Attribution in Anonymous Systems

by

Vishnu Teja Kilari

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved April 2026 by the
Graduate Supervisory Committee:

Guoliang Xue, Chair
Arunabha Sen
Yanchao Zhang
Satyajayant Misra

ARIZONA STATE UNIVERSITY

May 2026

ABSTRACT

Anonymous systems lie at the intersection of theoretical cryptography and practical privacy-preserving technologies. Attribution can make anonymous systems more practical by adding accountability, revocation, or reputation functionality. Incorporating attribution into such systems is challenging because their core purpose is to preserve anonymity and prevent identification. This dissertation addresses the issue of attribution in two types of anonymous systems: anonymous authentication systems and anonymous reputation systems.

Anonymous authentication systems enable a user to prove possession of valid credentials without revealing their identity, while revocation mechanisms ensure that misbehaving or no longer authorized users can be excluded from future access. These two goals are often in tension: strong anonymity can make revocation difficult, while efficient revocation mechanisms can introduce linkability or tracing features that weaken privacy. This dissertation studies the design, analysis, and implementation of revocable anonymous authentication systems in both pre-quantum and post-quantum settings, with a focus on practical constructions that preserve privacy under realistic trust and threat models.

These techniques are further extended to anonymous reputation systems, where users can accumulate and present trust signals anonymously without exposing their long-term identities or enabling unintended cross-session tracking. This dissertation develops separate frameworks for revocable anonymous authentication, anonymous reputation systems, and post-quantum revocable anonymous authenticated key exchange. The frameworks are analyzed extensively for their security and privacy properties, and detailed evaluations demonstrate the feasibility of these frameworks.

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my Ph.D. advisor, Dr. Guoliang Xue, for his unwavering guidance, support, and encouragement throughout my studies at Arizona State University. His insight, patience, expertise, and enthusiasm played an invaluable role in shaping my life and career. This dissertation would not have been possible without the guidance, feedback, and freedom he gave me over the course of this research.

My Ph.D. studies were supported by NSF grants 1461886, 1704092, and 1717197.

I am also sincerely grateful to my committee members, Dr. Arunabha Sen, Dr. Yanchao Zhang, and Dr. Satyajayant Misra, for their constructive comments and generous time. My heartfelt thanks go to my colleagues and friends: Buddha Puneeth Nandanoor, Anil Kumar Motupalli, Sandeep Narla, Arun Reddy Nelakurthi, Arun Balaji Buduru, Veda Prakash Kadiri, and Vivek Kopparthi. Their friendship and support kept me going over the years. I am fortunate to have worked with an incredible group of co-authors and collaborators. I want to thank my labmates, Xi Fang, Dejun Yang, Lingjun Li, Xinxin Zhao, Xiang Zhang, and Ruozhou Yu for their companionship, guidance, and encouragement. I also want to thank my colleagues at the Hispanic Research Center, Michael Sullivan, Ana Maria Regalado, and Vicente Magana for their kindness and encouragement.

I am deeply thankful to my wife and family for their endless love, patience, and sacrifice. Their belief in me and their encouragement carried me through the most challenging moments of this journey. I am very thankful to my parents, Jayachandra Kilari and Padmaja Kondeti, for their unconditional love and support throughout my life. I am thankful to my brother, Murali Krishna Teja Kilari, for his support. Finally, and most importantly, none of my achievements would have been possible without the love and patience of my wife, Harsha Undapalli. My heartfelt thanks to her for her endless support and care over the years.

TABLE OF CONTENTS

CHAPTER Page

CHAPTER Page

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

1.1 Motivation

Anonymous systems underpin many privacy-preserving technologies that are vital to modern digital life. In many applications, a system must verify that a participant is authorized, enrolled, or otherwise trustworthy, while simultaneously preventing disclosure of the participant's long-term identity. Examples include electronic voting, privacy-preserving access control, anonymous membership services, whistleblower platforms, and decentralized online communities. In these settings, the ideal authentication mechanism proves legitimacy without enabling profiling, surveillance, or tracking.

However, anonymity cannot be absolute in real-world systems because users may become compromised, lose authorization, or behave maliciously. Users in anonymous authentication systems may be incentivized to misbehave if they believe they cannot be identified. For example, anonymous social media services may struggle with bullying, abuse, and other harmful content. Therefore, a system that allows anonymity must also support revocation. In some settings, the system should additionally support traceability or accountability. The resulting design challenge is how to provide strong anonymity for honest users, efficient revocation for misbehaving users, and guarantees against forgery, collusion, and linkability. This dissertation studies that question through the lens of revocable anonymous authentication. It focuses on cryptographic mechanisms that allow users to authenticate

anonymously while enabling a distributed infrastructure to revoke the anonymity.

This problem is important not only in classical cryptographic settings, but also in the emerging post-quantum era, where cryptosystems must remain secure against adversaries equipped with large-scale quantum computation. Yet despite significant recent progress, the literature still lacks a broadly accepted standalone post-quantum framework that simultaneously provides strong anonymity, efficient revocation, and practical deployment. One reason is that many established anonymous authentication schemes rely on pairings, discrete logarithms, or related assumptions that are vulnerable to quantum algorithms. Post-quantum alternatives replace these with lattice-based, code-based, hash-based, or symmetric primitives. Although important building blocks have emerged, these remain only partially integrated, and some alternatives obtain practicality by weakening anonymity through linkability or related compromises rather than preserving the full functionality expected from revocable anonymous authentication. Thus, post-quantum revocable anonymous authentication is not simply a matter of replacing one assumption with another. Consequently, the development of a practical and fully post-quantum revocable anonymous authentication framework remains an important open challenge.

Anonymous reputation systems introduce an additional layer of complexity and utility. In many online environments, trust is built over time through repeated interactions, endorsements, or feedback. Yet conventional reputation systems rely on long-term identities, which can expose behavior patterns, enable surveillance, and create incentives for profiling. An anonymous reputation system seeks to reconcile these conflicting goals by allowing a participant to accumulate credibility while preventing observers from learning the participant's identity or linking all their actions. The challenge is how to bind the reputation to legitimate participation, how to prevent Sybil attacks and reputation inflation, and how to preserve unlinkability when reputation claims are presented repeatedly over time.

The literature has also emphasized the risks created by concentrated trust. Many privacy-

preserving authentication mechanisms rely on highly privileged authorities for enrollment, revocation, or signature opening. Concentrating revocation or opening power in a single authority creates a serious trust bottleneck. A solution is to distribute revocation or opening capability across multiple independent authorities and require threshold cooperation before revocation. Such a model makes unilateral abuse more difficult and improves robustness against coercion and corruption since an adversary must corrupt several authorities rather than only one.

1.2 Overview

In this dissertation, we study attribution in anonymous systems across four different scenarios. First, we propose and implement a revocable anonymous authentication framework for Vehicle-to-Grid (V2G) communications by leveraging federated trust management. As the number of electric vehicles (EVs) increases rapidly, there is a growing need to schedule EV charging at charging stations in advance. Charging scheduling requires communicating sensitive user information, such as location, charging status, and time of arrival, from an EV to a charging station. To protect user privacy, communications need to be anonymous. To defend against external attacks, communications need to be authenticated. However, the anonymity of users proven to be malicious must be revoked to protect against internal attacks. We consider these requirements and propose a revocable anonymous authentication framework leveraging federated trust management that is secure, scalable, and lightweight.

Second, we investigate application-level denial-of-service (DoS) attacks against the aforementioned revocable anonymous authentication framework. We augment the framework with a permit-based mechanism, which protects the system from application-level DoS attacks while providing revocable anonymous authentication and being secure, scalable, and robust.

Third, we consider anonymous reputation systems. A typical reputation system works

by tracking all information originating from a source along with the feedback to the information and its attribution to the source. The tracking of information and feedback could violate the privacy of users. Anonymous reputation systems have been designed to protect user privacy by ensuring user anonymity. Many anonymous reputation systems lack support for anonymous feedback updates and rely on a fully trusted authority. We propose an anonymous reputation system that ensures user anonymity while supporting anonymous feedback updates in a reputation system without the need for a fully trusted central authority. We also present a security analysis of our system against multiple types of attacks.

Finally, we propose and implement a secure, scalable, and efficient revocable anonymous authenticated key exchange in the post-quantum setting without relying on a central trusted authority. We distribute revocation capability and trust across multiple entities. We use NIST-recommended post-quantum cryptographic primitives while designing our system. We present a detailed security analysis of our system against multiple types of attacks against anonymous authentication systems and demonstrate through performance evaluation that our system is scalable and efficient.

1.3 Summarized Contributions

Addressing the aforementioned problems can be very challenging. This section summarizes our main contributions. Note that the concrete definitions of terms used, such as cryptographic primitives and protocols, are deferred to the corresponding chapters.

1.3.1 Revocable Anonymous Authentication in V2G Communications

We focus on designing a revocable anonymous authentication framework without a trusted central authority in V2G communications. Using our framework, an EV can communicate with the charging station using anonymous pseudonyms that cannot be linked to its identity by the charging station alone. If the EV's anonymity needs to be revoked after it

is proven malicious, the charging station can identify the EV with the help of the federated trust entities. We also present a proof mechanism that an EV can use to prove it did not misbehave. Our contribution to this problem [32], expanded in Chapter 2, is as follows:

- We proposed a secure, scalable, and efficient revocable anonymous authentication framework for V2G communications that is robust against attacks from malicious EVs.
- We also proposed a mechanism for an EV to prove its innocence in the event of a false accusation by the charging station.
- We analyzed the security of our framework and implemented it to demonstrate its security, scalability, and efficiency.

1.3.2 Robust Revocable Anonymous Authentication in V2G Communications

We designed a robust revocable anonymous authentication framework that defends against application-level denial-of-service (DoS) attacks. In our framework, application-level DoS attacks occur when malicious EVs authenticate themselves and launch successful DoS attacks before they are proven malicious and have their anonymity revoked and then penalized. We propose a permit-based mechanism, where each electric vehicle is issued only one anonymous permit per deposit at a time. A request is valid only if it contains a valid and unused permit, which protects the system from application-level DoS attacks. Our contribution to this problem [34], expanded in Chapter 3, is as follows:

- We uncovered and addressed application-level DoS attack for anonymously authenticated EV charging scheduling systems.
- We proposed a novel permit-based mechanism, which ensures the anonymity of EV users if they behave honestly, enables anonymity revocation in the event of mali-

cious behavior, and additionally defends against application-level DoS attacks on the charging infrastructure.

- We performed detailed security and privacy analysis of our framework, and evaluated its scalability and efficiency.

1.3.3 Anonymous Reputation System with Anonymous Feedback Update

We designed an anonymous reputation system that provides perpetual anonymous feedback updates without a trusted central authority. Our system enables users to provide information and feedback anonymously, attribute the feedback to the user's reputation anonymously, and perpetually update provided feedback anonymously without a trusted central authority. We performed detailed security analysis of our system against documented attacks against anonymous reputation systems and demonstrated its security. Our contribution to this problem [33], expanded in Chapter 4, is as follows:

- We designed a secure, scalable, and practical anonymous reputation system that implements all required functionalities (registration, information posting, feedback posting, and feedback update) while ensuring the anonymity and privacy of all users.
- Our system supports perpetual feedback updates anonymously.
- We performed detailed security analysis and implementation-based performance evaluation of our system.

1.3.4 Post-Quantum Revocable Anonymous Authenticated Key Exchange

We designed a revocable anonymous authenticated key exchange in the post-quantum setting without a trusted central authority. Using our framework, a user can anonymously authenticate and establish a shared key with a system using an ephemeral identity. If the

user's anonymity must be revoked because of misbehavior, the system can revoke the anonymity of the user using distributed trust entities. Our contribution to this problem, expanded in Chapter 5, is as follows:

- We proposed a secure, scalable, and efficient revocable anonymous authenticated key exchange in the post-quantum setting.
- We did not rely on a central trusted authority but distributed trust to prevent a single point of compromise.
- We analyzed the security of our framework and implemented it to demonstrate its security, scalability, and efficiency.

Chapter 2

REVOCABLE ANONYMOUS AUTHENTICATION IN V2G COMMUNICATIONS

2.1 Introduction

The smart grid is transforming modern electricity infrastructure. The increasing share of renewable energy in electricity generation introduces intermittency because these sources cannot consistently produce energy at all hours of the day. The increasing number of Electric Vehicles (EVs), combined with the intermittent nature of ever-increasing renewable energy sources, requires scheduling and coordination of EV charging to maintain grid stability and reliability. Efficient EV charging scheduling requires real-time interaction between EVs and grid components such as charging stations. These V2G communications include information such as estimated time of arrival and amount of charge required. Such data can be used to infer private information about an EV user, such as location, travel patterns, and charging patterns.

If a charging station or a network of charging stations managed by the same entity can correlate such information, it could lead to a significant privacy breach involving EV user data. To protect themselves and the grid, charging stations need to authenticate the communications from EVs to prevent attacks from external entities. Identity information of EVs cannot be stored in the charging stations to protect the privacy of EV users and to defend against insider attacks. Anonymous authentication systems solve both problems by providing anonymity and authentication.

The use of digital certificates has been proposed for enabling authenticated communication between EVs and charging stations [27]. However, this approach cannot provide anonymity. Pseudonym-based approaches provide anonymity [28] [39]. However, at least one of the system entities knows the original identity of the EV in these proposed approaches. Lynx [40] provides a completely anonymous authenticated framework for real-time reporting from EVs to charging stations using blind signatures. Token-based credential systems [11] allow a user to authenticate anonymously at most n times using n one-time tokens. This method uses an online zero-knowledge proof to verify the token, which is computationally expensive.

However, such completely anonymous authentication systems are vulnerable to attacks from malicious EVs. For example, an EV can communicate with a charging station, make a charging reservation, and then fail to honor the reservation. Anticipating the reservation, the charging station would have reserved the necessary charge from the grid. Such attacks can cause financial harm and grid instability if carried out at scale. Since completely anonymous authentication systems cannot identify the malicious EV(s) responsible for the attack due to the anonymity provided, these systems cannot take the required mitigation steps to defend against such attacks in the future.

A solution to this problem of completely anonymous authentication systems is to provide revocable anonymity. In revocable anonymous authentication systems, a participant's anonymity can be revoked by one or more system entities under specified conditions. One of the main challenges of such revocable anonymous authentication systems is to provide complete anonymity to all the participants while retaining the ability to revoke the anonymity of a participant when proven malicious. These two requirements are difficult to satisfy at the same time because to provide complete anonymity, no single system entity should be able to identify a participant independently. Furthermore, without any entity knowing the identity of a participant, revoking their anonymity will be challenging. An-

other challenge is to provide a process for the participants to prove their innocence without disclosing their identity to prevent misuse of the revocation mechanism.

Several anonymity revocation mechanisms have been proposed for anonymous authentication schemes. One of the popular cryptographic tools used to achieve this is group signatures [19]. However, in group signatures, the group manager knows the identity of a user, and if the manager is compromised or malicious, then the user's identity can be exposed. Traceable signatures [31] can also be leveraged to provide conditional anonymity. However, they are too computationally expensive to be used for communication between an EV and a utility. A revocable anonymity scheme based on threshold group signature schemes and blind signatures has been proposed [36]. This scheme uses a certificate generated by a user containing their identity as the basis of the protocol, which is verified by an intermediary before the start of the protocol. Communication between EVs and charging stations does not involve an intermediary that can verify the identity reported by an EV. In pseudonym-based protocols for conditional anonymous communication [28], the communication is truly anonymous since only the vehicle knows the pseudonyms. But this protocol contains a trusted third party (DMV), which knows the identity of each user to provide conditional anonymity. This third party is assumed to be trusted and immune to compromise, but it remains a single point of compromise.

In this chapter, we designed and evaluated a framework that can provide revocable anonymous authentication which is efficient, secure against malicious participants, and does not rely on central trusted authorities. Our framework is both lightweight and scalable. We validated its efficiency and scalability via implementations.

Our main contributions are summarized as follows:

- We proposed a secure, scalable, and efficient revocable anonymous authentication framework for V2G communications that is robust against attacks from malicious EVs.

- We also proposed a proof mechanism that an EV can use to prove it did not misbehave.
- Our framework does not have a central trusted authority but relies on federated trust entities for revocation.
- We analyzed the security of our framework and implemented it to demonstrate its security, scalability, and efficiency.

The rest of this chapter is organized as follows. Section 2.2 presents our system and threat model. Section 2.3 presents the cryptographic concepts needed to understand our framework and its protocols. Section 2.4 presents the detailed design of our framework. Section 2.5 presents the security analysis of our framework. Section 2.6 presents our performance evaluation results. Section 2.7 concludes this chapter.

2.2 System and Threat Model

2.2.1 System Model

The entities in our framework are a Certificate Authority (CA), a Department of Motor Vehicles (DMV), EVs and charging stations. In our system model, many EVs communicate with various charging stations. Because communications between different EVs and charging stations are independent, we consider a single EV and a single charging station when explaining and evaluating the framework. We assume that the CA, the DMV, and the charging station are always online and EVs can communicate with the charging station. In our framework, the registration is performed when the vehicle is brought into service. We assume that an EV registers its credentials with the CA and the DMV, and the EV is not compromised before and during this registration process. We assume that if an EV and the charging station have agreed on a reservation, then there is a guarantee that the charging

station has energy for charging of the EV.

2.2.2 Threat Model

We assume that the CA and DMV do not collude with each other or with the charging station, and that they cannot both be compromised. Both insider and external attackers are assumed to be computationally bounded. A malicious charging station may benefit from identifying an EV or learning its behavioral patterns. Denial of Service attacks are possible, i.e., a charging station can refuse to accept the messages or provide charging facilities to EVs, or a set of malicious EVs can make reservations for a large amount of energy and charging time to prevent benign EVs from charging. An attacker can compromise some of the EVs, charging stations, or both and turn them into malicious entities exposing their secrets. Possible attacks include privacy violation, misbehavior of users, man-in-the-middle attacks, forgery attacks, and replay attacks. All protocol messages contain fresh timestamps to prevent replay attacks.

2.3 Cryptographic Concepts

This section explains the cryptographic concepts needed to understand our framework and its protocols.

2.3.1 Blind Signatures

The blind signature protocol [18] allows a user v to get a signature $Sig_E(m)$ on a message m from a signer E by interacting with E. In the process of interaction, E does not know the message content and cannot link $Sig_E(m)$ with the protocol session during which $Sig_E(m)$ was created, or with the user that sent the message and received the signature.

2.3.2 Threshold Secret Sharing Schemes

Cryptographic (t, n)-threshold secret sharing schemes [48] are used to distribute sensitive information (the secret) among n parties such that the secret can be reconstructed only through the collaboration of at least t out of n parties.

2.3.3 Partial Message Proofs

Partial message proof [36] ensures that the signer E only signs a message m blindly if it can first verify a part p_m of the message m. This could be achieved by using a cut-and-choose protocol [43] or by selecting a blind signature scheme that incorporates zero-knowledge proofs on p_m. A cut-and-choose protocol works as follows. User U sends to E many blinded versions of the message m that must all contain a valid p_m. Signer E selects all but one of the messages which U has to unblind so that E can read them. Signer E signs the remaining blinded message m if all unblinded messages contain a valid p_m.

2.3.4 Implicitly Authenticated Diffie-Hellman Key Exchange

Diffie-Hellman Key Exchange (DHKE) is a method that allows two parties that have no prior knowledge of each other to jointly establish a shared secret key over an insecure channel. DHKE does not provide authentication and is vulnerable to the man-in-the-middle (MITM) attack. In order to thwart MITM attacks, Implicitly Authenticated DHKE (IADHKE) [24] uses digital signatures to authenticate the communicating parties.

2.4 Framework Overview

Our framework consists of four protocols: registration, token collection, pseudonym collection, and report and receipt generation. The token and pseudonym collection protocols are assumed to be executed offline while the EV is idle, for example at night. Reporting

Table 2.1: Table of Notations

$k(m)$	Message m encrypted with a key k
σ_E	Entity E's digital signature on the entire message
$Cert^v_{CA}$	Certificate issued to an EV v by CA during registration
τ^v_i	Token issued by a charging station to an EV v
δ^v_i	Pseudonym issued by a charging station to an EV v
ID_v	Identity of an EV v, like a VIN
$c(m, l)$	Commitment of message l with opening secret m
$b_e(m)$	Blinding factor e applied to message m
$u_e(m)$	Unblinding factor e applied to message m
$BS_k(m)$	Blind signature on m with key k
$EM^{k^A_i}_{k^E_i}(m)$	Message m encrypted with key k^E_i, then MAC is computed on the encrypted message with key k^A_i. Both the keys are derived from key k_i.
Pu_{CA}, Pr_{CA}	CA's public/private key pair used for communication
Pub_{CA}, Pri_{CA}	CA's public/private key pair used for signing/verifying the certificates

and receipt generation are done in real time. During the registration protocol, the EV registers with the DMV and the CA. The EV encrypts its identity (e.g., VIN) using a key and uses a threshold secret sharing scheme [48] to divide the key into three shares, giving one share each to the CA and the DMV along with the encrypted identity, while retaining the third share. In return, the EV receives a certificate from the CA to use as proof of identity. In the token collection protocol, the EV obtains multiple tokens from the charging station which are unlinkable to the EV. Each of these tokens is used in the next pseudonym collec-

tion protocol to obtain pseudonyms. The pseudonyms are unlinkable if the tokens are not reused. A request for a token consists of a certificate which is then included in the token issued.

In the pseudonym collection protocol, the EV uses an unused token as an anonymous credential to collect a pseudonym identity and a session key from the charging station for secure communication. In the reporting and receipt generation protocol, the EV chooses an unused pseudonym and its corresponding key to establish secure communication and exchange the required messages with the charging station. After a successful transaction between the EV and the charging station, the charging station issues a receipt to the EV. This receipt can be used by the EV to counter false complaints.

The unlinkability of the pseudonyms ensures that the charging station can verify that it is communicating with a valid EV, but it cannot identify the EV, resulting in anonymous authentication. If the anonymity of the EV needs to be revoked, the pseudonym used to communicate the malicious message is sent to the CA, which checks the certificate accompanying the pseudonym and decodes the private key (corresponding to the certificate issued by the CA) used to sign it. Based on this information, the CA adds the corresponding certificate to the blacklist and sends its share of the key corresponding to the encrypted ID associated with the certificate to the DMV. The DMV uses its own share and the share from the CA to recover the key, which is used to decrypt the ID. We present a penalty assessment mechanism that can be used by the DMV in cooperation with the charging station to disincentivize malicious behavior by imposing a penalty. We present a detailed overview of the protocols in our framework.

2.4.1 EV Registration Protocol

Protocol 1 presents the procedure followed by the EV v to register with the DMV and the CA and obtain a certificate from the CA. The EV v encrypts its identity ID_v with

a key k_v, and the key k_v is then divided into three shares k_v^1, k_v^2, and k_v^3 using the *(2,3)*-threshold secret sharing scheme. This ensures that at least 2 out of the 3 shares are needed to reconstruct the key k_v and decrypt ID_v. We use Blakley's secret sharing scheme [5].

The EV v establishes symmetric keys k_1 and k_2 with the DMV and the CA respectively using the IADHKE protocol. In Step 1, using key k_1, v securely sends a share of the key, k_v^1 and the encrypted ID, $k_v(ID_v)$ to the DMV. In Step 2, v uses key k_2 to securely send k_v^2 and $k_v(ID_v)$ to the CA and receives a certificate from the CA, denoted $Cert_{CA}^v$ in Step 3. The third share of the key, k_v^3 is retained by the EV v. The DMV and the CA each store their respective key shares along with the encrypted ID. The CA also stores the corresponding certificate.

Protocol 1 Registration Protocol for EV v

Input: Necessary information from the EV v.

Output: EV v registers itself with the DMV and the CA and obtains a certificate from the CA.

1: EV v sends the DMV its encrypted identity and the DMV share of the secret key (k_v^1) using a symmetric key k_1

$v \rightarrow DMV$: $k_1(k_v^1, k_v(ID_v))$.

2: EV v sends the CA its encrypted identity and the CA share of the secret key (k_v^2) using a symmetric key k_2

$v \rightarrow CA$: $k_2(k_v^2, k_v(ID_v))$.

3: The CA sends a signed certificate to the EV v

$CA \rightarrow v$: $k_2(Cert_{CA}^v)$.

2.4.2 Token Collection Protocol

Protocol 2 presents the procedure used by an EV v to obtain a token from the charging station $\mathcal{U}$. The EV v generates random numbers x_i and z_i and computes g^{x_i} along with a commitment $c(g^{x_i}, z_i)$, where g is a base for DHKE agreed upon by the EV and the charging station. This commitment commits z_i with the opening of g^{x_i}. The EV v sends the charging station $\mathcal{U}$ a blinded version of a message m, $b_e(m)$ containing the commitment and a certificate encrypted with the public key of the CA, $Pu_{CA}(Cert_i^v)$ to be signed by the charging station (Steps 1-3). The message also includes a digital signature on the message, σ_v. The EV v computes a framework-determined number of such messages and sends all of them to the charging station $\mathcal{U}$. $Cert_i^v$ is a certificate created by v and signed with the private key corresponding to $Cert_{CA}^v$, which is the certificate issued to v by the CA during the registration protocol.

The charging station $\mathcal{U}$ verifies that $Cert_i^v$ is a valid certificate using a partial message proof through cut-and-choose protocol. The charging station $\mathcal{U}$ requests the EV v to unblind all the messages except one that is chosen at random and the EV v complies by providing the unblinding factors for all but the chosen message. The charging station sends all the encrypted certificates in these unblinded messages to the CA. The CA decrypts them using its private key and verifies that they are valid certificates signed by the private key corresponding to one of the certificates issued by the CA, in this case $Cert_{CA}^v$. The CA notifies the charging station $\mathcal{U}$ that the certificates are valid and thus the charging station knows that the remaining unblinded message also has a valid certificate in it. The cut-and-choose protocol incurs little communication overhead because all the blinded messages are sent from v to $\mathcal{U}$ in one message and all the unblinding factors are provided to $\mathcal{U}$ in one message and all the encrypted certificates are sent to the CA by $\mathcal{U}$ in one message.

In Step 4, after confirming that the message actually contains a valid $Cert_i^v$ and the

digital signature σ_v is from a valid EV v, the charging station $\mathcal{U}$ performs a blind signature on the message and then digitally signs the whole message, $\sigma_{\mathcal{U}}$ and sends the blindly signed message along with the digital signature to the EV v in Step 5. In Step 6, after receiving the message from the charging station $\mathcal{U}$, the EV v confirms that the digital signature of the blindly signed message is a valid signature from the charging station $\mathcal{U}$ and then applies the unblinding factor and retrieves the charging station's signature on the message. The blind signature of the charging station $\mathcal{U}$ on the message is considered a token, τ_i^v (Step 7).

Protocol 2 Token Collection Protocol executed by EV v

Input: EV v has a valid certificate $Cert_{CA}^v$ from the CA.

Output: EV v acquires the tokens to interact with the charging station $\mathcal{U}$.

1: v calculates $m = c(g^{x_i}, z_i), Pu_{CA}(Cert_i^v)$.

2: v calculates $b_e(m)$, and the signature σ_v.

3: $v \rightarrow \mathcal{U}$: $b_e[c(g^{x_i}, z_i), Pu_{CA}(Cert_i^v)], \sigma_v$.

4: $\mathcal{U}$ verifies σ_v and blindly signs the message
$BS(b_e[c(g^{x_i}, z_i), Pu_{CA}(Cert_i^v)]), \sigma_{\mathcal{U}}$.

5: $\mathcal{U} \rightarrow v$: $BS(b_e[c(g^{x_i}, z_i), Pu_{CA}(Cert^v{}_i)]), \sigma_{\mathcal{U}}$.

6: v verifies $\sigma_{\mathcal{U}}$ and unblinds the message m
$u_e(BS(b_e[c(g^{x_i}, z_i), Pu_{CA}(Cert^v{}_i)]))$.

7: Unblinding the message results in blind signature of the charging station $\mathcal{U}$ on message m which is considered a token τ_i^v, $\tau_i^v = BS[c(g^{x_i}, z_i), Pu_{CA}(Cert_i^v)]$.

2.4.3 Pseudonym Collection Protocol

Protocol 3 presents the pseudonym collection protocol in which the EV v uses an unused token τ_i^v to acquire a pseudonym and a session key to use during the reporting and receipt generation protocol. In Step 1, the EV v sends g^{x_i} to the charging station $\mathcal{U}$. In Step

Protocol 3 Pseudonym Collection Protocol executed by EV v

Input: τ_i^v, g^{x_i}, $c(g^{x_i}, z_i)$, z_i, $Pu_{CA}(Cert_i^v)$.

Output: EV v has a pseudonym δ_i^v for communicating with the charging station $\mathcal{U}$.

1: $v \rightarrow \mathcal{U}$: g^{x_i}.

2: $\mathcal{U}$ generates random y_i, calculates g^{y_i} and computes k_i

$k_i = g^{y_i x_i}$.

3: $\mathcal{U}$ encrypts the message using k_i which contains the ordered pair of g^{x_i}, g^{y_i} along with its digital signature on the ordered pair and sends the encrypted message along with g^{y_i}.

4: $\mathcal{U} \rightarrow v$: $g^{y_i}, k_i(g^{x_i}, g^{y_i}, \sigma_{\mathcal{U}})$.

5: v takes g^{y_i}, computes $k_i = g^{x_i y_i}$ and decrypts the rest of the message using k_i and verifies $\sigma_{\mathcal{U}}$.

6: $v \rightarrow \mathcal{U}$: $k_i(g^{x_i}, g^{y_i}, c(g^{x_i}, z_i), Pu_{CA}(Cert_i^v), \tau_i^v, z_i)$.

7: $\mathcal{U}$ decrypts the message using k_i, opens the commitment using z_i and then checks that τ_i^v is a valid token.

8: $\mathcal{U}$ derives two keys k_i^E and k_i^A from key k_i using key derivation schemes and creates a pseudonym δ_i^v.

9: $\mathcal{U} \rightarrow v$: $EM_{k_i^E}^{k_i^A}(\delta_i^v, Pu_{CA}(Cert_i^v))$.

10: v decrypts the message and stores the pseudonym δ_i^v and $Pu_{CA}(Cert_i^v)$.

2, the charging station $\mathcal{U}$ generates a random y_i, calculates g^{y_i}, and computes k_i by $g^{y_i x_i}$. The charging station $\mathcal{U}$ sends a message to the EV v containing g^{y_i} and an ordered pair of (g^{x_i},g^{y_i}, $\sigma_{\mathcal{U}}$) encrypted with the key k_i (Steps 3-4). Here, $\sigma_{\mathcal{U}}$ is a signature of the charging station $\mathcal{U}$ on the ordered pair (g^{x_i}, g^{y_i}). After receiving the message from the charging station $\mathcal{U}$, in Step 5, the EV v uses g^{y_i} from the message and computes k_i using $g^{x_i y_i}$ and then proceeds to decrypt the rest of the message.

If the digital signature $\sigma_{\mathcal{U}}$ on the ordered pair (g^{x_i}, g^{y_i}) is from the charging station $\mathcal{U}$, then the charging station $\mathcal{U}$ has authenticated itself to the EV v. Now, in Step 6, the EV v sends the charging station a message containing g^{x_i}, g^{y_i}, $c(g^{x_i}, z_i)$, z_i, τ_i^v and $Pu_{CA}(Cert_i^v)$. In Step 7, the charging station $\mathcal{U}$ receives this message, decrypts it and finds that τ_i^v is indeed a blind signature on $c(g^{x_i}, z_i), Pu_{CA}(Cert_i^v)$ by itself and since an EV v has supplied it along with the commitment $c(g^{x_i}, z_i)$ and $Pu_{CA}(Cert_i^v)$ and the z_i that opens the commitment $c(g^{x_i}, z_i)$, the EV v has authenticated itself anonymously to the charging station $\mathcal{U}$. Now, the charging station $\mathcal{U}$ creates a pseudonym δ_i^v and derive two keys from the key k_i namely k_i^E and k_i^A using some Key Derivation Function (KDF) and sends a message to the EV v containing the pseudonym and $Pu_{CA}(Cert_i^v)$ (Steps 8-9). In Step 10, after receiving the message, EV v decrypts it and ensures that its MAC is correct and stores the pseudonym δ_i^v.

2.4.4 Reporting and Receipt Generation

When the EV v wants to communicate with the charging station $\mathcal{U}$ about a potential charging event, it crafts a message m containing the information, signs the message with the private key corresponding to the certificate associated with the pseudonym. It then sends the message, the signature along with the pseudonym and the encrypted certificate, $Pu_{CA}(Cert_i^v)$

$$v \rightarrow \mathcal{U}: EM_{k_i^E}^{k_i^A}(\delta_i^v, Pu_{CA}(Cert_i^v), m, \sigma_{Cert_i^v}^m)$$

where $\sigma_{Cert_i^v}^m$ denotes digital signature on m using the private key of $Cert_i^v$. The charging station $\mathcal{U}$ decrypts the message and verifies that the pseudonym included in the message is unused and $\sigma_{Cert_i^v}^m$ is a valid signature on m by sending the m and $\sigma_{Cert_i^v}^m$ to the DMV and $Pu_{CA}(Cert_i^v)$ to the CA. The CA retrieves the certificate and sends the private key corresponding to the certificate to the DMV who verifies the signature and notifies the charging station, which then proceeds to implement the required steps to prepare for the

arrival of the EV for charging.

If the EV v honors the reservation, the charging station $\mathcal{U}$ issues a receipt to the EV v acknowledging the completion of the transaction by issuing its digital signature on the reservation:

$$\mathcal{U} \rightarrow v\text{:}\, EM_{k_i^E}^{k_i^A}(\delta_i^v, Pu_{CA}(Cert_i^v), m, \sigma_{Cert_i^v}^m, \sigma_{\mathcal{U}})$$

where m is the message containing charging information sent by the EV v. The charging station only signs the request for reservation after the transaction is complete. Once the charging station signs the request for reservation, this signature acts as a receipt of the transaction.

2.4.5 Anonymity Revocation

If the EV does not honor the reservation made according to the message m, the charging station $\mathcal{U}$ sends a message to the CA containing the request of the EV and the pseudonym used

$$\mathcal{U} \rightarrow CA\text{:}\, Pu_{CA}(\delta_i^v, Pu_{CA}(Cert_i^v), m, \sigma_{Cert_i^v}^m).$$

The CA decrypts the message and then retrieves $(Cert_i^v)$. The CA then decodes the private key of the underlying certificate issued by itself, $Cert_{CA}^v$ used to sign this $Cert_i^v$ and sends out a message requesting the submission of corresponding receipt of the transaction. If the receipt is not produced within the required time, the CA proceeds to blacklist the $Cert_{CA}^v$ and then establishes a secure communication channel with the DMV and sends the key share associated with $Cert_{CA}^v$, k_v^2 to the DMV. Upon receiving this request, the DMV again broadcasts a request for the receipt corresponding to the reservation and in the absence of the receipt, the DMV uses its own share, k_v^1 and the share given to it by the CA, k_v^2 to reconstruct k_v and proceeds to decrypt the ID of the vehicle, $k_v(ID_v)$. If the charging station $\mathcal{U}$ generates a false report and the EV has a receipt for the transaction, the EV then

produces the receipt which is actually the charging station's signature on the request for reservation

$$EM_{k_i^E}^{k_i^A}(\delta_i^v, Pu_{CA}(Cert_i^v), m, \sigma_{Cert_i^v}^m, \sigma_{\mathcal{U}})$$

The CA proceeds to verify if it is indeed a valid signature of the charging station $\mathcal{U}$ in the receipt. If the signature is valid, the CA ignores the complaint of the charging station $\mathcal{U}$ on the EV v and starts appropriate proceedings to check the integrity of the charging station to determine if it is compromised.

2.4.6 Penalty

After anonymity revocation, the malicious EV v is subject to a penalty that covers the monetary loss suffered by the charging station, if any, and to deter the EV from misbehaving in the future. This penalty is computed as a function of the fixed price of the charge reserved for the EV and the market prices of the day. Let us assume that when the EV sent a message indicating its State of Charge and the charging station responded by reserving an amount of charge c at the market price of pr for each unit of the charge. When the EV does not honor the reservation, the penalty amount pe imposed on the EV is given by the function

$$pe = \beta(c \times pr) + c(pr_h - pr_l)$$

where β is assigned based on the nature of the disincentive mechanism, and pr_h and pr_l correspond to the highest and the lowest market price per unit charge of the day and are publicly known and can be validated. We assume that the penalty information is public knowledge.

2.5 Security Analysis

Our framework is immune to replay attacks since all the messages have fresh timestamps in them. The strength of the framework is not obvious in the face of some other

attacks that we discuss here.

Privacy: The charging station should not be able to link the pseudonym used for communication by an EV v to its original identity ID_v. The blind signature on the message by the charging station ensures that the charging station cannot in any way associate a token with identity of the EV that requested it. In the pseudonym collection protocol, the EV produces a token and gets a pseudonym. Since the token cannot be tied to an identity, it is not possible to link the pseudonym to an identity. If each unused token is used only once to acquire a pseudonym, the charging station cannot know that any two pseudonyms belong to the same EV. The pseudonyms of any EV v are temporally unlinkable and indistinguishable from the pseudonyms of other EVs. Thus, tracing of several pseudonyms cannot compromise an EV's privacy. The CA also cannot violate the privacy of an EV since it can only verify that the encrypted certificate provided by the EV is a valid certificate and it has no knowledge about the pseudonyms or tokens issued to an EV. Moreover, since each token contains a different certificate generated by an EV and signed with the private key corresponding to the certificate issued to the EV by the CA, a charging station cannot trace a certificate to a specific token or a certificate to an identity because only the CA knows the correspondence between the certificate and the identity.

Revocable anonymity: The blinded message provided to a charging station for its signature consists of a commitment and a valid certificate signed using the private key certified by the CA and encrypted with the CA's public key. The CA verifies that it is a valid certificate using cut-and-choose protocols. After the CA verifies that it is indeed a valid certificate, the blind signature, which is considered a token, is issued. The token contains this valid encrypted certificate which corresponds to the pseudonym issued for that token. Using this certificate, the charging station can report any malicious behavior of an EV. If an EV behaves maliciously, the malicious message along with the pseudonym used and corresponding certificate are sent to the CA which verifies that the report is true before

initiating the anonymity revocation procedure.

Man-in-the-middle attack: Our framework uses authentication during every protocol to prevent MITM attacks. During the token collection protocol, the EV authenticates itself to the charging station using its digital signature and the charging station authenticates itself to the EV by using its digital signature. This prevents MITM attacks in the token collection protocol. During the pseudonym collection protocol, the charging station authenticates itself using the digital signature on the ordered pair (g^{y_i}, g^{x_i}) and the EV authenticates itself using the commitment. During the reporting, authentication and non-repudiation are satisfied by having the message m signed with the certificate $Cert_i^v$. During the receipt generation, the signature of the charging station on the message ensures authentication and non-repudiation.

Forgery attack: An external attacker cannot forge tokens or pseudonyms because our system model assumes that an attacker cannot forge digital signatures. An EV trying to get the signature of a charging station on a real-time report so as to forge a receipt by the charging station will have to send the real-time report as a blinded message in the token collection protocol upon which the charging station can issue a blind signature. This is prevented by using cut-and-choose protocols to open the blinded message and check for the commitment inside the message and sending the encrypted certificate to the CA for verification. It can also be prevented by having the charging station sign the receipts with a different public/private key pair.

2.6 Performance Evaluation

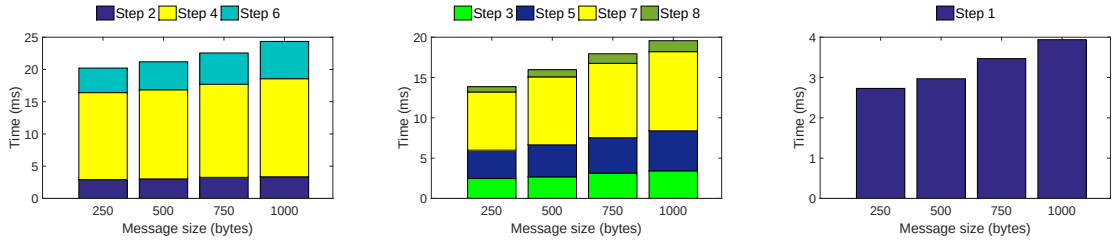
In this section, we describe the implementation of our protocols on real-world hardware and evaluate them to demonstrate that they can be implemented with reasonable latency and without any additional hardware or software requirements. We implemented our protocols using standard crypto libraries in Java. The protocol parts of the EV are implemented on an

Intel Core i5-2450M, 2.5 GHz machine with 8 GB RAM. Implementation of the charging station protocols is done on an Intel Core i7-6700K, 4.0 GHz machine with 32 GB RAM. All the results were averaged over 1000 runs. To evaluate the scalability and the robustness of the protocols, we measure the time taken by each step in the protocols with varying message sizes. This is important because the running time of several of the cryptographic primitives used in our protocols is dependent on message size.

Lynx [40] is the closest work to ours. However, Lynx does not provide revocable anonymity or non-repudiation. So, its reporting step has encryption and MAC operations while ours has a digital signature operation along with them to provide non-repudiation. Hence, we do not have any state-of-the-art work to compare against. Therefore, we do not compare our work with other approaches, but demonstrate the scalability of our framework in real-world settings.

We measure the time taken by each step in the token collection protocol, the pseudonym collection protocol, and the report and receipt generation protocol. We only consider the computation steps and do not consider the data transmission steps as transmission is dependent on network characteristics and communication technology used. Fig. 2.1a shows the time taken by the computation-intensive steps in the token collection protocol. Step 1, Step 3, and Step 5 are not shown since they do not affect the time taken by the protocol. Step 2 corresponds to the time taken by an EV for a blinding and a digital signature operation. Step 4 corresponds to the time taken by a charging station for a signature verification, a blind signature, and a digital signature operation. Step 6 corresponds to the time taken by an EV for a digital signature verification and an unblinding operation. Time taken by the token collection protocol for a 1,000-byte message is approximately 25 ms.

Fig. 2.1b shows the time taken by the computation-intensive steps in the pseudonym collection protocol. Step 3 shows the time taken by a charging station for a digital signature and an encryption operation. Step 5 corresponds to the time taken by an EV for a decryption



(a) In token collection protocol (b) In pseudonym collection protocol (c) During reporting by an EV

Figure 2.1: Time Taken by Various Steps in the Protocols of Our Framework

and a signature verification operation. Step 7 shows the time taken by a charging station for a decryption, a commitment verification and a digital signature verification operation. Step 8 corresponds to the time taken by a charging station for an encryption operation. Time taken by the pseudonym collection protocol for a 1,000-byte message is approximately 24 ms.

Fig. 2.1c shows the time taken in the report and receipt generation protocol. Since the receipt generation takes approximately the same amount of time as the reporting, we did not show it in our graphs.

As observed from our experiments, maximum time taken in each of the protocols is in the order of 10 ms. The token collection protocol and the pseudonym collection protocol can be executed when an EV is in idle state (during the night, in which an EV performs no functions but stays in a garage or a parking spot). Only the report and receipt generation protocol, which takes less than 10 ms, executes in real time. Step 4 in the token collection protocol and Step 7 in the pseudonym collection protocol require the highest amount of time, due to cryptographic operations involving digital signatures. The time taken by the steps in the protocols increases with the message size due to the dependence of cryptographic operations on the message size.

2.7 Conclusions

In this chapter, we proposed a revocable anonymous authentication framework for reporting of EV information that is robust against attacks by malicious EVs. Our framework allows EVs to authenticate themselves anonymously to a charging station for communicating real-time information. If an EV becomes malicious or misbehaves, our framework has mechanisms to revoke the anonymity of the EV to levy penalty on it after verifying that the report of maliciousness is accurate. Our framework is not only robust against attacks from users, but also efficient and scalable, as demonstrated by the results from our implementation.

Chapter 3

ROBUST REVOCABLE ANONYMOUS AUTHENTICATION FOR V2G COMMUNICATIONS

3.1 Introduction

The ever-increasing number of EVs can place significant load on the power grid through unscheduled charging events. Electric vehicle charging scheduling is necessary to ensure grid stability. Efficient EV charging scheduling requires interaction between the EVs and grid components like charging stations in real time. However, these communications can cause private information leakage of an EV user like location, travel patterns, and charging patterns. Anonymous information reporting can protect user privacy, but leaves charging stations vulnerable to attacks by unauthorized users. An anonymous authentication system can protect user privacy and defend against attacks by unauthorized users, but it cannot detect misbehaving authenticated users. A solution to this problem is a revocable anonymous authentication system that can revoke anonymity of malicious users after their misbehavior is proven.

Several anonymity revocation mechanisms have been proposed for anonymous authentication schemes. Group signatures [19] can be used to achieve anonymity revocation in anonymous authentication schemes. However, the group manager knows the identity of a user in group signatures and users can be deanonymized if the manager is compromised or malicious. PACP [28] is a framework in which vehicles interact with the Road Side Units

(RSUs) to generate pseudonyms for anonymous communications. However, this framework contains a trusted third party, the (DMV), which knows the true identity of each user. Traceable signatures [31] are another cryptographic tool to provide conditional anonymity but they are too computationally expensive to be used for communications between EVs and charging stations. Portunes [39] is a pseudonym-based authentication method, which allows at least one party to know an EV's identity. Li *et al.* [40] further proposed a method with complete anonymity for real-time reporting of EVs, using partially blind signatures. Saxena *et al.* [45, 46] proposed mutual authentication methods that provide forward privacy, identity anonymity, and untraceability. Afrin *et al.* [1] proposed an authentication method to achieve anonymity and time-flexibility in EV charging scheduling. However, all these solutions are vulnerable to insider attacks by malicious EVs. Our revocable anonymous authentication framework [32] provides anonymity revocation in case of EV misbehavior, but it is vulnerable to application-level DoS attacks.

Such revocable anonymous authentication systems are vulnerable to application-level denial-of-service (DoS) attacks. Consider a scenario where an anonymous authenticated EV simultaneously submits charging requests to all the charging stations in the vicinity for charging. Since the communication is anonymous until the EV is proven malicious, the charging stations cannot know that the requests came from the same EV. There is no limit to the number of requests each EV can place or to the number of charging stations these requests are directed at. The charging stations reserve the charge and wait for the EV. These requests will saturate the charge capacity of the charging stations, forcing them to deny charging requests from other EVs. This constitutes an application-level DoS attack. Even though the malicious EVs are eventually reported by charging stations and their anonymity revoked, this reporting and eventual revocation of anonymity takes time. After the complaint, the EV is given some time to refute a false accusation before its anonymity is revoked. The time between the charging request and the eventual revocation of anonymity

is the attack window. If multiple such EVs coordinate their attacks, they can mount a distributed denial-of-service (DDoS) attack.

A solution to this problem is a robust revocable anonymous authentication system that is resilient against application-level DoS attacks. One of the main challenges of such a system is to provide robustness against application-level DoS attacks without sacrificing the existing security and privacy properties. In this chapter, we designed a framework that provides robust, lightweight, and scalable revocable anonymous authentication that is secure against application-level DoS attacks.

Our main contributions are summarized as follows:

- We uncovered and addressed application-level DoS attack for anonymously authenticated EV charging scheduling systems.
- We proposed a novel permit-based mechanism, which ensures the anonymity of EV users if they behave honestly, enables anonymity revocation in the event of malicious behavior, and additionally defends against application-level DoS attacks on the charging infrastructure.
- Our framework does not have a central trusted authority but relies on federated trust entities for revocation.
- We performed detailed security and privacy analysis of our framework, and evaluated its scalability and efficiency.

The rest of this chapter is organized as follows. Section 3.2 presents our system and threat model. Section 3.3 presents the detailed design of our framework. Section 3.4 presents the privacy and security analysis of our framework. Section 3.5 presents our performance evaluation results. Section 3.6 concludes this chapter.

3.2 System and Threat Model

3.2.1 System Model

Our system consists of EVs, charging stations, and three Federated Trust Entities (FTEs): a Financial Authority (FA), a Certificate Authority (CA), and a Department of Motor Vehicles (DMV). We assume that each of these parties has its own unique public/private key pair. An entity such as a bank or a financial institution can act as the FA. We assume that the DMV conducts vehicle registration and it has no financial interest. We assume that each EV is uncompromised at the time of registration. We assume that the charging stations and the FTEs are always online, and communication channels exist between all parties. We assume that a charging station will reserve enough energy for an EV's charging request. For ease of illustration, we focus on the interactions among one EV, one charging station, and the FTEs and in the context of a single reservation involving one pseudonym only. An EV may obtain and possess multiple valid pseudonyms concurrently in our framework. To prevent traffic analysis attacks, we assume that all communications are carried over anonymous channels such as Tor.

3.2.2 Threat Model

The FTEs are assumed to be honest-but-curious: they follow all protocols honestly but may try to infer sensitive information regarding the EVs. A charging station is assumed to be honest-but-curious, i.e., it can benefit from identifying an EV and/or its patterns. Internal and external attackers are assumed to be computationally bounded. EVs and/or the charging stations can be compromised, and their secrets can be exposed. We assume that none of the FTEs (the CA, the FA, and the DMV) collude with each other or with the charging stations. We also assume that two or more of the FTEs do not become compromised at any time.

3.3 Framework Overview

Our framework to provide robust revocable anonymous authentication is based on distributing a permit to each EV. A permit is a blind token issued by a third-party authority. An EV submits its permit to the charging station along with the charging request. If the charging station accepts the request, it retains the permit until the EV visits and completes the transaction. After a successful transaction, the charging station provides a receipt to the EV, which can be redeemed by the EV to acquire another permit. To protect the anonymity of the EVs, the permits are issued, used, and redeemed with complete anonymity. To prevent an EV from accumulating permits and launching a DoS attack, every permit requires an upfront deposit.

Our framework consists of five protocols: registration, permit collection, token collection, pseudonym collection, and report and receipt generation. The token and pseudonym collection protocols are assumed to be executed offline during the time when the EV is idle (at night). Reporting and receipt generation are done in real time. The permit collection protocol is executed either when the EV makes its initial deposit, or after a random amount of time after the report and receipt generation protocol.

In the registration protocol, EV gets a certificate to use as its proof of identity after sharing the private information used in certificate generation among multiple FTEs. Specifically, the EV registers securely with both the DMV and the CA and collects the certificate from the CA. Registering with the DMV and the CA involves the EV using a secret sharing scheme [48] to divide the key used to encrypt its identity into three shares, giving two of them along with the encrypted identity to the CA and the DMV. The DMV and the CA working together (on proof of malicious behavior from a charging station) can revoke an EV's anonymity.

In the token collection protocol, an EV proves that it is authentic and requests and

Table 3.1: Table of Notations

$K(m)$	Message m encrypted with a key K
σ_E	Entity E's digital signature on the entire message
$Cert^v_{\text{CA}}$	Certificate issued by the CA to EV v at registration
p	A permit request from EV v to the FA
t	A token request from EV v to a charging station
ρ	A charging request from EV v to a charging station
ϕ^v	A permit issued by the FA to EV v
τ^v	A token issued to EV v by a charging station
δ^v	A pseudonym issued to EV v by a charging station
ID_v	Identity of EV v, *e.g.*, its VIN
$c(m, z)$	Commitment of message m with opening secret as z
$b_e(m)$	Blinding factor e applied to message m
$u_e(m)$	Unblinding factor e applied to message m
$BS_E(m)$	Blind signature of entity E on message m
$ETM^{K^A}_{K^E}(m)$	Message m encrypted with key K^E, and then attached with a Message Authentication Code (MAC) computed on the encrypted message with key K^A. Both keys are derived from a session key K.
$PK_{\text{CA}}, SK_{\text{CA}}$	CA's public/private key pair
$PK_{\text{DMV}}, SK_{\text{DMV}}$	DMV's public/private key pair
$PK_{\text{FA}}, SK_{\text{FA}}$	FA's public/private key pair
PK_v, SK_v	EV's public/private key pair

collects a blindly-signed token. This blindly signed token and its components prove the validity of EV and enable anonymity revocation of a proven malicious EV and are used to obtain a pseudonym and establish session keys during the pseudonym collection protocol. A token request consists of the certificate of the EV, which is subsequently included in the token.

To interact with a charging station, an EV needs a unique unused credential issued blindly by a charging station to the EV to preserve its anonymity; this credential is a pseudonym. In the pseudonym collection protocol, the EV uses an unused token to collect a pseudonym and associated session keys from the charging station. To prevent multiple pseudonyms from being linked to an EV, the tokens must not be reused. Before issuing this pseudonym, a charging station verifies that the requesting EV is not malicious. This verification is done using a blindly signed token by the charging station itself in the token collection protocol.

Since our framework allows each EV to collect multiple pseudonyms, an EV should not be able to use all the pseudonyms at once to prevent application-level DoS attacks. In our framework, to enforce this, an EV requires a permit to make a charging reservation. This permit is blindly signed by an FA and a new permit is only issued by the FA if the EV redeems a receipt of an old permit or makes a monetary deposit. The FA cannot link an EV to a permit since the request and issuance of the permit are made anonymously. During the charging request, when the EV sends a permit to the station, the station interacts with the FA to ensure that the permit is valid before reserving a slot for the EV.

In the report and receipt generation protocol, the EV establishes secure communications with the charging station using an unused pseudonym and its corresponding session keys. The EV uses a permit to make a charging reservation, and reports required information to the charging station. After a successful transaction between the charging station and the EV, the charging station issues a receipt to the EV. The EV can then redeem this receipt

for a new permit with another round of permit collection protocol or use it to counter false complaints by a charging station.

During the anonymity revocation protocol, a charging station complains about the malicious behavior of an EV to the CA and sends it the corresponding charging request. The CA verifies the certificate accompanying the pseudonym and decrypts the private key used to sign it. The CA and the DMV then work together to decrypt the ID of the malicious EV and impose punishment on it. The CA and the DMV broadcast a request for the receipt of the charging request from the corresponding EV after receiving the complaint and only proceed to subsequent steps if the receipt is not produced in a timely manner.

Now, we will describe the details of the protocols in our framework. For illustration, we use v and $\mathcal{U}$ to denote the EV and the charging station, respectively. In Table 3.1, we summarize the notations used in this chapter. Messages in all the protocols have timestamps, which are omitted in the protocol description for ease of illustration.

3.3.1 Registration

Protocol 4 Registration Protocol

Input: Encrypted identity $K_v(ID_v)$, key shares $K_v^{\text{DMV}}, K_v^{\text{CA}}$, symmetric keys $K_{v\text{DMV}}, K_{v\text{CA}}$.

Output: Certificate $Cert_{\text{CA}}^v$ from the CA.

1: $v \rightarrow$ DMV: $K_{v\text{DMV}}(K_v^{\text{DMV}} || K_v(ID_v))$.

2: DMV $\rightarrow v$: $PK_{CA}(SK_{DMV}(tkt))$.

3: $v \rightarrow$ CA: $K_{v\text{CA}}(Cert || PK_{CA}(SK_{DMV}(tkt)) || K_v^{\text{CA}} || K_v(ID_v))$.

4: CA $\rightarrow$ DMV: $PK_{DMV}(SK_{CA}(tkt))$.

5: CA $\rightarrow v$: $K_{v\text{CA}}(Cert_{\text{CA}}^v)$.

Protocol 4 explains the procedure followed by EV v to register with the DMV and the

CA to obtain a certificate from the CA. We assume that the EV is uncompromised during registration and physically present at the DMV. To achieve revocable anonymity, the EV's identity ID_v is encrypted with a secret key K_v, where the key is shared among three entities using a secret sharing scheme. Specifically, K_v is divided into three shares K_v^{DMV}, K_v^{CA}, and K_v^v using a $(2,3)$-threshold secret sharing scheme. The $(2,3)$-threshold secret sharing scheme requires at least 2 out of the 3 shares to reconstruct the key K_v which can then be used to decrypt ID_v. Our framework uses Blakley's secret sharing scheme [5].

Using the IADHKE protocol, EV v establishes secure communication channels with the DMV and the CA using symmetric keys $K_{v\text{DMV}}$ and $K_{v\text{CA}}$, respectively. In Step 1, the EV sends to the DMV the share K_v^{DMV} of the key and the encrypted ID $K_v(ID_v)$, both encrypted with the key $K_{v\text{DMV}}$. In Step 2, the DMV decrypts using $K_{v\text{DMV}}$, and stores the encrypted identity of v along with K_v^{DMV}. The DMV creates a message, tkt, containing the encrypted ID $K_v(ID_v)$ and a random nonce n, signs tkt using its private key SK_{DMV}, and then encrypts the message and its signature using the public key PK_{CA} of the CA. It sends this signed encrypted message to the EV. In Step 3, the EV sends to the CA the share K_v^{CA} of the key, the encrypted ID $K_v(ID_v)$, a certificate request ($Cert$), and the signed encrypted message given by the DMV, all of them encrypted with the key $K_{v\text{CA}}$. The certificate request, $Cert$, contains the public key of the EV, PK_v. In Step 4, the CA decrypts the message of the DMV using its private key SK_{CA} and verifies the signature of the DMV using DMV's public key PK_{DMV}.

If the signature is valid and the encrypted identity in the message tkt matches the encrypted identity sent by the EV, the CA signs the decrypted message tkt with its private key SK_{CA}, encrypts it with the public key PK_{DMV} of the DMV, and sends this signed encrypted message to the DMV. The DMV decrypts the message with its private key SK_{DMV}, verifies the signature of the CA using its public key PK_{CA}, matches the message it sent to the EV with the message it received from the CA, and notifies the CA that the EV is cur-

rently executing the registration protocol if the messages match and the EV is physically present at the DMV. If the CA receives the notification from the DMV that the EV with the encrypted identity $K_v(ID_v)$ is currently executing the registration protocol, the CA signs the certificate request and sends this certificate $Cert^v_{\text{CA}}$ to v.

The EV retains the third share of the key, K^v_v. This way, unless both the DMV and the CA are in agreement regarding the maliciousness of the EV, neither can obtain v's encryption key K_v and reveal its identity. The DMV and the CA both store their respective key shares along with the encrypted ID. The CA also stores the corresponding certificate.

To ensure that the EV is uncompromised during the registration protocol, the EV must be physically present at the DMV. In this case, the integrity of the software/firmware can be easily verified by the DMV. For example, the DMV can compute the checksum of the EV's software/firmware and compare it against the checksum provided by the vehicle's manufacturer. Recent advances in secure enclaves (e.g., Intel Software Guard Extensions (SGX) or ARM Trustzone) can be used to ensure that software/firmware of the vehicles can be attested inside the enclave in a vehicle to confirm that the process is not tampered with. Through this process, the DMV can ensure that the EV follows the secret sharing scheme in the registration protocol, which is crucial for enabling anonymity revocation. We make no assumptions regarding the integrity of the software/firmware of EV after the registration protocol.

3.3.2 Permit Collection

Before an EV attempts to make a charging request, it needs to execute Protocol 5 to obtain a permit anonymously from the FA. This is to ensure that each EV makes only one reservation at a time. When an EV v communicates with the FA to acquire a permit for the first time, it makes a deposit anonymously based on the amount required by the FA. To ensure anonymity during this step, the deposit must be anonymous and untraceable. This can

Protocol 5 Permit Collection Protocol

Input: Deposit or permit receipt, blinding & unblinding factors (b_f, u_f).

Output: Permit request p, permit ϕ^v.

1: v creates permit request $p = E||r$ where E is publicly known and r is a random secret.

2: v computes blinded message $b_f(p)$.

3: $v \rightarrow$ FA: (deposit / receipt, $b_f(p)$).

4: The FA verifies deposit or receipt and signs the blinded message: $BS_{\text{FA}}(b_f(p))$.

5: $FA \rightarrow v$: $BS_{\text{FA}}(b_f(p))$.

6: v obtains the permit ϕ^v by unblinding the signature on the message: $\phi^v = u_f(BS_{\text{FA}}(b_f(p))) = BS_{\text{FA}}(p)$.

be done using, for example, cash, prepaid cash cards, or privacy-preserving cryptocurrencies. In all the subsequent times, the EV presents a receipt for a previously issued permit to acquire a new permit anonymously. To defend against application-level DoS attacks, the initial deposit must be high enough to discourage the same EV from obtaining many permits that can be used to launch an application-level DoS attack. As long as the EV acts benignly, it will not lose anything.

To request a permit anonymously, the EV creates a permit request p by combining E and r, where E is a public number published by the FA, and r is a secret random number. The EV blinds the permit request p using a blinding factor b_f, resulting in a blinded permit request $b_f(p)$. Then, v sends this blinded permit request along with the initial deposit or a permit receipt to the FA. The FA verifies the deposit or the permit receipt, and signs the blinded permit request if it is valid. The EV applies the unblinding factor u_f to the blindly signed permit request, resulting in a blind signature of the FA on the permit request. Signature of the FA on the permit request p is considered as a permit. The EV can prove the ownership of the permit by revealing the r since only v knows about it.

3.3.3 Token Collection

Protocol 6 Token Collection Protocol

Input: Valid certificate $Cert^v_{\text{CA}}$ from the CA, blinding & unblinding factors (b_e, u_e).

Output: Token τ^v to interact with the charging station $\mathcal{U}$.

1: v generates random numbers x and z, and computes a commitment $c(g^x, z)$, a certificate $Cert^v$ signed by the public key certified in $Cert^v_{\text{CA}}$, and creates a token request $t = (c(g^x, z)||PK_{\text{CA}}(Cert^v))$.

2: v computes $b_e(t)$, and the signature σ_v.

3: $v \rightarrow \mathcal{U}$: $(b_e(t)||\sigma_v)$.

4: $\mathcal{U}$ verifies σ_v, verifies the certificate in t through a cut-and-choose protocol, signs the blind token request, and then signs the whole message: $(BS_{\mathcal{U}}(b_e(t))||\sigma_{\mathcal{U}})$.

5: $\mathcal{U} \rightarrow v$: $(BS_{\mathcal{U}}(b_e(t))||\sigma_{\mathcal{U}})$.

6: v verifies $\sigma_{\mathcal{U}}$ and obtains the token by unblinding the signature on t: $\tau^v = u_e(BS_{\mathcal{U}}(b_e(t))) = BS_{\mathcal{U}}(t)$.

The EV executes Protocol 6 to obtain a token anonymously from the charging station $\mathcal{U}$. This token will be used in the pseudonym collection protocol to obtain a pseudonym anonymously. In Step 1, the EV v and the charging station agree upon a base g as the base for DHKE; v then generates two random numbers x and z, and computes g^x and a commitment $c(g^x, z)$. The commitment is used to commit z to open g^x. EV v also creates a certificate $Cert^v$. To ensure its authenticity, it is signed with the public key corresponding to the certificate $Cert^v_{\text{CA}}$ issued to v by the CA during the registration protocol. The EV creates a token request t containing the commitment and the certificate encrypted with the public key of the CA, $PK_{\text{CA}}(Cert^v)$, to be signed by the charging station.

The EV blinds the token request t as $b_e(t)$ with a blinding factor e, and creates its signature σ_v upon that blind token request using its private key SK_v, in Step 2. In Step 3,

v sends the blind token request along with the signature to the charging station. For the cut-and-choose protocol, v computes a specific number (determined by the implementation) of blind token requests, and sends them to the charging station $\mathcal{U}$. Each blind token request has a unique certificate $Cert^v$ in it. To verify that $Cert^v$ is a valid certificate, the charging station $\mathcal{U}$ uses the partial message proof through the cut-and-choose protocol as follows. From all the blind token requests received by the charging station from EV v, the charging station selects a random blind token request and asks v to unblind all the blind token requests except the selected one.

With the exception of the selected blind token request, the EV provides all the other token requests and their blinding factors. The charging station verifies all the token requests against the corresponding blind token requests and sends all the encrypted certificates in the token requests to the CA. Since the certificates are encrypted using the public key of the CA, the CA decrypts them using its private key and verifies that they are valid certificates (signed by the public key corresponding to one of the certificates issued by the CA, in this case $Cert^v_{\text{CA}}$). After ensuring that all of the certificates are valid, the CA notifies the charging station $\mathcal{U}$ regarding their validity. Since the selection is random, the charging station knows that the selected blind token request has a valid certificate in it. The cut-and-choose protocol has little communication overhead, because all the blind token requests are sent to $\mathcal{U}$ from v in a single message.

In Step 4, after ensuring that the blind token request actually contains a valid certificate and a valid digital signature (σ_v is verified using the public key of the EV, PK_v), the charging station issues a digital signature on the blind token request and then issues a signature on the whole message ($\sigma_{\mathcal{U}}$). The charging station sends the blindly signed token request and the digital signature to EV v. After v receives the blindly signed token request from $\mathcal{U}$, it verifies that the digital signature on the blindly signed token request is a valid signature (from the charging station), and applies the unblinding factor to retrieve the charging

station's signature on the token request in Step 6. The charging station's signature on the token request t is considered a token, τ^v.

3.3.4 Pseudonym Collection

Protocol 7 Pseudonym Collection Protocol

Input: Token τ^v, commitment $c(g^x, z)$, proof of commitment (g^x,z), encrypted certificate $PK_{\text{CA}}(Cert^v)$.

Output: Pseudonym δ^v for reporting to $\mathcal{U}$.

1: $v \rightarrow \mathcal{U}$: g^x.

2: $\mathcal{U}$ generates random y, computes g^y and $K_{v\mathcal{U}} = g^{yx}$.

3: $\mathcal{U}$ generates a message containing $(g^x||g^y)$ and its signature $\sigma_{\mathcal{U}}$ on $(g^x||g^y)$, and encrypts it with $K_{v\mathcal{U}}$.

4: $\mathcal{U} \rightarrow v$: $(g^y, K_{v\mathcal{U}}(g^x||g^y||\sigma_{\mathcal{U}}))$.

5: v uses g^y and calculates $K_{v\mathcal{U}} = g^{xy}$ and decrypts the rest of the message using $K_{v\mathcal{U}}$ and verifies $\sigma_{\mathcal{U}}$.

6: $v \rightarrow \mathcal{U}$: $K_{v\mathcal{U}}(g^x||g^y||c(g^x, z)||PK_{\text{CA}}(Cert^v)||\tau^v||z)$.

7: $\mathcal{U}$ decrypts the message using $K_{v\mathcal{U}}$, opens the commitment using z and then checks that τ^v is a valid token.

8: $\mathcal{U}$ creates a pseudonym δ^v and derives two keys ${K_{v\mathcal{U}}}^E$ and ${K_{v\mathcal{U}}}^A$ from key $K_{v\mathcal{U}}$ using a KDF.

9: $\mathcal{U} \rightarrow v$: $ETM^{{K_{v\mathcal{U}}}^A}_{{K_{v\mathcal{U}}}^E}(\delta^v||PK_{\text{CA}}(Cert^v))$.

10: v decrypts the message and stores δ^v and $PK_{\text{CA}}(Cert^v)$.

Protocol 7 illustrates the pseudonym collection protocol. In this protocol, EV v uses an unused token τ^v to acquire a pseudonym and a session key to be used during the report and receipt generation protocol. The EV sends a part of the commitment, g^x, to the charging

station $\mathcal{U}$ in Step 1. As mentioned before, g is the agreed base of the DHKE protocol by both v and $\mathcal{U}$. After receiving g^x, the charging station generates a random number y, computes g^y, and calculates the session key $K_{v\mathcal{U}} = g^{yx}$ in Step 2. In Step 3, the charging station creates a message containing g^y and $(g^x||g^y||\sigma_{\mathcal{U}})$ encrypted with the key $K_{v\mathcal{U}}$, where $\sigma_{\mathcal{U}}$ is a signature of $\mathcal{U}$ on $(g^x||g^y)$. The charging station sends this message to v in Step 4.

In Step 5, after receiving the message from $\mathcal{U}$, v computes $K_{v\mathcal{U}} = g^{xy}$ by using g^y from the message. After computing $K_{v\mathcal{U}}$, v proceeds to decrypt the rest of the message. The EV verifies the validity of digital signature $\sigma_{\mathcal{U}}$ on $(g^x||g^y)$ to ensure that it is from $\mathcal{U}$. If it is valid, then $\mathcal{U}$ has authenticated itself to v. In Step 6, v sends $\mathcal{U}$ an encrypted message containing g^x, g^y, $c(g^x, z)$, z, τ^v, and $PK_{\text{CA}}(Cert^v)$. After receiving this message, $\mathcal{U}$ decrypts it and verifies that τ^v is its signature on $c(g^x, z)||PK_{\text{CA}}(Cert^v)$ in Step 7. Since v provided it along with the commitment $c(g^x, z)$ and the secret z that opens the commitment, v has authenticated itself anonymously to $\mathcal{U}$. In Step 8, $\mathcal{U}$ creates a pseudonym δ^v, and derives two keys from the key $K_{v\mathcal{U}}$, namely ${K_{v\mathcal{U}}}^E$ and ${K_{v\mathcal{U}}}^A$, using a Key Derivation Function (KDF).

We use Encrypt-then-MAC (ETM) to ensure that only untampered messages are read by both the EV and the charging station. To provide general security of the protocols, we use KDF to generate two keys for the ETM. The key ${K_{v\mathcal{U}}}^E$ is used for encryption while the key ${K_{v\mathcal{U}}}^A$ is used for Message Authentication Code (MAC). This ensures that if the key in either one of the encryption or the authentication scheme is compromised, then the other scheme is not automatically compromised. In Step 9, $\mathcal{U}$ sends a message to v containing the pseudonym and $PK_{\text{CA}}(Cert^v)$. After receiving the message, v verifies that its MAC is correct, decrypts it, and stores the pseudonym δ^v, in Step 10.

3.3.5 Report and Receipt Generation

The report and receipt generation protocol is shown in Protocol 8. When v wants to

Protocol 8 Report and Receipt Generation Protocol

Input: Permit ϕ^v, permit request p, pseudonym δ^v, encrypted certificate $PK_{\text{CA}}(Cert^v)$, session key $({K_{v\mathcal{U}}}^E, {K_{v\mathcal{U}}}^A)$, charging request information $info$.

Output: Transaction receipt $\sigma_{\mathcal{U}}^{\rho}$, permit receipt $\sigma_{\mathcal{U}}^{\phi^v}$.

1: v creates message $cred = ((info||\phi^v)||PK_{\text{FA}}(p))$, and charging request $\rho = (\delta^v||PK_{\text{CA}}(Cert^v)||cred||\sigma_{Cert^v}^{cred})$.

2: $v \rightarrow \mathcal{U}$: $ETM_{{K_{v\mathcal{U}}}^E}^{{K_{v\mathcal{U}}}^A}(\rho)$.

3: $\mathcal{U}$ decrypts the message and verifies that δ^v is unused.

4: $\mathcal{U} \rightarrow$ DMV: $(cred||\sigma_{Cert^v}^{cred})$.

5: $\mathcal{U} \rightarrow$ CA: $PK_{\text{CA}}(Cert^v)$.

6: CA $\rightarrow$ DMV: $Cert^v$.

7: The DMV verifies that $\sigma_{Cert^v}^{cred}$ is the signature of $Cert^v$ on $cred$, and notifies $\mathcal{U}$.

8: $\mathcal{U} \rightarrow$ FA: $(\phi^v||PK_{\text{FA}}(p))$.

9: The FA decrypts $PK_{\text{FA}}(p)$, verifies that the permit ϕ^v is valid and unused, and notifies $\mathcal{U}$.

10: If both $\sigma_{Cert^v}^{cred}$ and the permit are valid, $\mathcal{U}$ reserves the charge and waits for EV arrival.

11: After v finishes the transaction, $\mathcal{U}$ issues v the receipts:

$$\mathcal{U} \rightarrow v: ETM_{{K_{v\mathcal{U}}}^E}^{{K_{v\mathcal{U}}}^A}(\rho||\sigma_{\mathcal{U}}^{\rho}||\sigma_{\mathcal{U}}^{\phi^v}),$$

where $\sigma_{\mathcal{U}}^{\rho}, \sigma_{\mathcal{U}}^{\phi^v}$ are $\mathcal{U}$'s signatures on ρ, ϕ^v respectively.

communicate with $\mathcal{U}$ to schedule a charging service, it crafts a message $cred$ containing the request information, the permit, and the permit request encrypted with the FA's public key. Note that to make a reservation, the EV must use a unique and unused pseudonym each time. Otherwise, the anonymity of the EV may be broken if the adversary can link multiple reservations using the same pseudonym. It then creates a charging request ρ using the pseudonym δ^v, the certificate $Cert^v$ encrypted with the CA's public key, the message $cred$, and the signature of $cred$ using the private key corresponding to $Cert^v$. The request is then sent to $\mathcal{U}$, encrypted and authenticated with MAC.

The charging station decrypts the request and verifies that the included pseudonym is unused, and that $\sigma^{cred}_{Cert^v}$ is a valid signature on $cred$ by sending $cred$ and $\sigma^{cred}_{Cert^v}$ to the DMV and $PK_{\text{CA}}(Cert^v)$ to the CA. The CA retrieves the corresponding certificate and sends it to the DMV, which then verifies the signature and notifies $\mathcal{U}$. The charging station then extracts the permit and the encrypted permit request, and sends them to the FA securely. The FA decrypts the permit request, verifies its own signature on the permit as well as that the permit has never been used, and notifies $\mathcal{U}$. Upon receiving positive verification from all FTEs, the charging station then proceeds to preparing for the arrival of v for charging.

If v arrives for the reservation and completes the transaction, $\mathcal{U}$ issues a receipt $\sigma^{\rho}_{\mathcal{U}}$ to v by signing the entire request ρ. It also issues a permit receipt $\sigma^{\phi^v}_{\mathcal{U}}$ by signing only on the permit ϕ^v. $\mathcal{U}$ only signs the request and the permit after the transaction is complete. This ensures that the EV will not leave the charging station before the transaction completes.

3.3.6 Anonymity Revocation

If v does not show up for charging according to the request ρ, the anonymity revocation protocol (Protocol 9) will be executed. First, $\mathcal{U}$ sends a complaint to the CA containing the request of v and the pseudonym used in the request. Since the permit is not signed, the malicious EV will not have a permit receipt, and hence it cannot acquire another permit

Protocol 9 Anonymity Revocation Protocol

Input: Malicious charging request ρ.

Output: ID_v of v which issued ρ.

1: $\mathcal{U}$ complains to the CA with the unattended request ρ: $\mathcal{U} \rightarrow$ CA: ρ.

2: The CA gets $Cert^v$ and decodes which $Cert^v_{\text{CA}}$ signed it.

3: The CA broadcasts a request for receipt of the transaction pertaining to $Cert^v$. If the CA does not receive the receipt in a given amount of time, the CA blacklists $Cert^v_{\text{CA}}$.

4: CA $\rightarrow$ DMV: $(Cert^v_{\text{CA}} || K_v^{\text{CA}})$.

5: The DMV retrieves K_v from $(K_v^{\text{CA}}, K_v^{\text{DMV}})$ and uses the key K_v to decrypt the identity of ID_v.

6: If v provides the receipt, then the CA verifies the validity of the receipt and aborts the complaint from $\mathcal{U}$.

without paying another deposit. The CA decrypts the message and then retrieves $Cert^v$. The CA decodes the private key of $Cert^v_{\text{CA}}$ used to sign $Cert^v$, and sends out a message requesting for the receipt of this request. If v does not provide the receipt within a time limit, the CA blacklists $Cert^v_{\text{CA}}$.

The CA then establishes a secure channel with the DMV, and sends the key share K_v^{CA} corresponding to $Cert^v_{\text{CA}}$ to the DMV. Upon receiving this information, the DMV uses its own share K_v^{DMV} and the share K_v^{CA} given by the CA to reconstruct K_v, and proceed to decrypt the encrypted ID of the vehicle, $K_v(ID_v)$. If the charging station generates a false complaint and the EV possesses a receipt for the transaction, the EV provides the receipt, which is actually the charging station's signature on the request for reservation, to the CA:

$$v \rightarrow \text{CA}: (\rho || \sigma^{\rho}_{\mathcal{U}}).$$

The CA proceeds to verify if it is indeed a valid signature of the charging station in the receipt. If the signature is valid, the CA ignores the complaint of the charging station $\mathcal{U}$

on the EV. A separate procedure can later be executed to check if the charging station is compromised or malfunctioning, which is out of the scope of this dissertation.

3.3.7 Multiple Permits

In practice, there are legitimate use-cases which require an EV user to hold multiple permits. This requires an EV to make multiple reservations in advance before fulfilling any of them. Because the main focus of our work is to defend against the application-level DoS attack by a malicious EV, we require that each EV can only obtain and possess one permit with one deposit, each permit redeemable for one reservation at any time.

To facilitate multiple charging events, we allow an EV to obtain multiple permits for making multiple concurrent reservations, but the EV must make multiple deposits as well. To prevent an EV with multiple deposits from launching application-level DoS attack, punishment can be enforced after the EV's malicious behavior is confirmed and its identity is revealed through anonymity revocation. For example, all the deposits associated with the permits that were used to launch an attack are forfeited to the FA.

3.4 Privacy and Security Analysis

In this section, we perform a detailed privacy analysis of our framework, and discuss the security of the framework.

3.4.1 Privacy Analysis

Privacy from the charging station: The charging station cannot link a pseudonym or a permit used by an EV v to its original identity ID_v. Unlinkability of a token and a permit with the identity of the EV is ensured by the blind signature scheme used during the token collection protocol and the permit collection protocol.

In the pseudonym collection protocol, the EV produces a token and receives a pseudonym anonymously. Since a token cannot be tied to an identity, it is not possible to link the pseudonym to an identity. If each token is used only once to acquire a pseudonym, the charging station cannot link any two pseudonyms belonging to the same EV. Further, the blind signatures used in our framework ensure that the pseudonyms of any EV are unlinkable and indistinguishable from the pseudonyms of other EVs. Hence, tracking of multiple pseudonyms of a single EV or multiple EVs cannot compromise an EV's privacy either.

Privacy from FTEs: First, since the CA and the DMV only verify the validity of the encrypted certificates sent by the charging station, they do not know about the permits, tokens, or pseudonyms issued to an EV from the normal protocols. Specifically, the cut-and-choose protocols will prevent the CA from acquiring the specific certificate in the actual token used to acquire a pseudonym, as each token contains a different certificate generated by the EV itself.

The FA verifies the validity of permits, which are anonymized at issuance, so it cannot violate the privacy of the EV. The secret sharing scheme prevents either the DMV or the CA from unilaterally obtaining the EV's identity without the agreement from the other due to confirmed maliciousness. Note that the registration protocol also employs anonymized communications. Since it happens before any permit or charging request, it does not contain any information regarding the permits, tokens and pseudonyms of an EV.

3.4.2 Security Analysis

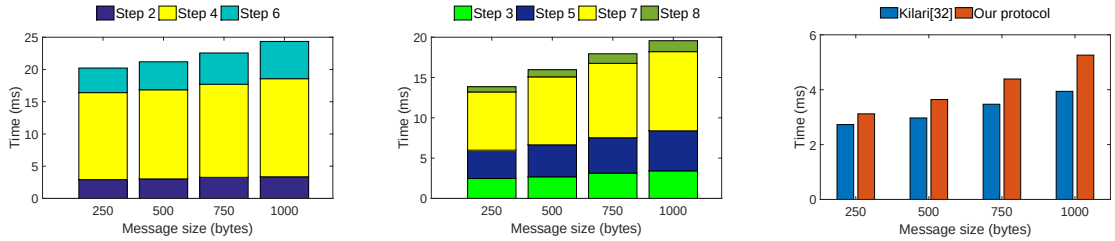
DoS/DDoS attacks: A malicious EV may launch DoS attacks either on a single charging station, or on all charging stations in an area, which can prevent other EVs from receiving charging services. In our framework, however, launching such an attack requires the user to possess a large number of valid permits at the same time. This can be done through accumulating permits using one or multiple EVs.

However, since each permit requires a deposit of a high-enough amount, such an attack can only be launched by an attacker with huge financial resources, making the attack economically infeasible for most common attackers. Moreover, once an attack is detected, the user's initial deposit can be lost (due to inability to obtain another permit), and the CA can blacklist the certificate(s) of one or multiple involved EVs and broadcast to the whole network, so that future requests by the EVs will not be accepted network-wide. Both punishments make an attack extremely costly for the attacker, economically preventing the attack from happening.

Forgery attacks: Our system model assumes that an attacker cannot forge digital signatures. Unless an external attacker can forge digital signatures, she cannot forge tokens or pseudonyms. An EV may try to get the signature of a charging station on a real-time report so as to forge a receipt. To do so, it needs to send the real-time report as a blinded message in the token collection protocol, upon which the charging station can issue a blind signature. This is prevented by the cut-and-choose protocol, which checks for the commitment inside the message and sends the encrypted certificate to the CA for verification. It can also be prevented by having the charging station sign the receipts with a different public/private key pair.

Man-in-the-middle attacks: In order to prevent the MITM attacks, our framework uses authentication in every protocol. For example, authentication between the EV and the charging station during the token collection protocol is achieved using their digital signatures on their messages. The charging station authenticates itself using the digital signature on (g^x, g^y) and the EV authenticates itself using the commitment in the pseudonym collection protocol. The EV authenticates to the charging station by signing the message with the certificate $Cert^v$ while reporting. During the receipt generation, the charging station authenticates through its signature on the message.

Replay attacks: Since all the messages are associated with timestamps, the framework



(a) In token collection protocol (b) In pseudonym collection protocol (c) During reporting by an EV

Figure 3.1: Time Taken by Various Steps in the Protocols of Our Framework

is immune to replay attacks.

Collusion attacks: Collusion between malicious EVs and charging stations cannot affect the anonymity of the honest EVs. If there are at least two benign EVs, the charging station cannot differentiate between their pseudonyms and by extension cannot infer their private information.

3.5 Performance Evaluation

In this section, we analyze and evaluate the performance of our framework.

3.5.1 Computation Overhead

We first show the computation overhead of our protocols through real-world implementations. Our framework can be implemented without any special hardware. We implemented our protocols using standard crypto libraries in Java. The EV parts of the protocols were implemented on an Intel Core i5-2450M (2.5 GHz) machine with 8 GB RAM. The charging station parts of the protocols were implemented on an Intel Core i7-6700K (4.0 GHz) machine with 32 GB RAM. We measured the time taken by each computation-intensive step in the protocols in order to evaluate the scalability of the protocols. All the results were averaged over 1000 runs.

We used the following implementations of the cryptographic primitives in our proto-

cols. For asymmetric encryption and blind signatures, we used RSA with 2048-bit keys. For digital signatures, we used SHA256 with RSA. For symmetric encryption and ETM (Encrypt-then-MAC), we used AES-CBC (Cipher Block Chaining) with 256-bit key and AES-CBC-MAC for MAC with 256-bit key.

Besides evaluating our proposed framework alone, we compared our framework to the framework in our preliminary work [32]. To the best of our knowledge, our framework in [32] is the state-of-the-art revocable anonymous authentication framework for EV information reporting. Since the major difference between [32] and the framework proposed in this chapter is the report and receipt generation protocol, the comparison was done regarding this protocol alone.

We compared the time taken by computation-intensive steps in the token collection protocol, the pseudonym collection protocol, and the report and receipt generation protocol. Fig. 3.1a shows the time taken by the computation-intensive steps in the token collection protocol. Step 2 corresponds to the time taken by an EV for a blinding and a digital signature operation. Step 4 corresponds to the time taken by a charging station for a signature verification, a blind signature, and a digital signature operation. Step 6 corresponds to the time taken by an EV for a digital signature verification and an unblinding operation. Time taken by the token collection protocol for a 1,000-byte message was approximately 25 ms.

Fig. 3.1b shows the time taken by the computation-intensive steps in the pseudonym collection protocol. Step 3 corresponds to the time taken by a charging station for a digital signature and an encryption operation. Step 5 corresponds to the time taken by an EV for a decryption and a signature verification operation. Step 7 corresponds to the time taken by a charging station for a decryption, a commitment verification and a digital signature verification operation. Step 8 corresponds to the time taken by a charging station for an encryption operation. Time taken by the pseudonym collection protocol for a 1,000-byte message was approximately 24 ms.

Fig. 3.1c compares the time taken by our report and receipt generation protocol against the time taken by the same protocol in [32]. Since receipt generation took approximately the same amount of time as reporting, we did not show it in the figures. As we can see, our protocol took slightly more time than the protocol in Kilari *et al.* [32]. This additional time was due to the encryption of the permit requests. During receipt generation, the additional time was due to computing the signature of the charging station on the permit. In our experiments, the maximum time taken in each protocol was in the order of 10 ms. The token collection and the pseudonym collection protocols can be executed when an EV is idle. Only the report and receipt generation protocol, which took less than 10 ms in experiments, is executed in real time.

Step 4 in token collection and Step 7 in pseudonym collection took the longest time, due to operations involving digital signatures. Time taken by the steps increased with the message size, due to the dependence of cryptographic operations on the message size. The token collection and the pseudonym collection protocols took a combined 49 ms (25 + 24) per EV. This means that a charging station can serve a million EVs in 14 hours. Even including the time required for the cut-and-choose protocols, the time required for our framework is less than an average idling time of an EV (15 mins, for every 24 hours), demonstrating the scalability of our framework.

3.5.2 Communication Overhead

Below, we further analyze the communication overhead of our framework in terms of the messages sent and received by the EV and the charging station during the execution of our normal operation protocols, where all asymmetric operations are performed on a group of a prime order of M bits. *CS* represents a charging station serving N EVs simultaneously. R is the number of messages used in the cut-and-choose protocol. D is the size of deposit during initial permit collection. P is the size of the pseudonym. C is the size of the

commitment key. Since the registration protocol and the anonymity revocation protocol are executed infrequently, we omit their communication overhead in the analysis. Unlike the above experiments, we do not assume specific implementations of our protocols for generality. We summarized the communication overhead of our normal operation protocols in Table 3.2. For simplicity, we omit the sizes of the payload message and the MAC.

We can see that the token collection protocol has the highest communication overhead. This is due to the use of the cut-and-choose protocol to verify the validity of EV's token requests. Assuming $M = 4096$ bits for the prime q of the group, $R = 10$ for the cut-and-choose protocol, $P = 4096$ bits for the size of the pseudonym, and $N = 1000$ EVs in the system, the worst-case size for the charging station is Protocol 3 with total size 18.55 MB while receiving and Protocol 5 with total size 6.35 MB while sending. For the EV, the token collection protocol has the largest worst-case send size of 19 KB while sending and the report and receipt generation protocol with total size 3.5 KB while receiving. Alternatively, we can use Zero Knowledge Proofs (ZKPs) to replace the cut-and-choose protocol, which may reduce the communication overhead but may lead to excessive computational overhead.

Only the report and receipt generation protocol must be executed in real time during the time of reservation, while all other protocols can be executed offline. The report and receipt generation protocol has the worst-case send message size of 6.35 MB and receive message size of 2.44 MB for a charging station. For an EV, report and receipt generation protocol has the worst-case send message size of 2.5 KB and receive message size of 3.5 KB. Note that it is highly unlikely that all EVs will be interacting with the charging station at the same time, hence the worst-case is very rare. The worst-case communication overhead for the EV for the real-time protocol is low. Also, since the only protocol that presents a significant communication overhead to the EV and/or the charging station is token collection protocol which is executed offline, and the communication overhead of both the EV and the charging

station for the only real-time protocol (report and receipt generation protocol) is sufficiently low, our framework is easily scalable.

Table 3.2: Communication Overhead

Protocol	**Entity**	**# Messages**		**Total Size**	
		Send	**Recv**	**Send**	**Recv**
Protocol 2	EV	1	1	$\max(D, 2M) + M$	M
Protocol 3	EV	2	2	$(4R - 2) \cdot M$	$R + 2M$
	CS	$2N$	$2N$	$(R + 2M) \cdot N$	$(4R - 2) \cdot MN$
Protocol 4	EV	2	2	$7M + C$	$5M + P$
	CS	$2N$	$2N$	$(5M + P) \cdot N$	$(7M + C) \cdot N$
Protocol 5	EV	1	1	$4M + P$	$6M + P$
	CS	$4N$	N	$(12M + P) \cdot N$	$(4M + P) \cdot N$

3.6 Conclusions

In this chapter, we proposed a revocable anonymous authentication framework that is robust against application-level DoS attacks. Our framework allows EVs to authenticate themselves anonymously to a charging station for reporting real-time information. To prevent malicious EVs from anonymously attacking the system, we equipped our framework with the ability to revoke the anonymity of an EV with verified maliciousness. To further protect charging stations and benign EVs from application-level DoS attacks by malicious EVs, we designed a permit-based mechanism to impose a high cost and penalties for launching an attack, making such attacks economically infeasible. We thoroughly analyzed the security and privacy properties of our framework, and evaluated its performance through analysis and implementations in real-world settings. The analysis and experiments showed that our framework is both efficient and scalable.

Chapter 4

ANONYMOUS REPUTATION SYSTEM WITH ANONYMOUS FEEDBACK UPDATE

4.1 Introduction

Reputation systems ensure the quality of information and the credibility of information sources, making them a vital component in various types of online systems. Community platforms such as Stack Overflow and Reddit, review platforms such as TripAdvisor and Yelp, and e-commerce sites such as Amazon and eBay all employ them to evaluate the quality of information posted by users. This evaluation is crucial in establishing and maintaining trust between information producers and information consumers. A reputation system tracks and publicizes the information along with the reputation of the information source. The information can be original, or it can be feedback from other sources regarding some information. The reputation system maintains a record of all user identities, their reputation, and the information originating from them. This tracking and maintenance allow a reputation system to attribute feedback or changes in feedback to a user's reputation perpetually.

Since user identities are tied to their reputation and the information originating from them, the reputation system or other users of the reputation system can violate the privacy of the users by tracking their information, feedback, or reputation. Such privacy violations can have legal, financial, and social implications and can also reduce user participation in these systems. To protect the privacy of information sources, reputation systems should

function effectively without relying on the true identities of the sources. As a solution, anonymous reputation systems have been proposed.

In P2P networks, anonymous reputation systems based on electronic cash (e-cash) have been proposed [2, 12]. These systems rely on anonymity in e-cash to ensure the unlinkability of users' transactions. However, these systems cannot support negative feedback.

Some anonymous reputation systems [21, 51, 6] rely on a central trusted authority to function. This central trusted authority can identify the users that might result in violation of user privacy when the authority can benefit from private user information or if it is compromised. The proposed blind signature based anonymous reputation and trust system [51] and group signature based anonymous reputation system [6] trust a central server or a group manager to be honest. Anonrep [53] is a tracking-resistant anonymous reputation system using verifiable shuffles, linkable ring signatures, and homomorphic cryptography. However, it follows the anytrust model in which it assumes at least one of the many servers of the system is trusted. Another drawback of Anonrep is that it operates in a series of message and feedback rounds that last for an arbitrary amount of time with a finite time interval. In many everyday activities, feedback is given perpetually, which makes Anonrep impractical. A blockchain-based trustless reputation system [47] has been proposed but it is vulnerable to ballot stuffing attacks.

Some anonymous reputation systems allow only one feedback submission per service access. Beaver [50] is a decentralized anonymous marketplace with a secure reputation system which provides strong client anonymity and weak vendor anonymity and requires the clients to purchase products from the vendor to be able to leave feedback. Other proposed anonymous reputation systems [35, 9] allow a user who accessed the service to rate a service only once. Such anonymous reputation systems are impractical because they deny users the option of changing their feedback.

There are many challenges that hinder the adoption and use of anonymous reputation

systems on present-day online platforms. First, most of the proposed systems are not sufficiently general, scalable, or practical for real-world deployment. Second, many rely on trust models that are not reflective of present-day requirements. Third, most of the systems do not support perpetual feedback updates. In this chapter, we designed a scalable and practical anonymous reputation system that provides anonymity during user registration, information posting, perpetual feedback updates, and feedback attribution from malicious users and a curious system operator.

Our main contributions are summarized as follows:

- We designed a secure, scalable, and practical anonymous reputation system, EARS, which implements all required functions while ensuring the anonymity and privacy of all users.
- Our system supports anonymous perpetual feedback updates.
- We performed detailed security analysis and implementation-based performance evaluation of our system.

The rest of this chapter is organized as follows. Section 4.2 presents our system and threat model. Section 4.3 presents the detailed design of our system. Section 4.4 presents the security analysis of our system. Section 4.5 presents our performance evaluation results. Section 4.6 concludes this chapter.

4.2 System and Threat Model

4.2.1 System Model

We assume that our system is implemented on a server controlled by the system operator. Many users can interact with the system, and because each user interacts with it

independently, we describe and evaluate its performance in terms of the interactions between a specific user and the system. However, our security analysis considers all users in the system. We classify users as ***posters***, who provide information, and ***voters***, who provide feedback. A user may play either or both roles. Since each of the roles has different responsibilities, we describe the relevant protocols for these two roles independently. We assume that multiple servers may be used for load-balancing or backup in our system. Since a user can only interact with a single server per session, the assumption of multiple servers for load-balancing will not have any impact on the security and privacy guarantees provided by our system. Since all the servers have the same functionality, our discussion applies to all servers in the system. Since the security of our system does not rely on having multiple servers, we assume only one server in our narrative for brevity. Our system and its users rely on Public Key Infrastructure (PKI). The system has a public-private key pair. All the communications happen over established secure channels to ensure the confidentiality and integrity of the communications.

4.2.2 Threat Model

We model the server deploying our system as honest-but-curious [42], and the users of our system as *malicious*. The server follows the defined protocol but will attempt to learn all possible information from messages received. The semi-honest-server assumption is realistic because practical factors may discourage protocol deviations that prevent the system operator from deviating from the protocols, like oversight by regulatory authorities and the reputational damage. Users can be malicious; they can deviate from the protocol, collude among themselves, or try to violate the anonymity of other users by tracking reputation scores, pseudonyms, and reputation updates. We assume all adversaries are computationally bounded and the cryptographic primitives like asymmetric key encryption, symmetric key encryption, and hash functions are all correctly implemented. We also assume that

the network and upper-layer protocols do not leak the users' identifiable information. To thwart traffic analysis attacks, we assume that the network connections between the users and the system are established over anonymous communication channels such as Tor [25].

4.3 System Overview

Table 4.1: Table of Notations

$h(m)$	Hash of the message m
$(m)_k$	Signature on message m using key k
ϕ^V	Voting token
τ^P	Posting token
π^P	Feedback collection token
γ^V	Feedback update token
$b_e(m)$	Blinding factor e applied to message m
$u_e(m)$	Unblinding factor e applied to message m
$BS(m)$	Blind Signature on m
$PBS(m)_x$	Partial Blind Signature on m with value x
(pu, pr)	Public/Private key pair of EARS

In this section, we provide an overview of the protocols of our anonymous reputation system. A user can be a poster (information provider) and/or a voter (feedback provider). Each user performs account initialization while joining the system, which associates the user identity, its initial reputation, and a credential token (blindly) signed by the system. Our system consists of the following protocols: poster registration, voter registration, posting, voting, reputation update, and feedback update.

During poster registration, a poster provides their credentials to the server and requests an anonymous posting token coupled with a post to enable feedback attribution. The sys-

tem verifies the credentials, checks that the reputation tokens for all the posts associated with this poster are redeemed recently and issues an anonymous posting token to the poster certifying their identity, reputation, and intention to post, along with an anonymous feedback collection token to collect the feedback. In the posting protocol, the poster provides the posting token along with the information they wish to post. After verifying that the posting token is valid, the system posts the information along with the reputation of the poster enclosed in the posting token. The poster token was blinded during registration to ensure the anonymity of the poster.

Like the poster registration protocol, in the voter registration protocol, a voter provides their credentials and requests an anonymous voting token to enable feedback updates. The system verifies the credentials and issues an anonymous voting token to the voter certifying their identity, reputation, and intention to vote on a specific post along with an anonymous feedback update token for updating this feedback in the future. Then, during the voting protocol, the voter provides the voting token along with the feedback they wish to post. The system verifies the voting token and posts the feedback given by the voter along with the reputation of the voter. The blinded voting token protects the anonymity of the voter during the voting protocol.

During the reputation update protocol, the poster submits the feedback collection token to the system. The system verifies the feedback collection token and then tallies all feedback attributed to the post up to that point and issues a partially blind signature on the reputation token and the timestamp. This reputation token is only redeemable by the poster who posted the information. After the feedback changes, the system issues a new anonymous reputation token reflecting the updated feedback score. The poster applies the unblinding factor to the partially blind signature on the reputation token and retrieves the signed reputation token. This token must be redeemed within a specific time (a week) and the poster submits this signed reputation token to the system along with their credentials

while redeeming. After verifying the credentials are valid and the time to redeem the token has not expired, the system updates the reputation score of the poster based on the reputation token and issues new credentials to the poster reflecting the updated reputation score.

In the feedback update protocol, the voter will submit the feedback update token issued during voter registration along with the updated feedback. After verifying the token, the system updates the feedback of a post corresponding to the token with the new feedback. We will explain the protocols of our system in detail.

4.3.1 Poster Registration Protocol

Protocol 10 Registration Protocol for Poster P

Input: ID, r, $h(ID_P||r)_{pr}$, and $h(ID_P||t)_{pr}$ of poster P.

Output: Poster P acquires a posting token and a feedback collection token from S

1: P sends its credentials along with blind tokens to S:

$P \rightarrow S$: $ID_P, r, h(ID_P||r)_{pr}, p, h(ID_P||t)_{pr}$

$b_k(a_k, r_k, h(a_k||r_k), B_k(h(ID_P)), h(a_k||B_k(h(ID_P))))$.

2: P sends the feedback collection token to S:

$P \rightarrow S$: $b_{n+1}(x), b_{n+2}(h(a_k||x))$.

3: After S verifies the blind token, it issues a partially blind signature on a posting token (k^{th} token) and a blind signature on the feedback collection token to P:

$S \rightarrow P$: $PBS(b_k(a_k, r_k, h(a_k||r_k), B_k(h(ID_P)),$

$h(a_k||B_k(h(ID_P))))_p, BS(b_{n+1}(x)),$

$BS(b_{n+2}(h(a_k||x)))$.

Protocol 10 shows the registration protocol, which is used to verify a poster's credentials and credibility and issue an anonymous posting token and an anonymous feedback col-

lection token to the poster P by S. In Step 1, a poster P sends S its identity (ID_P), a reputation score (r), the credential ($h(ID_P||r)_{pr}$) and a credibility token ($h(ID_P||t)_{pr}$) where t is the number of reputation tokens redeemed during previous week. This credibility token is issued by the system to the poster after verifying that the number of reputation tokens redeemed by the user is the same as the number of posting tokens issued to them and all the reputation tokens are redeemed recently. This can be easily verified using the timestamp on the reputation tokens. The poster also sends a blinded posting token, $b_k(a_k, r_k, h(a_k||r_k), B_k(h(ID_P)), h(a_k||B_k(h(ID_P))))$ and a value p. The term (b_k, B_k) corresponds to the blinding values; a_k is a random nonce; r_k corresponds to a reputation score less than the actual reputation score r; and p is a tag informing the system that this is a request for a posting token by the user (to perform posting). Then, S verifies the validity of the arguments inside the blind posting token using cut-and-choose protocols. The verification using the cut-and-choose protocols involves P sending n blind tokens to S. Each of the n blind tokens will have unique nonces, unique reputation scores less than the actual reputation score, and unique blinding values.

In Step 2, after S ensures that arguments in the blind token from P are valid using the cut-and-choose protocol, P also sends feedback collection token, $b_{n+1}(x)$ and $b_{n+2}(h(a_k||x))$. In Step 3, S issues a partially blind signature on the posting token,

$PBS(b_k(a_k, r_k, h(a_k||r_k), B_k(h(ID_P)), h(a_k||B_k(h(ID_P))))_p$, and a blind signature on the feedback collection token, $BS(b_{n+1}(x)), BS(b_{n+2}(h(a_k||x)))$. The poster P applies the unblinding factors to retrieve the signature of S on the posting token which is denoted as $\tau_i^U = PBS(a_k, r_k, h(a_k||r_k), B_k(h(ID_P)), h(a_k||B_k(h(ID_P))))_p$, and the feedback collection token which is denoted as $\pi_i^U = BS(x), BS(h(a_k||x))$.

The poster registration protocol can be executed offline since it does not require any details regarding the information to be posted. It can also be executed multiple times in advance to collect multiple tokens. Since the tokens are anonymously obtained and unlink-

able, accumulation of tokens will not violate the anonymity of the poster. While posting the information, the poster can select one of the unused tokens and submit the information to be posted along with it.

4.3.2 Posting Protocol

Protocol 11 Posting Protocol for Poster P

Input: Posting token issued by S to the poster P.

Output: Poster P posts a message anonymously.

1: P sends its credentials a_k, r_k, the message $post_k$ and $B_k(h(ID_P))$ along with the posting token to S:

$P \rightarrow S$: $a_k, r_k, post_k, B_k(h(ID_P))$,

$PBS(a_k, r_k, h(a_k||r_k)$,

$B_k(h(ID_P)), h(a_k||B_k(h(ID_P))))_p$.

2: S posts message $post_k$ with id_k and reputation score r_k.

Protocol 11 presents the procedure for a poster to anonymously post information using an obtained posting token. As described in Table 4.1, $PBS(q)_y$ denotes the partially blind signature on q, with y as the mutually agreed upon value by both the requester (P) and the issuer (S) of the partially blind signature. In Step 1, P presents the posting token $PBS(a_k, r_k, h(a_k||r_k), B_k(h(ID_P)), h(a_k||B_k(h(ID_P))))_p$ to S along with values a_k,r_k,$B_k(h(ID_P))$ and $post_k$ where a_k, r_k and $B_k(h(ID_P))$ are the random nonce, reputation score and blinded-hash of the identity (which serves as the reputation token) respectively, and $post_k$ is the information to be posted. Then, S compares its signature on the entities a_k, r_k and $B_k(h(ID_P))$ with the submitted token, and if verified, posts the information $post_k$ with a unique identity id_k and the reputation score r_k. This identity id_k denotes the identity of the post and not the identity of the poster who posted this post.

4.3.3 Voter Registration Protocol

Protocol 12 Registration Protocol for Voter V

Input: ID_V, r_V, and $h(ID_V \| r_V)_{pr}$ of the voter V.

Output: V obtains voting token and feedback update token.

1: V sends its credentials along with blind tokens to S:

$V \rightarrow S$: $ID_V, r_V, h(ID_V \| r_V)_{pr}, v, id_l$

$b_l(a_l, r_{Vl}, h(a_l \| r_{Vl}))$.

2: V sends the feedback update token to the S:

$V \rightarrow S$: $b_{m+1}(z), b_{m+2}(h(a_l \| z))$.

3: After S verifies the blind token, S issues a partially blind signature on voting token (l^{th} token) and a blind signature on the feedback update token to V:

$S \rightarrow V$: $PBS(b_l(a_l, r_{Vl}, h(a_l \| r_{Vl})))_{(v, id_l)}$,

$BS(b_{m+1}(z)), BS(b_{m+2}(h(a_l \| z)))$.

Protocol 12 details the procedure for a voter V to obtain from S an anonymous voting token to provide feedback for a post of identity id_l, and an anonymous feedback update token for potential feedback update in the future.

In Step 1, V presents to S its identity ID_V, and reputation r_V along with the credential $h(ID_V \| r_V)_{pr}$ accompanied by a blinded voting token $b_l(a_l, r_{Vl}, h(a_l \| r_{Vl}))$ and a bit v (tag signifying intent to vote) and the identity of the post it wants to vote on, id_l. Here, b_l corresponds to the blinding value, a_l is a random nonce and r_{Vl} is a reputation score less than V's actual reputation score r_V. The system S uses the cut-and-choose protocol to ensure the validity of the arguments. The verification using the cut-and-choose protocols involves V sending m blind tokens to S. Each of the m blind tokens will have unique nonces, unique reputation scores less than the actual reputation score, and unique blinding values. In Step 2, after S ensures that arguments in the blind to-

ken from V are valid using the cut-and-choose protocol, V sends the feedback update token, $b_{m+1}(z)$, $b_{m+2}(h(a_l||z))$. In Step 3, S issues a partially blind signature on the voting token, $PBS(b_l(a_l, r_{Vl}, h(a_l||r_{Vl})))_{(v,id_l)}$, and a blind signature on the feedback update token, $BS(b_{m+1}(z))$, $BS(b_{m+2}(h(a_l||z)))$. The voter V applies the unblinding factors to retrieve the signature of S on the voting token, which is considered as ϕ_i^V, $\phi_i^V = PBS(a_l, r_{Vl}, h(a_l||r_{Vl}))_{(v,id_l)}$, and then applies unblinding factors to the feedback update token, which is considered as γ_i^V, $\gamma_i^V = BS(z), BS(h(a_l||z))$.

4.3.4 Voting Protocol

Protocol 13 Voting Protocol for Voter V

Input: Voting token issued by S to the voter V.

Output: Voter V votes for a posting anonymously.

1: V sends its credentials a_l and r_{Vl} along with the signed blind tokens to S:
$V \rightarrow S$: $a_l, r_{Vl}, vote_l, PBS(a_l, r_{Vl}, h(a_l||r_{Vl}))_{(v,id_l)}$.

2: S validates signature, and posts $vote_l$ for post id_l and the reputation score r_{Vl} of the voter.

Protocol 13 explains the procedure followed by the voter V to provide anonymous feedback for a post with identity id_l using the voting token acquired from S. In Step 1, V presents the voting token $PBS(a_l, r_{Vl}, h(a_l||r_{Vl}))_{(v,id_l)}$ to S along with values a_l, r_{Vl} and vote, $vote_l$ where a_l is the random nonce, r_{Vl} is the reputation score, and $vote_l$ is the feedback of the voter V for id_l. Then, S compares its signature on the entities a_l and r_{Vl} with the submitted token, and after verifying that the signature is valid, it assigns $vote_l$ to the post associated with the id_l along with the reputation score of V, r_{Vl}.

4.3.5 Reputation Update Protocol

Protocol 14 Reputation Update Protocol for Poster P

Input: Feedback collection token issued by S to poster P.

Output: Updated reputation score of the poster P with the feedback f corresponding to the post id_k.

1: P sends the feedback collection token along with the credentials in the token to S:
$P \rightarrow S$: $a_k, x, BS(x), BS(h(a_k \| x))$.

2: S verifies arguments and its signature on the token and issues a partial blind signature on the reputation token:
$S \rightarrow P$: $PBS(B_k(h(ID_P)))_{(f,w)}$.

3: P sends the new reputation token to S and requests a new feedback collection token:
$P \rightarrow S$: $b_c(c), b_{c+1}(h(a_k \| c)), B_c(h(ID_P))$.

4: S replaces the reputation token with the new reputation token and blind signs the new feedback collection token:
$S \rightarrow P$: $BS(b_c(c)), BS(b_{c+1}(h(a_k \| c)))$.

5: P submits the signature of S on reputation token along with its credentials:
$P \rightarrow S$: $ID_P, r, h(ID_P \| r)_{pr}, PBS(h(ID_P))_{(f,w)}$.

6: S verifies received credentials and signature and issues updated reputation score and credentials to P:
$S \rightarrow P$: $ID_P, r + f, h(ID_P \| r + f)_{pr}$.

In the reputation update protocol, a poster interacts with S to update its reputation based on feedback it has received from voters on its posts. The poster submits the feedback collection token of the post to S. After verifying the token, S tallies all the feedback received by the post and issues a partially blind signature on the reputation token, which is an anonymous token containing the current feedback score of the poster and the timestamp, and sends it to the poster. This token is redeemable only by the poster who posted the information. Only one token for a post is issued per update period (*e.g.*, a week, decided by the system operator). Subsequent changes to the feedback result in issuance of a new anonymous reputation token reflecting the changes to the feedback score. This protocol is executed by the poster for each of its posts every update period.

Protocol 14 explains the procedure followed by S to update the reputation of the poster P based on the feedback (votes) its post id_k has received. Here, id_k is the identity of the post and S does not know the identity of P or the voters that voted on the post; thus their anonymity is preserved throughout the process. In Step 1, P presents to S the feedback collection token $\pi_i^U = BS(x), BS(h(a_k||x))$ (obtained during poster's registration for the post) along with the credentials in the token a_k, x. The system S signs x and matches it with $BS(x)$, checks the posts associated with the value a_k, computes $h(a_k||x)$ and compares its signature on $h(a_k||x)$ with the submitted token ($BS(h(a_k||x))$). If all the credentials and the tokens submitted by P are valid, then P is the poster of this information. In Step 2, S performs a partially blind signature on the reputation token ($B_k(h(ID_P))$) of the post associated with a_k with the feedback f (the total feedback received for this post) and the time stamp in terms of week (the week out of 52 weeks), w and sends this partially blind signature, $PBS(B_k(h(ID_P)))_{(f,w)}$ to P. Here, the time stamp w is added to ensure that the reputation update protocol will be executed for each post regularly. On receiving the partial blind signature, P applies the unblinding factor and retrieves the signature on the reputation token ($PBS(h(ID_P))_{(f,w)}$).

In Step 3, P sends the new reputation token to S and requests a new feedback collection token for the next (future) reputation update. Then, S updates the post associated with a_k with the new reputation token and issues a new feedback collection token to P in Step 4. In Step 5, to update their reputation, P sends its credentials along with the signed reputation token to S. In Step 6, S verifies the credentials and its signature on the reputation token and updates the reputation score r of P associated with the identity ID_P with feedback f and issues updated reputation score and corresponding credentials to P.

4.3.6 Feedback Update Protocol

Protocol 15 Feedback Update Protocol for Voter V

Input: Feedback update token issued by S to the voter V.

Output: Updated feedback of the voter V on the post id_l.

1: V presents the blindly signed feedback update token, the credentials in the token, and the updated feedback to S:
$V \rightarrow S$: $a_l, z, BS(z), BS(h(a_l \| z)), vote_{updated}$.

2: S verifies the argument and the signature on the token, and updates the vote associated with a_l on id_l by V.

3: V requests a new feedback update token from S:
$V \rightarrow S$: $b_d(d), b_{d+1}(h(a_l \| d))$.

4: S issues a new feedback update token to V:
$S \rightarrow V$: $BS(b_d(d)), BS(b_{d+1}(h(a_l \| d)))$.

Feedback update is one of the core functionalities that we need to support in an anonymous reputation system. In our system, V can update its past feedback on a post using the feedback update token that it received during the voter registration protocol. Protocol 15 details the procedure followed by V to update their feedback on a post identified

with the identity id_l. Voter V sends the feedback update token they obtained during voter registration, along with the credentials in the token and the updated feedback ($vote_{updated}$): $a_l, z, BS(z), BS(h(a_l||z)), vote_{updated}$ to S in Step 1. S verifies that its signature on z matches with the feedback update token submitted ($BS(z)$), checks the vote associated with the value a_l, computes $h(a_l||z)$, and compares its signature on the $h(a_l||z)$ with the submitted token ($BS(h(a_l||z))$). If all the credentials and the tokens submitted by V are valid, then V is the voter who provided this feedback. S then updates the old feedback $vote$ with the new feedback $vote_{updated}$. In Step 3, V requests a new feedback update token from S for a future feedback update, which S issues in Step 4.

4.4 Security Analysis

In this section, we perform detailed security analysis of EARS.

Proposition 1: *Poster anonymity is guaranteed in EARS.*

During poster registration, the poster P sends their credentials ID_P, r, and $h(ID_P||r)_{pr}$ along with the credibility token, $h(ID_P||t)_{pr}$ to the anonymous reputation system S. Then, S verifies the submitted credentials with its database of users, checks the validity of the credibility token and if they match, S knows that P is authentic. Poster P sends n blinded tokens; each token is composed of the following components that are unique to each token: a random nonce; a reputation value less than P's actual reputation; a commitment with the nonce and the reputation value; a reputation token; and the commitment of the reputation token with the nonce. S uses cut-and-choose protocol to verify these messages, and then issues a partially blind signature on one of the tokens with p.

The security of partially blind signature ensures that S cannot know the nonce, reputation value, feedback collection token, and their commitments, which are signed by it for the kth token despite unblinding the other $(n - 1)$ tokens. The blind signature on the feedback collection tokens also ensures that these values are concealed from S when it issues

its signature on them. So, S issues an anonymous blind token which contains valid arguments (reputation score, feedback token, and feedback collection token) to an authenticated poster. In the posting protocol, P sends the anonymous blind token (with the signature of S) along with the arguments in the token: a_k, r_k, $B_k(h(ID_P))$ to S.

Since the signature itself can only be verified and cannot be compared to anything in the past (due to the application of unblinding factors to the partially blind signature), S cannot know the identity of the poster to which this signature was issued. The arguments do not leak any information regarding the identity of the poster because the nonce is random, the reputation score is less than the original reputation score (which is not known by the system), and the blinded hash is a result of a cryptographic irreversible one-way function.

During the reputation update protocols, P presents the signature of S on the feedback collection token along with the arguments in it: a_k, x. Then, S verifies that the signature is valid on the submitted arguments. Since this signature is obtained by unblinding the blind signature on the feedback collection token, S cannot connect this signature or the arguments with the identity of the poster to which this signature was issued. The reputation token ($B_k(h(ID_P))$) itself was blinded with blinding factor B_k, preventing S from knowing the identity of the poster to which this token was going to be issued. The signature of S on this token is a partially blind signature with feedback score f and timestamp w. When P redeems this reputation token, S can keep track of all the reputation tokens of value f it issued and try to correlate that with the posters that redeemed those tokens. Since many posters might have been issued the reputation tokens of value f, deanonymizing the identity of a specific poster using only the value of reputation tokens is difficult (especially for large anonymity sets).

Proposition 2: *Voter anonymity is guaranteed in EARS.*

During registration, a voter V sends their credentials ID_V, r_V, and $h(ID_V \| r_V)_{pr}$ to the system S. Then, S verifies the submitted credentials as with the poster. Then V sends

m blinded tokens; each token is composed of the following components that are unique to each token: a random nonce, a reputation value less than the actual reputation value of the voter, and a commitment with the nonce and the reputation value. The system uses cut-and-choose protocol to verify that all these arguments are valid, and then S issues a partially blind signature on one of the tokens with (id_l, v) (indicating that the token is for voting on a post, id_l). Since it is a partially blind signature, S cannot know the nonce, reputation value, and their commitments it had signed.

The system cannot infer these values from the values obtained from $(m - 1)$ unblinded tokens because they are unique to each token. The blind signature on the feedback update token also ensures that these values are concealed from S when it issues its signature on them. So, S issues an anonymous blind token which contains valid arguments (reputation score, and feedback update token) to an authenticated voter V. During the voting protocol, V sends the anonymous blind token (signature of S on its token) and also the arguments in the token: a_l, r_{Vl} to S. Since the signature itself can only be verified and not compared to anything stored in the past (due to the application of unblinding factors to the partial blind signature), S cannot know the identity of V.

The arguments do not leak any information regarding the identity of the voter because the nonce is random, and the reputation score is less than the original reputation score (which is not known by the system). The vote or the identity of the post that is being voted on using this token cannot identify the voter. So, S receives only a signed token, which contains arguments necessary to vote on a specific post anonymously – the voter is not identifiable. In the feedback update protocol, V presents the signature of S on the feedback update token along with the arguments in it: a_l, z. Then, S verifies that the signature is valid on the submitted arguments. Due to the blind signature on the feedback update token, S cannot connect this signature or the arguments of the token with V's identity.

Proposition 3: *EARS cannot link the various tokens of the poster or the voter.*

Since the tokens issued by the system S to a user (a poster or a voter) are always anonymous, S cannot differentiate between the tokens issued to the same user or a different user. As such, the system has no information that would allow linking the various tokens of a user.

Proposition 4: *A poster cannot use higher reputation score of another poster for posting.*

The commitment of the random nonce (used to post the information) with the reputation score during the poster registration protocol prevents a poster from using a higher reputation score of another poster to post. Only the poster who obtained a token with their identity and a reputation score less than their reputation score will be able to use the token issued during poster registration protocol to post some information. Reputation scores cannot be created unless the poster interacts with S, further the several interactions between P and S end up functioning as challenge-response ensuring that a simple reputation of somebody else cannot be replayed or stolen and used. Even if another poster obtains this token and posts some information, the hash of the identity of the poster (who obtained the token) in the reputation token will ensure that reputation update based on the feedback for the post will only occur for the poster whose token has been used to post.

Proposition 5: *A poster can only redeem their feedback token once.*

The hash of the identity of the poster (who posted the information) in the feedback token will ensure that no other poster can redeem this feedback token. The unique signature of the system on this feedback token will prevent double spending of this token – once a feedback token is redeemed, the system S keeps track of it so that it cannot be redeemed again.

Proposition 6: *EARS is immune to bad mouthing and ballot stuffing.*

Since EARS enables weighted feedback, the reputation will play a vital role in determining the effect of the feedback, which in turn decides the reputation of the information source. As such, unless the reputation of a malicious voter is significant enough or a large group of voters collude, the effect of authentic feedback will outweigh the fake feedback.

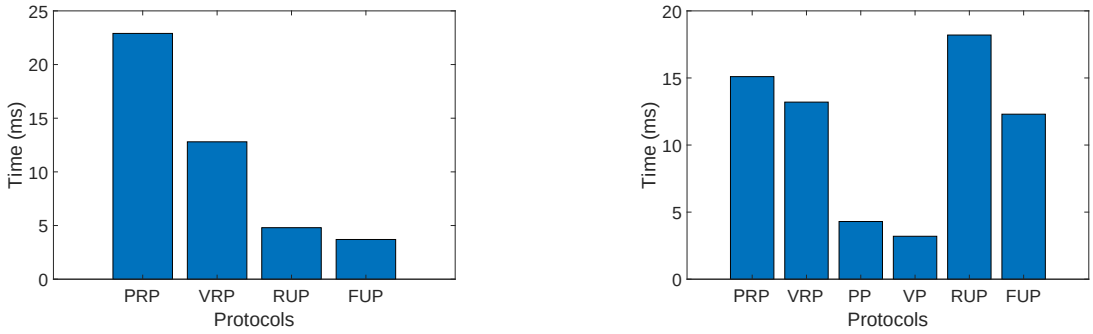
Thus, attempts on bad mouthing and/or ballot stuffing are not viable.

Non-goals: Our system cannot detect the malicious users but is resilient against such users. Like many previous works on anonymous reputation systems, our system is not immune to Sybil attacks, but it mitigates their effect. Our system cannot defend against network-level DoS and DDoS attacks.

4.5 Performance Evaluation

In this section, we discuss the implementation of EARS and analyze the results to demonstrate that it is efficient, robust, and scalable, and that it can be implemented without any additional hardware or software requirements. We implemented EARS using standard crypto libraries in Java. To emulate real-world hardware, the user-side protocols were run on an Intel Core i5-2450M, 2.5 GHz machine with 8 GB RAM. To emulate EARS, we implemented the server on an Intel Core i7-6700K, 4.0 GHz machine with 32 GB RAM. All the results were averaged over 1000 runs. As we mentioned, the interaction of a user (poster/voter) with EARS is independent of the interactions of other users with EARS – the system can interact with multiple users in parallel. This would ensure that EARS is scalable, since increase in the number of users would only increase the computation requirements of EARS linearly.

For better scalability, in EARS more servers can be added to serve an increasing number of clients with low latency. We measure the time taken by the user and the server for each protocol in our system. We consider only the computation steps and not the data transmission steps as they are dependent on the nature and traffic in the network. We also do not consider the storage requirements because they are trivial (in the order of several kilobytes for each client). Fig. 4.1a shows the time taken at the user during the poster registration protocol (PRP), the voter registration protocol (VRP), the reputation update protocol (RUP), and the feedback update protocol (FUP). Fig. 4.1b shows the time taken at



(a) Time taken by a user in various protocols.

(b) Time taken by the server in various protocols.

Figure 4.1: Time Taken by Various Entities of Our System

the server during PRP, VRP, the posting protocol (PP), the voting protocol (VP), RUP, and FUP.

As shown in Fig. 4.1a, the user part of the poster registration protocol takes only 22.9 ms (n = 10). It involves creating the blind tokens which in turn involve hashing, and applying blinding factors. After receiving the posting token, the poster applies the corresponding unblinding factor to retrieve the signature of the system on the posting token and the feedback collection token. The user part of the voter registration protocol takes only 12.8 ms (m = 10). It involves creating the blind tokens which involve hashing and applying blinding factors. Since the voter registration protocol does not involve reputation token, the number of blinding and hashing operations is less compared to the poster registration protocol. This is the reason that the time required for the voter registration protocol is less than that of the poster registration protocol.

After receiving the feedback update token, the voter applies the corresponding unblinding factor to retrieve the signature of the system on the feedback update token. The user part of the posting protocol and the voting protocol are not mentioned because there are no computation steps for the users in these protocols. They submit the tokens (which are signed by the system) and the arguments in the tokens along with the posts or votes to the system. The users do not need to perform any computations in these protocols. So, they

are omitted from Fig. 4.1a.

The reputation update protocol takes 4.8 ms; this is because it only requires a few blinding and unblinding operations. The feedback update protocol requires the voter to submit the feedback update token along with the arguments in it and the updated feedback (vote) to the system. The voter also applies blinding factor to the new feedback update token request. After receiving the blind signature on the new feedback update token, it applies unblinding factor to retrieve the new signed token. The user takes 3.7 ms to complete feedback update.

As shown in Fig. 4.1b, the server part of the poster registration protocol takes 15.1 ms. It encompasses the unblinding and verification of tokens in cut-and-choose protocols which involves hashing. After verifying that the tokens are valid, the system performs a partially blind signature on the posting token and a blind signature on the feedback collection token and send them to the poster. The server part of the voter registration protocol takes 13.2 ms. It constitutes the unblinding and verification of tokens in cut-and-choose protocols which encompasses hashing. After verifying that the tokens are valid, the system issues a partially blind signature on the voting token and a blind signature on the feedback update token. The amount of time taken by the system for the poster registration protocol is more than the voter registration protocol because of the presence of the reputation token, which increases the number of hashing and verification operations.

In the posting protocol, the system verifies its own signature on the posting token and verifies the arguments in the posting token with the arguments submitted to it by the poster, which involves hashing operations. The server takes 4.3 ms to execute the posting protocol and 3.2 ms for the voting protocol. The voting protocol involves the system verifying its own signature on the voting token and verifying the arguments inside that token with the arguments submitted to it by the voter, which involves a hashing operation. In the reputation update protocol, the server takes 18.2 ms. It requires several complex operations, such as verifying signature on the feedback collection token and integrity of the arguments inside

Table 4.2: Execution Time for a Single Server with Varying Clients

# Clients	VRP	PP	VP	FUP
10	88.7 ms	23.9 ms	23.6 ms	92.8 ms
100	743.5 ms	215.8 ms	212.3 ms	848.1 ms
1000	6811.9 ms	2059.6 ms	2047.7 ms	8191.9 ms

the token, and issuance of a partially blind signature on the reputation token.

After receiving a request for a new feedback collection token, the system issues a blind signature on the new feedback collection token. When the poster submits a reputation token for redemption, the system verifies its signature on the reputation token and the integrity of the arguments inside the token. After verifying, the system updates the reputation score of the poster and creates new credentials representing the updated reputation score of the poster and sends these credentials to the poster. This step involves signature of the system on the hash of the identity of the poster and new reputation score. In the feedback update protocol, the server takes 12.3 ms. The system part of the feedback update protocol involves the system verifying its signature on the feedback update token submitted by the voter and the integrity of its arguments. It also involves issuing blind signature on the new feedback update token submitted by the voter. As mentioned previously, the poster registration protocol and the reputation update protocol can be executed offline. The voter registration protocol, the posting protocol, the voting protocol and the feedback update protocol are the only protocols that need to be executed in real time.

As shown above, the voting protocol and the posting protocol take very little time to execute. The maximum amount of time taken among them is 4.3 ms. A processor core of a server running these protocols can service 200 users simultaneously in less than a second in real time. Use of cloud computing can easily enable our system to scale economically and efficiently. The other protocols which can be executed offline can be run during the off-

peak hours of the system. Table 4.2 shows the time taken by the server to simultaneously serve varying number of users during the various online protocols of the system. The poster registration and the reputation update protocols are not shown in Table 4.2, because they can be executed offline. The experimental results from the implementation of EARS demonstrate its efficiency and scalability.

4.6 Conclusions

In this chapter, we proposed a secure, practical, and scalable anonymous reputation system named EARS. EARS allows users to post information anonymously. It also allows users who want to vote (voters) to obtain a voting token for a specific post anonymously and to vote anonymously. In both cases, the reputation of the user is associated with the post/vote it submits to the system. After receiving the feedback on a post, our system allows the reputation of the posting user to be updated with the feedback received for that post. This reputation update is also anonymous. In case a voter decides to update their feedback on a specific post, our system enables them to do so anonymously as well. Our system operates under the assumption that no server in the reputation system is trusted. It also does not trust the users. Detailed security analysis of our system shows that it is immune to attacks from malicious users or any attempt to violate the anonymity of the user by the system. We note that our system is not fully immune to Sybil attacks, which is an orthogonal problem to what we solved. However, since our system enables weighted feedback, such a feature can be leveraged to mitigate the effect of Sybil accounts.

Chapter 5

REVOCABLE ANONYMOUS AUTHENTICATION IN A POST-QUANTUM WORLD

5.1 Introduction

The rapid expansion of digital services has made authentication a central security primitive in modern communication systems. From online banking and e-government services to vehicular networks, Internet of Things (IoT) platforms, and cloud-based environments, users and devices increasingly rely on authentication protocols. Anonymous authentication systems are used to preserve user privacy during authentication. To prevent the misuse of anonymity by malicious or misbehaving users, revocable anonymous authentication systems have been proposed. With the advent of quantum computing, the need for stronger security has become even more urgent. Public-key cryptographic assumptions that underpin many classical authentication schemes, such as integer factorization and discrete logarithms, are vulnerable to Shor's algorithm. This threat affects digital signatures, key exchange, and certificate-based protocols.

The classical foundation was established primarily through anonymous credential systems and group-signature-based accountability. In the issuer–holder–verifier model, credentials are bound to holder secrets and later shown through selective disclosure and zero-knowledge proofs, allowing users to satisfy verifier policies without disclosing their full identity [13] [15]. Revocation in this setting has been realized through several mechanisms, including certificate revocation lists, online status checking, and designated opening

authorities, each reflecting a different balance among scalability, privacy, and accountability [44] [8] [3]. Dynamic accumulators became especially influential because they enable compact non-revocation proofs without requiring verifiers to process large revocation lists directly [14].

Although large quantum computers are not yet available, multiple reasons provide impetus to design and implement post-quantum cryptographic primitives as soon as possible. The first and most important reason is that cryptographic migration takes many years. The second reason is due to the rise of Store Now, Decrypt Later (SNDL) or Harvest Now, Decrypt Later (HNDL) attacks. In SNDL or HNDL attacks, an attacker steals the encrypted data today, storing it until they possess a quantum computer capable of breaking current encryption. This type of attack renders data with long-term value vulnerable now.

As a result, cryptographic constructions that remain secure against quantum computers are needed. Classical anonymous authentication schemes often rely on pairing-based cryptography or discrete-log based primitives that are not post-quantum safe. In the context of anonymous authentication and revocable anonymous authentication, this would require replacing underlying primitives so that anonymity and revocability continue to hold under quantum attack models. Replacing these with post-quantum primitives is the first challenge, because techniques and algebraic structures used to establish anonymity may no longer apply directly. Researchers have developed lattice-based zero-knowledge arguments, lattice-based accumulators for anonymous credential revocation, and lattice-based group signature constructions that move important privacy-preserving mechanisms into the post-quantum domain [41] [29] [20]. The field still lacks a broadly accepted standalone framework that simultaneously offers strong anonymity, efficient revocation, and deployment-ready semantics under post-quantum assumptions [37]. The difficulty lies not only in replacing individual primitives, but also in recovering the compactness and composability that classical schemes derived from algebraically structured constructions.

Researchers have explored post-quantum or privacy-enhanced authentication in specific application domains. Recent proposals address blockchain-based IoT, intelligent transportation, edge computing, autonomous truck platooning, e-health, and the Internet of Drones [30] [52] [49] [17] [38] [54]. These systems demonstrate sustained interest in traceability, cross-domain trust, remote identification, group key establishment, and blockchain-assisted coordination. However, most are tailored to particular deployment settings rather than to a general revocable anonymous credential model. Complementary work on deniable post-quantum authenticated key exchange and stronger security notions for KEM-based protocols has also provided protocol-level insights that are valuable for secure deployment [22] [23]. Taken together, the literature shows clear progress across classical foundations, post-quantum building blocks, and domain-specific design. However, a single mature framework that unifies strong anonymity, efficient revocation, distributed accountability, and practical post-quantum deployment remains missing in the literature. The first challenge is replacing classical primitives without losing anonymity and revocability.

The second challenge is supporting revocation efficiently while avoiding large verification overhead. The third challenge is maintaining practical performance. Post-quantum primitives commonly increase bandwidth, key sizes, and computational cost. Revocable anonymous authentication schemes tend to be more complex than standard signature schemes. If the resulting protocols are secure but too heavy for constrained environments like IoT, vehicular systems, and mobile applications, then they might not be adopted.

The fourth challenge is balancing privacy and accountability. Systems often require some trusted authority to revoke anonymity and trace abusive behavior. However, such capabilities must be narrowly controlled to avoid undermining anonymity and trust. The literature [19] [7] [4] [16] [26] has also emphasized the risks created by concentrated trust. Concentrating revocation or opening power in a single authority creates a serious trust bottleneck. Compromise or malicious behavior of that authority can undermine the

privacy guarantees that these systems are intended to provide [26] [13] [3]. A solution is to distribute revocation or opening capability across multiple independent authorities and require threshold cooperation before revocation is executed [10] [26]. Such a model makes unilateral abuse more difficult and improves robustness against compromise of a single trusted authority since an adversary must corrupt several authorities rather than only one.

In this chapter, we designed a post-quantum-secure revocable anonymous authentication framework in which revocation authority is distributed across multiple independent parties rather than concentrated in a single entity. The framework is built from post-quantum cryptographic components to provide resistance against quantum-capable adversaries. Moreover, as demonstrated by our experimental results, the design is computationally efficient enough for deployment in real-world use cases.

Our main contributions are summarized as follows:

- We proposed a secure, scalable, and efficient revocable anonymous authenticated key exchange in the post-quantum setting.
- We did not rely on a central trusted authority but distributed trust to prevent a single point of compromise.
- We analyzed the security of our framework and implemented it to demonstrate its scalability and efficiency.

The rest of this chapter is organized as follows. Section 5.2 presents our system and threat model. Section 5.3 presents the cryptographic concepts needed to understand our framework and its protocols. Section 5.4 presents the detailed design of our framework. Section 5.5 presents the security analysis of our framework. Section 5.6 presents our performance evaluation results. Section 5.7 concludes this chapter.

5.2 System and Threat Model

5.2.1 System Model

The entities in our framework are a Certificate Authority (CA), a Tracing Authority (TRA), users and the server. We assume that multiple servers may be used for load-balancing or for backup. Since a user can only interact with a single server per session, the assumption of multiple servers to provide load-balancing will not have any impact on the security and privacy guarantees. Since the security of our system does not rely on having multiple servers, we assume only one server in our narrative for brevity. In our system model, many users communicate with many servers. Since the communications of different users with various servers are independent of each other, we only consider communication between a user and a server while explaining and evaluating our framework. We assume that the CA, the TRA, and the server are always online and users can communicate with the system.

5.2.2 Threat Model

We assume that the CA and the TRA neither collude with each other nor with the server, and that they are not both compromised simultaneously. Both insider and external attackers are modeled as quantum-capable adversaries. A server can be malicious and benefit from identifying a user. Denial of Service attacks are possible, i.e., a server may refuse to accept the messages or start the key exchange with a user. An attacker can compromise some of the users and/or some of the servers and turn them into malicious entities exposing their secrets. The possible attacks are privacy violation, man-in-the-middle attacks, forgery attacks, and replay attacks.

5.3 Cryptographic Concepts

This section explains the cryptographic concepts needed to understand our framework and its protocols.

5.3.1 Key Encapsulation Mechanism (KEM)

The Key Encapsulation Mechanism (KEM) schemes let two parties establish a shared secret over a public channel providing Diffie-Hellman-like secrecy guarantees. In KEM schemes, a user A trying to establish a shared secret with user B generates a pair of KEM keys (public and private) and performs a key encapsulation on public key of user B resulting in a key K and a ciphertext ct. User A retains the key and sends the ciphertext ct to user B over the public channel. User B decapsulates the received ciphertext ct using his secret key resulting in key K. For a public key pk and secret key sk, encapsulation and decapsulation are:

$$(K, ct) \leftarrow \mathsf{KEM.Encaps}(pk), \qquad K \leftarrow \mathsf{KEM.Decaps}(sk, ct).$$

5.3.2 Split Key Encapsulation Mechanism (split-KEM)

A split-KEM provides implicit authentication by requiring both parties' secret keys in the key exchange. Unlike a standard KEM where only the decapsulator uses a secret key, a split-KEM uses the encapsulator's secret key during encapsulation and the decapsulator verifies using the encapsulator's public key during decapsulation. For server public key $pk_S^{\mathsf{Split\text{-}KEM}}$, server secret key $sk_S^{\mathsf{Split\text{-}KEM}}$, and user ephemeral key pair $(pk_U^{\mathsf{ephem}}, sk_U^{\mathsf{ephem}})$:

$$(K, ct) \leftarrow \mathsf{Split\text{-}KEM.Encaps}(pk_S^{\mathsf{Split\text{-}KEM}}, sk_U^{\mathsf{ephem}}),$$

$$K \leftarrow \mathsf{Split\text{-}KEM.Decaps}(sk_S^{\mathsf{Split\text{-}KEM}}, pk_U^{\mathsf{ephem}}, ct).$$

We instantiate this using FrodoKEX+, an LWE-based split-KEM derived from FrodoKEM.

5.3.3 Revocation Accumulator

The CA maintains a set of revocation handles for users. These are arranged in a Merkle tree. For a list of leaves $(h_1, \ldots, h_n)$, the CA computes:

$$\mathsf{acc\_root} \leftarrow \mathsf{MerkleRoot}(h_1, \ldots, h_n).$$

For each user, the CA can provide a Merkle authentication path path_U that can be used to verify membership:

$$\mathsf{MerkleVerify}(h_U, \mathsf{path}_U, \mathsf{acc\_root}) = 1.$$

5.3.4 RISC Zero zkVM

We treat RISC Zero zkVM as providing a non-interactive zero-knowledge proof system for arbitrary computations. A program (the "guest") checks that a relation R over public inputs x and a private witness w holds. The prover produces a proof π and public output y. The verifier checks that:

$$\mathsf{Verify}(x, y, \pi) = 1,$$

without learning anything about w beyond what is implied by (x, y).

RISC Zero supports *proof composition*, whereby one guest program can record an *assumption* that another guest's proof exists with a specific journal. Concretely, a guest calls $\mathsf{env.verify}(\mathsf{image\_id}, \mathsf{journal})$ to assert that a proof for guest program $\mathsf{image\_id}$ with output $\mathsf{journal}$ has been (or will be) produced. The host supplies the inner receipt via $\mathsf{add\_assumption}$ when proving the outer guest; the composed receipt is valid only if both proofs are valid and the assumption matches.

We exploit this mechanism to decompose the zero-knowledge proof into a *static proof* (expensive, cached) and a *session proof* (lightweight, per-session). The static proof verifies the user's long-term ML-DSA certificate inside the zkVM once and is reused across ses-

sions. The session proof assumes the static proof via composition and proves only Merkle non-revocation, ephemeral key binding, and trace-token commitment.

5.4 System Overview

Table 5.1: Table of Notations

$h(m)$	Hash of the message m
$(pk_i^{\mathsf{sig}}, sk_i^{\mathsf{sig}})$	Long-term ML-DSA-65 key pair of User U_i
$(pk_{\mathsf{CA}}, sk_{\mathsf{CA}})$	ML-DSA-65 key pair of Certificate Authority CA
$(pk_S^{\mathsf{KEM1}}, sk_S^{\mathsf{KEM1}})$	ML-KEM-1024 KEM key pair of Server S
$(pk_S^{\mathsf{Frodo}}, sk_S^{\mathsf{Frodo}})$	FrodoKEX+ split-KEM key material of Server S
seed_S	FrodoKEX+ shared seed published by Server S
(pk_T, sk_T)	ML-KEM-1024 key pair of Tracing Authority TRA
$\mathsf{link\_nonce}$	Random nonce held by User for cross-proof linking

This section describes the design of our revocable anonymous authentication system in a post-quantum environment. The system combines post-quantum digital signatures (ML-DSA-65, FIPS 204), key encapsulation mechanisms (ML-KEM-1024 for KEM1 and TRA encryption, FrodoKEX+ split-KEM for KEM2 with implicit authentication), a tracing authority (TRA), and a zero-knowledge virtual machine (RISC Zero) with proof composition to provide revocable anonymous authentication for client-server communication.

Table 5.1 shows the cryptographic material possessed by the four entities of the system. The CA maintains a revocation structure over user handles using a Merkle tree. The TRA is responsible for deanonymization and revocation when required. Each user additionally holds a random $\mathsf{link\_nonce}$ used to cryptographically link the cached static proof to per-session proofs.

5.4.1 Global Parameters

Let λ denote the security parameter. The system is parameterized by:

- A hash function $H : \{0,1\}^* \to \{0,1\}^{256}$ instantiated as SHA2-256 with domain-separated hashing for Merkle tree leaves and internal nodes.
- A key derivation function $\mathsf{KDF} : \{0,1\}^* \to \{0,1\}^{256}$, instantiated using HKDF with SHA-256.
- A post-quantum digital signature scheme ML-DSA-65 = $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ (FIPS 204).
- A post-quantum KEM ML-KEM-1024 (FIPS 203) = $(\mathsf{KeyGen}, \mathsf{Encaps}, \mathsf{Decaps})$ for the first KEM instance.
- A post-quantum split-KEM FrodoKEX+ = $(\mathsf{GenPP}, \mathsf{GenA}, \mathsf{GenB}, \mathsf{Encaps}, \mathsf{Decaps})$ for the second KEM instance, providing implicit mutual authentication through the split-KEM structure where both parties' secret keys contribute to the shared secret.
- A hybrid encryption scheme for TRA trace tokens, combining ML-KEM-1024 (FIPS 203) for key encapsulation with ChaCha20Poly1305 for authenticated encryption.
- A zero-knowledge proof system instantiated via RISC Zero zkVM with proof composition, decomposed into a cached static proof and a per-session proof.

We denote by acc_root the current Merkle tree root committing to the set of non-revoked users. The protocols in our system are User Registration, Anonymous Ephemeral Key Issuance (with two-phase ZK proving), Anonymous Authenticated Key Exchange, and Revocation. We will explain our protocols in detail.

Protocol 16 User Registration Protocol for user U

Input: $(pk_U^{\mathsf{sig}}, sk_U^{\mathsf{sig}})$

Output: User U acquires long-term identity attestation from CA

1: User U generates a long-term ML-DSA-65 key pair

$(pk_U^{\mathsf{sig}}, sk_U^{\mathsf{sig}}) \leftarrow \mathsf{ML\text{-}DSA.KeyGen}()$.

2: U sends the public key and metadata to CA:

$U \rightarrow CA: pk_U^{\mathsf{sig}}, \mathsf{meta}_U$.

3: After CA verifies the public key, it computes a revocation handle

$h_U = H(pk_U^{\mathsf{sig}} \parallel \mathsf{meta}_U)$.

4: The CA encodes a long-term certificate blob

$\mathsf{LTBlob}_U = \mathsf{encode}(pk_U^{\mathsf{sig}}, h_U)$.

5: CA signs the long-term certificate blob with its secret key

$\mathsf{cert}_U^{\mathsf{LT}} = \mathsf{ML\text{-}DSA.Sign}(sk_{\mathsf{CA}}, \mathsf{LTBlob}_U)$.

6: CA sends the signed long-term certificate blob to U:

$CA \rightarrow U: h_U, \mathsf{cert}_U^{\mathsf{LT}}$.

5.4.2 User Registration

Each user must register with the CA to obtain a certified long-term identity. Protocol 16 details the procedure for a user U to register with the CA, and obtain an attestation of long-term identity.

In Step 1, a user U generates a long-term ML-DSA-65 key pair: $(pk_U^{\mathsf{sig}}, sk_U^{\mathsf{sig}})$. In Step 2, the user sends the public key pk_U^{sig} and auxiliary metadata meta_U to the CA over an authenticated registration channel. In Step 3, after verifying the public key of the user U, the CA computes a revocation handle h_U, $H(pk_U^{\mathsf{sig}} \parallel \mathsf{meta}_U)$, and inserts h_U as a leaf in the Merkle tree of non-revoked users. The CA maintains and periodically publishes the Merkle root, $\mathsf{acc\_root}$, $\mathsf{MerkleRoot}(\{h_V : V \text{ non-revoked}\})$.

In Step 4, the CA encodes a long-term certificate blob, LTBlob_U, $\mathsf{encode}(pk_U^{\mathsf{sig}}, h_U)$. In Step 5, the CA signs the long-term certificate blob using its ML-DSA-65 secret key, $\mathsf{ML\text{-}DSA.Sign}(sk_{\mathsf{CA}}, \mathsf{LTBlob}_U)$ and sends it to user U in Step 6. After receiving the long-term certificate blob, the user's long-term state consists of: $(pk_U^{\mathsf{sig}}, sk_U^{\mathsf{sig}}, h_U, \mathsf{cert}_U^{\mathsf{LT}})$.

5.4.3 Anonymous Ephemeral Key Issuance

Before initiating a key exchange session, a user must obtain a CA-certified ephemeral public key without revealing their long-term identity. The Anonymous Ephemeral Key Issuance protocol achieves this through three phases: (1) a one-time static proof that verifies the user's long-term certificate inside a zkVM, (2) a per-session proof that binds a fresh ephemeral key to the user's identity while proving non-revocation, and (3) a certificate issuance exchange in which the CA verifies the composed proof and signs the ephemeral key. We describe each phase below.

5.4.3.1 Static Proof

Protocol 17 is executed once when the user first registers or whenever the long-term certificate changes. In Step 1, the user samples a fresh linking nonce to blind the identity link. In Step 2, the user runs the static guest program with the private witness containing the long-term certificate material. In Step 3, the guest verifies four conditions inside the zkVM: CA key binding, handle derivation correctness, blob structure integrity, and the ML-DSA certificate signature. In Step 4, the guest commits a journal containing only hashes and the blinded identity link - no secret values are revealed. In Steps 5-6, the user obtains and caches the resulting static proof and the nonce for reuse in future sessions.

The ML-DSA signature verification (Step 3.4) is the dominant cost. However, this cost is amortized, the resulting $\mathsf{receipt}_{\mathsf{static}}$ is cached to disk and reused for all subsequent sessions until the certificate is re-issued. The cached $\mathsf{receipt}_{\mathsf{static}}$ remains valid as long as

Protocol 17 Static Proof Generation

Input: $(pk_U^{\mathsf{sig}}, sk_U^{\mathsf{sig}}, h_U, \mathsf{cert}_U^{\mathsf{LT}}, pk_{\mathsf{CA}})$

Output: User U obtains a static proof π_{static}

1: User U samples a random linking nonce

$\mathsf{link\_nonce} \leftarrow \{0,1\}^{256}$.

2: User U runs the *static guest* program inside the zkVM with private witness $(pk_U^{\mathsf{sig}}, \mathsf{meta}_U, h_U, \mathsf{LTBlob}_U, \mathsf{cert}_U^{\mathsf{LT}}, pk_{\mathsf{CA}}, \mathsf{link\_nonce})$.

3: The static guest verifies inside the zkVM:

1. $H(pk_{\mathsf{CA}}) = \mathsf{pk\_CA\_hash}$ (CA key binding)
2. $h_U = H(pk_U^{\mathsf{sig}} \parallel \mathsf{meta}_U)$ (handle derivation)
3. $\mathsf{LTBlob}_U = \mathsf{encode}(pk_U^{\mathsf{sig}}, h_U)$ (blob structure)
4. $\mathsf{ML\text{-}DSA.Verify}(pk_{\mathsf{CA}}, \mathsf{LTBlob}_U, \mathsf{cert}_U^{\mathsf{LT}}) = 1$ (certificate signature)

4: The static guest commits to the journal:

$\mathsf{journal}_{\mathsf{static}} = (H(pk_{\mathsf{CA}}),\ H(\mathsf{cert}_U^{\mathsf{LT}}),\ \mathsf{identity\_link})$

where $\mathsf{identity\_link} = H(pk_U^{\mathsf{sig}} \parallel \mathsf{link\_nonce})$.

5: User obtains static proof π_{static} with receipt $\mathsf{receipt}_{\mathsf{static}}$.

6: User caches $(\mathsf{receipt}_{\mathsf{static}}, \mathsf{link\_nonce})$ to persistent storage.

the user's long-term certificate $\mathsf{cert}_U^{\mathsf{LT}}$ has not been re-issued or the CA's signing key pk_{CA} has not been rotated. Changes to the Merkle root acc_root (from other users being added or revoked) do not require re-proving the static proof, since Merkle membership is verified in the session guest. The user may optionally refresh link_nonce for forward security of the linking tag. The identity_link = $H(pk_U^{\mathsf{sig}} \parallel \mathsf{link\_nonce})$ serves as a blinded commitment that links the static proof to session proofs without revealing pk_U^{sig}.

5.4.3.2 Session Proof

Protocol 18 is executed once per ephemeral identity to generate a session proof that binds a fresh ephemeral key to the user's long-term identity while proving non-revocation, without repeating the expensive ML-DSA certificate verification. In Step 1, the user generates a fresh FrodoKEX+ key pair, $(pk_U^{\mathsf{ephem}}, sk_U^{\mathsf{ephem}})$, from the server's published seed, $seed_S$. This ephemeral public key will be certified by the CA at the end of the issuance protocol. In Step 2, the user chooses a fresh session identifier sid and constructs an identity token, $\mathsf{id\_token} = \mathsf{encode}(pk_U^{\mathsf{sig}}, h_U, \mathsf{sid})$, which encodes the user's long-term public key, revocation handle, and session identifier for potential future deanonymization by the TRA.

In Step 3, the user encrypts this identity token to the TRA's public key (pk_T) using ML-KEM-1024 key encapsulation followed by ChaCha20Poly1305 authenticated encryption, producing the trace token, C_{trace}. This ensures that only the TRA can recover the user's identity, and neither the CA nor the server can decrypt it. In Step 4, the user obtains from the CA a Merkle authentication path, path_U for the current accumulator root acc_root, which will be used inside the zkVM to prove that the user has not been revoked. In Step 5, the user runs the session guest program inside the zkVM, which verifies five conditions: (1) the composition assumption referencing the cached static proof via env.verify, (2) the cross-proof linking check that $H(pk_U^{\mathsf{sig}} \parallel \mathsf{link\_nonce}) = \mathsf{identity\_link}$, ensuring the same long-term key is used across both proofs, (3) CA consistency between the session input

Protocol 18 Session Proof Generation

Input: $(pk_U^{\mathsf{sig}}, sk_U^{\mathsf{sig}}, h_U, \mathsf{cert}_U^{\mathsf{LT}}, \mathsf{receipt}_{\mathsf{static}}, \mathsf{link\_nonce})$

Output: User U obtains a session proof π_{session}

1: User U generates a fresh FrodoKEX+ key pair from the server's published seed $(pk_U^{\mathsf{ephem}}, sk_U^{\mathsf{ephem}}) \leftarrow \mathsf{FrodoKEX+.GenB}(\mathsf{seed}_S)$.

2: User U chooses a fresh session identifier sid and constructs an identity token: $\mathsf{id\_token} = \mathsf{encode}(pk_U^{\mathsf{sig}}, h_U, \mathsf{sid})$.

3: User U encrypts this identity token to the TRA using ML-KEM-1024 and ChaCha20Poly1305:

$(ct_{\mathsf{kem}}, K) \leftarrow \mathsf{ML\text{-}KEM.Encaps}(pk_T)$,

$ct_{\mathsf{aead}} \leftarrow \mathsf{ChaCha20Poly1305.Enc}(K, \mathsf{id\_token})$,

$C_{\mathsf{trace}} = (ct_{\mathsf{kem}}, ct_{\mathsf{aead}}, \mathsf{nonce})$.

4: User U obtains from CA a Merkle authentication, path path_U, such that $\mathsf{MerkleVerify}(h_U, \mathsf{path}_U, \mathsf{acc\_root}) = 1$.

5: User U runs the *session guest* program inside the zkVM, verifying:

1. $\mathsf{env.verify}(\mathsf{static\_guest\_id}, \mathsf{journal}_{\mathsf{static}})$
2. $H(pk_U^{\mathsf{sig}} \parallel \mathsf{link\_nonce}) = \mathsf{identity\_link}$ (cross-proof linking)
3. $H(pk_{\mathsf{CA}})$ matches between session input and static journal (CA consistency)
4. $h_U = H(pk_U^{\mathsf{sig}} \parallel \mathsf{meta}_U)$ (handle derivation)
5. $\mathsf{MerkleVerify}(h_U, \mathsf{path}_U, \mathsf{acc\_root}) = 1$ (non-revocation)

6: The session guest commits to the journal, $\mathsf{journal}_{\mathsf{session}}$:

$H(pk_{\mathsf{CA}})$, $\mathsf{acc\_root}$, $H(pk_U^{\mathsf{ephem}})$, $\mathsf{bind\_commitment}$, $H(C_{\mathsf{trace}})$, sid, $\mathsf{identity\_link}$.

7: User obtains session proof π_{session} with receipt $\mathsf{receipt}_{\mathsf{session}}$.

and the static journal, (4) correctness of the revocation handle derivation, and (5) Merkle membership of h_U in the current acc_root. Notably, the session guest does not re-verify the ML-DSA certificate signature; instead, it relies on the composition assumption resolved in the first condition. The host resolves this assumption by supplying $\mathsf{receipt}_{\mathsf{static}}$ during proving, and the resulting composed receipt is valid only if both the static and session proofs are individually valid and the identity_link matches between them.

In Step 6, the session guest commits to the session journal, which contains $H(pk_{\mathsf{CA}})$, acc_root, $H(pk_U^{\mathsf{ephem}})$, bind_commitment, $H(C_{\mathsf{trace}})$, sid, identity_link. The host precomputes $H(pk_U^{\mathsf{ephem}})$ and passes only the 32-byte hash to the guest. The session guest computes the bind_commitment, which ties the user's long-term identity to the ephemeral key without revealing (pk_U^{sig}). In Step 7, the user obtains the session proof π_{session} with receipt $\mathsf{receipt}_{\mathsf{session}}$.

Protocol 19 Anonymous Ephemeral Key Issuance Protocol

Input: $(pk_U^{\mathsf{sig}}, sk_U^{\mathsf{sig}}, h_U, \mathsf{cert}_U^{\mathsf{LT}}, \mathsf{receipt}_{\mathsf{static}}, \mathsf{link\_nonce})$

Output: User U obtains an ephemeral public key certified by CA

1: User U sends to CA:

$U \to CA$: $(pk_U^{\mathsf{ephem}}, C_{\mathsf{trace}}, \mathsf{sid}, \mathsf{receipt}_{\mathsf{session}})$.

2: CA verifies $\mathsf{receipt}_{\mathsf{session}}$ (which transitively verifies the static proof via composition) and checks that $\mathsf{journal}_{\mathsf{session}}$ fields match $(pk_{\mathsf{CA}}, \mathsf{acc\_root}, pk_U^{\mathsf{ephem}}, C_{\mathsf{trace}}, \mathsf{sid})$.

3: CA constructs an ephemeral certificate payload

$\mathsf{EphemPayload}_U = \mathsf{encode}(pk_U^{\mathsf{ephem}}, C_{\mathsf{trace}}, t_{\mathsf{exp}})$

4: CA signs the ephemeral certificate payload

$\mathsf{Cert}_U^{\mathsf{ephem}} = \mathsf{ML\text{-}DSA.Sign}(sk_{\mathsf{CA}}, \mathsf{EphemPayload}_U)$

5: CA sends the ephemeral certificate payload and the certificate to User U

$CA \to U$: $(\mathsf{EphemPayload}_U, \mathsf{Cert}_U^{\mathsf{ephem}})$

Protocol 19 describes the interaction between the user and the CA in which the user presents the ephemeral key and composed proof, and the CA issues a signed ephemeral certificate. In Step 1, the user sends to the CA the ephemeral public key pk_U^{ephem}, the encrypted trace token C_{trace}, the session identifier sid, and the session receipt $\mathsf{receipt}_{\mathsf{session}}$. In Step 2, the CA verifies $\mathsf{receipt}_{\mathsf{session}}$, which transitively verifies the static proof via composition, and checks that the session journal fields match the expected values for $(pk_{\mathsf{CA}}, \mathsf{acc\_root}, pk_U^{\mathsf{ephem}}, C_{\mathsf{trace}}, \mathsf{sid})$. The CA also verifies the $H(pk_U^{\mathsf{ephem}})$ outside the proof, confirming that the ephemeral key presented matches the one committed to in the session journal. If any check fails, the CA aborts the protocol.

In Step 3, the CA constructs an ephemeral certificate payload $\mathsf{EphemPayload}_U$ by encoding $(pk_U^{\mathsf{ephem}}, C_{\mathsf{trace}}, t_{\mathsf{exp}})$, embedding the trace token and an expiration time. In Step 4, the CA signs this payload using its ML-DSA-65 secret key, producing $\mathsf{Cert}_U^{\mathsf{ephem}}$. In Step 5, the CA sends the ephemeral certificate payload and the signed certificate to the user. Now, the user's ephemeral state consists of $(pk_U^{\mathsf{ephem}}, sk_U^{\mathsf{ephem}}, \mathsf{EphemPayload}_U, \mathsf{Cert}_U^{\mathsf{ephem}})$, which is used as input to the Anonymous Authenticated Key Exchange protocol.

5.4.4 Anonymous Authenticated Key Exchange

Protocol 20 describes the anonymous authenticated key exchange protocol between a user U and the server S. The server S holds an ML-KEM-1024 key pair $(pk_S^{\mathsf{KEM1}}, sk_S^{\mathsf{KEM1}})$ and FrodoKEX+ split-KEM key material $(pk_S^{\mathsf{Frodo}}, sk_S^{\mathsf{Frodo}})$ generated from a published seed, seed_S. The public keys $(pk_S^{\mathsf{KEM1}}, pk_S^{\mathsf{Frodo}}, \mathsf{seed}_S)$ are known to all users. The use of ML-KEM-1024 (a standard KEM) combined with FrodoKEX+ (a split-KEM) provides both defense in depth and implicit mutual authentication: even if one KEM instance is compromised, the session key remains secure, and the split-KEM ensures both parties' secret keys contribute to the session key derivation.

In Step 1, the user encapsulates to the server's ML-KEM-1024 public key, resulting

Protocol 20 Anonymous Authenticated Key Exchange Protocol

Input: $(pk_U^{\mathsf{ephem}}, sk_U^{\mathsf{ephem}}, \mathsf{EphemPayload}_U, \mathsf{Cert}_U^{\mathsf{ephem}})$

Output: User U and Server S establish a shared key K

1: User U encapsulates to the ML-KEM-1024 public key of S

$(K_1, ct_1) \leftarrow \mathsf{ML\text{-}KEM.Encaps}(pk_S^{\mathsf{KEM1}})$.

2: User U performs FrodoKEX+ split-KEM encapsulation using the server's public key and the user's ephemeral secret key

$(K_2, ct_2) \leftarrow \mathsf{FrodoKEX+.Encaps}(pk_S^{\mathsf{Frodo}}, sk_U^{\mathsf{ephem}})$.

3: User U samples a fresh nonce nonce_U and derives the session key.

$K = \mathsf{KDF}(K_1 \parallel K_2 \parallel nonce_U)$.

4: User U sends the ClientHello message to S

$U \rightarrow S$: $(pk_U^{\mathsf{ephem}}, \mathsf{EphemPayload}_U, \mathsf{Cert}_U^{\mathsf{ephem}}, ct_1, ct_2, \mathsf{nonce}_U)$.

5: S performs Ephemeral certificate verification

$\mathsf{ML\text{-}DSA.Verify}(pk_{\mathsf{CA}}, \mathsf{EphemPayload}_U, \mathsf{Cert}_U^{\mathsf{ephem}}) = 1$.

6: After the verification, S performs ML-KEM-1024 decapsulation

$K_1 = \mathsf{ML\text{-}KEM.Decaps}(sk_S^{\mathsf{KEM1}}, ct_1)$.

7: Then S performs FrodoKEX+ split-KEM decapsulation using its secret key and the user's CA-certified ephemeral public key

$K_2 = \mathsf{FrodoKEX+.Decaps}(sk_S^{\mathsf{Frodo}}, pk_U^{\mathsf{ephem}}, ct_2)$.

8: S derives the session key using KDF

$K = \mathsf{KDF}(K_1 \parallel K_2 \parallel nonce_U)$.

in (K_1, ct_1). In Step 2, the user performs FrodoKEX+ split-KEM encapsulation using the server's public key pk_S^{Frodo} and the user's ephemeral secret key sk_U^{ephem}, resulting in (K_2, ct_2). This provides implicit authentication: only the holder of sk_U^{ephem} (matching the CA-certified pk_U^{ephem}) can produce a valid K_2. In Step 3, the user samples a fresh nonce nonce_U and derives the session key, K, using HKDF with K_1, K_2, and nonce_U as inputs. The user sends the ClientHello message to the server in Step 4.

After receiving ClientHello message, the server performs ephemeral certificate verification in Step 5. If the check fails or the policy/expiry is not satisfied, the server aborts the protocol. In Step 6, the server performs ML-KEM-1024 decapsulation. In Step 7, the server performs FrodoKEX+ split-KEM decapsulation using its secret key sk_S^{Frodo} and the user's CA-certified ephemeral public key pk_U^{ephem}, providing implicit authentication of the server. In Step 8, the server derives the session key using KDF with (K_1, K_2, and nonce_U) as inputs which matches the user's key K if both parties are honest and the KEMs are correct. The server stores the trace token for potential deanonymization. All subsequent application data is protected using an AEAD scheme under K.

5.4.5 Revocation

For each session, the server extracts C_{trace} from the ephemeral certificate payload and logs it together with session metadata (time, IP address, protocol version, etc.). The server does not know the user's identity, as C_{trace} is encrypted under pk_T using ML-KEM-1024 key encapsulation and ChaCha20Poly1305 authenticated encryption. In the event of misbehavior, the server initiates the Revocation Protocol described by Protocol 21. In Step 1, the server sends the logged C_{trace} to the TRA along with the proof of misbehavior. In Step 2, the TRA verifies the proof and then decapsulates the ML-KEM ciphertext to recover the shared secret, then decrypts the AEAD ciphertext to recover the identity token. The TRA recovers the long-term identity of the user $(pk_U^{\mathsf{sig}}, h_U, \mathsf{sid})$ in Step 3.

Protocol 21 Revocation Protocol

Input: C_{trace}

Output: Anonymity revocation of misbehaving User U

1: S sends the encrypted trace token and proof of misbehavior to the TRA.

$S \rightarrow TRA$: C_{trace}.

2: TRA decrypts the trace token:

K = ML-KEM.Decaps(sk_T, ct_{kem}),

id_token = ChaCha20Poly1305.Dec(K, ct_{aead}, nonce).

3: TRA recovers the long-term identity $(pk_U^{\mathsf{sig}}, h_U, \mathsf{sid})$ from the id_token.

4: TRA sends the revocation request to the CA

$TRA \rightarrow CA$: $(pk_U^{\mathsf{sig}}, h_U, \mathsf{sid})$.

In Step 4, the TRA sends a revocation request along with the proof of misbehavior to the CA. Upon receiving a revocation request for user U, the CA verifies the proof of misbehavior and marks Merkle tree handle of the user, h_U, as revoked, by replacing it with a zero-hash in the Merkle tree. The CA then recomputes the Merkle tree and updates acc_root. The CA publishes the updated acc_root to all users. Subsequent anonymous ephemeral key issuance attempts by U fail, as the user can no longer produce a valid Merkle path for h_U in the current acc_root. Thus, U can no longer obtain ephemeral certificates. Note that the user's cached $\mathsf{receipt}_{\mathsf{static}}$ becomes useless after revocation, since the session proof will fail Merkle verification against the updated root.

5.5 Security Analysis

We now analyze the security of the framework against several classes of attacks.

Post-quantum security: All primitives (ML-DSA-65, ML-KEM-1024, FrodoKEX+, SHA2-256, ChaCha20Poly1305) used to design the revocable anonymous authentication

system are chosen to provide security against quantum adversaries. ML-DSA-65 (FIPS 204) and ML-KEM-1024 (FIPS 203) are NIST FIPS-standardized. FrodoKEX+ is a conservative split-KEM construction based on the Learning With Errors (LWE) problem over unstructured lattices, derived from FrodoKEM (a NIST PQC alternate candidate); while not itself FIPS-standardized, its conservative parameter choices and LWE-based security provide strong post-quantum guarantees complementary to the structured-lattice schemes. The KDF combines the ML-KEM-1024 shared secret with the FrodoKEX+ shared secret ($K = \mathsf{KDF}(K_1 \parallel K_2 \parallel nonce_U)$), providing hybrid quantum resilience across both structured and unstructured lattice assumptions.

Privacy: The trust is distributed between the TRA and the CA because the TRA can unmask the identity of the user but cannot prevent the user from obtaining more ephemeral identities since that is dependent on the membership in the Merkle tree. Only the CA can revoke the membership in the Merkle tree, not the TRA. Furthermore, the CA cannot unmask the identity of a user because the trace payload can only be decrypted by the TRA. Based on our assumption that the CA and the TRA do not collude and both of them do not collude with the server, the anonymity of the user is guaranteed in our system. The CA sees only the composed receipt and session journal, which contain only hashes and commitments, never the user's long-term key or handle.

Forgery attack: An external attacker cannot forge ephemeral identities because our system model assumes that an attacker cannot defeat quantum-safe primitives. The static proof verifies the ML-DSA certificate signature, ensuring that the user provably holds a valid CA-signed certificate. The cross-proof identity link ($\mathsf{identity\_link} = H(pk_U^{\mathsf{sig}} \parallel \mathsf{link\_nonce})$) and hash-based binding ($\mathsf{bind\_commitment} = H(pk_U^{\mathsf{sig}} \parallel H(pk_U^{\mathsf{ephem}}))$) ensure that only the entity with knowledge of pk_U^{sig} can produce valid session proofs.

Man-in-the-middle attack: Our framework resists man-in-the-middle attacks through multiple layers of cryptographic binding across its protocols. During the Anonymous

Ephemeral Key Issuance protocol, an adversary who attempts to substitute a fraudulent ephemeral public key cannot produce a valid composed zero-knowledge proof: the session guest commits the hash of ephemeral public key to the session journal, and the CA verifies that this commitment matches the ephemeral key presented outside the proof. Any tampering with the ephemeral key breaks this binding, causing the CA to reject the issuance request.

During the Anonymous Authenticated Key Exchange, the dual-KEM structure provides two independent barriers against interception. The first KEM layer (ML-KEM-1024) ensures that only the server holding the secret key of $KEM1$ can recover $K1$ from the ciphertext $ct1$. The second layer (FrodoKEX+ split-KEM) provides implicit mutual authentication: the encapsulation uses the user's ephemeral secret key while the decapsulation uses the server's secret key, so an adversary positioned between the two parties cannot derive $K2$ without knowledge of both secret keys. Since the session key K depends on both shared secrets, compromising a single KEM instance is insufficient to recover K.

Furthermore, the user's ephemeral public key is certified by the CA, which the server verifies in Step 5 of Protocol 20. An adversary who substitutes a different ephemeral key cannot produce a valid CA signature on that key without compromising the CA's signing key. An active adversary who attempts to relay or modify the ClientHello message will cause the server to derive a different session key than the user, resulting in detectable failure when application-layer AEAD decryption is attempted under the mismatched key.

Security of the issuance protocol: The bind_commitment serves as a hash-based binding that ties the certified long-term identity to the session's ephemeral key. The static proof has already established, via in-guest ML-DSA verification, that the user holds a valid CA-signed certificate for a public key, and the cross-proof identity ensures the same public key is used in both proofs. The CA verifies the composed receipt, ensuring the static proof's validity is checked transitively, but the CA does not learn any identifying information — it

sees only hashes, commitments, and the encrypted trace token, which is decryptable only by the TRA. Additionally, the FrodoKEX+ split-KEM provides implicit authentication during the subsequent AAKE: the user's ephemeral secret key is used in encapsulation, ensuring that only the holder of the certified ephemeral key can derive the correct session key. This prevents an adversary from using a stolen ephemeral certificate without the corresponding secret key.

Composition soundness: RISC Zero's composition mechanism ensures that the composed receipt is valid if and only if both the static and session proofs are individually valid and the assumption recorded by the session guest matches the static receipt's claim. An adversary cannot substitute a different static proof without breaking the collision resistance of the identity link hash.

Implicit authentication via split-KEM: The AAKE protocol achieves implicit authentication through the FrodoKEX+ split-KEM construction. Both the server and the user contribute secret keys to the shared secret derivation: the server uses its secret key sk_S^{Frodo} in decapsulation, while the user uses its secret key sk_U^{ephem} in encapsulation. An adversary who intercepts the public parameters (seed, b_S, b_U) cannot derive the shared secret without knowledge of either party's secret key. Since the user's FrodoKEX+ public key b_U is the CA-certified ephemeral key pk_U^{ephem}, the shared secret is implicitly bound to the certified identity. Combined with the ML-KEM-1024 KEM layer, the hybrid KDF provides security even if one lattice assumption is broken.

5.6 Performance Evaluation

In this section, we describe the implementation of our protocols on real hardware and evaluate them to demonstrate that they can be implemented with reasonable latency and without any additional hardware or software requirements. We implemented our protocols using standard crypto libraries in Rust. To compare the results of the system over different

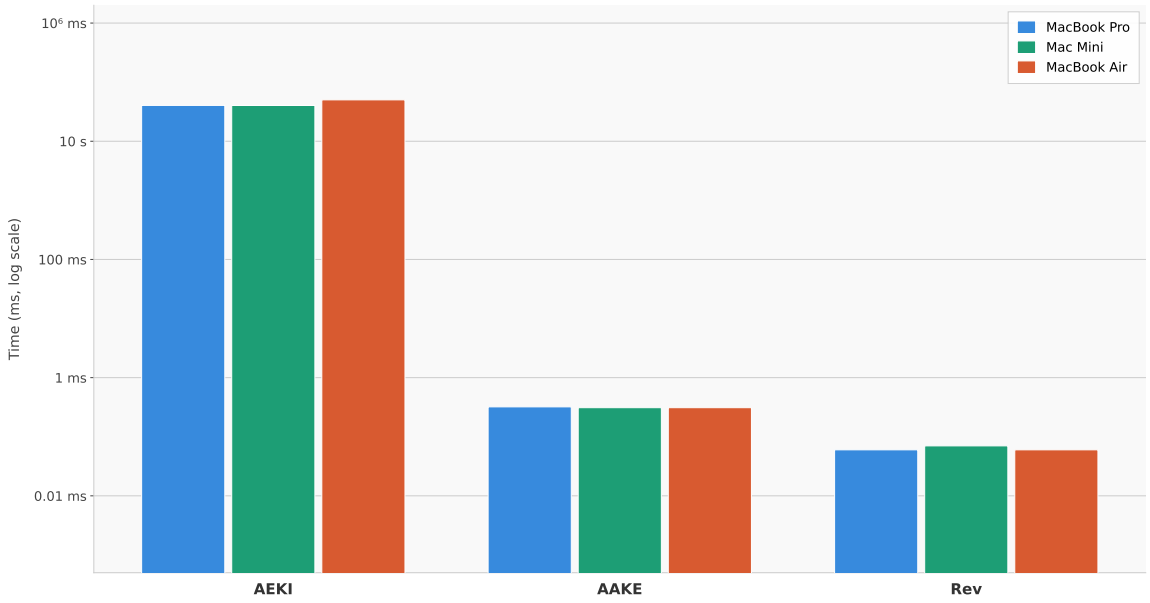


Figure 5.1: Time Taken by Various Protocols of Our System

Table 5.2: Key Sizes of Cryptographic Schemes

Scheme	Public Key	Secret Key	Cipher Text	Signature
ML-DSA-65	1952 B	4032 B		3309 B
ML-KEM-1024	1568 B	3168 B	1568 B	
FrodoKEX+	23,232 B	23,232 B	50,022 B	

hardware, we ran all the protocols in our system on MacBook Air with M4 processor/16 GB RAM, Mac Mini with M4 processor/16 GB RAM, and MacBook Pro with M4 processor/32 GB RAM. All the results were averaged over 100 runs. To evaluate the scalability and the robustness of the protocols, we measure the time taken by protocols of the system. We do not have any state-of-the-art work to compare against. Therefore, we do not compare our work with other approaches but demonstrate the scalability of our framework in real-world settings.

We measure the time taken by each phase in the Anonymous Ephemeral Key Issuance (AEKI) protocol, the Anonymous Authenticated Key Exchange (AAKE) protocol, and

Table 5.3: Step-Level Execution Times

Protocol	Steps	Time
AEKI	Ephem. Key + Trace Token	9.52 ms
	Static Proof — cached	108.20 ms
	Session Proof	40,045.04 ms
	ZK Verification	119.30 ms
	CA Ephem. Cert Issue	0.56 ms
AAKE	ClientHello	0.18 ms
	Server Process	0.13 ms
Revocation	TRA Deanonymization	0.06 ms
	CA Revocation	0.01 ms

the Revocation protocol. The User Registration protocol (Protocol 16) takes a negligible amount of time (in the order of microseconds) and is not shown. Fig. 5.1 shows the aggregate time taken by each protocol across the three hardware platforms on a logarithmic scale. Table 5.3 presents a detailed step-level breakdown of the execution times (averaged across all three devices).

The AEKI protocol (Protocols 17-19) is the most computationally intensive protocol in our framework, taking approximately 40.28 seconds. This cost is dominated by the session proof generation, which takes approximately 40.05 seconds. The remaining phases of the AEKI protocol are comparatively lightweight: ephemeral key generation and trace token construction takes 9.52 ms, loading the cached static proof from disk takes 108.20 ms, ZK verification by the CA takes 119.30 ms, and the CA's ephemeral certificate issuance takes 0.56 ms.

The static proof (Protocol 17) contains the expensive ML-DSA-65 certificate verification inside the zkVM. However, this cost is a one-time cost that is amortized over the

lifetime of the user's long-term certificate, which typically spans days or weeks. In our measurements, the static proof is loaded from a cached receipt on disk, taking only 108.20 ms. The first time computation of static proof takes 600 seconds. Since the static proof does not need to be regenerated unless the user's long-term certificate is re-issued or the CA's signing key is rotated, this amortized cost is negligible in practice.

The AAKE protocol (Protocol 20) is significantly faster than the AEKI protocol. The ClientHello construction by the user, which includes ML-KEM-1024 encapsulation and FrodoKEX+ split-KEM encapsulation, takes only 0.18 ms. The server-side processing, which includes ephemeral certificate verification, ML-KEM-1024 decapsulation, FrodoKEX+ split-KEM decapsulation, and session key derivation, takes 0.13 ms. The total AAKE protocol time of 0.31 ms demonstrates that the key exchange itself is highly efficient and suitable for real-time interactive sessions.

The TRA deanonymization in the Revocation protocol (Protocol 21), which involves ML-KEM-1024 decapsulation and ChaCha20Poly1305 decryption of the trace token, takes 0.06 ms. The CA-side revocation, which involves marking the user's handle as revoked and recomputing the Merkle tree root, takes 0.01 ms. The total revocation time of 0.07 ms confirms that the revocation mechanism introduces negligible computational overhead.

As observed from our experiments, the performance characteristics across the three hardware platforms are consistent. The MacBook Pro with 32 GB RAM shows marginally better performance on the session proof generation due to higher memory bandwidth, but the differences across devices are small, indicating that our framework does not require high-end hardware. The key observation is that the computationally expensive operation (session proof generation) is a per-session one-time cost that occurs before the actual key exchange. Once the ephemeral certificate is obtained, the real-time key exchange (AAKE) completes in under a millisecond, making the framework practical for interactive applications.

Table 5.4: Storage Space

Item	Bytes
User long-term state	7,313
CA key material	1,984
TRA key material	1,632
Server key material	50,208
Ephemeral keypair (per-session)	46,464
Merkle proof (per-session)	66
Static proof receipt (cached)	2,519,466
Session proof receipt (per-session)	2,789,088
Long-term certificate	5,297
Ephemeral certificate	3,0173

Table 5.2 shows the key sizes of the various cryptographic schemes used in our protocols. The FrodoKEX+ split-KEM has the largest key sizes (23,232 bytes each for public and secret keys), which is expected for an unstructured-lattice-based scheme that provides conservative post-quantum security. Table 5.4 shows the storage space needed for various artifacts in our protocols. The server key material is the largest per-entity storage requirement at 50,208 bytes, primarily due to the FrodoKEX+ key pair. The user's long-term state requires only 7,313 bytes, and the CA and TRA key materials require 1,984 bytes and 1,632 bytes, respectively. The per-session ephemeral key pair requires 46,464 bytes.

The RISC Zero zkVM proof sizes are approximately 2.66 MB for the session STARK receipt and 2.40 MB for the cached static receipt. The per-session communication overhead is approximately 2.77 MB (session receipt + ephemeral certificate exchange + ClientHello). While this is larger than a typical TLS handshake (which is on the order of a few kilobytes), the overhead is a one-time cost per ephemeral identity and is comparable to other zkVM-

based authentication approaches. For applications where bandwidth is constrained, the session receipt constitutes the bulk of the communication cost; future improvements in STARK proof compression (such as RISC Zero's Groth16 wrapping, which reduces proofs to approximately 256 bytes) could significantly reduce this overhead.

The key sizes, proof sizes, and storage requirements demonstrate that the storage and communication overhead for our protocols is manageable and practical for deployment in real-world settings, including desktop, laptop, and mobile environments.

5.7 Conclusions

In this chapter, we designed and implemented a secure, scalable, and efficient revocable anonymous authenticated key exchange in the post-quantum setting without relying on a single trusted central authority. We integrated FrodoKEX+, a split-KEM construction based on unstructured LWE, to achieve implicit authentication in the AAKE protocol, where both parties contribute secret keys to the shared secret derivation. We analyzed the security of our framework and implemented it to demonstrate its scalability and efficiency. Our framework allows users to obtain ephemeral identities to authenticate themselves anonymously to a server for real-time key exchange. If a user misbehaves, our system has a mechanism to revoke the anonymity of the user. Our framework is not only robust against attacks by malicious users, but also efficient and scalable, as the results from our evaluation indicate.

Chapter 6

CONCLUSIONS

Anonymous systems underpin many of the privacy-preserving technologies vital to modern digital life, yet pure anonymity, without any form of attribution, can limit their practical deployment. This dissertation investigated attribution in anonymous systems across four complementary scenarios, spanning pre-quantum and post-quantum cryptographic settings, authentication and reputation contexts, and increasingly sophisticated threat models.

In Chapter 2, we proposed a revocable anonymous authentication framework for V2G communications. The framework enables an EV to authenticate anonymously to a charging station using unlinkable pseudonyms derived from blindly signed tokens, while a federated trust architecture comprising a Certificate Authority and a Department of Motor Vehicles can jointly revoke the anonymity of a proven malicious EV. We also introduced a receipt-based mechanism that allows an EV to prove its innocence in the face of false accusations. Security analysis and implementation-based evaluation demonstrated that the framework is secure, lightweight, and scalable.

In Chapter 3, we identified and addressed application-level DoS attacks against the revocable anonymous authentication framework. We augmented the framework with a permit-based mechanism that requires each EV to hold a valid, single-use permit backed by a monetary deposit before making a charging reservation. This economic enforcement,

combined with the existing cryptographic protections, makes large-scale DoS attacks economically infeasible. We introduced a third Federated Trust Entity, the Financial Authority, and provided detailed privacy, security, and performance analysis showing that the added robustness comes with minimal overhead.

In Chapter 4, we designed EARS, an anonymous reputation system that supports perpetual anonymous feedback updates without relying on a fully trusted central authority. EARS enables users to post information, provide feedback, and update feedback over time—all anonymously—while attributing feedback to the poster's reputation through partially blind signatures and cut-and-choose protocols. We performed detailed security analysis against documented attacks on anonymous reputation systems, including bad mouthing, ballot stuffing, and deanonymization, and demonstrated through implementation that EARS is efficient and scalable.

In Chapter 5, we proposed a revocable anonymous authenticated key exchange in the post-quantum setting. The framework combines ML-DSA-65 for digital signatures and ML-KEM-1024 and FrodoKEX+ for key encapsulation, with a Tracing Authority and RISC Zero zkVM for zero-knowledge proof of non-revocation. This design eliminates the reliance on pairings or discrete logarithm assumptions that are vulnerable to quantum algorithms, while preserving the strong anonymity, efficient revocation, and distributed trust properties established in earlier chapters. Implementation-based evaluation confirmed that the post-quantum framework is practical and scalable.

The problem of attribution in anonymous systems extends well beyond the specific applications studied in this dissertation. Whistleblower platforms, anonymous voting systems, privacy-preserving access control, decentralized online communities, and anonymous marketplaces all face the fundamental tension between protecting user privacy and maintaining accountability. The frameworks and design principles developed in this dissertation provide a foundation for addressing this tension in these and other emerging contexts. The

post-quantum dimension of this work is particularly timely: as the threat of quantum computation to existing cryptographic infrastructure grows, ensuring that privacy-preserving authentication systems remain secure in the post-quantum era is essential for the long-term viability of digital privacy.

Beyond solving specific problems in selected contexts, this dissertation proposes general and extensible frameworks that can adapt to changes in trust models and use cases. We believe that the design principles of distributed trust, unlinkability, and modular protocol composition will continue to be valuable as the landscape of anonymous systems evolves. We hope that this work inspires further research at the intersection of anonymity and attribution, and contributes to digital systems that are both privacy-preserving and trustworthy. Future work will focus in part on extending the Chapter 5 solution, which demonstrates that revocable anonymous authentication can be realized using post-quantum primitives. A natural next step is to extend this approach to the anonymous reputation setting of Chapter 4. Designing a post-quantum anonymous reputation system that supports perpetual feedback updates while maintaining the security and privacy guarantees of EARS would be a significant contribution.

Overall, the dissertation demonstrates that attribution need not be incompatible with anonymity. With careful protocol design, distributed trust, and narrowly scoped revocation or update mechanisms, anonymous systems can support accountability while preserving privacy for honest participants.

REFERENCES

[1] Afrin, S. and A. Kwasinski, "A Privacy-Preserving Method with Flexible Charging Schedules for Electric Vehicles in the Smart Grid", in "Proc. IEEE ANTS", pp. 1–6 (2017).

[2] Androulaki, E., S. G. Choi, S. M. Bellovin and T. Malkin, "Reputation systems for anonymous networks", in "Proc. PETS", pp. 202–218 (2008).

[3] Bellare, M., D. Micciancio and B. Warinschi, "Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions", in "International conference on the theory and applications of cryptographic techniques", pp. 614–629 (Springer, 2003).

[4] Bellare, M., H. Shi and C. Zhang, "Foundations of group signatures: The case of dynamic groups", in "Cryptographers' track at the RSA conference", pp. 136–153 (Springer, 2005).

[5] Blakley, G. R., "Safeguarding cryptographic keys", in "Proceedings of the AFIPS National Computer Conference", pp. 313–317 (IEEE, 1979).

[6] Blömer, J., J. Juhnke and C. Kolb, "Anonymous and publicly linkable reputation systems", in "Proc. FC", pp. 478–488 (2015).

[7] Boneh, D., X. Boyen and H. Shacham, "Short group signatures", in "Annual international cryptology conference", pp. 41–55 (Springer, 2004).

[8] Boneh, D. and H. Shacham, "Group signatures with verifier-local revocation", in "Proceedings of the 11th ACM conference on Computer and communications security", pp. 168–177 (2004).

[9] Busom, N., R. Petrlic, F. Sebé, C. Sorge and M. Valls, "A privacy-preserving reputation system with user rewards", Elsevier Journal of Network and Computer Applications **80**, 58–66 (2017).

[10] Camenisch, J., M. Drijvers, A. Lehmann, G. Neven and P. Towa, "Short threshold dynamic group signatures", in "International conference on security and cryptography for networks", pp. 401–423 (Springer, 2020).

[11] Camenisch, J., S. Hohenberger, M. Kohlweiss, A. Lysyanskaya and M. Meyerovich, "How to win the clonewars: efficient periodic n-times anonymous authentication", in "ACM CCS", pp. 201–210 (2006).

[12] Camenisch, J., S. Hohenberger and A. Lysyanskaya, "Balancing accountability and privacy using e-cash", in "Proc. SCN", pp. 141–155 (2006).

[13] Camenisch, J. and A. Lysyanskaya, "An efficient system for non-transferable anonymous credentials with optional anonymity revocation", in "International conference on the theory and applications of cryptographic techniques", pp. 93–118 (Springer, 2001).

[14] Camenisch, J. and A. Lysyanskaya, "Dynamic accumulators and application to efficient revocation of anonymous credentials", in "Annual international cryptology conference", pp. 61–76 (Springer, 2002).

[15] Camenisch, J. and A. Lysyanskaya, "Signature schemes and anonymous credentials from bilinear maps", in "Annual international cryptology conference", pp. 56–72 (Springer, 2004).

[16] Canard, S., B. Schoenmakers, M. Stam and J. Traoré, "List signature schemes", Discrete Applied Mathematics **154**, 2, 189–201 (2006).

[17] Chaudhary, D., P. Santhi, M. Durgarao, A. Padmavathi, M. M. Hassan and B. F. Alkhamees, "Module lattice-based post quantum secure blockchain empowered authentication framework for autonomous truck platooning", IEEE access **12**, 105219–105233 (2024).

[18] Chaum, D., "Blind signatures for untraceable payments", in "Advances in Cryptology", pp. 199–203 (Springer, 1983).

[19] Chaum, D. and E. Van Heyst, "Group signatures", in "Advances in Cryptology, EUROCRYPT", pp. 257–265 (Springer, 1991).

[20] Chen, L., C. Dong, C. J. Newton and Y. Wang, "Sphinx-in-the-head: group signatures from symmetric primitives", ACM transactions on privacy and security **27**, 1, 1–35 (2024).

[21] Christin, D., C. Roßkopf, M. Hollick, L. A. Martucci and S. S. Kanhere, "Incognisense: An anonymity-preserving reputation framework for participatory sensing applications", Proc. IEEE PerCom **9**, 3, 353–371 (2013).

[22] Collins, D., L. Huguenin-Dumittan, N. K. Nguyen, N. Rolin and S. Vaudenay, "{K-Waay}: Fast and deniable {Post-Quantum}{X3DH} without ring signatures", in "33rd USENIX Security Symposium (USENIX Security 24)", pp. 433–450 (2024).

[23] Cremers, C., A. Dax and N. Medinger, "Keeping up with the kems: Stronger security notions for kems and automated analysis of kem-based protocols", in "Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security", pp. 1046–1060 (2024).

[24] Diffie, W., P. C. Van Oorschot and M. J. Wiener, "Authentication and authenticated key exchanges", Designs, Codes and Cryptography **2**, 107–125 (1992).

[25] Dingledine, R., N. Mathewson and P. Syverson, "Tor: The Second-Generation Onion Router", in "Proc. USENIX Security", pp. 303–320 (2004).

[26] Gennaro, R., S. Goldfeder and B. Ithurburn, "Fully distributed group signatures", https://www.orbs.com/white-papers/fully-distributed-group-signatures/ (2019).

[27] Hsiao, H.-C., A. Studer, C. Chen, A. Perrig, F. Bai, B. Bellur and A. Iyer, "Flooding-resilient broadcast authentication for vanets", in "ACM MobiCom", pp. 193–204 (2011).

[28] Huang, D., S. Misra, M. Verma and G. Xue, "PACP: an efficient pseudonymous authentication-based conditional privacy protocol for vanets", IEEE Transactions on Intelligent Transportation Systems **12**, 736–746 (2011).

[29] Kemmoe, V. Y., A. Lysyanskaya and N. K. Nguyen, "Lattice-based accumulator and application to anonymous credential revocation", Cryptology ePrint Archive (2025).

[30] Khare, D. K., D. Dangi, A. Bhagat, S. Malviya and D. K. Dixit, "Post-quantum traceable anonymous cross-domain authentication for blockchain-based iot", IEEE Transactions on Consumer Electronics (2025).

[31] Kiayias, A., Y. Tsiounis and M. Yung, "Traceable signatures", in "Advances in Cryptology, EUROCRYPT", pp. 571–589 (Springer, 2004).

[32] Kilari, V. T., S. Misra and G. Xue, "Revocable Anonymity Based Authentication for Vehicle to Grid (V2G) Communications", in "Proc. IEEE SmartGridComm", pp. 351–356 (2016).

[33] Kilari, V. T., R. Yu, S. Misra and G. Xue, "Ears: Enabling private feedback updates in anonymous reputation systems", in "2020 IEEE Conference on Communications and Network Security (CNS)", pp. 1–9 (IEEE, 2020).

[34] Kilari, V. T., R. Yu, S. Misra and G. Xue, "Robust revocable anonymous authentication for vehicle to grid communications", IEEE Transactions on Intelligent Transportation Systems **21**, 11, 4845–4857 (2020).

[35] Kokoschka, A., R. Petrlic and C. Sorge, "A reputation system supporting unlinkable, yet authorized expert ratings", in "Proc. ACM SAC", pp. 2320–2327 (2015).

[36] Köpsell, S., R. Wendolsky and H. Federrath, "Revocable anonymity", in "Emerging Trends in Information and Communication Security", pp. 206–220 (Springer, 2006).

[37] Krenn, S., O. Mir and D. Slamanig, "Structure-preserving compressing primitives: Vector commitments and accumulators and applications", Cryptology ePrint Archive (2024).

[38] Latif, R., B. M. Yakubu, N. S. M. Jamail, A. M. Talib and F. O. Alomary, "Bbas: A blockchain-based authentication system for e-health with multi-factor authentication, access control, and post-quantum security", Scientific Reports (2026).

[39] Li, H., G. Dán and K. Nahrstedt, "Portunes: Privacy-preserving fast authentication for dynamic electric vehicle charging", in "IEEE SmartGridComm", pp. 920–925 (2014).

[40] Li, H., G. Dán and K. Nahrstedt, "Lynx: Authenticated anonymous real-time reporting of electric vehicle information", in "IEEE SmartGridComm", pp. 599–604 (2015).

[41] Libert, B., S. Ling, K. Nguyen and H. Wang, "Lattice-based zero-knowledge arguments for integer relations", in "Annual International Cryptology Conference", pp. 700–732 (Springer, 2018).

[42] Paverd, A., A. Martin and I. Brown, "Modelling and automatically analysing privacy properties for honest-but-curious adversaries", Tech. Rep. (2014).

[43] Rabin, M. O., "Digitalized signatures", Foundations of secure computation **78**, 155–166 (1978).

[44] Santesson, S., M. Myers, R. Ankney, A. Malpani, S. Galperin and C. Adams, "X. 509 internet public key infrastructure online certificate status protocol-ocsp", Tech. rep. (2013).

[45] Saxena, N. and B. J. Choi, "Authentication Scheme for Flexible Charging and Discharging of Mobile Vehicles in the V2G Networks", IEEE Transactions on Information Forensics and Security **11**, 7, 1438–1452 (2016).

[46] Saxena, N., S. Grijalva, V. Chukwuka and A. V. Vasilakos, "Network Security and Privacy Challenges in Smart Vehicle-To-Grid", IEEE Wireless Communications **24**, 4, 88–98 (2017).

[47] Schaub, A., R. Bazin, O. Hasan and L. Brunie, "A trustless privacy-preserving reputation system", in "Proc. IFIP SEC", pp. 398–411 (2016).

[48] Shamir, A., "How to share a secret", Communications of the ACM **22**, 612–613 (1979).

[49] Shamshad, S., S. Belguith and A. Oracevic, "A quantum-secure framework for iod: Strengthening authentication and key-establishment", in "Proceedings of the 20th ACM Asia Conference on Computer and Communications Security", pp. 313–326 (2025).

[50] Soska, K., A. Kwon, N. Christin and S. Devadas, "Beaver: A decentralized anonymous marketplace with secure reputation.", IACR Cryptology ePrint Archive **2016**, 464 (2016).

[51] Wang, X. O., W. Cheng, P. Mohapatra and T. Abdelzaher, "Artsense: Anonymous reputation and trust in participatory sensing", in "2013 Proceedings IEEE INFOCOM", pp. 2517–2525 (IEEE, 2013).

[52] Xu, S., T. Wang, A. Sun, Y. Tong, Z. Ren, R. Zhu and H. H. Song, "Post-quantum anonymous, traceable and linkable authentication scheme based on blockchain for intelligent vehicular transportation systems", IEEE Transactions on Intelligent Transportation Systems **25**, 9, 12108–12119 (2024).

[53] Zhai, E., D. I. Wolinsky, R. Chen, E. Syta, C. Teng and B. Ford, "Anonrep: towards tracking-resistant anonymous reputation", in "Proc. USENIX NSDI", pp. 583–596 (2016).

[54] Zhang, S., Z. Yan, W. Liang, K.-C. Li and B. Di Martino, "Bcae: A blockchain-based cross domain authentication scheme for edge computing", IEEE Internet of Things Journal **11**, 13, 24035–24048 (2024).