/, 2021. Accessed: 2023-12-12.

[190] Wang, J., Chen, H., Zhou, F., Sun, M., Huang, Z., and Zhang, Z. A-DECS: Enhanced collaborative edge–edge data storage service for edge computing with adaptive prediction. *Computer Networks 193* (2021), 108087.

[191] Waze Mobile Ltd. Waze Website. https://www.waze.com/, 2007. Accessed: 2023-12-12.

[192] Wei, W., Xu, F., and Li, Q. Mobishare: Flexible privacy-preserving location sharing in mobile online social networks. In *Proceedings of the IEEE International Conference on Computer Communications* (Orlando, FL, USA, Mar. 2012).

[193] Weichbrodt, N., Kurmus, A., Pietzuch, P., and Kapitza, R. Asyncshock: Exploiting synchronisation bugs in Intel SGX enclaves. In *Proceedings of the European Symposium on Research in Computer Security* (Heraklion, Greece, Sept. 2016).

[194] Whyte, W., Weimerskirch, A., Kumar, V., and Hehn, T. A security credential management system for V2V communications. In *Proceedings of the IEEE Vehicular Networking Conference* (Boston, MA, USA, Dec. 2013).

[195] Yahoo! Inc. Yahoo! Weather Website. `https://mobile.yahoo.com/weather/`, 1994. Accessed: 2023-12-12.

[196] Yang, S.-R., Su, Y.-J., Chang, Y.-Y., and Hung, H. Short-term traffic prediction for edge computing-enhanced autonomous and connected cars. *IEEE Transactions on Vehicular Technology* (2019), 3140–3153.

[197] Yelp Inc. Yelp Website. `https://www.yelp.com`, 2004. Accessed: 2023-12-12.

[198] ZDNET Special Feature. What is the IoT? Everything you need to know about the Internet of Things right now. `https://www.zdnet.com/article/what-is-the-internet-of-things-everything-you-need-to-know-about-the-iot-right-now/`, 2020. Accessed: 2022-02-08.

[199] Zhang, J., Chen, B., Zhao, Y., Cheng, X., and Hu, F. Data security and privacy-preserving in edge computing paradigm: Survey and open issues. *IEEE Access* (2018), 18209–18237.

[200] Zhang, Y., You, W., Jia, S., Liu, L., Li, Z., and Qian, W. EnclavePoSt: A practical proof of storage-time in cloud via intel SGX. *Security and Communication Networks* (2022).

[201] Zhou, W., Jia, Y., Peng, A., Zhang, Y., and Liu, P. The effect of IoT new features on security and privacy: New threats, existing solutions, and challenges yet to be solved. *IEEE Internet of Things Journal 6*, 2 (2018).