Wrocław University of Science and Technology

FIELD OF SCIENCE: Engineering and Technology

DISCIPLINE OF SCIENCE: Information and Communication Technology

DOCTORAL DISSERTATION

Secure Multi-Party Schemes for Intelligent Connected Infrastructures

Mr. Hannes Salin, MSc., Eng.

Supervisor/Supervisors:
D.Eng. hab. Łukasz Krzywiecki

Keywords: ephemeral leakage, cryptography, signatures, key split, pairings, bilinear maps, VANET

WROCŁAW 2024

Secure Multi-Party Schemes for Intelligent Connected Infrastructures

Hannes Salin

The Swedish Transportation Administration
Department of Digitalization and Transport Systems

A thesis submitted for the degree of Doctor of Philosophy under the supervision of D.Eng. hab. Łukasz Krzywiecki at Wrocław University of Science and Technology, Department of Fundamentals of Computer Science

Abstract

This thesis explores the security challenges within connected infrastructures such as cooperative intelligent transport systems where vehicles and infrastructure has the ability to communicate efficiently via both short-range and cellular technologies. The primary focus is on the vulnerabilities of weak devices with poor randomness and limited computational resources. The research is two-fold: firstly, it investigates the security of efficient signature and authentication schemes in multi-node environments by proposing novel cryprographic schemes; secondly, we assess the feasibility and performance of these cryptographic solutions through proof of concept implementations for IoT and low-powered devices. Theoretical and practical approaches are thus used, with formal security proofs and software implementations primarily in Python. Our goal is to bridge the gap between theoretical cryptography and industry implementation, providing secure and feasible solutions for intelligent connected infrastructures, such as vehicular ad-hoc networks.

The research output was five core papers $\mathsf{P_I}$ to $\mathsf{P_V}$, proposing enhanced schemes used in several different use cases. The schemes in papers $\mathsf{P_I}$, $\mathsf{P_{II}}$ and $\mathsf{P_V}$ are proven secure against ephemeral key leakage. In papers $\mathsf{P_I}$ and $\mathsf{P_{II}}$ the original schemes are shown vulnerable in the stronger security models, referred to as cryptanalysis, whereas in paper $\mathsf{P_V}$ we provide a stronger version of cryptanalysis where the original scheme is shown weak under the current security model. We use in particular three different security enhancement techniques: exponentiation, key split and key refresh mechanisms. These techniques are used in papers $\mathsf{P_I}, \mathsf{P_{II}}, \mathsf{P_{III}}$ and $\mathsf{P_V}$. However, in core paper $\mathsf{P_{IV}}$ we also provide a novel scheme for source hiding - particularly important for privacy in connected vehicle systems - in a multi-party environment using standard re-encryption, mixing and ring signature primitives. In addition to the core papers' results, we also provide an additional proof of security for core paper $\mathsf{P_I}$, using the asymmetric pairing setup, whereas the previous proofs are in the symmetric setup. Finally, we also provide as an additional contribution, the performance analysis of all schemes developed and run in laboratory equipment from the Swedish Transport Administration.

Acknowledgments

A seed was planted in my mind during a walk in Macau almost ten years ago, and it further developed in thought, years later at a conference in the United Arab Emirates. Then, only a couple of months before the pandemic struck the world, I began my research journey for real. None of these events that led me from an idea to actually taking action could have happened without Łukasz Krzywiecki—my supervisor, mentor, and friend. This thesis, and all the papers published along this journey, would not have existed without him, and I am grateful for all the encouragement and talks, discussions, mini-seminars, and meetings we have had over the years, which helped me grow my knowledge and understanding of cryptography.

I owe all my work to Julia, who always gave me the space and support during these years when work and family life competed with my research. Working full-time while being a student has been a struggle at times, but with dedication and a supportive family, the last five years have proved to me that anything is possible.

To my parents, my brother, Julia and my beloved children - Alma and Saga - thank you.

List of Publications

This thesis is based on the following papers that were published during the research:

Paper I Krzywiecki, Ł., Salin, H., Panwar, N., Pavlov, M. *Proxy Signcryption Scheme for Vehicle Infrastructure Immune to Randomness Leakage and Setup Attacks*. In 2020 IEEE 19th International Symposium on Network Computing and Applications (NCA) (pp. 1-8). IEEE.

Paper II Krzywiecki, Ł., Salin, H., Jachniak, M. *Certificateless Multi-Party Authenticated Encryption Mitigating Ephemeral Key Leakage*. In 2021 IEEE 20th International Symposium on Network Computing and Applications (NCA) (pp. 1-8). IEEE.

Paper III Krzywiecki, Ł., Salin, H. *Short Signatures via Multiple Hardware Security Modules with Key Splitting in Circuit Breaking Environments*, 2022 IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), Wuhan, China.

Paper IV Salin, H., Krzywiecki, Ł. *A Source Hiding Protocol for Cooperative Intelligent Transportation Systems (C-ITS)*. 18th International Conference, ISPEC 2023, Copenhagen, Denmark, August 24–25, 2023, Proceedings.

Paper V Krzywiecki, Ł., Salin, H. (2022). *How to Design Authenticated Key Exchange for Wearable Devices: Cryptanalysis of AKE for Health Monitoring and Countermeasures via Distinct SMs with Key Split and Refresh*. In: Beresford, A.R., Patra, A., Bellini, E. (eds) Cryptology and Network Security. CANS 2022. Lecture Notes in Computer Science, vol 13641. Springer, Cham.

The ordering is such that papers $\mathsf{P_I}$ to $\mathsf{P_{IV}}$ are directly applicable to transportation use-cases. Paper $\mathsf{P_V}$ used another use-case scenario when published, but the scheme itself is highly applicable in the connected infrastructure context since it relates to authenticated key exchange, very much

needed in the world of connected infrastructure. During the research, a set of additional papers in other topics were published but not included in this thesis:

A B. Drzazga, Ł. Krzywiecki and H. Salin, *Cryptanalysis of Deterministic and Probabilistic Multi-Copy PDP Schemes For Cloud Storage - Attacks and Countermeasures,* 2021 IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), Shenyang, China, 2021, pp. 172-179.

B Ł. Krzywiecki, A. Połubek and H. Salin, *Multi-Signature Scheme Resistant to Randomness Injection Attacks - A Bitcoin Case,* 2021 IEEE 20th International Symposium on Network Computing and Applications (NCA), Boston, MA, USA, 2021, pp. 1-4.

C Salin, H., Fokin, D., Johansson, A. (2022). *Authenticated Multi-proxy Accumulation Schemes for Delegated Membership Proofs.* In: Luo, B., Mosbah, M., Cuppens, F., Ben Othmane, L., Cuppens, N., Kallel, S. (eds) Risks and Security of Internet and Systems. CRiSIS 2021. Lecture Notes in Computer Science, vol 13204. Springer, Cham.

D Salin, H. and Fokin, D. (2022). *An Authenticated Accumulator Scheme for Secure Master Key Access in Microservice Architectures.* In Proceedings of the 7th International Conference on Internet of Things, Big Data and Security - IoTBDS, pages 119-126.

E Salin, H. (2023). *AccA: A Decentralized and Accumulator-Based Authentication and Authorization Architecture for Autonomous IoT in Connected Infrastructures.* In Proceedings of the 8th International Conference on Internet of Things, Big Data and Security - IoTBDS, pages 170-177.

F Salin, H. (2023). *Pseudonym Swapping with Secure Accumulators and Double Diffie-Hellman Rounds in Cooperative Intelligent Transport Systems.* In: Kallel, S., Jmaiel, M., Zulkernine, M., Hadj Kacem, A., Cuppens, F., Cuppens, N. (eds) Risks and Security of Internet and Systems. CRiSIS 2022. Lecture Notes in Computer Science, vol 13857. Springer, Cham.

Contents

Chapter 1

Introduction

1.1 Background

As urbanization and information technologies continue to evolve, the number of devices, entities, and infrastructure that are interconnected is growing rapidly. The Cloud has enabled numerous new efficient applications and solutions for previously challenging problems, such as moving computational tasks to distributed architectures and decentralizing security.

The transportation sector in particular, including railway, road and maritime transportation, has been the subject of extensive research in various cross-functional areas, such as machine learning, wireless communication, and cybersecurity. Therefore, with cloud technology and an increase in collaborative transportation scenarios, the need for security and privacy is more important than ever. Intelligent infrastructure and vehicles, where automation, collaboration, and smart solutions for safety and traffic efficiency, have been the focus of many studies in several research fields. Despite the progress made, the field is still relatively new, especially from a security perspective, and there are still many open questions. Additionally, many ongoing initiatives in intelligent transportation are taking place in Europe [142], but few of them have a specific focus on security and the necessary cryptography, both from theoretical and practical perspectives. Ongoing projects currently utilize trust and privacy using standard public key infrastructures (PKI), but academic research offers a wide range of complementary and innovative approaches such as certificateless and identity-based public key crypto systems [8, 25]. Issues such as secure scalability of trust and leakage resilience in cryptographic protocols are not addressed by traditional PKI, which motivates our pursuit of finding secure and computationally feasible solutions. This thesis therefore addresses the intersection of intelligent transportation and

connected infrastructures, with cybersecurity and cryptography. The main objective is to provide solid theoretical output for academia and practical results that can be considered by the industry. Furthermore, we bridge the gap between theory and practice by providing proof-of-concept implementations and performance evaluations of the theoretical part of the research.

1.2 Research Objective

Cooperative intelligent transport systems (C-ITS) or connected infrastructure ecosystem, comprises a variety of moving and stationary elements, referred to as *nodes*, that interact with each other to facilitate data exchange, coordination, and control of traffic systems (a more thorough description of C-ITS is detailed in Sec. 2.1). Nodes include vehicles, traffic signal devices, roadside units, and control centers, which are each represented by one or several *devices* performing specific functions within the system. These devices can range from simple sensors and actuators to more complex computing systems. A critical aspect to consider is the presence of *weak devices*, which possess poor randomness and limited computational resources. These devices are particularly vulnerable to security threats, as powerful adversaries may exploit their weaknesses to impersonate vehicles or other nodes within the system. To address these security challenges, *multi-party communication* plays a crucial role in facilitating secure data exchange among nodes. This process involves digital signatures and secure protocols for authentication, which ensure that the identity and integrity of the nodes are preserved while preventing unauthorized access and tampering. Moreover, in complex systems with ad-hoc networks there is a need for *proxy* nodes that can support long-range communication and computational tasks for other parties. These proxies may not be fully trusted, thus secure protocols for providing authenticity and integrity of the sent data must be ensured. Additionally to the above mentioned security aspects, we also consider the perspective of *anonymity* between nodes in the connected eco-system.

In order to design and implement cryptographically secure protocols, it is essential to formalize the real-world setting of these complex systems into *formal models*. This allows for a comprehensive analysis of security properties and potential vulnerabilities. Furthermore, evaluating the theoretical work through the implementation of proof of concepts in Internet of Things (IoT) and low-powered devices provides valuable insights into the practicality and effectiveness of the proposed solutions. This study therefore aims to develop secure solutions that effectively address the unique security challenges of these systems while ensuring the safety and efficiency of transportation

networks. Furthermore, we *only use relevant security models where the adversaries are considered as real-world type of attackers in the industrial setting*; in particular the *ephemeral leakage setup* where randomness is leaked or injected by an attacker. The overall objective with the conducted research in this thesis was two-folded; to find efficient and secure signature and authentication schemes in multi-node environments for connected infrastructures, and to investigate feasibility and performance of such cryptographic solutions via proof of concept implementations. The research has focused primarily on two categories of applications:

- Digital signatures, authentication protocols with proxies and anonymity features in intelligent transport systems, i.e. providing secure functionality to vehicles or trains via one or several third-party nodes.
- Mitigation of ephemeral leakage in otherwise secure signature schemes, used in IoT-based connected infrastructures and intelligent transport systems.

For both categories, a theoretical and practical approach was used. For scheme propositions, formal security proofs in the random oracle model would be the primarily used method. From a cryptographic perspective we propose and develop a set of ***security enhancing techniques*** to build our schemes, namely *secure exponentiation, key split* and *key refresh*. We also consider other secure building blocks to provide node *anonymity* in a system, such as re-encryption and mixing combined with ring signatures. The theoretical work has then been evaluated via developed software in Python, WASM and Java for a variety of devices in the published papers. However, an additional contribution to this thesis is the *re-implementation of the protocols in Python, using a low-powered* ***laboratory device*** for accurate simulation of road- and railway specific equipment, provided by the Swedish Transport Administration.

1.2.1 Research Questions

Given the stated objective of this thesis, a number of research questions have been formulated as a basis for our work. They are the following:

RQ1: *How can we mitigate and improve vulnerable signature and authentication schemes subject to weak devices (low-powered and vulnerable against ephemeral injection/leakage), to be used in the connected infrastructure domain?*

RQ2: *What level of feasibility and performance will the improved and proposed schemes have, i.e. are they suitable for real-world implementations?*

We supplement the stated research objective and formulated research questions with high-level hypotheses:

HT1: *The conducted research will contribute to bridge the gap between theoretical cryptography and industry implementation, in the domain of connected infrastructure and vehicles.*

HT2: *The conducted research will provide evidence for the feasibility of novel and strongly secure cryptographical solutions in the domain of connected infrastructure and vehicles.*

Next, we formulate clearly stated goals that our research aim to fulfill. These goals thus connect the research questions and hypothesises:

GL1: *Propose how to mitigate and improve secure schemes that are vulnerable to weak devices and impersonation attacks, and can be used in the domain of connected infrastructure and cooperative intelligent transportation systems.*

GL2: *Implement and carefully evaluate the proposed schemes from a performance perspective, aligned with relevant hardware for the connected infrastructure and cooperative intelligent transportation systems domain.*

GL3: *Address the major security challenges identified in this thesis with applicable and theoretically sound solutions. The specific security challenges are* authenticity of C-ITS data, trust models, impersonation attacks, authentication via proxy *and* leakage resilience*, and denoted* Sec1-Sec5 *respectively. These challenges are further described in Sec. 2.1.5.*

Using the RQ1, RQ2 and HT1, HT2 as the fundamental layer for this thesis, the goal is to traverse the path of the conducted research, resolving the research questions, validating or rejecting the hypothesises and thus reach each of the specified goals GL1-GL3.

1.3 Contribution

The conducted research has provided new insights and knowledge via the published papers in several ways, alongside the additional contributions of new security proofs in the asymmetric configuration, and the re-implementation

of all proposed schemes in provided laboratory equipment from the Swedish Transport Administration. The proposed solutions build upon our defined and utilized *security enhancement techniques* which are cryptographic adjustments in the protocols, hence part of the thesis's contribution. We give a brief introduction to these techniques in the subsequent section and delve in more detail when describing each protocol in the papers $\mathsf{P_I}$-$\mathsf{P_V}$. The reminder of this section specifies the author's individual contributions to the published papers.

1.3.1 Security Enhancing Techniques

We briefly describe the three main security enhancing techniques that were used in the research papers $\mathsf{P_I}, \mathsf{P_{II}}, \mathsf{P_{III}}$ and $\mathsf{P_V}$, to immune the proposed schemes against *randomness leakage attacks*. For some cryptographic schemes, e.g. those based on the Schnorr construction, random values and long-term secret keys mutually mask each other. If that masking forms a linear relation, then randomness leakage could compromise the secret key. Regarding the hardware architecture, we apply those techniques to devices with *Hardware Security Modules* (HSM). This refers to secure parts of the device's architecture, including memory and computation, separated from the regular code execution area. In Sec. 3.1 a more in-depth introduction will be given.

Exponentiation

The method of *exponentiation*, as we call it, can be used in schemes which security is based on the discrete logarithm problem assumption (DLP) in appropriate groups of computation. Here users hold a pair of keys: a secret key sk and the public key $\mathsf{pk} = g^{\mathsf{sk}}$ for a group generator g. For example, in the regular Schnorr signing procedure [143] we generate a random value r, compute $R = g^r$, and $h = \mathcal{H}(m, R)$ for a message m, where $\mathcal{H} : \{0,1\}^* \to \mathbb{Z}_q$ is a secure hash function. Next, compute $s = r + \mathsf{sk} \cdot h$ and return signature $\sigma = (R, s)$. During verification we check if $(R, s) = g^s \cdot \mathsf{pk}^h$. However, if the random value r is leaked to the attacker, then sk could be computed as $(s - r)/h$. To illustrate the idea of the security enhancement with the *exponentiation* technique, we consider a signing device for the *modified Schnorr* signature scheme, equipped with an HSM, where a secret key sk is located and used. But this time, the scheme is set in appropriate pairing-friendly groups, e.g. suitable for a symmetric pairing. Now, instead of returning s as a part of the signature, we return it in the exponent of $S = \hat{g}^{r+\mathsf{sk}\cdot h}$ for some new group generator $\hat{g} = \mathcal{H}_g(m, R)$, where a hash function $\mathcal{H}_g$ maps inputs to

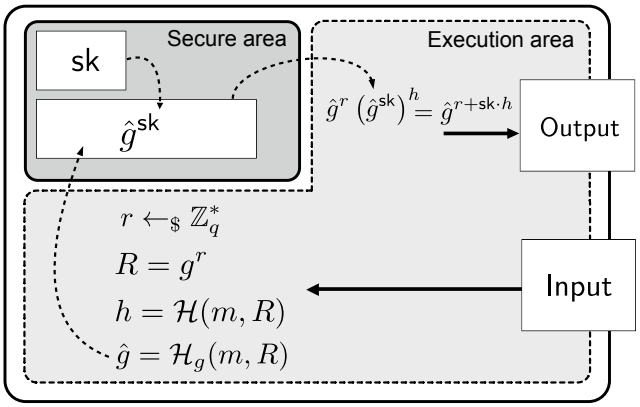

Figure 1.1: Device using the exponentiation technique. The group element $\hat{g}$ is computed by a hash function $\mathcal{H}_g$ that is sent to the secure area as input. The final exponentiation is computed after receiving $\hat{g}^{\mathsf{sk}}$.

group elements. As a result, even if randomness r is leaked, the attacker does not get the secret key sk, but $\hat{g}^{\mathsf{sk}}$, which immune against subsequent forgeries. On the other hand, such a new signature $\sigma = (R, S)$ can be verified using a pairing function: $\hat{e}(S, g) == \hat{e}(\hat{g}, R \cdot \mathsf{pk}^h)$. This is shown in Fig. 1.1. For a detailed description of using this techniques with symmetric pairing in papers $\mathsf{P_I}, \mathsf{P_{II}}$ and $\mathsf{P_{III}}$, see sections 4.2.4, 4.2.4 and 4.3.5 respectively. For the asymmetric solution in paper $\mathsf{P_I}$ specifically see Sec. 4.1.10.

Key Split

The *key split* technique could be used as an additional layer of security by using several HSMs mitigating the possibility of compromising one of them. This method involves dividing a secret key sk into two or more shares, each share being stored separately, thereby reducing the risk of total key compromise from a single point of failure. This can be particularly effective in scenarios where different HSMs comes from different vendors, but not all of them can be equally trusted. The process is shown in Fig. 1.2, where BLS signature [27] with the secret key additive split is computed: $\mathsf{sk} = \mathsf{sk}_1 + \mathsf{sk}_2$. The final signature is computed after being processed by each HSM, namely given σ_1 and σ_2: $\sigma_1 \cdot \sigma_2 = \mathcal{H}(m)^{\mathsf{sk}_1} \cdot \mathcal{H}(m)^{\mathsf{sk}_2} = \mathcal{H}(m)^{\mathsf{sk}_1+\mathsf{sk}_2} = \mathcal{H}(m)^{\mathsf{sk}}$.

Key Refresh

The key split technique could be enhanced further by using the *key refresh* concept. Namely, in each HSM individual shares are updated, per signing session, with new random values based on a secure source of randomness,

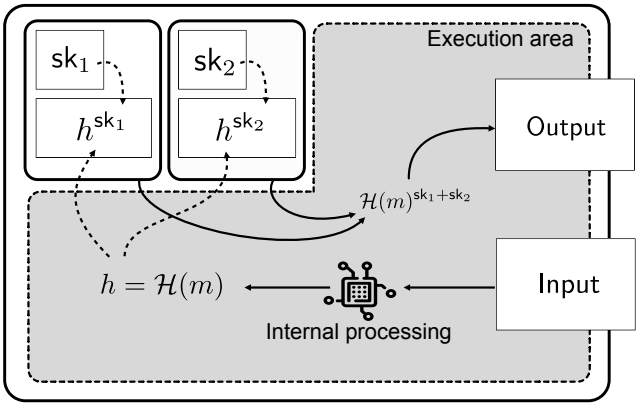

Figure 1.2: Device using key split when computing a signature. Each key part is inside a separate secure area.

preferably sampled from the environment. This helps in mitigating attacks where the adversary could potentially target both HSMs, but tamper only one of them per signing session. The regular refreshment of key components, where the keys are split across different security modules, relies on synchronization that could be based on a shared environmental sampling. As shown

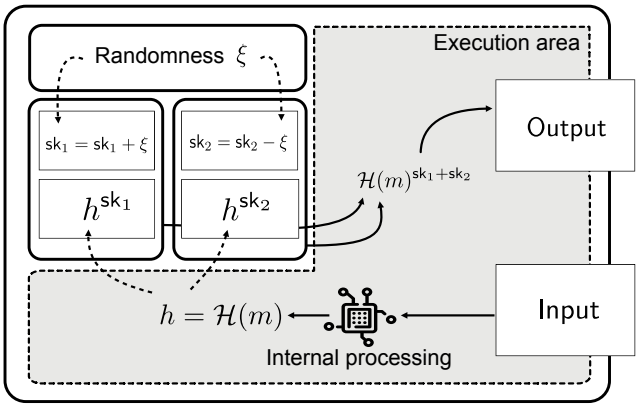

Figure 1.3: Device using key split and refresh when computing a signature. Each key part is inside a separate secure area, and each such area uses a shared secure randomness ξ.

in Fig. 1.3 we depict BLS signatures with split secret shares updated with a fresh value ξ sampled from the environment. The synchronization provides that: $\mathsf{sk}_1 + \xi + (\mathsf{sk}_2 - \xi) = \mathsf{sk}_1 + \mathsf{sk}_2 = \mathsf{sk}$. We silently assume that the attacker is not able to access the environmental randomness ξ, since this would trivialize its attacks.

1.3.2 Specific Contributions of Published Research

Papers $\mathsf{P_I}$-$\mathsf{P_V}$ provides an answer to RQ1 since they show the weakness of previously published schemes, including cryptanalysis in $\mathsf{P_I}, \mathsf{P_{II}}$ and $\mathsf{P_V}$, and introduce new, secure solutions in stronger security models. The feasibility section in this thesis provide clear answers to RQ2. All papers addresses all three goals, where papers $\mathsf{P_I}$ and $\mathsf{P_{II}}$ in particular drives towards GL1 and GL2, whereas papers $\mathsf{P_{III}}$ and $\mathsf{P_{IV}}$ broadens the perspective and addresses security challenges other than ephemeral leakage. We also highlight paper $\mathsf{P_V}$ which during publication did not have a vehicle infrastructure scenario associated, however, the proposed protocols are highly suitable for any connected infrastructure domain that uses low-powered and short-range devices. The main method of security enhancements is the key split and refresh technique, also used in paper $\mathsf{P_{III}}$.

Following is a condensed summary of the publications included in this thesis with individual and overall contributions. We also point out a set of distinct contributions that indicates the fulfillment of the formulated goals GL1-GL3; we denote 5 such contribution points as CPi for $i = 1, ..., 5$ with possible suffix E indicating that the security enhancing technique of exponentiation was used, and KR for key split and refresh. The description of these contribution points are summarized in Tab. 1.1.

Code	Description
CP1	Formulating a novel security model.
CP2	Cryptanalysis breaking a vulnerable scheme, either in a stronger security model or under current security model.
CP3	Propose a novel scheme under a new security model.
CP3-E	Applying exponentiation security enhancement specifically.
CP3-KR	Applying key split and refresh enhancement specifically.
CP4	Security analysis and proof.
CP5	A proof of concept implementation with performance analysis.

Table 1.1: Description of contribution points.

Paper I: Krzywiecki, Ł., **Salin, H.**, Panwar, N., Pavlov, M. *Proxy Signcryption Scheme for Vehicle Infrastructure Immune to Randomness Leakage and Setup Attacks.* In 2020 IEEE 19th International Symposium on Network Computing and Applications (NCA) (pp. 1-8). IEEE.

This paper introduces a new proxy signcryption scheme designed for multi-party settings, with a particular focus on high-traffic environments such as modern railway infrastructure and vehicle-to-vehicle/ infrastructure (V2X) systems. The proposed scheme, built on a double

Schnorr signature approach, is unique in its ability to resist randomness leakage and setup attacks—a major improvement over existing models which assume that initial party procedures are performed in a controlled, secure environment. The key contributions our paper provides are:

- A novel security model for proxy signcryption schemes where ephemeral keys may be leaked at the initiator node or the proxy node.
- The demonstration of vulnerabilities in a typical proxy signcryption scheme, given in a stronger security model, highlighting its inability to withstand ephemeral setup attacks.
- Proposing a new proxy signcryption scheme resistant to such attacks.
- Providing a security proof for the scheme within the proposed stronger security model.
- A performance assessment with a proof of concept implementation of the proposed scheme.

These contributions fulfills contribution points: CP1, CP2, CP3-E, CP4 and CP5.

Individual contribution: Article writing, reviewing, proof verification and conceptualization.

Paper II: Krzywiecki, Ł., **Salin, H**., Jachniak, M. *Certificateless Multi-Party Authenticated Encryption Mitigating Ephemeral Key Leakage*. In 2021 IEEE 20th International Symposium on Network Computing and Applications (NCA) (pp. 1-8). IEEE.

This paper examines the security needs of infrastructure reliant on IoT devices, with a focus on 5G-enabled systems such as the European Rail Traffic Management System (ERTMS). We address 5G security challenges, particularly in the context of Narrowband IoT (NB-IoT), proposing an enhanced multi-party authenticated encryption protocol. Our solution targets key issues such as ephemeral key leakage and vulnerabilities to side-channel attacks. The given environment and ecosystem for our solution aligns well in the novel approach combining certificateless cryptography to overcome traditional PKI scalability issues. The contribution of this paper is the following:

– We introduce a new security model for multi-party authentication, in which ephemeral keys can be exposed or set by an attacker.
– We show that a typical scheme, as proposed in [165], is not secure in the proposed security model.
– We propose two versions of the improved multi-party authentication scheme, asynchronous and synchronous, which are immune to ephemeral key leakage.
– We provide security proofs for our schemes in the proposed stronger security model.
– We provide comparison benchmarks, based on our proof-of-concept implementations.

These contributions fulfills contribution points: CP1, CP2, CP3-E, CP4 and CP5.

Individual contribution: Article writing, reviewing, proof verification, conceptualization of the synchronous proposed scheme, partly implementation, summary and analysis of the proof of concept experiments.

Paper III: Krzywiecki, Ł., **Salin, H**. *Short Signatures via Multiple Hardware Security Modules with Key Splitting in Circuit Breaking Environments*, 2022 IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), Wuhan, China.

This paper introduces the notion of *circuit breaking environments* where devices that collaborate, e.g. in multisignature procedures, are continuously synchronized, and with any interruption a complete desynchronization occurs. Moreover, we introduce a key-split scheme for the secret keys within the devices' hardware security modules, enabling a secure non-leakable signature scheme. This technique is useful for connected train carriages transporting highly sensitive cargo which may need continuous verification. In the given environment, our scheme provides a secure mechanism that would trigger any tampering with the target cargo if the devices are connected between carriage and cargo. The key contributions are:

– Proposing a BLS-like signature scheme with additive secret key split over two hardware security modules. augmented with a synchronous refreshment mechanisms.

– Proving the scheme secure within a strong adversary model, which allows for multiple partial secret leakages, providing that one leakage occurs in one chosen hardware security module per signing.
– Propose a generalization of the refresh mechanism for regular BLS multi-signatures with two distinct devices, each having one hardware security module for the refreshed secret signing key.

These contributions fulfills contribution points: CP1, CP3-KR, CP4 and CP5.

Individual contribution: Article writing, reviewing, proof verification, scheme conceptualization, implementation of the software and experiments.

Paper IV: **Salin, H.**, Krzywiecki, Ł. *A Source Hiding Protocol for Cooperative Intelligent Transportation Systems (C-ITS).* 18th International Conference, ISPEC 2023, Copenhagen, Denmark, August 24–25, 2023, Proceedings.

This paper presents a novel protocol for the secure, source-hiding delivery of ring signatures from a group of users to a server, immune to routing path tracing and ring signature de-anonymization via source address attacks. The proposed protocol, based on a secure encryption scheme and an unforgeable anonymous ring signature scheme, is modular and does not rely on an additional trusted third party. Furthermore, the study introduces two versions of the protocol: a basic scheme designed for the honest-but-curious adversary model, and an extended version that can withstand malicious users attempting to replace signed messages with forged ones. The key contributions our paper provides are:

– We propose a *source-hiding* protocol SHP immune to routing path tracing and ring signature de-anonymization via source address attacks.
– The proposed scheme does not require an additional *trusted-third-party* and is entirely self-contained within the group of registered participants.
– We propose two versions of the SHP scheme: the basic scheme π_{SHP} secure in the *honest-but-curious* adversary model, and the extended version π_{SHP2} secure against malicious users trying to substitute signed messages with a forged ones.

– We perform a benchmark performance analysis of each scheme, from a proof of concept implementation in Python.

These contributions fulfills contribution points: CP1, CP3, CP4 and CP5.

Individual contribution: Article writing, reviewing, proof verification, implementation of the software and experiments.

Paper V: Krzywiecki, Ł., **Salin, H**. (2022). *How to Design Authenticated Key Exchange for Wearable Devices: Cryptanalysis of AKE for Health Monitoring and Countermeasures via Distinct SMs with Key Split and Refresh.* In: Beresford, A.R., Patra, A., Bellini, E. (eds) Cryptology and Network Security. CANS 2022. Lecture Notes in Computer Science, vol 13641. Springer, Cham.

This paper presents a cryptanalysis of an authenticated key exchange (AKE) protocol tailored for wearable devices, leading to the development of a stronger variant of the SIGMA protocol. This enhanced protocol incorporates robust signature blocks and leverages an additional out-of-bound channel, alongside a novel signature scheme that divides the signing key additively and introduces a key split and refresh mechanism. The security analysis is based on the assumption that split key components reside within separate security hardware modules in the signing device, offering protection against key compromise. Our comprehensive security assessment is framed within an augmented Canetti-Krawczyk (CK) model, which factors in the splitting and periodic renewal of long-term secret keys across dual signature security modules.

The key contributions our paper provides are:

– We successfully provide cryptanalysis on the AKE scheme proposed in [163].

– We propose a generic solution to mitigate similar attacks, based on provable secure modular AKE protocols augmented with a long-term secret key split and refresh mechanism, stored in two distinct security modules.

– We propose a stronger modified Canetti-Krawczyk (mCK) model, in which the adversary can issue additional type of queries per session, called "Partial-Key-Reveal", which returns computation results from both security modules.

– We propose a modified SIGMA based on the regular SIGMA from [36], and on the Schnorr signatures with additive secret key split and refreshment from [130].
– We provide a formal security analysis of our schenme in our strengthen (mCK) model.

These contributions fulfills contribution points: CP1, CP2, CP3, CP3-KR and CP4.

Individual contribution: Article writing, reviewing, cryptanalysis, proof verification.

	CP1	CP2	CP3	CP4	CP5
Paper I	✓	✓	E	✓	✓
Paper II	✓	✓	E	✓	✓
Paper III	✓		KR	✓	✓
Paper IV	✓		✓	✓	✓
Paper V	✓	✓	KR	✓	

Table 1.2: Summary of the scientific contributions addressed in all core papers.

I have summarized my main contributions in the papers described above in terms of contribution points, and the additional research conducted as part of this thesis in Tab. 1.2. Moreover, a complementary depiction of the contribution is shown in Tab. 1.3. The symbol ● means that I was the main contributor but not necessarily the only one, ◐ that I was partly contributing together with at least one other co-author, and ○ that I did not contribute at all on the specified part. Finally, in general the conducted research has consisted of regular video conference meetings via Zoom with the supervisor and co-authors since 2020 up to the finalization of the thesis, but also scheduled on-site meetings both in Wroclaw and at conference venues. All papers in this thesis has been presented by the author at each conference, including preparing posters, presentation slides and video recordings.

1.4 Limitations

Given the research objectives described in Sec. 1.2, we narrow the scope as follows: algorithmic optimizations for the proposed schemes are not considered, neither the network complexity that may arise in different scenarios.

	Writing	Concept	Imp1	Imp2	Security analysis
Paper I	◐	◐	○	●	◐†
Paper II	◐	◐	◐	●	◐
Paper III	●	●	●	●	◐
Paper IV	●	●	●	●	◐
Paper V	●	◐	○	●	◐

Table 1.3: Summary of the author's contribution to the core papers of this thesis. **Imp1** refers to implementation for the conference paper for various devices and programming languages, and **Imp2** refers to full implementation by the author in Python for the laboratory device, specifically for this thesis as an additional contribution. † Additional proof by the author is provided specifically for this thesis as contribution,

The focus is on cryptographic solutions and computational feasibility evaluation of the proposed schemes and architectures. The primary implementation scope is via proof of concept software, illustrating the core procedures of the proposed schemes, and not sub-optimizing code. Moreover, we are not considering different type of security models than the explicitly formulated ones in the core papers. This implies that any other security model not outlined in this research may lead to theoretical vulnerabilities under different type of adversaries. This is however expected and natural given provable security best practices.

1.5 Thesis Outline

The thesis structure is divided into four main chapters together with an introductory chapter and a conclusion chapter, as follows:

Introduction This chapter introduces the aim, goals and research scope. The chapter consists of the following sections:

- Section 1.1: Background and introduction to the thesis subject.
- Section 1.2: Research objective, goals and research questions.
- Section 1.3: Summary of contribution.
- Section 1.4: Limitations of the research.
- Section 1.5: This section.

Real-world Setting This chapter introduces relevant background, security challenges, use cases and system settings from an industry point of view. We introduce problems to be addressed via scenarios, and identified vulnerabilities described from an industry setting with real-world devices, architectures and systems. The chapter consists of the following sections:

- Section 2.1: Introduction of cooperative intelligent transport systems technology, security challenges and scenarios, including system architectures.
- Section 2.2: Related work in terms of authentication and signatures in connected infrastructures, and other relevant cryptography and technology areas.

Formal Setting In this chapter we give a formal and mathematical description of the system settings and the challenges. From this chapter we thus bridge the industrial perspective into a mathematical description to be used for security analysis. The chapter consists of the following sections:

- Section 3.1: Introduces the necessary mathematical preliminaries.
- Section 3.2: Introduces the underlying cryptography theory used in our research.
- Section 3.3: Introduces the cryptographic setting where the theory is applied and security models.
- Section 3.4: Provides a walk-through of the general proving techniques the be used later in the thesis.
- Section 3.5: Introduces the industrial setting, showcasing how the formal models are used in real-world scenarios.

Solutions In this chapter we introduce the produced research including security analysis and performance evaluations. These solutions are linked to previously described challenges, and further detailed how they solve them. The chapter consists of the following sections:

- Section 4.1-4.5: Published research, recalling the four core papers of the thesis, describing scenarios $\mathsf{P_I}$-$\mathsf{P_V}$ together with the proposed schemes. This chapter also includes new contributions

that has not been published; security proofs in the asymmetric configuration for paper $\mathsf{P_I}$.

Implementation This chapter is an additional contribution except the core papers $\mathsf{P_I} - \mathsf{P_V}$ by providing proof of concept implementations for all schemes but on a provided laboratory equipment.

- Section 5.1: This section includes description of the experiments, practical considerations and equipment analysis.
- Section 5.2: Provides the complexity and performance analysis of the proof of concept implementations, including feasibility analysis.

Conclusion This chapter summarizes the thesis, describes future work and make concluding remarks that leads answering the formulated reserch questions.

1.5.1 Relations

In Fig.1.4 we highlight how the identified security challenges drive the formulation of research questions RQ1 and RQ2, which in turn makes us to formulate the hypotheses. These are then rejected or accepted by providing the contributions of our research, leading to either fulfilling the goals GL1, GL2, GL3 or not. Note that we view hypotheses HT1 and HT2 in a meta perspective, and not in the traditional statistical sense, meaning that we use them for better clarity in communicating our research. In Fig. 1.5 we depict how each core paper of the thesis relates to each other and to what challenges they contribute solutions for. Each paper concerns primarily connected vehicle- and railway infrastructures, where ad-hoc networking is the basis.

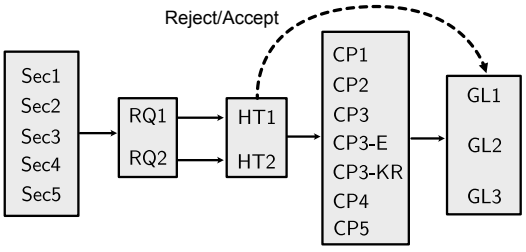

Figure 1.4: Relation between security challenges, research questions, hypotheses, contributions and goals of the research.

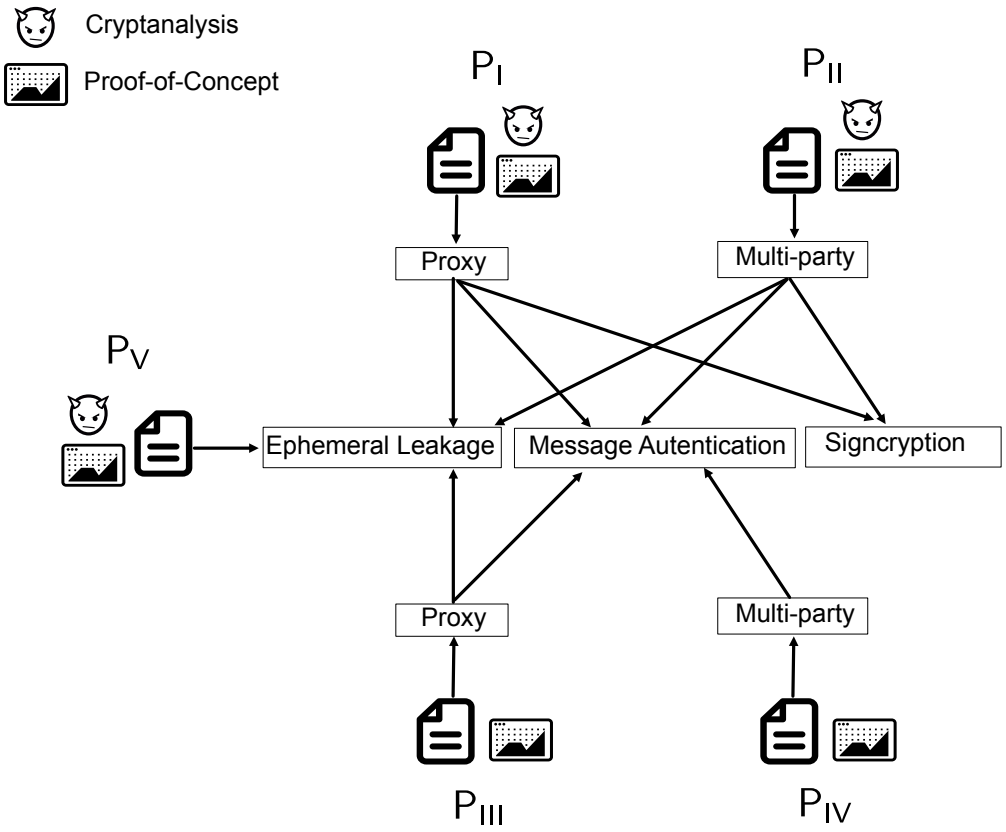

Figure 1.5: Core paper descriptions, where each paper is marked if it contains a cryptanalysis and/or a proof of concept implementation.

Chapter 2

Real-World Systems

2.1 The Domain of Cooperative Intelligent Transport Systems

C-ITS is the collection of technologies that utilizes communication and information exchange between vehicles, road- and railway infrastructures, and other users to enhance the safety, efficiency, and sustainability of transportation systems [44, 4]. C-ITS therefore use different types of communication technology stacks, such as cellular networks, wifi, and Dedicated Short Range Communication (DSRC). These technologies enable vehicles, trains and different type of connected infrastructure to exchange data in real-time. The data can then be used to support a wide range of applications, such as collision warning systems, traffic management systems, monitoring and emergency response systems. C-ITS thus have the potential to significantly improve both the safety and efficiency of transportation eco-systems by providing real-time information to drivers and other road users, and by enabling more efficient and coordinated use of the road network [112]. Moreover, C-ITS can also be used for reducing fuel consumption and emissions, and to support the integration of electric and autonomous vehicles into the transportation system. Therefore, C-ITS technologies are being developed and deployed in several countries around the world, and is expected to play a significant role in the future of transportation [144]. The number of C-ITS initiatives and proof of concept implementations increases in both the US and in Europe, however several technical challenges remain such as C-ITS interoperability and harmonization [90]. C-ITS typically involves a range of connected nodes that can potentially collaborate. Some of the key entities that are involved in a roadside C-ITS system include connected infrastructure such as traffic lights, electric road signs, sensors, cameras, and

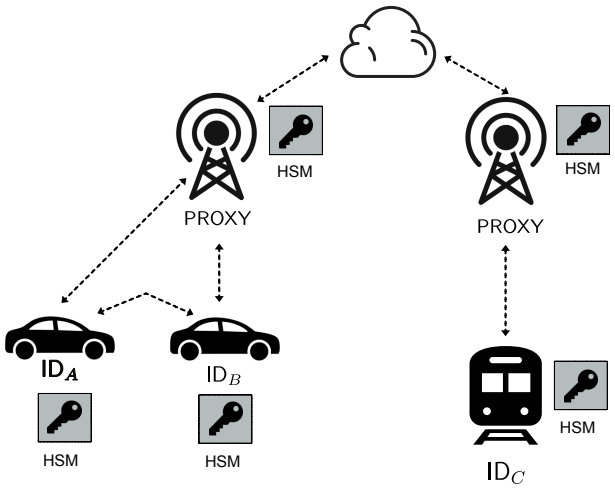

Figure 2.1: Connected infrastructure across modes of transport, using proxies and cloud connectivity. Each node's secret key is stored in a hardware security module named HSM.

other devices that are located along the road infrastructure (typically in tunnels, bridges and conjunctions in urban areas). Similarly, for railroad systems there are railside infrastructure components such as balises, sensors and cameras. These are equipped with communication devices that allow data exchange between vehicles and other infrastructure entities. A Roadside (or railside) Unit (RSU) is a specific device that provides communication services to vehicles (or train carriages) within a C-ITS, but is stationary in the infrastructure. it can be servers, relay stations but also sensor equipment that also provides relay functions. The vehicles in turn has Onboard Units (OBU), which are devices installed in the vehicle and is used to communicate with other vehicles and RSUs. RSUs and OBUs are typically equipped with wireless communication technologies, such as DSRC or other type of short-range technologies, but also cellular communication. Such technology is often referred to as V2X (vehicle-to-anything). RSUs can be used to provide a variety of services to vehicles, including traffic management, safety alerts, and location-based services. They can also be used to collect data from vehicles and other infrastructure entities such as proxy servers or base stations, to support traffic analysis functions. For railway scenarios the primary functionality is more on interoperability services and data collection for maintenance services of tracks. An overview is shown in Fig. 2.1 where both road and railway infrastructures can be inter-connected using proxies.

For road and vehicle systems, the involved nodes can be setup in a Vehicular Ad-Hoc Network (VANET), which is a type of wireless communication

network that is designed to enable communication between vehicles and infrastructure dynamically.

2.1.1 Safety Concerns and Accident Reduction

The number of traffic fatalities in Europe has fallen significantly since the early 2000's, however the current trend is not aligned with the European goal of 2030 [55]. As shown in Fig. 2.2 the number of traffic fatalities are slightly above the target goal. The corresponding railway fatality statistics shows a similar trend, with decreasing deaths between 2010 and 2020, but then a slight increase (808 reported deaths in 2022) according to Eurostat [61].

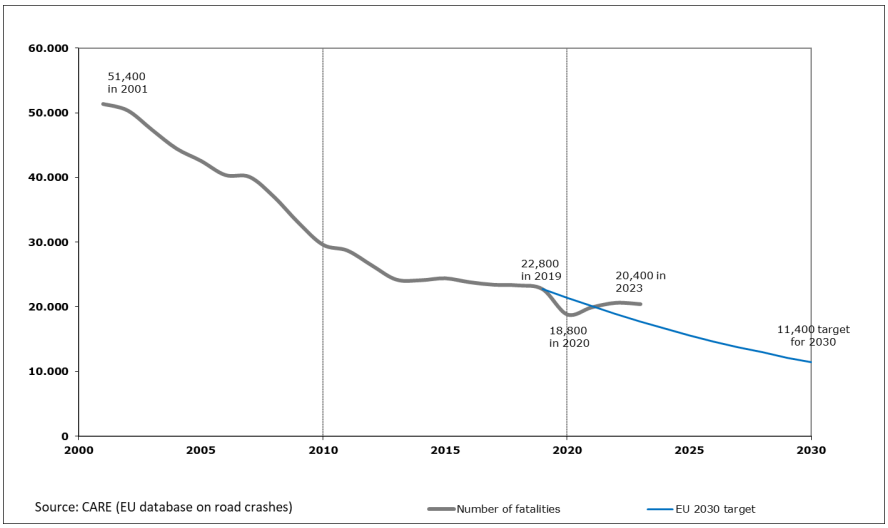

Figure 2.2: Road safety statistics 2023 in EU [55]

There are many factors that influence traffic safety, where technology is one of the more significant ones that can be used for mitigation. In urban environments the function of traffic management is highly important for avoiding traffic congestion, inform vehicles and constantly monitor the infrastructure for hazards and accidents. It is also well established that C-ITS has an important role in terms of safety, where technology and connected infrastructure combined with intelligent vehicles can indeed save lives [89, 146, 38], even the Swedish Transport Administration's latest report on connected infrastructure points out that one of four prioritized areas for development is traffic safety [142]. C-ITS and connected infrastructure are definitely important components in reducing accidents and avoiding traffic fatalities. Therefore, providing robust technological solutions in such critical infrastructure must then have adequate security mechanisms, since any

attack or faulty system may lead to catastrophe - injury, fatality and huge societal costs. It is thus not only important to mitigate active attackers that tries to inject predictable randomness or stealing leaking keys from vehicles and infrastructure devices, but also consider vulnerable infrastructures due to indeliberate events such as environmental catastrophes, malfunctioning equipment and more.

2.1.2 Swedish Transport Administration and C-ITS

The Swedish Transport Administration (STA) is a government agency in Sweden responsible for the planning, construction, operation, and maintenance of the national transport system [5]. STA oversees and is responsible for enabling various forms of transportation, including roads, railways, maritime transport, and aviation. As of 2022, STA reported it is involved in 81 different ITS initiatives at both the national and international level [142]. Many of these initiatives are/were centered on the efficient use of road-, traffic-, and travel data and maintaining consistency in ITS services for traffic management. While many of the initiatives in STA's report tackle traffic safety from various angles, none of them specifically address cybersecurity, privacy, or data protection. Consequently, there is a significant deficiency of cybersecurity research in current C-ITS initiatives from a Swedish (and potentially Scandinavian) standpoint. Moreover, European directives and international standards such as the Network and Information Security (NIS) [56] and ISO 62443 [42], has been developed further to strengthen the implementation of cybersecurity in general for infrastructure-related areas such as operational technology (OT). Thus, it is crucial to find ways to integrate security and cryptography research with industry C-ITS initiatives. This thesis endeavors to bridge that gap.

Intelligent and Connected Railway Infrastructure

STA is heavily involved in the European Rail Traffic Management System (ERTMS) which is an initiative led by the EU, aiming to harmonize and modernize the train control and communication systems across Europe [59]. The goal is to create an interoperable railway system that facilitates cross-border train operations, with improved safety and increased efficiency of railway transportation. The European Train Control System (ETCS) and the Global System for Mobile Communications-Railway (GSM-R) are two major technology components in ERTMS. Current discussions in the international union of railways regarding the successor to GSM-R has not yet been decided, however one promising candidate is the Future Railway Mobile Communi-

cation System (FRMCS) [81]. Therefore, stakeholders such as STA need to be involved and follow the progress of technology- and standardization work closely fur future implementation.

Shift2Rail is a European research and innovation project aimed at accelerating both the development and deployment of new technologies, solutions, and services in the railway sector [60]. Launched in 2014, this project is a public-private partnership that brings together the EU, railway industry stakeholders, research institutions, and small and medium-sized enterprises. The project focuses on several key areas, such as advanced traffic management and control systems, intelligent freight and passenger transport, and the enhancement of the use of digital technologies to improve the management of railway operations. STA is both part of and closely observing the development of the Shift2Rail project as well.

Intelligent and Connected Road Infrastructure

STA is a key participant and collaborator in multiple Scandinavian and European C-ITS projects, including the recently completed NordicWay initiative [132] and C-ROADS [31]. The NordicWay project is a research and development initiative aimed at advancing C-ITS in the Nordic countries and is being carried out by a consortium of partners from academia, industry, and government organizations in the Nordic region. The main goal of the project is to demonstrate the benefits and potential of C-ITS technologies, such as ITS-G5 short-range V2X communication. It also aims to encourage the market for the deployment and uptake of these technologies in the Nordic region. To achieve this goal, the project is conducting a number of activities, including the development of new C-ITS applications and services, the testing and evaluation of these applications and services in real-world environments, and the dissemination of the results and findings of the project to a wider audience.

The National Access Point Coordination Organisation for Europe (NAPCORE) is another important initiative, formed to coordinate and harmonise several mobility data platforms across Europe [124]. The ITS Directive of EU together with a set of delegated regulations forms the basis of the creation and development of this initiative [124]. A data platform has been developed and is called the National Access Point (NAP) which is a node where ITS-related data is collected and further published in the infrastructure. These NAP nodes are thus crucial in enabling C-ITS services since data sharing and coordination is fundamental to almost all use cases. Moreover, the adoption of cloud technology, proxy solutions and decentralization of services are tested in the overall architectural view in these type of initiatives.

2.1.3 C-ITS Technology

One of the key technologies used in C-ITS in the European context, is defined in ETSI ITS-G5 [57], which is a European standard for wireless communication between vehicles and roadside infrastructure based on the IEEE 1609 series and IEEE 802.11p standards [3, 1]. ITS-G5 thus refers to a wireless short-range technology stack. The IEEE 802.11p standard operates in the 5.9 GHz frequency band and defines the physical and medium access control layers of the communication protocol. One of the key features of 802.11p is its ability to support safety-critical applications, such as collision avoidance, emergency vehicle notification, and cooperative driving. Moreover, to ensure reliable communication, mechanisms for prioritizing safety-critical messages, as well as error detection and correction techniques, are included.

Another important technology for C-ITS are cellular V2X technologies, which has the ability to integrate with 4G and 5G cellular networks [40]. These are widely available in Europe and can provide high data rates and low latency. Cellular V2X is being developed and promoted by industry groups such as the 3rd Generation Partnership Project and the 5G Automotive Association. This type of technology is expected to play a key role in the future of connected and autonomous vehicles, as well as in the development of smart cities and intelligent transportation systems; however, comparative studies have shown that short-range ITS-G5 technology still have benefits over long-range cellular technology [121]. Regardless, C-ITS is a mix of different standards and technologies that has not yet harmonized on a European level.

2.1.4 Laws and Regulations for C-ITS Security

EU directives are legislative acts aimed at harmonizing laws and regulations across its member states. Once a directive is issued by the EU, each member state is required to create national laws that align with the objectives and guidelines set forth in the directive.

The growing number of vehicles on European roads and the need for greater mobility among citizens place significant pressure on the current infrastructure, and contribute to higher energy consumption, resulting in environmental concerns. In order to tackle these issues, the EU has decided to develop and employ a regulated and standardized framework for ITS and similar services. Hence, the European Commission has therefore formulated a directive to its member states, specifically for the ITS domain, namely the *ITS Directive (2010/40/EU)* [44]. This directive is also aimed towards the harmonization and increased adoption of ITS for roadside infrastructures,

but also including the conjunction with other types of traffic types. Some key points in these focus areas are to maximizing the utilization of road-, traffic-, and travel information, ensuring seamless ITS services for traffic management and freight transportation. Moreover, to also enhance traffic safety and secure transport by integrating vehicles with (connected) transport infrastructure.

2.1.5 Security Challenges

Although extensive research has been conducted for C-ITS, there are still technical challenges related to security that has not yet been fully addressed. This section provides a brief overview of such challenges and current issues, both from the perspective of the industry, government sector and academia. These challenges are used to motivate our research further and highlight the pressing need for secure C-ITS solutions in practice.

Security in C-ITS primarily concerns the safeguarding of connected infrastructure, traffic flows, vehicle safety and reliable transportation [20]. Therefore, a C-ITS is considered a highly sensitive environment that requires protection due to the significant involvement of people's safety and privacy. Furthermore, the technology stacks for wireless communications and IoT/OT devices evolves rapidly, making the security aspect highly dynamic and thus require constant adaptation [76].

From both a government and industry standpoint, there is a significant incentive to address security and privacy concerns as the widespread adoption of new technology is dependent on citizens' acceptance. Technical specifications and standards have been proposed [58, 82], as well as government policy papers to support progress in research and development of open security issues within C-ITS [126]. Also, the Conference of European Directors of Roads (CEDR) announced in 2022 a call for proposals regarding trust models in C-ITS [43], and Shift2Rail have recently conducted cybersecurity projects for smart and connected railway [60]. However, from a technology readiness level it seems that cybersecurity components of C-ITS is lagging behind; we have communication technology and efficient devices, but secure protocols are still investigated. To conclude, these standardization initiatives, European-funded projects, and industry research have collectively emphasized the importance of addressing a wide range of cybersecurity challenges in the C-ITS domain. These challenges need to be addressed in order to ensure secure deployment of connected infrastructure solutions.

There is no standardized, general definition of cybersecurity. In this thesis, the term refers to activities required to protect network- and information systems, their users, and other individuals or devices affected by cyber

threats. These incidents may be intentional or unintentional and may include the disclosure of sensitive data, attacks against devices or critical infrastructure, and theft of personal data. This definition captures a relevant subset of the European Union Agency for Cybersecurity's (ENISA) definition towards ITS [2]. Next, much of cybersecurity relies on cryptography, i.e. cryptography is an essential component as it provides a mechanism for securing data and communications. The use of cryptography allows sensitive information to be encrypted and decrypted securely, authenticate the identity of users/devices, and to provide non-repudiation, meaning that the sender of a message cannot later deny having sent it. These are only a few examples since the field of cryptography is broad and contains many different techniques. Overall, much the security we are looking for in everyday use can be reduced to cryptography and mathematics. For the purposes of this thesis and in the context of C-ITS, security solutions will be primarily referred to as *protocols* or *schemes*. These are sets of procedures or algorithms that use cryptographic primitives (i.e. building blocks) to provide security measures. These terms will be used interchangeably. In some cases we also refer to secure architecture, meaning the devices and/or primitives used in protocols, that together constitutes a secure solution.

We summarize relevant security challenges for C-ITS and connected infrastructure that has strong requirements for cryptography-based solutions, as identified by the industry and academia, in Tab. 2.1. We also note a specific challenge - *leakage resilience* - found as an additional identified challenge, but not frequently mentioned in the connected infrastructure and C-ITS cryptography literature. However, several secure schemes that can be used in that domain falls into a category of being vulnerable to ephemeral leakage attacks [92, 99]. Hence, implicitly these types of schemes are also part of the challenges the C-ITS domain is facing. Each challenge is linked to the research outcome of this thesis, showing which papers address which challenge. The challenges are discussed in more detail in the subsequent sections.

Security Challenge	**Source**	**Article**
Sec1: Authenticity of C-ITS Data	[126, 70, 144, 10, 103, 116]	P_I, P_{II}, P_{III},P_{IV}
Sec2: Trust Models	[43, 32, 147]	P_{II}
Sec3: Impersonation Attacks	[144, 10, 72, 103, 116]	P_I,P_{II}, P_{III},P_{IV}
Sec4: Authentication via Proxy	[110, 162, 149]	P_I
Sec5: Leakage Resilience	[92, 99]	P_I,P_{II},P_{III}

Table 2.1: Summary of industry, government and academic research that indicates cybersecurity challenges for connected infrastructure and C-ITS, and a mapping of our papers in our research that addresses each challenge.

Challenge 1: Authenticity and Privacy of C-ITS Data

C-ITS and smart vehicle technology have the potential to generate both more and reliable data as ever before, thus implying a need for data privacy [125]. Not only for future scenarios, but current data flows in most C-ITS architectures relies on authenticated data [58] between vehicles and stationary nodes. Moreover, data that flows through intermediate or third-party nodes in the system, different privacy and authenticity requirements may apply to the different nodes handling the data. It is not clear how either government or third party companies can or should access subsets of (sensitive) data such as location-, bio-metrical- or safety related data [125]. In particular, it is of high importance to ensure authenticity of any type of messages for safety purposes, otherwise collisions or the injection of malicious data could occur between vehicles and infrastructure [144]. For this reason, secure digital signature schemes must be the foundation of a secure C-ITS architecture, and the challenge is to construct protocols that ensure authenticity and privacy of the user data, that also are provably secure.

Challenge 2: Trust models for C-ITS

We underline that a trust model here is referred to as the technological solution for providing trust in a system, such as PKI. There is interest in investigating non-standard trust models, such as certificateless cryptography [43, 32, 147], due to the scalability challenge of key management in traditional PKI. Also, certificate-based systems such as PKI need to manage the trust hierarchies with certificates. In certificateless crypto systems a semi-trusted third party, referred to as the *Key Generation Center* (KGC) issues partial private keys. Each user generates their complete private key by combining a random, secret value with their public value. The end result is then solving the key escrow problem and the user's private key is not stored within the KGC. There is an increase of research papers in this field directed towards C-ITS, but more pilots and proof-of-concepts are likely needed before the industry will adopt these non-standard trust models on a large scale [140]. Therefore, this challenge is two-folded: to investigate the feasibility and security of certificateless solutions of trust and data authenticity in C-ITS, and to help driving the research field and industry into evaluating the technology further.

Challenge 3: Impersonation Attacks

Since a typical C-ITS eco-system contains multiple nodes with identities, also in the form of pseudonyms [70, 10], there are incentives for an adversary

to impersonate a node in order to either steal, capture or inject data [103]. This type of privacy can also be subdivided into *location privacy* and *identity privacy* [70], where both areas aim for protecting a vehicle's or driver's identity in a VANET. Man-in-The-Middle attacks (MiTM) also falls under this category of attacks since it compromises a user's identity in a system.

This challenge, of establish secure protocols where participants cannot be impersonated, is a central topic in this thesis. This challenge is also tightly connected to all other challenges where the data needs to be authenticated in various ways, e.g., from leaked keys or exploits in otherwise secure protocols.

Challenge 4: Authentication and Signatures via Proxy

In dynamic and ad-hoc networks such as VANET:s, there might be cases where cooperative computations are needed, also with the property of ensuring the privacy of each participant. Moreover, when sending messages over larger geographical areas there might be necessary to make use of one or several proxies for relaying messages, and where cellular technologies does not suffice. These proxies can be other vehicles or stationary units in the infrastructure. In any of these cases, we can also consider that the proxy can receive a delegation on behalf of the original sender, thus new secure protocols are needed. The challenge is to construct suitable signature schemes and protocols that provides security both when using un-trusted or delegated proxies, and still being efficient enough for transport scenarios.

Challenge 5: Leakage Resilience

Many studies in C-ITS underlines the importance of mitigating information leakage and privacy preservation, e.g. comprehensive studies such as [144] and [103]. However, not much of the current literature in C-ITS points specifically to secret key leakage attacks, where a party may leak whole or parts of the ephemeral keys, e.g. due to faulty pseudo-random generators or tampered hardware devices for cryptographic computations [49, 50, 87]. This threat put requirements on both the algorithmic part, as well on the hardware the devices use. From a cryptography perspective we need to ensure that the protocols use primitives secure against such leakage (or conversely, vulnerable to randomness injection). In summary, in a vehicle or infrastructure node where there might be several connected devices from different vendors, where some of them may be vulnerable to leakage attacks, this would pose a dangerous threat to both the vehicle and the dynamic network it participates in. Also, in the current standardization work for C-ITS there is no specific part that focus on key leakage resilience. This would be a potential issue

since it heavily relates to the harmonization and interoperability of C-ITS, i.e. participants from different areas or environments that need to collaborate implies an agreement of algorithms and protocols. This is therefore considered an open problem in this thesis, i.e. to coordinate and standardize this area of leakage resilience in C-ITS.

2.1.6 System Architecture and Scenarios

To illustrate the security challenges in a suitable context, we consider a C-ITS setup with a VANET and two connected vehicles ID_A and ID_B. In general, each participant ID_i holds a key pair with private and public keys $(\mathsf{sk}_i, \mathsf{pk}_i)$. These can be used for both encryption and signatures, but may also be complemented with additional temporary keys (session keys). Authentication and signatures can also be computed via a proxy P. This proxy could act as a KGC in scenarios with certificateless solutions, or it may act as a delegated signer. The proxy also generates and utilizes a key pair $(\mathsf{sk}_P, \mathsf{pk}_P)$ including any potential session- or delegate keys. We refer to any data that is transferred between nodes as *messages*; typically ID_A need to send an authenticated message m to ID_B or a proxy P. The procedure to securely transmit m and a signature σ over that message is done using a scheme. Depending on the scheme's construction there might be some interaction between the participating nodes, e.g., the initiating node - the *initiator* - might need to send and receive several messages with the responding node - the *responder*. In that case, the initiator is the node that starts the scheme. The initiator might also be the *prover* of the authenticity of the message, i.e., proving that σ is the corresponding signature. There is also a *verifier* of the authenticated message, which typically is the responder. However, there can also be third parties that only have an interest in verifying messages without participating in the interactive parts of the scheme. An adversary $\mathcal{A}$ may then mount several different attacks; central for this thesis are the impersonation- and ephemeral key leakage attacks. A simple depiction of this overview is given in Fig. 2.3. We will expand Fig. 2.3 into a set of scenarios, addressing each challenge, thus getting a series of use cases. For simplicity, we will use vehicle C-ITS architectures in the following depictions.

Scenario P_I: Ephemeral Leakage Mitigation Using Signcryption in a Proxy Environment

We consider a scenario in an environment with a connected infrastructure such as a VANET, but also trans-transportation environments where vehicle and railway infrastructures could be interconnected. Typically in such envi-

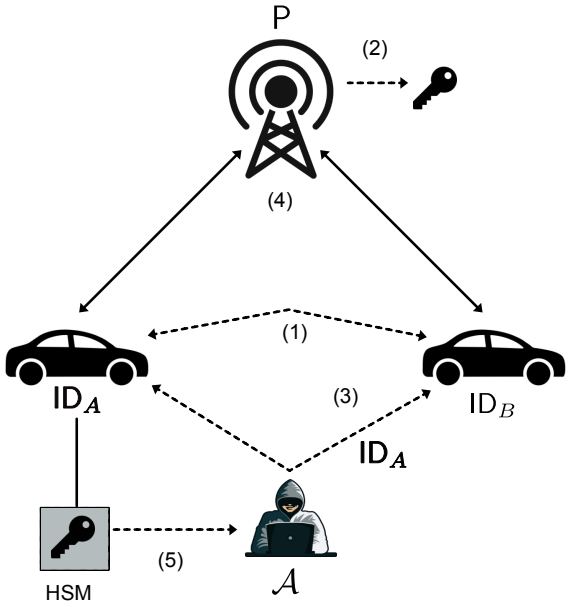

Figure 2.3: Security challenges, (1) data authenticity, (2) non-standard trust model, (3) impersonation attacks, (4) authentication via proxy and (5) ephemeral leakage attacks. The HSM is a secure memory area in the vehicle where secret keys are stored.

ronments, there are several type of sensors and operational technology, e.g., cameras, balises and radar equipment. These devices collects and send data to traffic centrals to inform status of the infrastructure. However, due to long distances and possible disturbances, these devices need to rely their data via proxy servers, to reach the traffic central. Therefore, the need for a secure protocol to ensure the data is both signed and encrypted when sending via a (delegated) proxy, is high. We can regard a device as an initiator $\mathsf{ID}_{\mathcal{I}}$ that needs to send a signed and secure message m to a receiver $\mathsf{ID}_{\mathcal{R}}$, via a proxy $\mathsf{ID}_{\mathcal{P}}$. Due to the challenge of ephemeral leakage (as described in C5), the receiver must be ensured that the protocol is still secure against randomness injections or leakages in either the device $\mathsf{ID}_{\mathcal{I}}$ or the proxy server $\mathsf{ID}_{\mathcal{P}}$.

This scenario primarily addresses challenges Sec1, Sec3, Sec4 *and* Sec5.

Scenario P_{II}: Multi-Party Signcryption for Cellular Networking Infrastructures Secure Against Ephemeral Leakage

We consider any transportation infrastructure, including roads and railways, that are integrating connected devices like IoT, IIoT, sensory equipment, cameras, and other type of low-powered devices. Especially those that uses

technology that can be used via cellular service operations specifically tailored for IoT devices, e.g., Narrowband Internet of Things (NB-IoT) and the 5G. In such eco-system, these devices need to send data to different verifying nodes, e.g., authorities or infrastructure components that need to analyze the data. Some clusters of devices need to aggregate the data before sending it further, which can be done using one or several aggregation nodes (which can be part of the infrastructure or delegated to one of the devices). Hence, being able to authenticate and encrypt the data is important. While many protocols that have been developed for encryption and authentication in LTE and 5G architectures, challenges remain, such as the absence of adequate identity protection. One mitigation is authenticated encryption - signcryption - which offers a dual service of encrypting a message while also confirming its authenticity. This secure function must be enabled in this multi-party setting where several devices $\mathsf{ID}_1, ..., \mathsf{ID}_n$ are connected and need to send aggregated (authenticated and encrypted) data to a verifier ID_V, via an aggregator node ID_A. In the eco-system, a third party for key generation is used as well, the key generation center. This node enables certificateless public key structures that are efficient and can be scaled. However, leakage of ephemeral keys from a device is a real threat; there might be a poor random generator, side-channel attacks or malevolent hardware producers who might incorporate data leakage functions. Despite advancements in encryption algorithms tailored for the NB-IoT in 5G frameworks, the risk posed by the leakage of ephemeral values persists.

This scenario primarily addresses challenges Sec1-Sec3 *and* Sec5.

Scenario $\mathsf{P}_{\mathsf{III}}$: Secure and Leakage-Free Authentication with Collaborative Signatures for Circuit-Breaking Transportation Infrastructures

In a connected railway infrastructure scenario there may be transports of highly important or sensitive cargo. Consider a train ID_T with a carriage ID_C containing sensitive cargo equipped with IoT device(s) $\mathsf{ID}_{C'}$ for continuous monitoring, directly connected to the carriage network system. If there is a breach in the cargo, or an attempt of tampering the network, a circuit-breaking signal triggers and different security (and safety) measures are initiated. For this setup we need to verify that the delivery is intact or not disconnected from its route. The network environment this scenario can utilize is what we refer to as a circuit-breaking environment where close-by devices can sample the same surrounding environment using sensory data such as pressure, weather, radiation etc. when seeding each device's internal pseduorandom generators with the same values. This particular feature

will thus allow for the circuit breaking function where the cluster of connected devices will alert if any of the nodes are removed or the randomness is desynchronized. For this scenario, we need a way to continuously generate and verify signatures computed by the cargo device $\mathsf{ID}_{C'}$ and its linkage to the connected carriage ID_C. Each device contains their own hardware security modules used for signature generation and key storage. For this setup we then need a secure protocol allowing ID_C and $\mathsf{ID}_{C'}$ synchronize and compute/verify signatures without being vulnerable to partial key leakage.

This scenario primarily addresses challenges $\mathsf{Sec1}, \mathsf{Sec3}$ *and* $\mathsf{Sec5}$.

Scenario $\mathsf{P_{IV}}$: Source Hiding of Anonymous Signers

This scenario consider a VANET where a group of moving nodes - vehicles - needs to collectively generate a ring signature to transmit a signed message to a RSU or other vehicle. The complexity is that the original signer's identity must remain anonymous, hence preserving the anonymity of the vehicle that initiated and originally signed the message. However, ring signatures only safeguards the signer's identity at the application layer, leaving its logical and underlying identities, such as the IP address on the network layer, potentially exposed. As such, a secure protocol that can effectively mitigate tracing attacks becomes essential in providing the overall network security and anonymity of the vehicles. Moreover, this protocol must function without relying on a trusted third party (TTP), enhancing the autonomous and distributed nature of the VANET. More formally, in this particular VANET setup we then consider n *participants* $\mathsf{ID}_1, ..., \mathsf{ID}_n$ and one *proxy server* which is *not* a TTP, i.e. can be considered untrusted. There is a protocol π which provides a ring signature computation between the participants, and then an interactive sub-protocol execution between the server and each ID_i. The sub-protocol ensure that each participant's underlying identity L_i, i.e. the IP address, remains undisclosed during the whole process of signing a message and having any *verifier* that holds the public keys of the group of signers, to verify the signature successfully.

This scenario primarily addresses challenges $\mathsf{Sec1}$ *and* $\mathsf{Sec3}$.

Scenario $\mathsf{P_V}$: Leakage-Resilient Authenticated Key Exchange for Short-Range Devices

This scenario consider any type of connected infrastructure that uses two short-range devices that need to collaborate or communicate. A plausible scenario can be a smartphone making a temporary connection to a VANET hub or information relying server, or two IoT devices that need to share

monitoring and data analysis tasks. In any case, the devices need to perform an authenticated key exchange protocol in order to authenticate and continue the data transmissions. During such protocol execution, there must be mitigation against key leakage, otherwise there will be a high risk of impersonation attacks. Typically, an attacker can block or intercept the protocol execution, and in combination with partial key leakage attacks, the devices are compromised and the attacker can act as one of the parties. This would be highly dangerous if the devices are used for traffic safety monitoring purposes.

This scenario primarily addresses challenges Sec3 *and* Sec5.

2.2 Related Work

Cryptography plays a critical role in ensuring the security of connected infrastructures such as C-ITS environments, and is of primary focus of this thesis. There has been much research in this area, particularly focusing on VANET systems [114, 85, 147, 123]. In this section, we discuss different types of solutions and their underlying cryptographic schemes that have been proposed within the C-ITS context, in order to give a comprehensive and necessary background to our research. Also, for the given scenarios $\mathsf{P_I} - \mathsf{P_V}$, the specific related work in terms of similar solutions or other type of approaches are summarized in more detail in Chapter 4 .

2.2.1 Authentication and Signatures in C-ITS

Authentication and digital signatures are vital in maintaining the integrity and reliability in C-ITS environments; these cryptographical tools are used for safeguarding against several of the potential security threats described previously. Over the past two decades, the field of C-ITS has evolved significantly, paralleled by an increase in the development of authentication and signature schemes. The field is extensive, and for the curious reader we refer to any of the following survey articles [123, 51, 148] for more information. Now, authentication of a node ID_i's data m within a C-ITS environment is done via a signature σ over m that is sent to a recipient for verification. Typically, in these type of ad-hoc environments many participants sends or even broadcast messages frequently. For this reason, efficient and secure signature schemes are required. For C-ITS environments several different type of schemes has been proposed, including collaborative multi-party protocols using *group signatures, ring signatures* and *signcryptions*. Group signatures and ring signatures are cryptographic primitives designed for anonymous

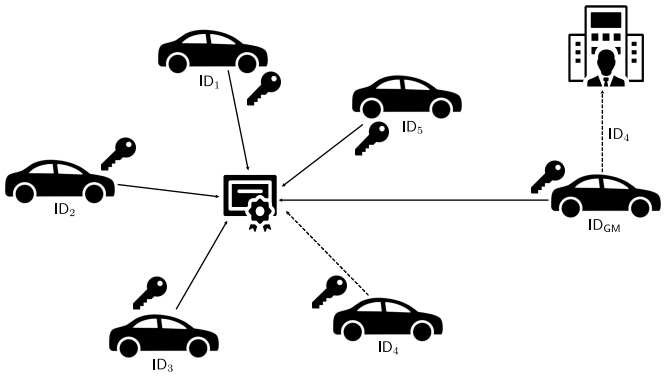

Figure 2.4: Conceptual schematic depiction of how group signatures work, where signers $\mathsf{ID}_1, .., \mathsf{ID}_5$ is a group and ID_4 initiates a signature. The group manager $\mathsf{ID}_{\mathsf{GM}}$ is able to reveal the identity to a authority on request.

signing, but they differ in their foundational principles and applications [135].

In group signature protocols, members of a designated group can sign a message m, on the group's behalf, thus producing a verifiable signature σ. Any third party can validate that it was signed by some member of the group, without revealing the signer's identity. There is a group manager that is able to reveal the signer's identity in necessary circumstances, ensuring accountability; this is in particular useful in use-cases where authorities might need to track previous events. We depict the concept in Fig. 2.4. Ring signatures operate without a fixed group, i.e., a signer can spontaneously form a "ring" of potential signers $\mathsf{ID}_1, \mathsf{ID}_2, ..., \mathsf{ID}_n$. The resulting signature σ verifies that one member of this ring signed m. Anyone can verify that a signer from the ring signed m, but cannot determine which specific signer. Additionally, there is no entity that can revoke the signer's anonymity, making it fully anonymous. Within C-ITS environments, group signatures prove beneficial when vehicles need to anonymously share messages such as road statuses or traffic alerts, with a centralized system. The system can authenticate a message m sent by a vehicle using σ without identifying the specific sender. If the scenario demand, like safety or regulatory needs, the vehicle's identity can be exposed using the group manager. Meanwhile, ring signatures can be suited for peer-to-peer communication in C-ITS. This ensures privacy as vehicles relay vital messages. For instance, a vehicle detecting a hazard might communicate a warning message m to its peers, producing a corresponding signature σ. Other vehicles can then trust the message's origin without knowing the exact sender, upholding privacy in these type of data exchanges.

Secure group- and ring signature schemes have been proposed within the field of C-ITS e.g., [105, 83, 11, 164, 109, 29, 122, 73]. However, several protocols lack proper security proofs, and few schemes are leakage resilient, whereas our research provides that as one of the main contributions.

2.2.2 Signcryption

Signcryption is a cryptographic method that combines the processes of signing and encrypting into one single operation. The method was introduced by Zheng in the late 1970's [166], to mitigate the computational and communication overheads associated with traditional procedures of *sign-then-encrypt* and *encrypt-then-sign*. In these traditional methods, a signature scheme might be used to sign a message m, followed by an encryption scheme with new sets of keys, to encrypt both the message and the signature. Instead, computing a signcryption of m would encapsulate both signing and encryption in the same scheme with the same keys. When a message is signcrypted, it could be signed by the sender's private key and encrypted with the recipient's public key in one step. Upon receipt, the receiver can decrypt the message and verify the signature at the same time. Therefore, both confidentiality and authenticity of m can be ensured. The security attributes derived from signcryption include confidentiality, unforgeability, and non-repudiation. For instance, in a scenario where ID_A wants to send a confidential and authenticated message to ID_B, she could employ a signcryption scheme. When ID_B receives the signcrypted message, he utilizes an unsigncryption algorithm to access the original message and also verify ID_A's signature. Executing this in one step each way makes signcryption and unsigncryption algorithms more efficient than separate signature and encryption algorithms. This enhanced efficiency makes signcryption particularly useful in settings where computational resources are constrained, such as in IoT devices or mobile communications; thus particularly suitable for C-ITS environments. In the current literature many signcryption schemes have been proposed, however, for C-ITS environments specifically only a limited amount of research is available, e.g., [45, 69, 131, 161], where much of the proposed schemes are within certificateless public key infrastructures. Moreover, our research provides signcryption schemes using proxy nodes and also possess the properties of being secure against ephemeral leakage attacks. This is developed using our proposed security enhancement techniques, applied to existing vulnerable schemes.

2.2.3 Leakage Resilience

As previously noted, many existing surveys and literature reviews on security challenges for C-ITS and its subdomains identify privacy preservation as one significant concern. This typically relates to the leakage of sensitive data, as referenced in [144, 103], rather than the specific leakage of secret keys. Furthermore, it does not mention temporary session keys, also known as ephemeral keys. This is very interesting since the technical architecture of C-ITS relies on secure storage of keys in dedicated hardware; tamper-resistant memory units of hardware security modules inside OBU:s and RSU:s [70, 159]. Now, assuming these hardware-based key storage units can be compromised we are able to model powerful adversaries, exploiting different type of leakage attacks. For authenticated key exchange (AKE) schemes, the well-known Canetti-Krawczyky (CK) model was proposed [35], where adversaries are allowed to compromise a key holder's device, and extracting the key. Later on, an extension of the model was proposed by LaMacchia et al. to handle the case of providing security guarantees for a certain session if the ephemeral key of any of the participants of the protocol has been leaked [101]. This extension - the eCK-model - is particularly interesting in a C-ITS system where many different vendors of technology can interact, e.g. vehicles, train carriages and infrastructure with different hardware security modules. Having many different devices and modules, not necessarily following the same security testing compliance frameworks, increases the risk of having compromised or tampered hardware. This in turn could then allow for the ephemeral leakage attack during a protocol run. Even if a protocol is not a typical AKE protocol, the security model still applies well in C-ITS with many devices and ad-hoc connections.

Chapter 3

Formal Setting

This chapter will provide necessary mathematical theory and theoretical cryptography as a basis for the reminder of this thesis. A short introduction to the field of provable security is given, followed by the theory of pairing-based cryptography which includes necessary elliptic curve theory and algebra. Finally, we introduce some useful security notions and theory of architectural security and cryptography needed for subsequent chapters.

3.1 Mathematical Preliminaries

We denote $\lambda \in \mathbb{N}$ as a security parameter used as input for the initialization of cryptographic schemes, i.e. determine the bit-level of security the scheme will provide. For example, in the initialization phase of a protocol, all participating parties may need to agree on several setup parameters where λ is typically one such parameter. We refer to $\mathbb{G}$ as an algebraic group (typically finite field groups or elliptic curve groups) without specifying the group operations. These are naturally understood when given the context, e.g. for a scheme based on elliptic curve points the operations are curve point addition and multiplication. The notation $\langle g \rangle = \mathbb{G}$ means an element of the group that generate the entire group $\mathbb{G}$ by repeated application of the group's operation. As for λ, one or several secure groups may also be agreed on during a protocol setup phase. The notion of a secure group is a group $\mathbb{G}$ which has certain properties where some hardness problems (to be elaborated further in Sec. 3.2.1) can be assumed, i.e. for certain problem formulations within the group it is computationally infeasible to solve that problem.

We denote the set of bit strings of length λ as $\{0,1\}^\lambda$, typically used when defining the domain of a function, e.g. $\mathcal{H} : \{0,1\}^* \rightarrow \{0,1\}^\lambda$ is a hash function $\mathcal{H}$ that takes any length bit string as input and produces a λ-bit

long output. All hash functions we consider in this thesis are defined as cryptographically secure [77], meaning $\mathcal{H}$ is efficient (fast) to compute but exponentially difficult to invert, i.e. given $\mathcal{H}(m)$ for a message m it is hard to compute m, and collision resistant, i.e. it is hard to find messages m_1 and m_2 such that $\mathcal{H}(m_1) = \mathcal{H}(m_2)$.

In cryptography it is common to use the conceptualization of a so called *probabilistic polynomial-time* (PPT) algorithm when modeling certain adversaries. It is a key concept in computational complexity theory, and formal definitions can be found in any book on the topic, e.g. Goldreich's *Computational Complexity: A Conceptual Perspective* [65]. We will use typical notation used in [65]. Informally we describe a PPT as an algorithm that runs efficiently (in polynomial time) and is able to make true random choices during its computations. In our context, the PPT algorithm is polynomial over the security parameter λ. For our purposes, we define a PPT as follows (note that we implicitly view the algorithm as a Turing machine):

Definition 1 (Probabilistic Polynomial-Time Algorithm)**.** *We say that $\mathcal{A}$ is a probabilistic polynomial-time algorithm if there exists a polynomial $p(\cdot)$ for which $\mathcal{A}$ always halts on any input bit string $x \in \{0,1\}^*$, within at most $p(|x|)$ steps.*

When we construct security proofs for a our schemes, the notion of a negligible function is needed. In general terms, if we are able to prove that an adversary - typically modeled as a PPT algorithm - cannot break a predefined security game in the sense that it is *negligible* in probability for the adversary, the proof is completed. It means that the adversary can only break the security game in such a small probability that we can consider it unbreakable.

Definition 2 (Negligible function)**.** *A function ϵ is called* negligible *if for any $c > 0$, there exists some N_c such that for any $N > N_c$ it holds that $\epsilon(N) < \frac{1}{N_c}$.*

Finally, much of our schemes relies on a source of randomness, e.g., when generating keys, ephemeral values and more. Throughout our research we denote such a source as a *pseudo-random generator* PRG or equivalently *pseudo-random number generator* PRNG, which generate bytes indistinguishable from a true random generator. When we draw a random value x from a set $\mathbb{A}$ which is uniformly distributed, we denote that as $x \leftarrow_{\$} \mathbb{X}$.

Important note: When generating random bits to be used for ephemeral keys or secure exponents to group elements, we assume a simple yet efficient

mitigation of *edge cases*, i.e,. we assume that the PRNG will re-generate its output in the unlikely event of generate a 0, 1 or other very small number. If the $x \leftarrow_{\$} \mathbb{Z}_q$ and it turns out $x = 0$ then for example $g^x = 1$ and the scheme collapses. The mitigation would be a single line of code checking for trivial values when generating the random numbers.

3.2 Provable Security

Cryptography is a multi-disciplinary field containing several areas in mathematics and computer science. A secure protocol such as an encryption- or signature scheme is constructed out of both a mathematical part and an algorithmic part. So how can one ensure that the scheme is actually secure, and more specifically: secure against what? To answer these questions we have what is referred to as *provable security* which provide cryptographers with the theoretical tools for proving certain security properties of a scheme. Briefly, the process of constructing a scheme and prove it secure builds on finding (or choosing) a hard problem, i.e. a mathematically or computationally intractable problem, which the proof shows that for a certain security (adversary) model it is at least equally hard to break the scheme as it is to break that hard problem. That is a reduction proof. It is highly important to choose a security model, i.e. a clear description of the power of the adversary and in what way such adversary may attack the scheme; sometimes called *security games*. We will introduce detailed security models used in this work in later chapters.

A central concept in formal security analysis, is the *random oracle model*, in which a hash function is modeled as a random function - or an oracle $\mathcal{O}$ - that takes an input and returns a fixed-size output [18]. The hash function is considered random because its output for any given input is indistinguishable from a truly random value. This concept of an oracle is further developed into other types of oracles, e.g. signature or encryption oracles that are able to return valid signatures and encryption/decryption values of a message, run in an instance of a certain scheme. In contrast to ROM there is another security model called *the standard model* which does not assume the existence of such oracle, however in our work we only consider the ROM.

3.2.1 Fundamental Assumptions

As mentioned previously, modern cryptography relies on certain assumptions of intractable mathematical (or rather computational) problems. We will introduce a set of these hardness assumptions used throughout the research.

However, we need to establish some preliminary notation and concepts first. Recall the notion of an oracle $\mathcal{O}$, i.e. an algorithm with the ability to efficiently answer queries to some computational problem, providing random output distinguishable from a true random source. For an oracle with a specific task, e.g. $\mathcal{O}_{\mathsf{DDH}}$ - a Decisional Diffie-Hellman oracle - it would produce a solution to the otherwise intractable problem; specifically, the $\mathcal{O}_{\mathsf{DDH}}$ would return an answer to a given instantiation of the Decisional Diffie-Hellman problem, as described in Def. 5. Let $\mathcal{G}(\lambda)$ be an algorithm that generates a secure group $\mathbb{G}$ and corresponding parameters p, q, g where p, q are primes and g a generator to $\mathbb{G}$. We note that $\mathcal{G}$ uses the security parameter λ as input, thus reflecting the size of the groups that can be generated. We define a set of commonly used assumptions used in our research.

Definition 3 (Discrete Logarithm Problem (DLP))**.** *For any PPT algorithm* $\mathcal{A}_{\mathsf{DLP}}$ *in* $\langle g \rangle = \mathbb{G}$*:*

$$\Pr[\mathcal{A}_{\mathsf{DLP}}(\mathbb{G}, g^x) = x | \mathbb{G} \leftarrow_{\$} \mathcal{G}(\lambda), x \leftarrow_{\$} \mathbb{Z}_q^*] \leq \epsilon_{\mathsf{DLP}}(\lambda),$$

where $\epsilon_{\mathsf{DLP}}(\lambda)$ *is negligible.*

Definition 4 (Computational Diffie-Hellman (CDH))**.** *For any PPT algorithm* $\mathcal{A}_{\mathsf{CDH}}$ *in* $\langle g \rangle = \mathbb{G}$*:*

$$\Pr[\mathcal{A}_{\mathsf{CDH}}(\mathbb{G}, g^x, g^y) = g^{xy} | \mathbb{G} \leftarrow_{\$} \mathcal{G}(\lambda), x \leftarrow_{\$} \mathbb{Z}_q^*, y \leftarrow_{\$} \mathbb{Z}_q^*] \leq \epsilon_{\mathsf{CDH}}(\lambda),$$

where $\epsilon_{\mathsf{CDH}}(\lambda)$ *is negligible.*

Definition 5 (Decisional Diffie-Hellman (DDH))**.** *We define* $D_0 = (\mathbb{G}, g^x, g^y, g^{xy})$ *and* $D_1 = (\mathbb{G}, g^x, g^y, g^z)$ *where* $x \leftarrow_{\$} \mathbb{Z}_q^*, y \leftarrow_{\$} \mathbb{Z}_q^*$ *and* $z \leftarrow_{\$} \mathbb{Z}_q^*$*, and* $\mathbb{G} = \mathcal{G}(\lambda)$*. For any PPT algorithm* $\mathcal{A}_{\mathsf{DDH}}$ *in* $\langle g \rangle = \mathbb{G}$*:*

$$|\Pr[\mathcal{A}_{\mathsf{DDH}}(D_0) = 0] - \Pr[\mathcal{A}_{\mathsf{DDH}}(D_1) = 0]| \leq \epsilon_{\mathsf{DDH}}(\lambda),$$

where $\epsilon_{\mathsf{CDH}}(\lambda)$ *is negligible.*

Definition 6 (Decisional Diffie-Hellman Oracle ($\mathcal{O}_{\mathsf{DDH}}$))**.** *We compute* $\mathbb{G} = \mathcal{G}(\lambda)$ *and* $x \in \mathbb{Z}_q^*, y \in \mathbb{Z}_q^*, z \in \mathbb{Z}_q^*$*. We define* $\mathcal{O}_{\mathsf{DDH}}$ *as an PPT algorithm for which* $\mathcal{O}_{\mathsf{DDH}}(\mathbb{G}, g^x, g^y, g^z) = 1$ *if and only if* $z = xy \bmod q$*.*

Definition 7 (Gap Computational Diffie-Hellman (GDH))**.** *For any PPT algorithm* $\mathcal{A}_{\mathsf{GDH}}$ *having access to the oracle* $\mathcal{O}_{\mathsf{DDH}}$*:*

$$\Pr[\mathcal{A}_{\mathsf{CDH}}^{\mathcal{O}_{\mathsf{DDH}}}(g^x, g^y) = g^{xy}] \leq \epsilon_{\mathsf{DDH}}(\lambda),$$

where $\epsilon_{\mathsf{DDH}}(\lambda)$ *is negligible.*

Definition 8 (Computational co-Diffie-Hellman ($\mathsf{co}-\mathsf{CDH}$)). *Let $\mathbb{G} = \langle g_1 \rangle$ and $\mathbb{G}_2 = \langle g_2 \rangle$ be cyclic groups of prime order $q > 2^\lambda$ with generators g_1 and g_2 respectively. Then*

$$\Pr[(\alpha, \beta) \leftarrow_\$ \mathbb{Z}_q^2,\ g_1^{\alpha\beta} \leftarrow \mathcal{A}(g_1^\alpha, g_1^\beta, g_2^\beta)] \leq \epsilon(\lambda),$$

for any polynomial-time algorithm $\mathcal{A}$, where $\epsilon(\lambda)$ is negligible.

Above assumptions were used in the published core papers. However, we also use an additional assumption specifically for this thesis, that is needed when proving the security of the asymmetric setup of our schemes, whereas the published papers uses the symmetric setup (a detailed description of theses setups are given in Sec. 3.2.3). We therefore recall the Computational Diffie-Hellman Problem for Product Groups (PG-CDH) assumption [155] as follows:

Definition 9 (Computational Diffie-Hellman Problem for Product Groups (PG-CDH)). *Given $g_i, g_i^x, g_i^y, g_{3-i}, g_{3-i}^x, g_{3-i}^y$ it is then infeasible to compute g_i^{xy}, where $i \in \{1, 2\}$.*

3.2.2 Elliptic Curve Cryptography

Elliptic Curve Cryptography (ECC) is an approach to public key cryptography based on the algebraic structure of *elliptic curves* over finite fields. ECC offers equivalent security with smaller key sizes compared to other established public key cryptosystems, such as RSA, which results in faster computations and reduced resource usage [77].

In ECC specifically, an elliptic curve is defined by an equation of the form:

$$y^2 \equiv x^3 + ax + b \pmod{p},$$

where $a, b \in \mathbb{F}_p$, p is a prime number, and the curve is defined over the finite field $\mathbb{F}_p$. The points on the curve, together with a point at infinity, form an abelian group E under a specific point addition operation.

The security of ECC relies on the difficulty of the Elliptic Curve Discrete Logarithm Problem (ECDLP), which involves finding a scalar k given two points P and Q on the curve, such that $Q = kP$. It is computationally infeasible to solve the ECDLP efficiently, making ECC an attractive choice for secure communication and digital signatures.

Definition 10 (Elliptic Curve Discrete Logarithm Problem (ECDLP)). *Let E be an elliptic curve defined over a finite field $\mathbb{F}_q$ and P be a point of order n on E. Given a point Q on E such that $Q = kP$ for some integer k with $0 \leq k < n$, the Elliptic Curve Discrete Logarithm Problem is to find k.*

3.2.3 Pairing-Based Cryptography

Pairing-based cryptography was first introduced by Boneh and Franklin in 2001 paper [25], although the mathematical branch of bilinear maps was discovered much earlier. Pairings are structurally defined over *elliptic curves* and *bilinear maps*, which are defined slightly different in the literature, e.g., in additive or multiplicative notation. We use the definitions by Boneh and Franklin, which is the multiplicative version [25]:

Definition 11. Pairing*: Let $\mathbb{G}_1 = \langle P \rangle, \mathbb{G}_2 = \langle Q \rangle$ be additive cyclic groups, and $\mathbb{G}_T$ a multiplicative cyclic group, all of prime order q. Then $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ are* asymmetric bilinear map groups *if there exists a bilinear map:*

$$\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T \tag{3.1}$$

such that the following conditions hold:

- *(bilinearity) $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$ for all $(P, Q) \in \mathbb{G}_1 \times \mathbb{G}_2$ and all $\forall a, b \in \mathbb{Z}$.*
- *(non-degeneracy) For all $P \in \mathbb{G}_1, P \neq 0$ there is an element $Q \in \mathbb{G}_2$ such that $\hat{e}(P, Q) \neq 1$. Similarly, for all $Q' \in \mathbb{G}_2, Q' \neq 0$ there exists some $P' \in \mathbb{G}_1$ such that $\hat{e}(P', Q') \neq 1$.*
- *(computability) $\hat{e}$ can be efficiently computed.*
- *(isomorphism) There exist an efficient computable isomorphism $\phi : \mathbb{G}_2 \to \mathbb{G}_1$ such that $\phi(Q) = P$ for $P \in \mathbb{G}_1$ and $Q \in \mathbb{G}_2$.*

If $\mathbb{G}_1 = \mathbb{G}_2$ and ϕ is the identity mapping, we call the pairing symmetric*, otherwise it is called* asymmetric*.*

Typically for cryptography purposes, the bilinear map is defined over elliptic curve subgroups. Such subgroup is a cyclic group over an elliptic curve group $E(\mathbb{F}_p)$ with a target group over a finite field $\mathbb{F}_{p^\alpha}$, for some prime p and $\alpha \in \mathbb{N}$. For a better readability, we use a more compact notation for pairings. Namely, we write the bilinearity property as $\hat{e}(g_1^a, g_2^b) = \hat{e}(g_1, g_2)^{ab}$. Thus, instead of using the notation aP for a generator P, we rather use the notation g^a for a generator g. This notation is widely used in the literature and we use it in the remaining of the thesis.

To illustrate the utility of pairings, we consider the BLS signature scheme, named after the creators Boneh, Lynn, and Shacham [27]. Let the pairing be symmetric, i.e., $\mathbb{G}_1 = \mathbb{G}_2$. A user's public key pk is derived by applying an exponent d to a generator $g \in \mathbb{G}_1$, i.e., $\mathsf{pk} = g^d$ while the signature on a message

m is the element $\sigma = \mathcal{H}(m)^d$ where $\mathcal{H} : \{0,1\}^* \rightarrow \mathbb{G}_1$ being a cryptographic hash function mapping messages to a point in $\mathbb{G}_1$ To verify the signature we check if the equality $\hat{e}(\sigma, g) == \hat{e}(\mathcal{H}(m), \mathsf{pk})$ holds. The correctness of this simple, yet effective scheme, lies in the bilinearity property, namely that $\hat{e}(\sigma, g) = \hat{e}(\mathcal{H}(m)^d, g) = \hat{e}(\mathcal{H}(m), g)^d = \hat{e}(\mathcal{H}(m), g^d) = \hat{e}(\mathcal{H}(m), \mathsf{pk})$.

3.3 Cryptographic Setting

Symmetric cryptography, is a technique in which the same secret key sk is used for both encryption and decryption of the data. The sender and receiver of the data must thus share the same secret key sk and keep it securely stored. One of the main benefits of symmetric cryptography is typically faster and more efficient computations compared to asymmetric encryption schemes, making it well-suited for applications that require high-speed encryption and decryption. On the other hand, asymmetric cryptography, also known as public key cryptography (PKC), uses two separate keys for encryption and decryption. A key pk, called the public key, is publicly distributed and can be used for encryption, while the other key sk, called the private key, is kept secret and used for decryption. Asymmetric cryptography thus eliminates the need for a shared secret key. However, it is typically slower and more computationally expensive than symmetric cryptography, making it better suited for applications that require higher levels of security but can tolerate slower processing speeds, such as digital signatures and secure key exchange [78]. Overall, the choice between symmetric and asymmetric cryptography depends on the specific requirements of the application, including the level of security required, the amount of data being transferred, and the speed and efficiency of the cryptographic algorithms used. However, in settings such as C-ITS with VANET and other type of dynamically changing networks, the issue of scaling solutions based on symmetric encryption schemes can be highly inefficient. The reason for this is that in a network with n nodes, a maximum of $n(n-1)/2$ key distributions is required, while only n distributions are required when using PKC; with a growing n, the key distribution becomes a real problem if using solutions entierly based on symmetric encryption.

Another very important distinction we have is between symmetric and asymmetric pairings in pairing-based cryptography, as defined in Def. 3.2.3. We use both symmetric and asymmetric pairings in our research. We provide an additional proof for paper $\mathsf{P_I}$, which is previously proven secure in the symmetric pairing configuration, but for this thesis also a proof in the asymmetric pairing configuration.

3.3.1 Hardware Security Modules

The notion of a *Hardware Security Module* (HSM) is a specific piece of hardware (can also be software-based), designated for secure storage of cryptographic keys and execution of secure computations. HSM can be used in many type of architectures, e.g. servers, mobile devices and IoT devices, but also in OBUs [160] and train control systems [137]. Using multiple HSMs can bring advantages, as it enables load-balancing for key management and further segregates the device architecture for solutions that uses more than one key (or protocol). For a protocol that relies on several secret keys and thus additional ephemeral values (session keys), enhancing the security layer by using a separate HSM for each key, would be a sensible approach. Preferably HSMs from different vendors to spread the risk of having compromised hardware. This reasoning is based on the premise that if a solution utilizes several HSMs and a vendor turns out to be malicious, the entire solution would not be compromised; only a subset of the HSMs might be successfully attacked. As we will explore later, there are specific cryptographic protocols that can still remain secure even if this type of attack occurs. We introduce the definition of a *minimal function* as follows:

Definition 12. *We say that a function* $\mathbf{f}$ *provides* minimal functionality *if it performs a set of cryptographic operations and returns either a value needed for input to a cryptographic protocol or the final output of a cryptographic protocol.*

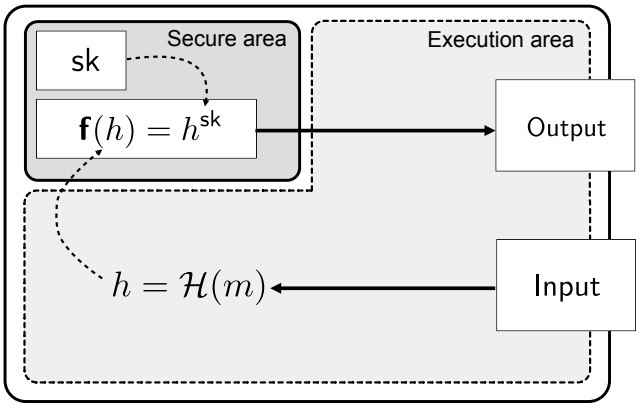

Figure 3.1: Example of a HSM in a device, realizing the minimal function $\mathbf{f}$ as a BLS signature computation using secret key sk.

Naturally, a minimal function $\mathbf{f}$ should be executed in a secure code area within the device, typically inside a HSM. Functions to perform are encryption operations and/or operations part of signature schemes, e.g. a function

$h = \mathbf{f}(m)$ where m is an input sent as input to the HSM and x a secret key stored inside the HSM, then the computation could be $\mathbf{f}(m) = h^x$ and the HSM outputs h^x to the user space in the device. We depict a conceptual sketch of the HSM and another $\mathbf{f}$ in Fig. 3.1. In the device's untrusted area (user space), the computation of a hash digest h of a message m, is then used as input to the HSM which is a trusted area of the architecture. The minimal function $\mathbf{f}$ outputs h^{sk}, i.e., the BLS signature of m. Such minimal functions are conceptualized further and used in papers $\mathsf{P_{II}}$ and $\mathsf{P_{III}}$.

3.3.2 Security Models

The *advantage* of an adversary $\mathcal{A}$ is a measure used in cryptography to quantify the success of $\mathcal{A}$ breaking a cryptographic scheme. It is generally defined as the difference between the $\mathcal{A}$'s probability of success and the probability of success by random guessing. This method is usually formalized by formulating so called *security experiments* where the success of the adversary is measured against a random guess. A negligible advantage indicates a higher level of security for the scheme. With a notation based on [28] we illustrate the concept of analyzing the security of an encryption scheme using the advantage notion (also denoted semantic security). Assume there is an experiment Exp which consists of $\mathcal{A}$ generating two messages m_0 and m_1, from which a challenger then randomly chooses one of the messages, say m_i for $i \in \{0, 1\}$. The challenger then encrypts it into ciphertext c. This ciphertext is sent to $\mathcal{A}$. The experiment now continues as $\mathcal{A}$ tries to compute i, namely given c which message was encrypted by the challenger. Let $\Pr(\mathsf{Exp}_0)$ denote the probability that the adversary $\mathcal{A}$ computes $i = 0$ and correspondingly $\Pr(\mathsf{Exp}_1)$ if $\mathcal{A}$ computed $i = 1$. Then, the advantage of $\mathcal{A}$ is defined as:

$$\mathrm{Adv}(\mathcal{A}, \mathsf{Exp}) = \left|\Pr(\mathsf{Exp}_0) - \Pr(\mathsf{Exp}_1)\right|. \tag{3.2}$$

We use the concept of advantage in all of our security analyses to prove the security of the proposed schemes. In order to utilize this concept in the security analysis, we usually adopt the *game approach*. The game approach is a widely-used method in cryptography to prove the security of a scheme. It involves designing a security game, similar to above example with semantic security, that models the interactions between an adversary and the cryptographic scheme, allowing the adversary to make queries to access certain functionalities or information related to the scheme. The goal of the adversary is to break the security property being analyzed, such as distinguishing between ciphertexts or forging signatures. To prove the security of the scheme using the game approach, the goal is to show that any adversary's

advantage in winning the game is negligible, meaning it is not significantly better than random guessing. The advantage is used as a measure to quantify the adversary's success in the game, as previously defined.

We also have indistinguishability models for encryption schemes, that provide the modeling of various levels of adversary capabilities; we base the remaining overview on Bellare and Rogaway's descriptions [19]. The IND-CPA model, or *indistinguishability under chosen plaintext attacks*, ensures that an encryption scheme cannot be compromised by an adversary by observing encrypted messages. The IND-CCA model - *indistinguishability under chosen ciphertext attacks*, further strengthen the security requirements by giving the adversary the ability to decrypt ciphertexts of their choosing, with the exception of the specific ciphertext c under attack. This model has two subcategories: IND-CCA1, which limits the adversary to a non-adaptive role where decryption queries are only possible until the challenge ciphertext is obtained. In the IND-CCA2 model, the adversary can continue to query the decryption oracle adaptively even after receiving the challenge. For any usage of these and similar/adjusted models we state them in detail in the respective papers for full self-containment, e.g., in $\mathsf{P_{II}}$ and $\mathsf{P_{III}}$.

3.4 Proving Techniques for Asymmetric Pairing Configurations

In our published research we constructed secure schemes using the symmetric pairing setup, i.e., $\mathbb{G}_1 = \mathbb{G}_2$ for the pairing. However, the proof of concept implementations were done in the asymmetric setup since the most efficient programming libraries only provided functionality for asymmetric pairings. Therefore, this section will illustrate step-by-step our method for proving schemes secure in the asymmetric setup as well, showcasing how we construct the proofs for the modified Schnorr scheme. Finally, we illustrate our proving strategy for the asymmetric configuration which is an additional contribution of this thesis and detailed in $\mathsf{P_I}$. The fundamental techniques are within the field of provably security using the ROM. To illustrate the concept we will begin with recalling how to prove the security of impersonation in the modified Schnorr identification scheme [96] in the symmetric pairing configuration, then bridge our proving technique into the asymmetric pairing configuration. We will use a symmetric pairing $\hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ over the secure groups $\mathbb{G}, \mathbb{G}_T$, with generator $\langle g \rangle = \mathbb{G}$. Similarly, for the asymmetric case we use the pairing $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ over secure groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$. In our setup we use two central parties who interacts during a security exper-

iment, namely the prover $\mathcal{P}$ and the verifier $\mathcal{V}$. These two parties interacts during a run of protocol π, denoted $\pi(\mathcal{P}, \mathcal{V})$ where the aim of $\mathcal{P}$ is to convince $\mathcal{V}$ of a certain statement, e.g., that a signature is valid and originates from the correct signer. We use security *experiments* denoted Exp which consists of several stages where the aim of running the experiment is to evaluate if an adversary $\mathcal{A}$ is able to break the protocol, given pre-defined security goals, e.g., impersonation.

In the forthcoming experiments we will use an adversary $\mathcal{A}$ that is able to run a protocol π multiple times (in particular up to a polynomial ℓ number of times), using given adversarial powers such as access to keys or ability to eavesdrop certain messages. We recall that $\mathcal{A}$ is a PPT (see Def. 1), i.e. a Turing machine with the ability to generate random choices, hence it could be realized as software program on a modern computer. A protocol message is referred to as a *transcript* T_i and the set of recorded transcripts is called the *view* $V^{\mathcal{P},\mathcal{V},\ell} = \bigcup\{T_i\}$ given ℓ number of executions of π. This is the adversary's knowledge in the experiment that can be used for breaking the protocol. Initially before the experiment starts $V = \emptyset$.

3.4.1 Modified Schnorr

We recall the original Schnorr identification scheme (IS) [143], based on DLP as in Def. 3. Let $a \leftarrow_{\$} \mathbb{Z}_q^*$ be the secret key sk and $A = g^a = \mathsf{pk}$ of the prover $\mathcal{P}$. The protocol consists of two subroutines used for the initial setup of the protocol, namely ParGen which generates necessary security parameters given λ, and KeyGen which generates the above mentioned keys. The goal is for $\mathcal{P}$ to prove its identity to verfier $\mathcal{V}$ in the following protocol steps, constituting protocol $\pi(\mathcal{P}(a, A), \mathcal{V}(\mathcal{A}))$:

Definition 13 (Schnorr IS)**.** *The Schnorr Identification Scheme is defined in the following steps:*

1. $\mathcal{P}$*:* $r \leftarrow_{\$} \mathbb{Z}_q^*, R = g^r$ *and sends* R *to* $\mathcal{V}$*.*
2. $\mathcal{V}$*:* $c \leftarrow_{\$} \mathbb{Z}_q^*$ *and sends* c *to* $\mathcal{P}$*.*
3. $\mathcal{P}$*:* $s = r + ac$ *and sends* s *to* $\mathcal{V}$*.*
4. $\mathcal{V}$*: accepts iff* $g^s == RA^c$ *holds.*

In the modified Schnorr [96], the protocol instead uses a symmetric bilinear pairing $\hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$, a secure hash function $\mathcal{H}_g : \{0,1\}^* \rightarrow \mathbb{G}$ mapping inputs into group elements of $\mathbb{G}$, and is based on GDH as in Def. 7. The key generating setup is same as in the original scheme. The protocol runs as follows:

Definition 14 (Mod Schnorr IS)**.** *The Modified Schnorr Identification Scheme is defined in the following steps:*

1. $\mathcal{P}$*:* $r \leftarrow_{\$} \mathbb{Z}_q^*, R = g^r$ *and sends* R *to* $\mathcal{V}$*.*
2. $\mathcal{V}$*:* $c \leftarrow_{\$} \mathbb{Z}_q^*$ *and sends* c *to* $\mathcal{P}$*.*
3. $\mathcal{P}$*:* $\hat{g} = \mathcal{H}_g(R, c), S = \hat{g}^{r+ac}$ *and sends* S *to* $\mathcal{V}$*.*
4. $\mathcal{V}$*:* $\hat{g} = \mathcal{H}_g(R, c)$ *and accepts iff* $\hat{e}(S, g) == \hat{e}(\mathcal{H}_g(R, c), RA^c)$ *holds.*

As we see in the modified version, the value s is now hidden via exponentiation as in $\hat{g}^s$ using $\mathcal{H}_g$, and the verification is possible using a bilinear pairing due to its properties of shifting exponents. We also note that in the third step computing S can be realized via HSM and minimal functions, i.e., $S = \hat{g}^s = \hat{g}^{r+ac} = \hat{g}^r \cdot \hat{g}^{ac} = \hat{g}^r \cdot (\mathbf{f}(\hat{g}))^c$ for some secure function $\mathbf{f}$ (computing the input with an exponent of the secret key a) programmed inside a HSM.

3.4.2 Security Proof

To introduce the general proof technique for the asymmetric case we must first begin with showing a similar proof in the symmetric configuration. We start with re-stating the proof for the security of the modified Schnorr under the Chosen Prover Ephemeral (CPE) model [96], to showcase the general approach. This means that the attacker is able to inject a chosen random ephemeral value $\bar{x}$ into the prover during to protocol run, hence being able to impersonate the prover. We first need to define the security experiment:

Definition 15. *We define the security experiment* $\mathsf{Exp}^{CPE,\lambda,\ell}$ *as three stages in the following way:*

`Init Stage` : $\mathsf{par} \leftarrow \mathsf{ParGen}(\lambda)$ *and* $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}()$*, and the adversary* $\mathcal{A}$ *is able to run the protocol with malicious algorithms* $\hat{\mathcal{P}}, \hat{\mathcal{V}}$ *using the public key* pk*.*

`Query Stage:` $\mathcal{A}$ *can run the protocol* $\pi(\mathcal{P}^{\bar{r}_i}(\mathsf{sk}), \hat{\mathcal{V}}(\mathsf{pk}, \bar{r}_i))$ *up to a polynomial number* ℓ *times. By doing so,* $\mathcal{A}$ *is able to record the transcripts from these protocol runs and collect the view* $V^{\mathcal{P},\hat{\mathcal{V}},\bar{r}(l)}$ *where* $\bar{r}_i \in \{\bar{r}_1, ..., \bar{r}_\ell\}$ *representing the different choices of ephemeral values injected by* $\hat{\mathcal{V}}$ *on the* i*th protocol execution.*

`Impersonation Stage:` *the adversary runs protocol* $\pi\left(\hat{\mathcal{P}}(\mathsf{pk}, V^{\mathcal{P},\hat{\mathcal{V}},\bar{r}(l)}), \mathcal{V}(\mathsf{pk})\right)$*.*

The advantage of adversary $\mathcal{A}$ in this experiment is defined as:

$$\boldsymbol{Adv}(\mathcal{A}, \mathsf{Exp}^{CPE,\lambda,l}) = \Pr[\pi(\hat{\mathcal{P}}(\mathsf{pk}, V^{\mathcal{P},\hat{\mathcal{V}},\bar{r}(l)}), \mathcal{V}(\mathsf{pk})) \rightarrow 1]$$

and the scheme is secure if $\boldsymbol{Adv}(\mathcal{A}, \mathsf{Exp}^{CPE,\lambda,l})$ is negligble in λ.

Now, given the defined security experiment $\mathsf{Exp}^{CPE,\lambda,l}$ we also need to setup any oracles to use within the ROM. We define the hash oracle as follows:

Definition 16. *Let $\mathcal{O}_{\mathcal{H}}$ be a hash oracle modelled as ROM, i.e., a programmable table with three columns: I, H and d for inputs, outputs and masking values. For input I_i the oracle chooses a random mask d_i and output $\hat{g}_i = g^{d_i}$, and the oracle stores $I_i, \hat{g}_i$ and d_i on row i in the table.*

The hash oracle is accessible during the security experiments and as per definition, any previous input query to the oracle will return the same output.

3.4.3 Simulation and Rewinding

A key concept in the proof is the ability to *simulate* the protocol without the secret key sk, i.e., we can simulate π as described in the query stage, such that the protocol runs and verifies correctly, thus generating the transcripts but without access to the secret. This holds since during a protocol run the following steps can be computed without the knowledge of a: the ephemeral $\bar{r}_i$ is injected and the prover produces $\bar{R} = g^{\bar{r}_i}$. The value c is generated randomly from $\mathbb{Z}_q^*$ and the hash oracle returns $\hat{g} = g^d$ with some random mask d, on input $(\bar{R}, c)$. Now, the value $S = \hat{g}^{\bar{r}_i + ac}$ requires a, however, since the oracle gives us $\hat{g} = g^d$ we see that the following relation holds:

$$S = \hat{g}^{\bar{r}_i + ac} = (g^d)^{\bar{r}_i + ac} = g^{d\bar{r}_i} g^{dac} = g^{d\bar{r}_i} A^{dc}. \tag{3.3}$$

Therefore, we are able to simulate the transcripts only using $\mathsf{pk} = g^a = A$, without using the secret key a. We also see that the verification step $\hat{e}(S, g) == \hat{e}(\mathcal{H}_g(\bar{R}, c), \bar{R}A^c)$ holds since from Eq. (1) we have that:

$$\hat{e}(\hat{g}, \bar{R}A^c) = \hat{e}(g^d, g^{\bar{r}_i} g^{ac}) = \hat{e}(g^d, g)^{\bar{r}_i + ac} = \hat{e}((g^d)^{\bar{r}_i + ac}, g) = \hat{e}(S, g). \tag{3.4}$$

To summarize, the protocol can be simulated only using the public key pk and where the oracle $\mathcal{O}_{\mathcal{H}}$ is used for querying inputs of $(\bar{R}, c)$; returning $\hat{g}$ values.

In the forthcoming proof we are using the *rewinding technique* [96]. In its essence, the rewinding refer to the notion of being able to fix an input value

r as well as R at the adversary during a protocol run, and after receiving and consuming a random value c from the verifier (given the fixed r from the prover), we are able to interrupt the protocol execution and step backwards in the protocol until the r was fixed. After rewinding the protocol runs again but from this starting point, which enforces the verifier to generate a new random value c'. This means that there are two random values c and c' for one fixed r, and more importantly, these c-values also constitutes S and S' which both verifies correctly during the verification step of the protocol.

We also consider for signature schemes, the closely related *forking lemma* [136] which specifically deals with the probability of creating two successful executions (forks) of a signature scheme that differ only in a certain random challenge. It specifically addresses how a signature, generated for a given message m is composed of $\sigma = (r, h, s)$ where r is a randomly chosen number from a large set, h is a hash that incorporates both the message m and the first random number r, and s is derived from m, h and r. The essence of the forking lemma lies in its ability to demonstrate that if an algorithm can produce a valid signature σ under certain conditions, then it is theoretically possible to "fork" this process to generate two distinct yet valid signatures σ and σ' for the same message m. This notion plays a critical role in assessing the security of signature schemes by highlighting potential vulnerabilities without requiring access to a secret key. In our upcoming proofs we will use the rewinding technique, we will utilize the forking lemma for the none interactive schemes.

We are now able to recall the security proof of the modified Schnorr identification scheme from [143], based on the symmetric pairing configuration:

Theorem 1. *The modified Schnorr identification scheme as given in Def. 14 is secure against ephemeral injection, i.e., for a malicious verifier $\hat{\mathcal{V}}$ to inject randomness $\bar{r}$ into prover $\mathcal{P}$ - in the symmetric pairing configuration.*

Proof. The proof is by contradiction; assuming the existence of an adversary $\mathcal{A} = (\hat{\mathcal{P}}, \hat{\mathcal{V}})$ which is able to compute $g^{\alpha\beta}$ given the GDH instance $(g, g^{\alpha}, g^{\beta})$, within the frame of experiment $\mathsf{Exp}^{CPE,\lambda,\ell}$. This means that we inject the GDH instance into the experiment where the attacker possibly wins if the instance can be broken, i.e., breaking the GDH assumption. We also note that α corresponds to the secret key a in the experiment. The experiment is then executed as follows:

`Init Stage:` The experiment starts by running $\mathsf{par} \leftarrow \mathsf{ParGen}(\lambda)$ such that CDH holds, and instantiate GDH with $(g, g^{\alpha}, g^{\beta})$. The adversary is given access to hash oracle $\mathcal{O}_{\mathcal{H}}$ and given the public key $\mathsf{pk} = g^{\alpha}$.

`Query Stage:` The adversary simulates the protocol $\pi(\mathcal{P}^{\bar{r}_i}(\mathsf{sk},\mathsf{pk}),\hat{\mathcal{V}}(\mathsf{pk},\bar{r}_i))$ ℓ number of times. In each protocol run, the malicious verifier $\hat{\mathcal{V}}$ has the ability to inject randomness $\bar{r}_i$ into $\mathcal{P}$ during the first step when computing R. Next, for each query with input I_i to the oracle $\mathcal{O}_{\mathcal{H}}$, it returns $H_i = g^{d_i}$, i.e. the $\hat{g}$ value, for a randomly chosen d_i and the values (I_i, g^{d_i}, d_i) are stored in the oracle's table. If an already queried input I_i is requested, the previously stored value H_i is returned. When the adversary chooses an ephemeral value $\bar{r}_i$ of its own choice and injects it into $\mathcal{P}$, the resulting computation is $\bar{R} = g^{\bar{r}_i}$, i.e., the first message sent in the protocol from the prover to the verifier. The simulation is possible since $\mathcal{O}_{\mathcal{H}}(\bar{R}, c_i) = H_i = \hat{g} = g^{d_i}$, but the value $S = \hat{g}^{\bar{r}_i + ac_i}$ requires a. However, since the oracle gives us $\hat{g} = g^{d_i}$ and the public key A is publicly known, we see that the following relation holds:

$$S = \hat{g}^{\bar{r}_i + ac_i} = (g^{d_i})^{\bar{r}_i + ac_i} = g^{d_i\bar{r}_i} g^{d_i ac_i} = g^{d_i\bar{r}_i} A^{d_i c_i}. \tag{3.5}$$

Therefore, the verification step also holds:

$$\hat{e}(\hat{g}, \bar{R}A_i^c) = \hat{e}(g^{d_i}, g^{\bar{r}_i} g^{ac_i}) = \hat{e}(g^{d_i}, g)^{\bar{r}_i + ac_i} \tag{3.6}$$

$$= \hat{e}((g^{d_i})^{\bar{r}_i + ac_i}, g) = \hat{e}(S, g). \tag{3.7}$$

`Impersonation Stage:` In this third stage we run $\pi(\hat{\mathcal{P}}(\mathsf{pk}, V^{\mathcal{P},\hat{\mathcal{V}},\bar{r}_i}), \mathcal{V}(\mathsf{pk}))$ where the verifier is honest and the prover is controlled by $\mathcal{A}$. After the first protocol step the malicious prover $\hat{\mathcal{P}}$ controlled by $\mathcal{A}$ fixes the randomness r and utilities the *rewinding technique* in the following way: $\hat{\mathcal{P}}$ sends R to $\mathcal{V}$ and receives c. After consuming c and making sure that (R, c) was not previous input to $\mathcal{O}_{\mathcal{H}}$ in the query stage, it rewinds back and trigger a new value c' from $\mathcal{V}$ with the same fixed R. The adversary has now consumed $c \neq c'$ such that the ROM table has registered $H_1 = (g^{\beta})^{d_1}$ and $H_2 = (g^{\beta})^{d_2}$ where d_1, d_2 are two randomly chosen masks returned when the oracle gets input (R, c) and (R, c') respectively. In the next step of the protocol, the adversary is now able to construct:

$$S = (H_1)^{r+ac}, \tag{3.8}$$

$$S' = (H_2)^{r+ac'}. \tag{3.9}$$

Now, let $\dot{S} = S^{\frac{1}{d_1}}$ and $\dot{S}' = S'^{\frac{1}{d_2}}$, we then see that:

$$\frac{\dot{S}}{\dot{S}'} = \frac{(H_1)^{\frac{r+ac}{d_1}}}{(H_2)^{\frac{r+ac'}{d_2}}} = \frac{(g^{\beta})^{d_1 \frac{r+ac}{d_1}}}{(g^{\beta})^{d_2 \frac{r+ac'}{d_2}}} = \frac{(g^{\beta})^{r+ac}}{(g^{\beta})^{r+ac'}}, \tag{3.10}$$

$$= \frac{(g^{\beta})^{r}(g^{\beta})^{ac}}{(g^{\beta})^{r}(g^{\beta})^{ac'}} = (g^{\beta})^{a(c-c')}. \tag{3.11}$$

Now, since both c and c' are known we can proceed with the computation:

$$\left(\frac{\dot{S}}{\dot{S}'}\right)^{\frac{1}{c-c'}} = (g^{\beta})^{a} = g^{\alpha\beta}. \tag{3.12}$$

This is due to $a = \alpha$, initiated in the public key $\mathsf{pk} = g^{\alpha}$ during the init stage. Therefore, GDH is broken since we were able to compute $g^{\alpha\beta}$, thus by contradiction such adversary $\mathcal{A}$ cannot exist and the protocol is secure.

□

3.4.4 Proving the Modified Schnorr in the Asymmetric Configuration

In the previous setup we proved the security of Mod Schnorr IS using the symmetric configuration. We are now considering the asymmetric pairing $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, hash function $\mathcal{H}_{g_1} : \{0,1\}^* \to \mathbb{G}_1$ and a similar protocol but for which the public key consists of components from both groups, namely $(a, A_1 = g_1^a, A_2 = g_2^a) \leftarrow \mathsf{KeyGen}$ where $\langle g_1 \rangle = \mathbb{G}_1$ and $\langle g_2 \rangle = \mathbb{G}_2$, and a, b are secret keys. The main difference in this protocol is during the final step, i.e. the verification $e(S, g_2) == e(\mathcal{H}_{g_1}(R, c), RA_2^c)$ where $S = \hat{g}_1^{r+ac}$, i.e. an element of $\mathbb{G}_1$, and $R = g_2^r$ for a random $r \in \mathbb{Z}_q^*$, i.e. an element of $\mathbb{G}_2$.

Theorem 2. *The modified Schnorr identification scheme proven in Thm. 1 is also secure in the asymmetric pairing configuration.*

Proof. The proof is by contradiction; assuming the existence of an adversary $\mathcal{A} = (\hat{\mathcal{P}}, \hat{\mathcal{V}})$ which is able to compute $g_1^{\alpha\beta}$ given the $\mathsf{PG\text{-}CDH}$ instance $(g_1, g_1^{\alpha}, g_1^{\beta}, g_2, g_2^{\alpha}, g_2^{\beta})$, within the frame of experiment $\mathsf{Exp}^{CPE,\lambda,\ell}$. The experiment is then executed as follows:

`Init Stage:` The experiment starts by running $\mathsf{par} \leftarrow \mathsf{ParGen}(\lambda)$ and instantiate $\mathsf{PG\text{-}CDH}$ with $(g_1, g_1^{\alpha}, g_1^{\beta}, g_2, g_2^{\alpha}, g_2^{\beta})$. The adversary is given access to hash oracle $\mathcal{O}_{\mathcal{H}}$ and given the public key components $\mathsf{pk} = (A_1 = g_1^{\alpha}, A_2 = g_2^{\alpha})$. We note that it is straightforward to check that $\hat{e}(A_1, g_2) = \hat{e}(g_1, A_2)$.

`Query Stage:` This stage is same as in the symmetric configuration, however, adjusted to the before mentioned differences of values in groups $\mathbb{G}_1$ and $\mathbb{G}_2$. Explicitly, the simulation of the protocol $\pi(\mathcal{P}^{\bar{r}_i}(\mathsf{sk}, \mathsf{pk}), \hat{\mathcal{V}}(\mathsf{pk}, \bar{r}_i))$ running up to ℓ number of times, works as follows: the oracle $\mathcal{O}_{\mathcal{H}}$

returns only elements belonging to $\mathbb{G}_1$. When the adversary choose an ephemeral value $\bar{r}_i$ to inject into $\mathcal{P}$, the resulting computation is $\bar{R} = g_2^{\bar{r}_i}$. The value is sent to the adversary controlled verifier which returns a random c as previously. The prover queries the oracle with $(\bar{R}, c)$ and receives from the ROM table the output $\hat{g}_1 = g_1^d$ for some random mask d. As we know, the value $S = \hat{g}_1^{\bar{r}_i + ac}$ but the simulation does not require a, since the relation

$$S = \hat{g}_1^{\bar{r}_i + ac} = (g_1^d)^{\bar{r}_i + ac} = g_1^{d\bar{r}_i} g_1^{dac} = g_1^{d\bar{r}_i} A_1^{dc} \tag{3.13}$$

holds, i.e., we only need public key A_1 and not rely on a. Moreover, the simulation is valid since the verification $\hat{e}(S, g_2) == \hat{e}(\mathcal{H}_{g_1}(\bar{R}, c), \bar{R}A_2^c)$ holds:

$$\hat{e}(\hat{g}_1, \bar{R}A_2^c) = \hat{e}(\hat{g}_1, g_2^{\bar{r}_i} g_2^{ac}) = \hat{e}(g_1^d, g_2^{\bar{r}_i + ac}) \tag{3.14}$$

$$= \hat{e}((g_1^d)^{\bar{r}_i + ac}, g_2) = \hat{e}(S, g_2). \tag{3.15}$$

`Impersonation Stage:` This stage is same as in the symmetric configuration, however, adjusted for the asymmetric configuration. Basically, the adversary controls the prover and interacts with an honest verifier, and fixes the injected randomness $\bar{r}$ which yields $\bar{R} = g_2^{\bar{r}}$, then making $\mathcal{V}$ return two different values c and c' by applying the rewinding technique. In this protocol that means that the adversary consumes c and c' and produces both $S = \hat{g}_1^{\bar{r}+ac}$ and $S' = \hat{g}_1'^{\bar{r}+ac'}$. We note that these values are produced during this stage after querying the oracle with fresh inputs $(\bar{R}, c)$ and $(\bar{R}, c')$, returning $H_1 = \hat{g} = (g_1^{\beta})^{d_1}$ and $H_2 = \hat{g}' = (g_1^{\beta})^{d_2}$ for the masks d_1, d_2 respectively. Here we also assume that there were two fresh queries returning the corresponding β just as in the previous proof. The adversary is now able to make the following constructions:

$$S = (H_1)^{\bar{r}+ac} = (g_1^{\beta d_1})^{\bar{r}+ac}, \tag{3.16}$$

$$S' = (H_2)^{\bar{r}+ac'} = (g_1^{\beta d_2})^{\bar{r}+ac'}. \tag{3.17}$$

Now, let $\dot{S} = S^{\frac{1}{d_1}}$ and $\dot{S}' = S'^{\frac{1}{d_2}}$, we then see that:

$$\frac{\dot{S}}{\dot{S}'} = \frac{(g_1^{\beta})^{d_1 \frac{\bar{r}+ac}{d_1}}}{(g_1^{\beta})^{d_2 \frac{\bar{r}+ac'}{d_2}}} = \frac{(g_1^{\beta})^{\bar{r}+ac}}{(g_1^{\beta})^{\bar{r}+ac'}}, \tag{3.18}$$

$$= \frac{(g_1^{\beta})^{\bar{r}} (g_1^{\beta})^{ac}}{(g_1^{\beta})^{\bar{r}} (g_1^{\beta})^{ac'}} = (g_1^{\beta})^{a(c-c')}. \tag{3.19}$$

Now, since both c and c' are known to the adversary it can simply proceed with:

$$\left(\frac{\dot{S}}{\dot{S}'}\right)^{\frac{1}{c-c'}} = (g_1^{\beta})^a = g_1^{\alpha\beta} \tag{3.20}$$

since $a = \alpha$, initiated in the public key $\mathsf{pk} = A_1 = g_1^{\alpha}$ during the init stage. Again, PG-CDH is broken, thus by contradiction such adversary $\mathcal{A}$ cannot exist and the protocol is secure.

□

We have shown a generalization of the proof technique first in the symmetric configuration, illustrated by proving the Mod Schnorr IS secure, followed by a similar approach but with the asymmetric configuration. In addition to paper $\mathsf{P_I}$ that will be elaborated in the next chapter, we also provide a new proof not previously published, proving the asymmetric configuration of the proposed protocol.

3.5 Industrial System Setting

In this section, we introduce the system modeling for the addressed scenarios in this thesis. A formal model of a system is a mathematically precise representation that captures the structure, behavior, and properties of the system using. We develop a formal model that involves describing the system components, such as nodes, devices, and communication channels, along with their properties. Moreover, the model provides formal descriptions of the protocols and algorithms used to better describe desired security properties, such as cryptographic primitives, key management, and authentication mechanisms. The model descriptions will specify the assumptions made about the system's environment and trust relationships among components.

3.5.1 The Environment Model

We will describe the environment model by detailing the participants, communication channels, and their relationships with one another. The fundamental concepts underpinning this model include *nodes, devices*, and *proxies*. We briefly define the roles and interactions of these entities within the system. The environment is denoted ENV and is the collection of moving and stationary entities that communicate and collaborate within a distinct logical and geographical area. It can represent a network, e.g. a VANET, or even an aggregation of several networks and systems such as a cluster of VANET:s

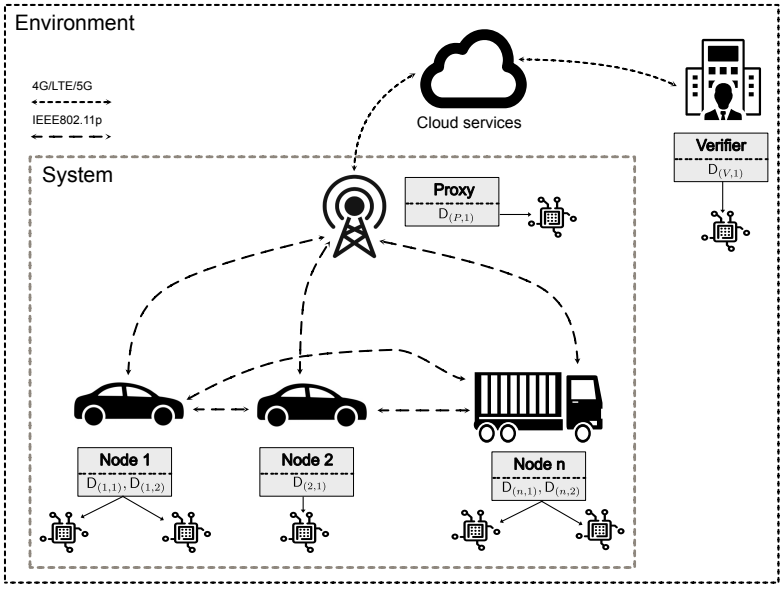

Figure 3.2: Example of how the environment contains a system with different nodes. Each node i may have several devices $\mathsf{D}_{(\mathsf{i},1)}, ..., \mathsf{D}_{(\mathsf{i},\mathsf{k})}$. Between one system and external nodes there might be cloud services or other types of network systems enabling communication channels.

connected to a certain infrastructure area. We use ENV to describe what type of networks and entities that makes a complete system. Therefore, a specified ENV may have certain system properties. For example, in paper $\mathsf{P}_{\mathsf{III}}$ we define a specific environment as a *Circuit Breaking Environment* which defines the property of the system to provide continuous verification of the connectivity between system entities. We show how the environment model relates to all components in Fig. 3.2.

System: In an environment ENV, the entity *system* refers to a collection of entities and technologies that collectively provide a specific function or several functions to a certain subset of the ENV. A system can be regarded as a fundamental component, which itself can be considered an entity within the model. The system encapsulates the interactions and relationships among the connected entities.

Node: The entity *node* represents an integral component within a system, which can be either moving or stationary. A node can model different type of entities, such as a vehicles, trains carriages, devices, and even people in the ENV. It is characterized by having a distinct identity and function within the system. Nodes can be classified as trusted or un-

trusted based on the level of confidenc. Trusted nodes are considered to be dependable and adhere to any agreed upon and established protocols in the system, while un-trusted nodes may have uncertain or potentially malicious intentions; these nodes may not follow a protocol execution fully. It is important to note that an adversary is typically a node within the environment.

Device: The entity *device* is associated with a node and is responsible for carrying out specific functions within the system. A node may have multiple devices associated, which can collaborate to fulfill the node's role in the environment. Typically, a device is an IoT/OT component, OBU or sensor that is capable of communication, computation, and data management, thus contributing to the overall functionality and performance of the node. A device can be considered *weak* if it possesses components with vulnerabilities that can be exploited by adversaries. Moreover, a weak device can also be a low-powered device which has computationally limitations, typically in IoT and OT devices. These vulnerabilities could stem from factors such as poor randomness generation or compromised hardware, which can impact the security and reliability of the device. We consider a device having *trusted* components, e.g. security modules that has the ability to store private keys, and *un-trusted* components, e.g. areas in the device where computations are executed without encryption, or where data leakage may occur.

Proxy: This entity refers to a node that can be either moving or stationary and is capable of relaying messages between two other nodes. Hence, a proxy acts as an intermediary data exchange node within the network. Proxies can be classified as trusted or un-trusted, depending on the level of confidence in their reliability and security. Additionally, a proxy may have delegated responsibilities, such as computing signatures on behalf of other nodes. This delegation of responsibility can enhance the efficiency and performance of the network in the system, especially in cases where certain nodes may have limited computational resources (weak devices) or there are communication challenges.

Channel: This is not an entity per se, but a necessary infrastructure that allows for data exchange between nodes, devices and/or internally of a device. A typical channel in a C-ITS environment are wireless communication via the IEEE 802.11p protocol and long-range communication links via 4G/5G. Also, RFID channels between two devices for close-range pairings are possible, i.e. for authentication purposes.

3.5.2 Atomic Operations

In this section, we describe functionalities the different nodes area able to perform. A device have a set of *atomic* operations that can be computed internally. We describe each such operation in Tab. 3.1 and note that certain devices may have different restrictions in what operations they can perform. Thus, the set of atomic operations a device has, defines it technical capability.

Cryptographic protocols are composed by different subroutines (or algorithms) that can be run, either separately or in sequence between two nodes as in a protocol. We denote a protocol with π. The subroutines involved in π are mathematically described at an abstract level, without detailing the specific implementations. For instance, a signature subroutine is modeled as an algorithm capable of generating a signature σ for a given message m, without specifying the precise method of signature computation. The realization of these subroutines are then detailed in the respective papers in chapter 4 and subsequently utilized in the formal security analysis of π. We also have to consider asymmetric-key encryption systems as part of some protocols. For these, we denote $\mathcal{K}$ the key-pair space, $\mathcal{M}$ the message space, and $\mathcal{C}$ the ciphertext space.

Network operation	**Description**
Generate data	Computing and encode data to be sent.
Send data	Sending data over a communication channel to a recipient.
Receive data	Receiving and decoding data sent from another device.
Computational operation	**Description**
Encrypt m	Encrypting a message m into a ciphertext c. Requires an encryption scheme E.
Decrypt c	Decrypting a ciphertext c into corresponding m. Requires a decryption scheme D.
Sign m	Compute a signature σ of a message m.
Verify σ	Verifying that σ is a valid signature of m.
Hash m	A secure hash computation of m into a hash digest $h(m)$. Each hash function maps into a specified domain, e.g., into $\mathbb{Z}_q^*$.
Rand	Generate a pseduorandom number from its internal PRNG, i.e., generating random bits within the device using some seed ξ.
Arithmetic operation†	**Description**
G: Addition	Group addition, e.g., ECC addition.
G: Multiplication	Group multiplication, e.g., ECC multiplication such as g^x for $g \in \mathbb{G}$ and $x \in \mathbb{Z}_q^*$.
G: Pairing	Bilinear pairing operation into a value of group $\mathbb{G}_T$; relies on efficient curve operations.
G: Exponentiation	Group exponentiation of elements in $\mathbb{G}_T$, e.g, t^x where $t = e(g_1, g_2), g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2$ and $x \in \mathbb{Z}_q^*$
G: Hash	Hashing of any message m into a group element in $\mathbb{G}$, e.g, $\mathbb{G}_1$ or $\mathbb{G}_2$.

Table 3.1: Atomic operations that can be performed by different devices. These operations are then further detailed as subroutines in different protocols. † these operations are computed in the asymmetric setup using the MCL library [120].

Chapter 4

Solutions

In this chapter, we will elaborate further on each scenario corresponding the the published research of papers $\mathsf{P_I}$-$\mathsf{P_V}$. The main consideration that the identified challenges in these scenarios are addressed are due to the safety of passengers and actors in the described environments, i.e. connected infrastructures. We have therefore focused our research on these particular scenarios to provide better and improved safety and security. This is provided by a set of proposed novel schemes, mainly based on the cryptographic primitive of pairings. In some cases we also show, via cryptanalysis, how previous attempts to provide secure schemes have failed. We then suggest improvements of how to patch the vulnerable scheme but also propose new and more secure schemes.

In Tab. 4.1 a summary of the proposed schemes is provided, detailing scheme properties such as type, proxy-based, what security enhancing technique is applied and underlying hardness assumption(s). In the subsequent

Scheme	**Type**	**Proxy**	**Technique**	**Assumptions**	**Paper**
π_{PSC}	Signcryption	✓	Exponent	GDH,PG-CDH	$\mathsf{P_I}$
$\pi_{\mathsf{MPAE-1}}$	Agg. Signcryption	✓	Exponent	GDH	$\mathsf{P_{II}}$
$\pi_{\mathsf{MPAE-2}}$	Agg. Signcryption	✓	Exponent	GDH	$\mathsf{P_{II}}$
π_{KeySplit}	Signature		Key split/ref	CO-CDH	$\mathsf{P_{III}}$
π_{SHP}	Sign-and-Encrypt	✓	Encryption	$\mathsf{RingSign}^{\dagger}, \mathsf{ES}^{\ddagger}$	$\mathsf{P_{IV}}$
π_{SHP2}	Sign-and-Encrypt	✓	Encryption	$\mathsf{RingSign}^{\dagger}, \mathsf{ES}^{\ddagger}$	$\mathsf{P_{IV}}$
$\pi_{\mathsf{LR-AKE}}$	Signature		Key split/ref	CDH	$\mathsf{P_V}$

Table 4.1: Summary of proposed schemes produced during the research conducted for this thesis. † Assuming a secure ring signature scheme RingSign. ‡ Assuming an encryption scheme ES secure under IND-CCA2.

sections we will recall each paper in a unified structure describing the ENV and system settings, security requirements, threat model, related work, pre-

liminary mathematics and cryptography, the proposed scheme(s), security analysis, performance analysis and concluding remarks. We have thus restructured the accepted and published papers, for the purpose of better reading fluency of the reading this thesis. Each paper is self-contained, but in this restructure we are also able to efficiently provide additional contributions such as a new, not yet published proof of scheme π_{PSC} in the asymmetric pairing configuration, but also the extended version of paper $\mathsf{P}_{\mathsf{III}}$ which includes a security proof that was accepted but excluded from publishing due to space constraints. Each paper falls under the copyright of

4.1 Scenario $\mathsf{P_I}$: Ephemeral Leakage Mitigation Using Signcryption in a Proxy Environment

The following section is based on the published paper $\mathsf{P_I}$[1].

System and ENV setup: We consider a VANET (vehicle or railway-based) setup with connected nodes, with installed devices such as sensors, cameras and any type of IoT that is part of a vehicle or infrastructure. There are three types of nodes: initiator node ($\mathcal{I}$), proxy node ($\mathcal{P}$), and, receiver node ($\mathcal{R}$); $\mathcal{I}$ generates messages (sensory data) which are sent to $\mathcal{P}$ that sends the data further as signcrypted messages to $\mathcal{R}$ for verification. Every node is able to communicate with via standard wireless VANET channels, e.g., IEEE 802.11p, and proxy nodes are able to use 4G/5G cellular technology for connection to cloud services or distant traffic centers.

We recall that scenario $\mathsf{P_I}$ is a trans-transportation environment where vehicle and railway systems can be connected for cooperative ITS. All type of sensory devices are part of the system and continuously generate measurement data that needs to be analyzed and verified. In certain areas, the data need to be transmitted long-distance, hence a proxy node is needed that can use long-range communication technology to rely data to traffic management servers. In the scenario we thus need a multi-party protocol for integrity-preserving message-relaying via proxies, in order to securely reach distant nodes. To provide such security *signcryption* can be used, i.e., encrypted and signed messages. The initiator node $\mathcal{I}$ generates messages, e.g., warning, driver assistance, measurement- or maintenance status, all of which belongs to a certain message space Ω that strictly defines which type and structure of messages to be generated. $\mathcal{R}$ can process the messages and utilize the data for assisted driving, safety measures or traffic management optimization. However, since the wireless connectivity is prone to fluctuations, a recipient node might fall out of coverage, and, not receive the urgent messages from initiator node $\mathcal{I}$. Therefore, the proxy node $\mathcal{P}$ act as an intermediate connector between $\mathcal{I}$ and $\mathcal{R}$, and is able to send messages from the space Ω defined by $\mathcal{I}$. Clearly, due to the indirect connectivity between $\mathcal{I}$ and $\mathcal{R}$, it requires the recipient node to verify the authenticity of the data. Moreover, if maliciously injected (weak) randomness is used in either node,

[1]Published at IEEE International Symposium on Network Computing and Applications conference and is under copyright © 2020 IEEE.

an external attacker would be able to completely forge any signcrypted data. Therefore, any proxy signcryption scheme to be used must be resistant to ephemeral key leakage.

4.1.1 Security Requirements

We formulate the following security requirements for the messages and signatures in the system:

- Sensitive data generated at the initiator node must be protected and authenticated by any receiving system, therefore the fundamental security requirement is to provide both *encryption* and *signatures* of messages, with minimal computational power since many of the nodes are using IoT or low-powered devices. Therefore, signcryption is a requirement.
- Since receiving nodes might be far away from the data generating nodes, the proxy nodes must be able to provide signcrypted data on behalf of the initiators. Hence, the requirement is that the protocol can handle *delegated signcryption*.
- Due to ephemeral leakage attacks [16, 12, 151], especially in eco-systems with many different type of devices with different vendors, the requirement for the protocol is to be secure against malicious randomness injections and ephemeral key leakage attacks, both at the initiator and proxy node.

Our protocol is based on the efficient Schnorr signature scheme that is otherwise vulnerable to attacks where the attacker controls the randomness, e.g. [113, 23, 54, 9], hence we provide a stronger and more secure scheme against such attacks. We show by example that a typical Schnorr-based scheme [53] is not secure in the proposed security model, and thus is improved by our proposed scheme. This scenario thus addresses challenges Sec1 and Sec3-Sec5, in a multi-party setting with ad-hoc network environments and where proxy nodes are used.

4.1.2 Related Work

There is both research and implementations of Schnorr-based signcryption schemes e.g. for mobile communication systems [54], [9] but without considering a proxy node in the system. Moreover, for ephemeral key leakage resistant schemes, especially in the context of connected vehicle and IoT-device based infrastructure, there is some research, e.g., [98, 100, 91]. However,

these type of schemes does not consider proxy nor signcryption solutions. A thorough analysis of proxy signcryption schemes can be found in [46].

4.1.3 Threat Model

The signcryption model we suggest, aimed at preserving message integrity, is constructed on a three party framework. During a potentially corrupted execution of this message-relaying process, the receiver has the responsibility to authenticate messages prior to utilizing them for vehicular services. A compromised message-relaying protocol refers to situations where a message m, which began its journey at the initiator node $\mathcal{I}$, undergoes unauthorized alterations and eventually reaches the recipient node as m'. Note that our approach does not consider how the message's content could be altered, but instead we assume it is possible. However, a corrupted session can be identified through our designated verification procedure. There are three long-term secret keys involved: the initiator's secret key ($\mathsf{sk}_{\mathcal{I}}$), the proxy's secret key ($\mathsf{sk}_{\mathcal{P}}$), and the receiver's secret key ($\mathsf{sk}_{\mathcal{R}}$). Additionally, the temporary session keys (ephemeral keys) employed by the initiator and the proxy are denoted as r_1 and r_2 respectively. The verification process proposed in our research has the resilience to withstand potential randomness leakage and injections of these cryptographic keys. Our proposed scheme thus addresses two attack types during a compromised message-relaying session from $\mathcal{I}$ through $\mathcal{P}$:

Ephemeral leakage at the initiator device: The vulnerability of the initiator device and the leakage of randomness r_1, i.e., a random value used for the specific process in the initiator device (see Tab. 4.2 for process details). In this scenario, the adversary, such as a malicious producer of that device, mounts an attack on the long term secret $\mathsf{sk}_{\mathcal{I}}$ with the knowledge of r_1, impersonating $\mathcal{I}$ and forging an arbitrary message space Ω^* to the proxy.

Ephemeral leakage at the proxy device: The vulnerability of the proxy device and the leakage of randomness r_2, also a random value used in a specific process within the proxy device (see Tab. 4.2 for process details). In this scenario, the adversary, such as a malicious producer of the proxy device, mounts an attack on the long term secret $\mathsf{sk}_{\mathcal{P}}$ with the knowledge of r_2, impersonating $\mathcal{P}$, and forging any message m^* from the legitimate space Ω to the receiver.

4.1.4 Preliminaries

Let λ be a security parameter. We denote negligible values as ϵ.

Definition 17. *A proxy signcryption scheme -* PSC *- is a system which consists of six algorithms:* ParGen, KeyGen, Id, Ver, SC, USC.

$\mathsf{ParGen}(\lambda) \to \mathsf{par}$**:** *The setup algorithm takes as input a security parameter* 1^k *and outputs any common parameters* par *required by the signcryption schemes. This include the description of a group* $\mathbb{G}$*, generators for that group, choices for hash functions* $\mathcal{H}$*, and an encryption scheme* $\mathcal{E}$*.*

$\mathsf{KeyGen}(\mathsf{par}, u) \to (\mathsf{sk}_u, \mathsf{pk}_u)$**:** *The key generation algorithm takes as input the common parameters* par*, the user identity* u *and outputs a pair of corresponding secret and public keys.*

$\mathsf{Id}(\mathsf{par}, \mathsf{sk}_{\mathcal{I}}, \Omega) \to \sigma_1$**:** *The identity algorithm takes as input the parameters* par*, the initiator private key* $\mathsf{sk}_{\mathcal{I}}$*, the message space* Ω*. It outputs the warrant* σ_1*.*

$\mathsf{Ver}(\mathsf{par}, \mathsf{pk}_{\mathcal{I}}, \sigma_1, \Omega) \to 1/0$**:** *The warrant verification algorithm takes as input the parameters* par*, the initiator public key* $\mathsf{pk}_{\mathcal{I}}$*, the warrant* σ_1*, and the message space* Ω*. It outputs* 1 *or* 0 *to indicate the acceptance or the rejection of the warrant.*

$\mathsf{SC}(\mathsf{par}, \mathsf{sk}_{\mathcal{P}}, \sigma_1, \Omega, \mathsf{pk}_{\mathcal{R}}, m) \to \sigma_2$**:** *the proxy signcrypt algorithm takes as input the common parameters* par*, the proxy secret key* $\mathsf{sk}_{\mathcal{P}}$*, the warrant* σ_1 *for the message space* Ω*, the receiver public key* $\mathsf{pk}_{\mathcal{R}}$*, and message* m *from space* Ω*. It outputs signcrypted message* σ_2*.*

$\mathsf{USC}(\mathsf{par}, \mathsf{sk}_{\mathcal{R}}, \mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{P}}, \sigma_2, \Omega) \to m/\perp$**:** *The unsign and decrypt algorithm takes as input the parameters* par*, the receiver secret key* $\mathsf{sk}_{\mathcal{R}}$*, the public key of the initiator* $\mathsf{pk}_{\mathcal{I}}$*, the public key of the proxy* $\mathsf{pk}_{\mathcal{P}}$*, the signcrypted message* σ_2*, the message space* Ω*. It outputs the decrypted message* m *or an error symbol* $\perp$*, which indicates that the plaintext cannot be restored or the restored message* m *has the wrong signature.*

Definition 18 (Correctness)**.** *The* PSC *scheme is correct iff for any* $m \in \Omega$*, and any parties* $\mathcal{I}, \mathcal{P}, \mathcal{R}$*:*

$$
\begin{aligned}
\mathtt{Pr}[\mathsf{USC}(\mathsf{par}, \mathsf{sk}_{\mathcal{R}}, \mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{P}}, \sigma_2, \Omega) = m \mid \\
\mathsf{par} \leftarrow \mathsf{ParGen}(\lambda), \\
(\mathsf{sk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{I}}) \leftarrow \mathsf{KeyGen}(\mathsf{par}, \mathcal{I}), \\
(\mathsf{sk}_{\mathcal{P}}, \mathsf{pk}_{\mathcal{P}}) \leftarrow \mathsf{KeyGen}(\mathsf{par}, \mathcal{P}), \\
(\mathsf{sk}_{\mathcal{R}}, \mathsf{pk}_{\mathcal{R}}) \leftarrow \mathsf{KeyGen}(\mathsf{par}, \mathcal{R}), \\
\sigma_1 \leftarrow \mathsf{Id}(\mathsf{par}, \mathsf{sk}_{\mathcal{I}}, \Omega), 1 \leftarrow \mathsf{Ver}(\mathsf{par}, \mathsf{pk}_{\mathcal{I}}, \sigma_1, \Omega), \\
\sigma_2 \leftarrow \mathsf{SC}(\mathsf{par}, \mathsf{sk}_{\mathcal{P}}, \sigma_1, \Omega, \mathsf{pk}_{\mathcal{R}}, m)] = 1.
\end{aligned}
$$

Definition 19 (Indistinguishability under adaptive chosen ciphertext attack)**.** *Let* $\mathsf{PSC} = (\mathsf{ParGen}, \mathsf{KeyGen}, \mathsf{Id}, \mathsf{Ver}, \mathsf{SC}, \mathsf{USC})$ *be a proxy signcryption scheme. We define the following security experiment:*

`Init stage` : *Challenger generates common parameters* $\mathsf{par} \leftarrow \mathsf{ParGen}(\lambda)$ *and the keys* $(\mathsf{sk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{I}}) \leftarrow \mathsf{KeyGen}(\mathsf{par}, \mathcal{I})$, $(\mathsf{sk}_{\mathcal{P}}, \mathsf{pk}_{\mathcal{P}}) \leftarrow \mathsf{KeyGen}(\mathsf{par}, \mathcal{P})$, $(\mathsf{sk}_{\mathcal{R}}, \mathsf{pk}_{\mathcal{R}}) \leftarrow \mathsf{KeyGen}(\mathsf{par}, \mathcal{R})$.

`Query stage` : *Adversary* $\mathcal{A}$ *runs on input* $(\mathsf{param}, \mathsf{y}_{\mathcal{I}}, \mathsf{y}_{\mathcal{P}}, \mathsf{y}_{\mathcal{R}})$. $\mathcal{A}$ *may query a signcryption oracle with messages of its choice* $m \in \Omega$ *to receive the signcryption* $\sigma_2 \leftarrow \mathcal{O}_{\mathsf{SC}}(\mathsf{par}, m, \Omega, \mathcal{I}, \mathcal{P}, \mathcal{R})$ *dedicated for the receiver* $\mathcal{R}$*, from the proxy* $\mathcal{P}$ *with the warrant from initiator* $\mathcal{I}$. $\mathcal{A}$ *may also query an unsigncryption oracle with a signcrypted messages* σ_2 *of its choice, to receive the message* $m \leftarrow \mathcal{O}_{\mathsf{USC}}(\mathsf{par}, \sigma_2, \mathcal{I}, \mathcal{P}, \mathcal{R})$ *if the message was successfully decrypted with* $\mathsf{sk}_{\mathcal{R}}$ *and verified against* $\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{P}}$*, or* $\perp$ *otherwise.*

`Challenge stage` : $\mathcal{A}$ *outputs two equal-length messages* m_1, $m_2 \in M$. *The challenger chooses* $b \in \{0, 1\}$ *at random and computes the challenge signcrypted message tuple* $\sigma_{2,b} \leftarrow \mathcal{O}_{\mathsf{SC}}(\mathsf{par}, m_b, \Omega, \mathcal{I}, \mathcal{P}, \mathcal{R})$.

`Response stage` : $\mathcal{A}$ *inputs the challenge signcrypted message tuple* $\sigma_{2,b}$. $\mathcal{A}$ *may query the signcryption and unsigncryption oracles as before, with the exception that it is forbidden to submit the ciphertext* $\sigma_{2,b}$ *to the unsigncryption oracle.* $\mathcal{A}$ *ends with the output of a bit* b'.

The adversary wins if $b' = b$*, the advantage is then* $|\Pr[b' = b] - 1/2| \leq \varepsilon$.

New stronger unforgeability model

To cover the scenario where ephemeral secrets are known by the adversary, we propose a new, stronger unforgeability model for signcryption schemes, based on similar but weaker models for three party protocols [100, 98]. We assume that the adversary $\mathcal{A}$ has all the public information. We have two scenarios. In the first one, $\mathcal{A}$ possesses additionally the secret key of a proxy $\mathcal{P}$ and sets the randomness of the initiator $\mathcal{I}$ each time it queries the oracle of the initiator. In the second scenario $\mathcal{A}$ has the secret of initiator $\mathcal{I}$ and sets the randomness of proxy $\mathcal{P}$ each time the proxy oracle is invoked. In our definition the scheme is secure if $\mathcal{A}$ cannot impersonate $\mathcal{I}$ in the first case, nor it impersonates $\mathcal{P}$ in the second case.

Definition 20 (Unforgeability under Impersonation Attacks)**.** *Let* $\mathsf{PSC} = (\mathsf{ParGen}, \mathsf{KeyGen}, \mathsf{Id}, \mathsf{Ver}, \mathsf{SC}, \mathsf{USC})$ *be a proxy signcryption scheme. We define the following security experiment:*

`Init` : $\mathsf{par} \leftarrow \mathsf{ParGen}(\lambda)$, $(\mathsf{sk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{I}}) \leftarrow \mathsf{KeyGen}(\mathsf{par}, \mathcal{I})$,
$(\mathsf{sk}_{\mathcal{P}}, \mathsf{pk}_{\mathcal{P}}) \leftarrow \mathsf{KeyGen}(\mathsf{par}, \mathcal{P})$, $(\mathsf{sk}_{\mathcal{R}}, \mathsf{pk}_{\mathcal{R}}) \leftarrow \mathsf{KeyGen}(\mathsf{par}, \mathcal{R})$.

Id `Oracle` : *The oracle* $\mathcal{O}^{\bar{r}}_{\mathsf{Id}}$ *accepts parameters* par*, a message space* Ω*, randomness* $\bar{r}$*, and outputs corresponding valid warrant* σ_1 *generated with the* $\bar{r}$ *on behalf of the initiator* $\mathcal{I}$*, i.e.* $\mathcal{O}^{\bar{r}}_{\mathsf{Id}}(\Omega) \rightarrow \sigma_1$*, such that a verification holds, i.e.,* $\mathsf{Ver}(\mathsf{par}, \sigma, \mathsf{pk}_{\mathcal{I}}, \Omega) = 1$*. The oracle models the device of the initiator in which the warrants are generated with injected ephemerals controlled externally by the adversary.*

SC `Oracle` : *The oracle* $\mathcal{O}^{\bar{r}}_{\mathsf{SC}}$ *accepts the ephemeral* $\bar{r}$*, parameters* par*, a valid warrant* σ_1 *over the space* Ω*, the public key* $\mathsf{pk}_{\mathcal{R}}$ *of the recipient* $\mathcal{R}$*, and a message* m*. It outputs a valid signcryption* σ_2 *generated with* $\bar{r}$ *on behalf of the proxy* $\mathcal{P}$*, i.e.* $\mathcal{O}^{\bar{r}}_{\mathsf{SC}}(\mathsf{par}, \sigma_1, \Omega, \mathsf{pk}_{\mathcal{R}}, m) \rightarrow \sigma_2$*, such that the following computation holds:* $\mathsf{USC}(\mathsf{par}, \mathsf{sk}_{\mathcal{R}}, \mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{P}}, \sigma_2, \Omega) = m$*. The oracle models the device of the proxy in which the messages are generated with injected ephemerals controlled externally by the adversary.*

`FT1` (`Forgery Type I`) : *The adversary generates a tuple:*
$(\Omega^*, \sigma_2^*) \leftarrow \mathcal{A}_{\mathtt{I}}^{\mathcal{O}^{\bar{r}}_{\mathsf{Id}}}(\mathsf{par}, \mathsf{sk}_{\mathcal{P}}, \mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{R}})$*, such that*
$\mathsf{USC}(\mathsf{par}, \mathsf{sk}_{\mathcal{R}}, \mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{P}}, \sigma_2^*, \Omega^*) = m^*$ *and* $m^* \in \Omega^*$*, and* Ω^* *was not previously queried to* $\mathcal{O}^{\bar{r}}_{\mathsf{Id}}$*. The attack reflects the scenario in which the adversary totally controls the proxy (possesses the secret key* $\mathsf{sk}_{\mathcal{P}}$ *of the proxy* $\mathcal{P}$*), and controls the randomness of the initiator device.*

`FT2` (`Forgery Type II`) : *The adversary generates a tuple:*
$(\Omega^*, \sigma_2^*) \leftarrow \mathcal{A}_{\mathtt{II}}^{\mathcal{O}^{\bar{r}}_{\mathsf{SC}}}(\mathsf{par}, \mathsf{sk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{P}}, \mathsf{pk}_{\mathcal{R}})$*, such that*
$\mathsf{USC}(\mathsf{par}, \mathsf{sk}_{\mathcal{R}}, \mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{P}}, \sigma_2^*, \Omega^*) = m^*$ *and* $m^* \in \Omega^*$*, and* m^* *was not previously queried to* $\mathcal{O}^{\bar{r}}_{\mathsf{SC}}$*. The attack reflects the scenario in which the adversary totally controls the initiator (possesses the secret key* $\mathsf{sk}_{\mathcal{I}}$ *of the initiator* $\mathcal{I}$*), and controls the randomness of the proxy device.*

We say that the scheme is secure if the probability of `FT1` *is negligible and the probability of* `FT2` *is negligible.*

4.1.5 Cryptanalysis In a Stronger Security Model

In Theorem 3 we show that a typical Schnorr based PSC [53] (left column of Tab. 4.2) is not secure in our model. This motivates our work in mitigating such attacks and provide a more secure proxy signcryption scheme in a stronger model.

Theorem 3. *The scheme presented in left column of Tab. 4.2 is not secure in our new strong unforgeability model.*

Proof. FT1: After system is initialized, the attacker $\mathcal{A}_{\mathtt{I}}^{\mathcal{O}_{\mathsf{Id}}^{\bar{r}}}(\mathsf{par}, \mathsf{sk}_{\mathcal{P}}, \mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{R}})$ selects $\bar{r}_1 \leftarrow_{\$} \mathbb{Z}_q$ and queries $\mathcal{O}_{\mathsf{Id}}^{\bar{r}}$ for an arbitrary space Ω, i.e. $(s_1, R_1) = \sigma_1 \leftarrow \mathcal{O}_{\mathsf{Id}}^{\bar{r}}(\Omega)$. It computes $\mathsf{sk}_{\mathcal{I}} = \frac{s_1 - \bar{r}_1}{\mathcal{H}_1(\Omega, R_1)}$. Thus, with the initiator key it can compute the new warrant over a new Ω^* and a valid signcryption over a new message $m^* \in \Omega^*$.
FT2: After system is initialized, the attacker $\mathcal{A}_{\mathtt{II}}^{\mathcal{O}_{\mathsf{SC}}^{\bar{r}}}(\mathsf{par}, \mathsf{sk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{P}}, \mathsf{pk}_{\mathcal{R}})$ selects $\bar{r}_2 \leftarrow_{\$} \mathbb{Z}_q$ and queries $\mathcal{O}_{\mathsf{SC}}^{\bar{r}}(\mathsf{par}, \sigma_1, \Omega, \mathsf{pk}_{\mathcal{R}}, m) \rightarrow \sigma_2 = (s_2, R_1, h_2, c)$. Then it computes: $R_2 = g^{\bar{r}_2}$, $k_2 = \mathcal{H}_3(R_2, 2)$, and $\mathsf{sk}_{\mathcal{P}} = \frac{s_2 - \bar{r}_2}{\mathcal{H}_2(m, k_2)} - s_1$. With the secret key $\mathsf{sk}_{\mathcal{P}}$ of the proxy it can signcrypt any new message m^* for any Ω^*. □

4.1.6 An Improved Signcryption Scheme

Proposed PSC construction

In the right column of Tab. 4.2 we propose our stronger proxy signcryption scheme. The construction mimics a typical scheme architecture with a double Schnorr signature approach, as seen in the left column. In that scheme, the first signature computed by Id procedure is performed by $\mathcal{I}$ over the message space Ω. The second signature occurs in the SC procedure, where a proxy $\mathcal{P}$ uses a linear combination of keys $\mathsf{sk}_{\mathcal{P}} + s_1$ as a secret key for the second Schnorr signature. Note that the leakage of randomness r_1 allows the attacker to obtain the long term key $\mathsf{sk}_{\mathcal{I}}$ and produce a warrant over an arbitrary message space Ω^*, hence s_1 is included plain as part of σ_1. Moreover, the leakage of r_2 enables the attacker to get $\mathsf{sk}_{\mathcal{P}} + s_1$ and signcrypt any message m^* from Ω over the given warrant σ_1. We state those vulnerabilities in Thm. 3. To mitigate these threats, we propose a scheme, set in a group $\mathbb{G}$ with a symmetric pairing function $\hat{e}$. In our proposition the linear equations for s_1 in Id, and s_2 in SC procedures, are shifted into exponents of two group elements: $\hat{S}_1 = \dot{g}^{s_1}$ and $\hat{S}_2 = \ddot{g}^{s_2}$, for two new ad-hoc, and deterministically (via the function $\mathcal{H}_g$) created generators of $\mathbb{G}$: $\dot{g}$ and $\ddot{g}$ respectively. The verification is performed via the bilinear property of the chosen pairing function $\hat{e}$, which we state in the Thm. 11. Here we highlight the advantage over other three-party schemes (identity-based signatures and authentication [98, 100, 91]), where just one new generator is used for the proxy party procedure, mitigating the leakage of the randomness from its device only.

Our scheme also increases the secrecy level over the example scheme [53]. In both schemes, the secrecy depends on the chosen encryption scheme $\mathcal{E}$,

Typical scheme [53]		**Proposed scheme**
$\mathsf{ParGen}(\lambda)$:		$\mathsf{ParGen}(\lambda)$:
$(\mathbb{G}, g, q) \leftarrow Gen(1^\lambda)$		$(\mathbb{G}, \mathbb{G}_T, g, g_T, q, \hat{e}) \leftarrow Gen_{BP}(1^\lambda)$
$\mathcal{E} = (E, D, K, \Omega)$		$\mathcal{E} = (E, D, K, \Omega)$
$\mathcal{H}_1 : \{0,1\}^* \rightarrow \mathbb{Z}_q$		$\mathcal{H}_1 : \{0,1\}^* \rightarrow \mathbb{Z}_q$
$\mathcal{H}_2 : \{0,1\}^* \rightarrow \mathbb{Z}_q$		$\mathcal{H}_2 : \{0,1\}^* \rightarrow \mathbb{Z}_q$
$\mathcal{H}_3 : \{0,1\}^* \rightarrow K$		$\mathcal{H}_3 : \{0,1\}^* \rightarrow K$
		$\mathcal{H}_g : \{0,1\}^* \rightarrow \mathbb{G}$
$\mathsf{par} = (\mathbb{G}, g, q, \mathcal{E}, \mathcal{H}_1, \mathcal{H}_2, h_3)$		$\mathsf{par} = (\mathbb{G}, \mathbb{G}_T, g, g_T, q, \mathcal{E}, \hat{e}, \mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_g)$
Id $(\mathsf{par}, \mathsf{sk}_\mathcal{I}, \Omega)$:		Id $(\mathsf{par}, \mathsf{sk}_\mathcal{I}, \Omega)$:
$r_1 \leftarrow_\$ \mathbb{Z}_q, R_1 = g^{r_1}$		$r_1 \leftarrow_\$ \mathbb{Z}_q, R_1 = g^{r_1}$
$s_1 = r_1 + \mathsf{sk}_\mathcal{I} \cdot \mathcal{H}_2(\Omega, R_1)$		$s_1 = r_1 + \mathsf{sk}_\mathcal{I} \cdot \mathcal{H}_1(\Omega, R_1)$
		$\dot{g} = \mathcal{H}_g(\Omega, R_1), \hat{S}_1 = \dot{g}^{s_1}$
$\sigma_1 = (s_1, R_1)$		$\sigma_1 = (\hat{S}_1, R_1)$
Ver $(\mathsf{par}, \mathsf{pk}_\mathcal{I}, \sigma_1, \Omega)$:		Ver $(\mathsf{par}, \mathsf{pk}_\mathcal{I}, \sigma_1, \Omega)$:
$h_1 = \mathcal{H}_1(\Omega, R_1)$		$\dot{g} = \mathcal{H}_g(\Omega, R_1)$, $h_1 = \mathcal{H}_1(\Omega, R_1)$
Accept iff		Accept iff
$g^{s_1} == R_1 \cdot \mathsf{pk}_\mathcal{I}^{h_1}$		$\hat{e}(\hat{S}_1, g) == \hat{e}(\dot{g}, R_1 \cdot \mathsf{pk}_\mathcal{I}^{h_1})$
SC $(\mathsf{par}, \mathsf{sk}_\mathcal{P}, \sigma_1, \Omega, \mathsf{pk}_\mathcal{R}, m)$:		SC $(\mathsf{par}, \mathsf{sk}_\mathcal{P}, \sigma_1, \Omega, \mathsf{pk}_\mathcal{R}, m)$:
$r_2 \leftarrow_\$ \mathbb{Z}_q, R_2 = g^{r_2}, \hat{R}_2 = \mathsf{pk}_\mathcal{R}^{r_2}$		$r_2 \leftarrow_\$ \mathbb{Z}_q, R_2 = g^{r_2}, \hat{R}_2 = \mathsf{pk}_\mathcal{R}^{r_2}, k = \mathsf{pk}_\mathcal{R}^{\mathsf{sk}_\mathcal{P}}$
$k_1 = \mathcal{H}_3(\hat{R}_2, 1), k_2 = \mathcal{H}_3(\hat{R}_2, 2)$		$k_1 = \mathcal{H}_3(R_2, k, 1), k_2 = \mathcal{H}_3(R_2, 2)$
$h_2 = \mathcal{H}_2(m, k_2)$		$\ddot{g} = \mathcal{H}_g(\Omega, R_1, k_2, m)$
$s_2 = r_2 + (\mathsf{sk}_\mathcal{P} + s_1) \cdot h_2$		$s_2 = r_2 + \mathsf{sk}_\mathcal{P} \cdot \mathcal{H}_2(\Omega, R_1, k_2, m)$
		$\hat{S}_2 = \ddot{g}^{s_2}, \hat{S} = \hat{S}_1 \hat{S}_2$
$c = E_{k_1}(m)$		$c = E_{k_1}(m)$
$\sigma_2 = (s_2, R_1, h_2, c)$		$\sigma_2 = (\hat{S}, R_1, \hat{R}_2, c)$
USC $(\mathsf{par}, \mathsf{sk}_\mathcal{R}, \mathsf{pk}_\mathcal{I}, \mathsf{pk}_\mathcal{P}, \sigma_2, \Omega)$:		USC $(\mathsf{par}, \mathsf{sk}_\mathcal{R}, \mathsf{pk}_\mathcal{I}, \mathsf{pk}_\mathcal{P}, \sigma_2, \Omega)$:
$h_1 = \mathcal{H}_1(\Omega, R_1)$		$R_2 = \hat{R}_2^{1/\mathsf{sk}_\mathcal{R}}, k = \mathsf{pk}_\mathcal{P}^{\mathsf{sk}_\mathcal{R}}, k_1 = \mathcal{H}_3(R_2, k, 1), k_2 = \mathcal{H}_3(R_2, 2)$
$\hat{R}_2 = ((g^{s_2})/(\mathsf{pk}_\mathcal{P} \cdot R_1 \mathsf{pk}_\mathcal{I}^{h_1})^{h_2})^{\mathsf{sk}_\mathcal{R}}$		$m = D_{k_1}(c)$
$k_1 = \mathcal{H}_3(\hat{R}_2, 1), k_2 = \mathcal{H}_3(\hat{R}_2, 2)$		$\dot{g} = \mathcal{H}_g(\Omega, R_1), \ddot{g} = \mathcal{H}_g(\Omega, R_1, k_2, m)$
$m = D_{k_1}(c)$		$h_1 = \mathcal{H}_1(\Omega, R_1), h_2 = \mathcal{H}_2(\Omega, R_1, k_2, m)$
Accept iff		Accept iff
$m \in \Omega$ and $h_2 == \mathcal{H}_2(m, k_2)$		$m \in \Omega$ and $\hat{e}(\hat{S}, g) == \hat{e}(\dot{g}, R_1 \mathsf{pk}_\mathcal{I}^{h_1}) \hat{e}(\ddot{g}, R_2 \mathsf{pk}_\mathcal{P}^{h_2})$

Table 4.2: Schnorr based scheme construction to the left and our proposed, improved construction in the right column.

and the way the symmetric key k_1 is established between the proxy $\mathcal{P}$, and the receiver $\mathcal{R}$. Note that in the example scheme the secrecy is broken once the ephemeral value r_2 is revealed. To mitigate that we propose the usage of an additional intermediate static Diffie-Hellman key $k = \mathsf{pk}_{\mathcal{R}}^{\mathsf{sk}_{\mathcal{P}}} = \mathsf{pk}_{\mathcal{P}}^{\mathsf{sk}_{\mathcal{R}}}$, computable locally and independently by the proxy and the receiver, and included as an additional input to the hash function while computing $k_1 = \mathcal{H}_3(R_2, k, 1)$. Note that this does not require to include any additional data in σ_2. This technique is typical for the authenticated key exchange protocols (thus the thorough secrecy discussion for k_1 is omitted due to space constraints).

4.1.7 Security Analysis

Theorem 4. *The* PSC *scheme proposed in the right-hand side of Tab. 4.2 is correct.*

Proof. Obviously $\hat{R}_2 = \mathsf{pk}_{\mathcal{R}}^{r_2} = g^{\mathsf{sk}_{\mathcal{R}} r_2}$, therefore $R_2 = g^{r_2} = \hat{R}_2^{1/\mathsf{sk}_{\mathcal{R}}}$.
We have: $k = \mathsf{pk}_{\mathcal{R}}^{\mathsf{sk}_{\mathcal{P}}} = \mathsf{pk}_{\mathcal{P}}^{\mathsf{sk}_{\mathcal{R}}}$, $k_1 = \mathcal{H}_3(R_2, k, 1)$ and $k_2 = \mathcal{H}_3(R_2, 2), m = D_{k_1}(c)$. For generators $\dot{g} = \mathcal{H}_g(\Omega, R_1)$, $\ddot{g} = \mathcal{H}_g(\Omega, R_1, k_2, m)$, and hashes $h_1 = \mathcal{H}_1(\Omega, R_1)$, $h_2 = \mathcal{H}_2(\Omega, R_1, k_2, m)$, it holds:

Ver correctness

$$\begin{aligned}\hat{e}(\hat{S}_1, g) &= \hat{e}(\dot{g}^{s_1}, g) = \hat{e}(\dot{g}, g^{s_1}) \\ &= \hat{e}(\dot{g}, g^{r_1 + \mathsf{sk}_{\mathcal{I}} h_1}) = \hat{e}(\dot{g}, R_1 \cdot \mathsf{pk}_{\mathcal{I}}^{h_1}),\end{aligned}$$

USC correctness

$$\begin{aligned}\hat{e}(\hat{S}, g) &= \hat{e}(\hat{S}_1 \hat{S}_2, g) = \hat{e}(\hat{S}_1, g) \cdot \hat{e}(\hat{S}_2, g) \\ &= \hat{e}(\dot{g}^{s_1}, g) \cdot \hat{e}(\ddot{g}^{s_2}, g) \\ &= \hat{e}(\dot{g}, g^{s_1}) \cdot \hat{e}(\ddot{g}, g^{s_2}) \\ &= \hat{e}(\dot{g}, g^{r_1 + \mathsf{sk}_{\mathcal{I}} h_1}) \cdot \hat{e}(\ddot{g}, g^{r_2 + \mathsf{sk}_{\mathcal{P}} h_2}) \\ &= \hat{e}(\dot{g}, R_1 \mathsf{pk}_{\mathcal{I}}^{h_1}) \cdot e(\ddot{g}, R_2 \mathsf{pk}_{\mathcal{P}}^{h_2}).\end{aligned}$$

□

Theorem 5. *The* PSC *scheme proposed in the right-hand side of Fig. 4.2 is unforgeable in the sense of Definition 20, i.e. is secure against* FT1.

Proof. `Init:` Let (g, g^α, g^β) be an instance of the GDH problem in $\mathsf{par} = (\mathbb{G}, \mathbb{G}_T, g, g_T, q, \hat{e})$. We setup the system s.t. $\mathsf{sk}_{\mathcal{P}} \leftarrow_\$ \mathbb{Z}_q, \mathsf{pk}_{\mathcal{P}} = g^{\mathsf{sk}_{\mathcal{P}}}$, $\mathsf{sk}_{\mathcal{R}} \leftarrow_\$ \mathbb{Z}_q, \mathsf{pk}_{\mathcal{R}} = g^{\mathsf{sk}_{\mathcal{R}}}$, and $\mathsf{pk}_{\mathcal{I}} = g^\alpha$. Thus the unknown $\mathsf{sk}_{\mathcal{I}}$ equals the unknown α. We run the adversary $\mathcal{A}_{\mathrm{I}}^{\mathcal{O}_{\mathsf{Id}}^{\bar{r}}}(\mathsf{par}, \mathsf{sk}_{\mathcal{P}}, \mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{R}})$ the access to $\mathcal{O}_{\mathsf{Id}}^{\bar{r}}$, and $\mathcal{O}_{\mathcal{H}_g}$ oracles.

`Serving` $\mathcal{O}_{\mathcal{H}_g}$ `Oracle` : We allow ℓ fresh inputs to the $\mathcal{O}_{\mathcal{H}_g}$ oracle. We choose the random index $j \leftarrow_\$ \{1, \ldots, \ell\}$, which denotes the j-th invocation of $\mathcal{O}_{\mathcal{H}_g}$, for which we assume the forgery will happen.

- On i-th ($i \neq j$) fresh input: $d \leftarrow_\$ \mathbb{Z}_q$, and g^d is registered as a hash value in the ROM table for $\mathcal{H}_g$. Return g^d as the output.
- On j-th fresh input: g^β is registered as a hash value in the ROM table for $\mathcal{H}_g$. Return g^β as the output.

`Serving` $\mathcal{O}_{\mathsf{Id}}^{\bar{\mathbf{r}}}$ `Oracle` : On input $\Omega, \bar{r}_1$ we compute $\bar{R}_1 = g^{\bar{r}_1}$, serve the call $\mathcal{O}_{\mathcal{H}_g}(\Omega, \bar{R}_1)$, namely locate and return $(\dot{g}_1 = g^d, d)$ if $\Omega, \bar{R}_1$ was not j-th fresh input to $\mathcal{O}_{\mathcal{H}_g}$, then compute $\hat{S}_1 = (\bar{R}_1 \mathsf{pk}_{\mathcal{I}}^{\mathcal{H}_1(\Omega, \bar{R}_1)})^d$, return $\sigma_1 = (\hat{S}_1, \bar{R}_1)$. Note that in this case the $\mathsf{Ver}(\mathsf{par}, \mathsf{pk}_{\mathcal{I}}, \sigma_1, \Omega) = 1$ as $\hat{e}(\hat{S}_1, g) = \hat{e}((\bar{R}_1 \mathsf{pk}_{\mathcal{I}}^{\mathcal{H}_1(\Omega, \bar{R}_1)})^d, g) = \hat{e}((g^d)^{\bar{r}_1 + \mathsf{sk}_{\mathcal{I}} \mathcal{H}_1(\Omega, \bar{R}_1)}, g) = \hat{e}(g^d, \bar{R}_1 \cdot \mathsf{pk}_{\mathcal{I}}^{h_1}) = \hat{e}(\dot{g}, \bar{R}_1 \cdot \mathsf{pk}_{\mathcal{I}}^{h_1})$.

`Forgery Stage` : Under the *Forking Lemma* for the hash $\mathcal{H}_1$ the attacker returns two valid signcryptions with the same randomness R_1: $\sigma_2 = (\hat{S}, R_1, \hat{R}_2, c)$, $\sigma_2' = (\hat{S}', R_1, \hat{R}_2', c')$, s.t. $\hat{S} = \hat{S}_1 \hat{S}_2$, $\hat{S}' = \hat{S}_1' \hat{S}_2'$, and the output $\mathcal{H}_1(\Omega, R_1)$ equals h_1 in $\hat{S}_1$ of σ_2 but equals $h_1' \neq h_1$ in $\hat{S}_1'$ of σ_2'. With the non-negligible probability $1/\ell$, the forking happens on j-th fresh input (Ω, R_1) to $\mathcal{O}_{\mathcal{H}_g}$, the $\dot{g}$ equals to g^β in both tuples σ_2, σ_2'. We compute $R_2 = \hat{R}_2^{1/\mathsf{sk}_{\mathcal{R}}}$, $k = \mathsf{pk}_{\mathcal{P}}^{\mathsf{sk}_{\mathcal{R}}}$, $k_1 = \mathcal{H}_3(R_2, k, 1)$, $k_2 = \mathcal{H}_3(R_2, 2)$, $m = D_{k_1}(c)$, locate $(\ddot{g} = g^d, d)$ in ROM table for $\mathcal{O}_{\mathcal{H}_g}(\Omega, R_1, k_2, m)$, compute $\hat{S}_2 = (R_2 \mathsf{pk}_{\mathcal{P}}^{\mathcal{H}_2(\Omega, R_1, k_2, m)})^d$, and eventually $\hat{S}_1 = \hat{S}/\hat{S}_2$. Compute $\hat{S}_1'$ from σ_2' in a similar way. We have $\hat{S}_1/\hat{S}_1' = \hat{g}_1^{\mathsf{sk}_{\mathcal{I}}(h_1 - h_1')} = (g^\beta)^{\mathsf{sk}_{\mathcal{I}}(h_1 - h_1')}$. Therefore we could compute $g^{\alpha\beta} = (\hat{S}_1/\hat{S}_1')^{(1/(h_1 - h_1'))}$, breaking the given instance of GDH.

□

Theorem 6. *The* PSC *scheme proposed in the right-hand side of Fig. 4.2 is unforgeable in the sense of Definition 20, i.e. is secure against* `FT2`.

Proof. `Init`: Let (g, g^α, g^β) be an instance of the GDH problem in $\mathsf{par} = (\mathbb{G}, \mathbb{G}_T, g, g_T, q, \hat{e})$. We setup the system s.t. $\mathsf{sk}_{\mathcal{I}} \leftarrow_\$ \mathbb{Z}_q, \mathsf{pk}_{\mathcal{I}} = g^{\mathsf{sk}_{\mathcal{I}}}$, $\mathsf{sk}_{\mathcal{R}} \leftarrow_\$ \mathbb{Z}_q, \mathsf{pk}_{\mathcal{R}} = g^{\mathsf{sk}_{\mathcal{R}}}$, and $\mathsf{pk}_{\mathcal{P}} = g^\alpha$. Thus the unknown $\mathsf{sk}_{\mathcal{P}}$ equals the unknown α. We run the adversary $\mathcal{A}_{\mathtt{II}}^{\mathcal{O}_{\mathsf{SC}}^{\bar{r}}}(\mathsf{par}, \mathsf{sk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{P}}, \mathsf{pk}_{\mathcal{R}})$ the access to $\mathcal{O}_{\mathsf{SC}}^{\bar{r}}$, and $\mathcal{O}_{\mathcal{H}_g}$ oracles.

`Serving` $\mathcal{O}_{\mathcal{H}_g}$ `Oracle` : This stage is same as previously, namely that we allow ℓ fresh inputs to the $\mathcal{O}_{\mathcal{H}_g}$ oracle. We choose the random index

$j \leftarrow_\$ \{1, \ldots, \ell\}$, which denotes the j-th invocation of $\mathcal{O}_{\mathcal{H}_g}$, for which we assume the forgery will happen.

- On i-th ($i \neq j$) fresh input: $d \leftarrow_\$ \mathbb{Z}_q$, and g^d is registered as a hash value in the ROM table for $\mathcal{H}_g$. Return the g^d as the output.
- On j-th fresh input: g^β is registered as ahash value in the ROM table for $\mathcal{H}_g$. Return g^β as the output.

`Serving` $\mathcal{O}^{\bar{\mathsf{f}}}_{\mathsf{SC}}$ `Oracle` : On input $\mathsf{par}, \sigma_1, \Omega, \mathsf{pk}_{\mathcal{R}}, m$, and randomness $\bar{r}_2$, we first verify σ_1: $\hat{e}(\hat{S}_1, g) = \hat{e}(\mathcal{H}_g(\Omega, \bar{R}_1), \bar{R}_1 \cdot \mathsf{pk}_{\mathcal{I}}^{\mathcal{H}_1(\Omega, \bar{R}_1)})$. We compute $\bar{R}_2 = g^{\bar{r}_2}$, $k = \mathsf{pk}_{\mathcal{P}}^{\mathsf{sk}_{\mathcal{R}}}$, $k_1 = \mathcal{H}_3(R_2, k, 1)$, $k_2 = \mathcal{H}_3(R_2, 2)$, $c = E_{k_1}(m)$. We serve the call $\mathcal{O}_{\mathcal{H}_g}(\Omega, R_1, k_2, m)$ and consider two cases: if (Ω, R_1, k_2, m) was not j-th fresh input to $\mathcal{O}_{\mathcal{H}_g}$ then locate and return $(\ddot{g} = g^d, d)$. Next compute $\hat{S}_2 = (\bar{R}_2 \mathsf{pk}_{\mathcal{P}}^{\mathcal{H}_2(\Omega, \bar{R}_1, k_2, m)})^d$, $\hat{S} = \hat{S}_1 \hat{S}_2$ and return $\sigma_2 = (\hat{S}, R_1, \hat{R}_2, c)$. In this case, as $\hat{e}(\hat{S}_2, g) = \hat{e}((\bar{R}_2 \mathsf{pk}_{\mathcal{P}}^{\mathcal{H}_2(\Omega, R_1, k_2, m)})^d, g) = \hat{e}((g^d)^{\bar{r}_2 + \mathsf{sk}_{\mathcal{I}} \mathcal{H}_2(\Omega, R_1, k_2, m)}, g) = \hat{e}(g^d, \bar{R}_2 \cdot \mathsf{pk}_{\mathcal{P}}^{h_2}) = \hat{e}(\ddot{g}, \bar{R}_2 \cdot \mathsf{pk}_{\mathcal{I}}^{h_2})$, the verification holds:

$$\hat{e}(\hat{S}, g) = \hat{e}(\mathcal{H}_g(\Omega, \bar{R}_1), \bar{R}_1 \cdot \mathsf{pk}_{\mathcal{I}}^{\mathcal{H}_1(\Omega, \bar{R}_1)}) \hat{e}(\ddot{g}, \bar{R}_2 \cdot \mathsf{pk}_{\mathcal{I}}^{h_2}).$$

Otherwise, if the call to $\mathcal{O}_{\mathcal{H}_g}$ was the j-th fresh input, we abort. Note that the probability of not aborting, but providing the verifiable signature is non-negligable.

`Forgery Stage` : Under the *Forking Lemma* for the hash $\mathcal{H}_2$ the attacker returns two valid signcryptions with the same randomness $\hat{R}_2$: $\sigma_2 = (\hat{S}, R_1, \hat{R}_2, c)$, $\sigma_2' = (\hat{S}', R_1', \hat{R}_2, c')$, s.t. $\hat{S} = \hat{S}_1 \hat{S}_2$, $\hat{S}' = \hat{S}_1' \hat{S}_2'$, and the output $\mathcal{H}_2(\Omega, R_1, k_2, m)$ equals h_2 in $\hat{S}_2$ of σ_2 but equals $h_2' \neq h_2$ in $\hat{S}_2'$ of σ_2'. With the non-negligible probability $1/\ell$, forking happens on j-th fresh input (Ω, R_1, k_2, m) to $\mathcal{O}_{\mathcal{H}_g}$, the $\ddot{g}$ equals to g^β in both tuples σ_2, σ_2'. We locate $(\dot{g} = g^d, d)$ in ROM table for $\mathcal{O}_{\mathcal{H}_g}(\Omega, R_1)$, compute $\hat{S}_1 = (R_1 \mathsf{pk}_{\mathcal{I}}^{\mathcal{H}_1(\Omega, R_1)})^d$, and eventually $\hat{S}_2 = \hat{S} / \hat{S}_1$. Compute $\hat{S}_2'$ from σ_2' in a similar way. We have $\hat{S}_2 / \hat{S}_2' = \hat{g}_2^{\mathsf{sk}_{\mathcal{P}}(h_2 - h_2')} = (g^\beta)^{\mathsf{sk}_{\mathcal{P}}(h_2 - h_2')}$. Therefore we could compute $g^{\alpha\beta} = (\hat{S}_2 / \hat{S}_2')^{(1/(h_2 - h_2'))}$, breaking the given instance of GDH.

□

4.1.8 Performance Analysis

Our proposed scheme is based on, and proven secure over symmetric pairings, however a similar scheme utilizing asymmetric pairings would still be secure

under similar security analysis (future work). We conclude from benchmarks, run on a proof-of-concept of such a scheme, that a version with asymmetric pairings, would be of interest since the average computational complexity lies within reasonable, practical limits. All testing was performed on a MacBook Pro, with an Intel Core i5 2,7GHz. We used the MCL library [120] with BLS12_381 curve for pairings. As for comparison, the original scheme was implemented entirely in group `G2`, while our modified scheme uses `G1` for exponent hiding.

The substantial operations together with the average timings are listed in Tab. 4.8. We compare their numbers for each procedure. New versions are marked with (*). The bottom row shows the total timings per each procedure, which includes all algebraic operations and encryption/decryption via one-time-pad. These are compliant to the standards e.g. in [127].

		Id	Id*	Ver	Ver*	SC	SC*	USC	USC*
`G1:exp`	0.110	-	1	-	-	-	1	-	-
`G2:exp`	0.204	1	1	2	1	2	2	4	3
`G1:hashTo`	0.353	-	1	-	1	-	1	-	2
`Pairing`	1.808	-	-	-	2	-	-	-	3
Total Time [ms]		1.019	1.271	1.237	4.956	0.431	1.915	1.450	7.729

Table 4.3: Complexity and time assessment (5000 runs).

4.1.9 Conclusion

In this paper we analyze a proxy signcryption scheme, resistant to randomness injection attacks for railway and traffic communication. We introduce a stronger model of security in which we consider potential malicious ephemeral setup at both the initiator and proxy parties, which may occur due to untrusted hardware usage. Therefore, we propose a novel scheme which withstands attacks on such weak devices, where physical tampering is likely to occur, for the initiator and the proxy. This is crucial for all levels of security and passenger safety where trains and vehicles interact with intermediate track- and roadside units, and further to cloud based servers and control centers. Our scheme is therefore resistant to *ephemeral key leakage/setup*, further implying secure to the threat of extracting the static long term secret key - as would not be the case in the original scheme. This is of critical importance since connected railway- and vehicle infrastructure is supposed to provide massive, distributed networking dependent on secure communication in an utterly heterogeneous running environment, but at the cost of signa-

ture verification for every message exchange. We therefore conclude that this paper have contributed to addressing security challenges Sec1 and Sec3-Sec5 since we provide secure means of transmitting connected infrastructure data (Sec1), mitigate impersonation attacks (Sec3) in a proxy environment (Sec4) using stronger ephemeral leakage model (Sec4).

4.1.10 An Additional Proof of Security in the Asymmetric Configuration

This subsection ***is not part of the previously published research*** and contains a proof of security with the asymmetric configuration, hence ***an additional contribution*** for this thesis. In this version we use a bilinear pairing $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$. We have $\langle g_1 \rangle = \mathbb{G}_1 \neq \langle g_2 \rangle = \mathbb{G}_2$, namely two different groups. The protocol is therefore adjusted into π'_{PSC}.

Definition 21. *The proposed protocol π_{PSC} in the symmetric configuration is adjusted into an asymmetric configuration with the the pairing $\hat{e}$, denoted as the protocol π'_{PSC}. All participants' keys are the same but extended with an additional part $\mathsf{pk} = g_2^{\mathsf{sk}}$ of the public key, i.e., for party i we would compute $\mathsf{sk}_i = a$ for $a \leftarrow_{\$} \mathbb{Z}$ and $\mathsf{pk}_i = (\mathsf{pk}_{(i,1)} = g_1^a, \mathsf{pk}_{(i,2)} = g_2^a)$. For π'_{PSC} we thus have the key pairs $\big(\mathsf{sk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{I}} = (\mathsf{pk}_{(\mathcal{I},1)}, \mathsf{pk}_{(\mathcal{I},2)})\big)$ for initiator $\mathcal{I}$, $\big(\mathsf{sk}_{\mathcal{P}}, \mathsf{pk}_{\mathcal{P}} = (\mathsf{pk}_{(\mathcal{P},1)}, \mathsf{pk}_{(\mathcal{P},2)})\big)$ for proxy $\mathcal{P}$ and $\big(\mathsf{sk}_{\mathcal{R}}, \mathsf{pk}_{\mathcal{R}} = (\mathsf{pk}_{(\mathcal{R},1)}, \mathsf{pk}_{(\mathcal{R},2)})\big)$ for receiver $\mathcal{R}$.*

Remark: The adjusted scheme π'_{PSC} is summarized in Tab. 4.4

Definition 22. *We re-define the security experiment for unforgeability under impersonation attacks (UIA) from the previous security analysis in the following way, denoted $\mathsf{Exp}^{UIA,\lambda,\ell}$:*

`Init`: *Security parameters and keys are generated, i.e.,* $\mathsf{par} \leftarrow \mathsf{ParGen}(\lambda)$,
$(\mathsf{sk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{I}} = (\mathsf{pk}_{(\mathcal{I},1)}, \mathsf{pk}_{(\mathcal{I},2)})) \leftarrow \mathsf{KeyGen}(\mathsf{par}, \mathcal{I})$,
$(\mathsf{sk}_{\mathcal{P}}, \mathsf{pk}_{\mathcal{P}} = (\mathsf{pk}_{(\mathcal{P},1)}, \mathsf{pk}_{(\mathcal{P},2)})) \leftarrow \mathsf{KeyGen}(\mathsf{par}, \mathcal{P})$,
$(\mathsf{sk}_{\mathcal{R}}, \mathsf{pk}_{\mathcal{R}} = (\mathsf{pk}_{(\mathcal{R},1)}, \mathsf{pk}_{(\mathcal{R},2)})) \leftarrow \mathsf{KeyGen}(\mathsf{par}, \mathcal{R})$.

The Id `Oracle`, SC `Oracle`, `FT1` *and* `FT2` *are same as in the experiment in the symmetric configuration. The scheme is secure if $\boldsymbol{Adv}(\mathcal{A}, \mathsf{Exp}^{UIA,\lambda,\ell})$ is negligible in λ for forgery types I and II.*

Theorem 7. *The proposed scheme π'_{PSC} is secure under the security model of* `Forgery Type I` *as π_{PSC} is, but with the asymmetric pairing configuration.*

Adjusted scheme π'_{PSC}

$\mathsf{ParGen}(\lambda)$:
$(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, g_T, q, \hat{e}) \leftarrow Gen_{BP}(1^\lambda)$
$\mathcal{E} = (E, D, K, \Omega)$
$\mathcal{H}_1 : \{0,1\}^* \rightarrow \mathbb{Z}_q$
$\mathcal{H}_2 : \{0,1\}^* \rightarrow \mathbb{Z}_q$
$\mathcal{H}_3 : \{0,1\}^* \rightarrow K$
$\mathcal{H}_{g_1} : \{0,1\}^* \rightarrow \mathbb{G}_1$
$\mathsf{par} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, g_T, q, \mathcal{E}, \hat{e}, \mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_{g_1})$

$\mathsf{Id}\ (\mathsf{par}, \mathsf{sk}_{\mathcal{I}}, \Omega)$:
$r_1 \leftarrow_\$ \mathbb{Z}_q, R_1 = g_2^{r_1}$
$s_1 = r_1 + \mathsf{sk}_{\mathcal{I}} \cdot \mathcal{H}_1(\Omega, R_1)$
$\dot{g}_1 = \mathcal{H}_{g_1}(\Omega, R_1), \hat{S}_1 = \dot{g}_1^{s_1}$
$\sigma_1 = (\hat{S}_1, R_1)$

$\mathsf{Ver}\ (\mathsf{par}, \mathsf{pk}_{\mathcal{I}}, \sigma_1, \Omega)$:
$\dot{g}_1 = \mathcal{H}_{g_1}(\Omega, R_1),\ h_1 = \mathcal{H}_1(\Omega, R_1)$
Accept iff
$\hat{e}(\hat{S}_1, g_2) == \hat{e}(\dot{g}_1, R_1 \cdot \mathsf{pk}_{(\mathcal{I},2)}^{h_1})$

$\mathsf{SC}\ (\mathsf{par}, \mathsf{sk}_{\mathcal{P}}, \sigma_1, \Omega, \mathsf{pk}_{\mathcal{R}}, m)$:
$r_2 \leftarrow_\$ \mathbb{Z}_q, R_2 = g_2^{r_2}, \hat{R}_2 = \mathsf{pk}_{(\mathcal{R},2)}^{r_2}, k = \mathsf{pk}_{(\mathcal{R},1)}^{\mathsf{sk}_{\mathcal{P}}}$
$k_1 = \mathcal{H}_3(R_2, k, 1), k_2 = \mathcal{H}_3(R_2, 2)$
$\ddot{g}_1 = \mathcal{H}_{g_1}(\Omega, R_1, k_2, m)$
$s_2 = r_2 + \mathsf{sk}_{\mathcal{P}} \cdot \mathcal{H}_2(\Omega, R_1, k_2, m)$
$\hat{S}_2 = \ddot{g}_1^{s_2}, \hat{S} = \hat{S}_1 \hat{S}_2$
$c = E_{k_1}(m)$
$\sigma_2 = (\hat{S}, R_1, \hat{R}_2, c)$

$\mathsf{USC}\ (\mathsf{par}, \mathsf{sk}_{\mathcal{R}}, \mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{P}}, \sigma_2, \Omega)$:
$R_2 = (\hat{R}_2)^{1/\mathsf{sk}_{\mathcal{R}}}, k = \mathsf{pk}_{(\mathcal{P},1)}^{\mathsf{sk}_{\mathcal{R}}}, k_1 = \mathcal{H}_3(R_2, k, 1), k_2 = \mathcal{H}_3(R_2, 2)$
$m = D_{k_1}(c)$
$\dot{g}_1 = \mathcal{H}_{g_1}(\Omega, R), \ddot{g}_1 = \mathcal{H}_{g_1}(\Omega, R_1, k_2, m)$
$h_1 = \mathcal{H}_1(\Omega, R_1), h_2 = \mathcal{H}_2(\Omega, R_1, k_2, m)$
Accept iff
$m \in \Omega$ and $\hat{e}(\hat{S}, g_2) == \hat{e}(\dot{g}_1, R_1 \cdot \mathsf{pk}_{(\mathcal{I},2)}^{h_1}) \cdot \hat{e}(\ddot{g}_1, R_2 \cdot \mathsf{pk}_{(\mathcal{P},2)}^{h_2})$

Table 4.4: The adjusted scheme π'_{PSC} using the asymmetric pairing configuration.

Proof. The proof is by contradiction; we assume the existence of an adversary $\mathcal{A}$ for which $\mathbf{Adv}(\mathcal{A}, \mathsf{Exp}^{UIA,\lambda,\ell})$ is non-negligible. The experiment is then executed as follows:

`Init Stage:` The experiment starts by running $\mathsf{par} \leftarrow \mathsf{ParGen}(\lambda)$ and instantiate PG-CDH with $(g_1, g_1^{\alpha}, g_1^{\beta}, g_2, g_2^{\alpha}, g_2^{\beta})$. The adversary is given access to hash oracle $\mathcal{O}_{\mathcal{H}}$ and given the public key components for party $\mathcal{I}$ we have $\mathsf{pk}_{\mathcal{I}} = (\mathsf{pk}_{(\mathcal{I},1)} = g_1^{\alpha}, \mathsf{pk}_{(\mathcal{I},2)} = g_2^{\alpha})$. Thus the unknown $\mathsf{sk}_{\mathcal{I}}$ equals the unknown α. The remaining parties setup their keys as per the protocol, i.e., $\mathsf{sk}_{\mathcal{P}} \leftarrow_{\$} \mathbb{Z}_q$ and $\mathsf{sk}_{\mathcal{R}} \leftarrow_{\$} \mathbb{Z}_q$, hence $\mathsf{pk}_{\mathcal{P}} = (\mathsf{pk}_{(\mathcal{P},1)} = g_1^{\mathsf{sk}_{\mathcal{P}}}, \mathsf{pk}_{(\mathcal{P},2)} = g_2^{\mathsf{sk}_{\mathcal{P}}})$ and $\mathsf{pk}_{\mathcal{R}} = (\mathsf{pk}_{(\mathcal{R},1)} = g_1^{\mathsf{sk}_{\mathcal{R}}}, \mathsf{pk}_{(\mathcal{R},2)} = g_2^{\mathsf{sk}_{\mathcal{R}}})$. We run the adversary $\mathcal{A}_{\mathtt{I}}^{\mathcal{O}_{\mathsf{Id}}^{\bar{r}}}(\mathsf{par}, \mathsf{sk}_{\mathcal{P}}, \mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{R}})$ the access to $\mathcal{O}_{\mathsf{Id}}^{\bar{r}}$, and $\mathcal{O}_{\mathcal{H}_{g_1}}$ oracles.

`Serving` $\mathcal{O}_{\mathcal{H}_{g_1}}$ `Oracle:` We first consider the serving of the $\mathcal{O}_{\mathcal{H}_{g_1}}$ oracle. We allow ℓ fresh inputs to the $\mathcal{O}_{\mathcal{H}_{g_1}}$ oracle. We choose the random index $j \leftarrow_{\$} \{1, \ldots, \ell\}$, which denotes the j-th invocation of $\mathcal{O}_{\mathcal{H}_{g_1}}$, for which we assume the forgery will happen.

- On i-th ($i \neq j$) fresh input: $d \leftarrow_{\$} \mathbb{Z}_q$, g_1^d is registered as a hash value in the ROM table for $\mathcal{H}_{g_1}$. Return g_1^d as output.
- On j-th fresh input: $\dot{g}_1 = (g_1^{\beta})$ and register that hash value in the ROM table for $\mathcal{H}_{g_1}$. Return g_1^{β} as output.

`Serving` $\mathcal{O}_{\mathsf{Id}}^{\bar{\mathtt{r}}}$ `Oracle` : Next, we consider the serving of the $\mathcal{O}_{\mathsf{Id}}^{\bar{r}}$ oracle. On input $\Omega, \bar{r}_1$ we compute $\bar{R}_1 = g_2^{\bar{r}_1}$, serve the call $\mathcal{O}_{\mathcal{H}_{g_1}}(\Omega, \bar{R}_1)$, namely locate and return $(\dot{g}_1 = g_1^d, d)$ if $\Omega, \bar{R}_1$ was not the j-th fresh input to $\mathcal{O}_{\mathcal{H}_{g_1}}$, then compute $\hat{S}_1 = (g_1^{\bar{r}_1} \cdot \mathsf{pk}_{(\mathcal{I},1)}^{\mathcal{H}_1(\Omega,\bar{R}_1)})^d$, return $\sigma_1 = (\hat{S}_1, \bar{R}_1)$. Note that in this case the $\mathsf{Ver}(\mathsf{par}, \mathsf{pk}_{\mathcal{I}}, \sigma_1, \Omega) = 1$ since

$$\hat{e}(\hat{S}_1, g_2) = \hat{e}(\Big(g_1^{\bar{r}_1} \cdot \mathsf{pk}_{(\mathcal{I},1)}^{\mathcal{H}_1(\Omega,\bar{R}_1)}\Big)^d, g_2) = \hat{e}(\big(g_1^d\big)^{\bar{r}_1 + \mathsf{sk}_{\mathcal{I}}\mathcal{H}_1(\Omega,\bar{R}_1)}, g_2) \quad (4.1)$$

$$= \hat{e}(\dot{g}_1, g_2^{\bar{r}_1 + \mathsf{sk}_{\mathcal{I}}\mathcal{H}_1(\Omega,\bar{R}_1)}) = \hat{e}(\dot{g}_1, g_2^{\bar{r}_1} \cdot g_2^{\mathsf{sk}_{\mathcal{I}}\mathcal{H}_1(\Omega,\bar{R}_1)}) \quad (4.2)$$

$$= \hat{e}(\dot{g}_1, \bar{R}_1 \cdot \mathsf{pk}_{(\mathcal{I},2)}^{h_1}) \quad (4.3)$$

`Forgery Stage:` Under the *Forking Lemma* for the hash function $\mathcal{H}_1$ the attacker returns two valid signcryptions with the same randomness as in R_1, namely $\sigma_2 = (\hat{S}, R_1, \hat{R}_2, c)$, $\sigma_2' = (\hat{S}', R_1, \hat{R}_2', c')$, s.t. $\hat{S} = \hat{S}_1\hat{S}_2$, $\hat{S}' = \hat{S}_1'\hat{S}_2'$, and the output $h_1 = \mathcal{H}_1(\Omega, R_1)$ in $\hat{S}_1$ of σ_2 but $h_1' \neq h_1$ in $\hat{S}_1'$ of σ_2'. With the non-negligible probability $1/\ell$, the forking happens on

the j-th fresh input (Ω, R_1) to $\mathcal{O}_{\mathcal{H}_{g_1}}$, the $\dot{g}_1$ equals to g^β in both tuples σ_2, σ_2'. We compute $R_2 = (\hat{R}_2)^{1/\mathsf{sk}_\mathcal{R}}$, $k = \mathsf{pk}_{(\mathcal{P},1)}^{\mathsf{sk}_\mathcal{R}}$, $k_1 = \mathcal{H}_3(R_2, k, 1)$, $k_2 = \mathcal{H}_3(R_2, 2)$, $m = D_{k_1}(c)$, locate $(\ddot{g}_1 = g_1^d, d)$ in the ROM table for $\mathcal{O}_{\mathcal{H}_{g_1}}(\Omega, R_1, k_2, m)$, compute $\hat{S}_2 = (R_2 \cdot \mathsf{pk}_{(\mathcal{P},1)}^{\mathcal{H}_2(\Omega, R_1, k_2, m)})^d$, and eventually $\hat{S}_1 = \hat{S}/\hat{S}_2$. Compute $\hat{S}_1'$ from σ_2' in a similar way. We now have

$$\frac{\hat{S}_1}{\hat{S}_1'} = \dot{g}_1^{\mathsf{sk}_\mathcal{I}(h_1 - h_1')} = (g_1^\beta)^{\mathsf{sk}_\mathcal{I}(h_1 - h_1')}. \tag{4.4}$$

Since we started with $\mathsf{sk}_\mathcal{I} = \alpha$ we can therefore compute

$$\frac{\hat{S}_1}{\hat{S}_1'}^{\frac{1}{(h_1 - h_1')}} = (g_1^\beta)^{\alpha \frac{h_1 - h_1'}{h_1 - h_1'}} = g_1^{\alpha\beta}, \tag{4.5}$$

thus breaking the given instance of PG-CDH.

□

Theorem 8. *The proposed scheme π'_{PSC} is secure under the security model of* `Forgery Type II` *as π_{PSC} is, but with the asymmetric pairing configuration.*

Proof. The proof is by contradiction; we assume the existence of an adversary $\mathcal{A}$ for which $\mathbf{Adv}(\mathcal{A}, \mathsf{Exp}^{UIA,\lambda,\ell})$ is non-negligible. The experiment is then executed as follows:

`Init Stage:` The experiment starts by running $\mathsf{par} \leftarrow \mathsf{ParGen}(\lambda)$ and instantiate PG-CDH with $(g_1, g_1^\alpha, g_1^\beta, g_2, g_2^\alpha, g_2^\beta)$. The adversary is given access to hash oracle $\mathcal{O}_\mathcal{H}$ and given the public key components for party $\mathcal{P}$ we have $\mathsf{pk}_\mathcal{P} = (\mathsf{pk}_{(\mathcal{P},1)} = g_1^\alpha, \mathsf{pk}_{(\mathcal{P},2)} = g_2^\alpha)$. Thus the unknown $\mathsf{sk}_\mathcal{P}$ equals the unknown α. The remaining parties setup their keys as per the protocol, i.e., $\mathsf{sk}_\mathcal{I} \leftarrow_\$ \mathbb{Z}_q$ and $\mathsf{sk}_\mathcal{R} \leftarrow_\$ \mathbb{Z}_q$, hence $\mathsf{pk}_\mathcal{I} = (\mathsf{pk}_{(\mathcal{I},1)} = g_1^{\mathsf{sk}_\mathcal{I}}, \mathsf{pk}_{(\mathcal{I},2)} = g_2^{\mathsf{sk}_\mathcal{I}})$ and $\mathsf{pk}_\mathcal{R} = (\mathsf{pk}_{(\mathcal{R},1)} = g_1^{\mathsf{sk}_\mathcal{R}}, \mathsf{pk}_{(\mathcal{R},2)} = g_2^{\mathsf{sk}_\mathcal{R}})$. We run the adversary $\mathcal{A}_\mathtt{I}^{\mathcal{O}_{\mathsf{Id}}^{\bar{r}}}(\mathsf{par}, \mathsf{sk}_\mathcal{I}, \mathsf{pk}_\mathcal{P}, \mathsf{pk}_\mathcal{R})$ the access to $\mathcal{O}_{\mathsf{SC}}^{\bar{r}}$, and $\mathcal{O}_{\mathcal{H}_{g_1}}$ oracles.

`Serving` $\mathcal{O}_{\mathcal{H}_{g_1}}$ `Oracle:` We first consider the serving of the $\mathcal{O}_{\mathcal{H}_{g_1}}$ oracle. We allow ℓ fresh inputs to the $\mathcal{O}_{\mathcal{H}_{g_1}}$ oracle. We choose the random index $j \leftarrow_\$ \{1, \ldots, \ell\}$, which denotes the j-th invocation of $\mathcal{O}_{\mathcal{H}_{g_1}}$, for which we assume the forgery will happen.

- On i-th ($i \neq j$) fresh input: $d \leftarrow_\$ \mathbb{Z}_q$ and g_1^d is registered as a hash value in the ROM table for $\mathcal{H}_{g_1}$. Return g_1^d as the output.

- On j-th fresh input: g_1^{β} is computed and registered as a hash value in the ROM table for $\mathcal{H}_{g_1}$. Return g_1^{β} as the output.

`Serving` $\mathcal{O}_{\mathsf{SC}}^{\bar{r}}$ `Oracle` : Next, we consider the serving of the $\mathcal{O}_{\mathsf{SC}}^{\bar{r}}$ oracle. On the provided input $(\mathsf{par}, \mathsf{sk}_{\mathcal{P}}, \sigma_1, \Omega, \mathsf{pk}_{\mathcal{R}}, m)$ and randomness $\bar{r}$ we compute $\bar{R}_2 = g_2^{\bar{r}_2}$, serve the call $\mathcal{O}_{\mathcal{H}_{g_1}}(\Omega, \bar{R}_1, k_2, m)$, namely locate and return $(\ddot{g}_1 = g_1^d, d)$ if $\Omega, \bar{R}_1, k_2, m$ was not the j-th fresh input to $\mathcal{O}_{\mathcal{H}_{g_1}}$, then compute $\hat{S}_2$ as $\hat{S}_2 = (g_1)^{dr_2}(g_1)^{d\mathsf{sk}_{\mathcal{P}}h_2} = (g_1)^{dr_2}\mathsf{pk}_{(\mathcal{P},1)}^{dh_2}$. This returns σ_2 and we note that in this case the USC verifies correctly since:

$$\hat{e}(\hat{S}, g_2) = \hat{e}(\hat{S}_1, g_2)\hat{e}(\hat{S}_2, g_2) = \hat{e}(\hat{S}_1, g_2)\hat{e}((g_1)^{d\bar{r}_2}\mathsf{pk}_{(\mathcal{P},1)}^{dh_2}, g_2) \tag{4.6}$$

$$= \hat{e}(\dot{g}_1^{r_1+\mathsf{sk}_{\mathcal{I}}h_1}, g_2)\hat{e}((g_1^d)^{\bar{r}_2+\mathsf{sk}_{\mathcal{P}}h_2}, g_2) \tag{4.7}$$

$$= \hat{e}(\dot{g}_1, (g_2)^{r_1+\mathsf{sk}_{\mathcal{I}}h_1})\hat{e}(\ddot{g}_1, (g_2)^{\bar{r}_2+\mathsf{sk}_{\mathcal{P}}h_2}) \tag{4.8}$$

$$= \hat{e}(\dot{g}_1, R_1 \cdot \mathsf{pk}_{(\mathcal{I},2)}^{h_1})\hat{e}(\ddot{g}_1, \bar{R}_2 \cdot \mathsf{pk}_{(\mathcal{P},2)}^{h_2}) \tag{4.9}$$

`Forgery Stage:` Now, according to the *Forking Lemma* we see that over the hash function $\mathcal{H}_2$, the attacker is able to return two valid signcryptions with the same randomness, namely $\sigma_2 = (\hat{S}, R_1, \hat{R}_2, c)$, $\sigma_2' = (\hat{S}', R_1', \hat{R}_2, c')$, such that $\hat{S} = \hat{S}_1\hat{S}_2$, $\hat{S}' = \hat{S}_1'\hat{S}_2'$, and the output $h_2 = \mathcal{H}_2(\Omega, R_1, k_2, m)$ in $\hat{S}_2$ of σ_2 but $h_2' \neq h_2$ in $\hat{S}_2'$ of σ_2'. With the non-negligible probability $1/\ell$, the forking happens on the j-th fresh input (Ω, R_1, k_2, m) to $\mathcal{O}_{\mathcal{H}_{g_1}}$, the $\ddot{g}_1$ equals to g^{β} in both tuples σ_2, σ_2'. We locate $(\dot{g}_1 = g_1^d, d)$ in the ROM table for $\mathcal{O}_{\mathcal{H}_{g_1}}(\Omega, R_1)$, compute $\hat{S}_1 = \left(R_1 \cdot \mathsf{pk}_{(\mathcal{P},1)}^{h_2}\right)^d$ and then $\hat{S}_2 = \hat{S}/\hat{S}_1$. It is now possible to compute $\hat{S}_2'$ from σ_2 in a similar way. We have the following relation:

$$\frac{\hat{S}_2}{\hat{S}_2'} = \ddot{g}_1^{\mathsf{sk}_p(h_2-h_2')} = (g^{\beta})^{\mathsf{sk}_p(h_2-h_2')}. \tag{4.10}$$

Therefore we could compute $g^{\alpha\beta} = (\hat{S}_2/\hat{S}_2')^{(1/(h_2-h_2'))}$, breaking the given instance of $\mathsf{PG\text{-}CDH}$.

□

4.2 Scenario P_{II}: Certificateless Multi-Party Signcryption in 5G-Connected Infrastructures

The following section is based on the published paper P_{II}[2].

System and ENV setup: We consider a 5G architecture, providing connected nodes to communicate via cellular means, known as Narrowband Internet of Things (NB-IoT), particularly designed for IoT devices (IOTD) requiring low cost, long battery life, increased coverage (good for building interiors, basements) and high connection density (thousands of devices in a small area). For a connected infrastructure where clusters of sensors need to cooperate, e.g. for a limited range of railway maintenance (cameras, radar, heat sensors etc.), each IOTD needs to send the sensor data to a track-side unit (dedicated collector for IOTD) for further processing. At this stage, all data need to be sent with intact integrity and privacy. We have three layers: IOTD in the *Endpoints* layer that run functions in the *Core Network* layer via an *Access Network* components layer where a base station is located. The Access and Mobility Management Function (AMF) handles key agreement and authentication requests, implements integrity protection algorithms, receives all connection and session related information from the user equipment and passes the session management requirements to the Session Management Function (SMF). The User Plane Function (UPF) is responsible for user-side configurations, allowing the data transfer component to be decentralized, connects mobile infrastructure and data network, and provides packet routing/forwarding. A brief depiction describes the connections in Fig. 4.1. Devices in this system uses architectures based upon two distinct HSMs from different vendors, for each secret key; HSM1 and HSM2 will thus realizing the minimal functionalities **f1**, **f2**, i.e., securely computing the exponent of a value with the secret key, namely the exponentiation technique.

In this paper, we focus on connected eco-systems such as road- and railway infrastructures reliant on connected devices such as sensors, cameras, and control units, emphasizing the need for low-powered, efficient IoT networking. A key initiative in this area is the European Rail Traffic Management System (ERTMS), currently deploying 5G technology across Europe [59].

[2]Published at IEEE International Symposium on Network Computing and Applications conference and is under copyright © 2021 IEEE.

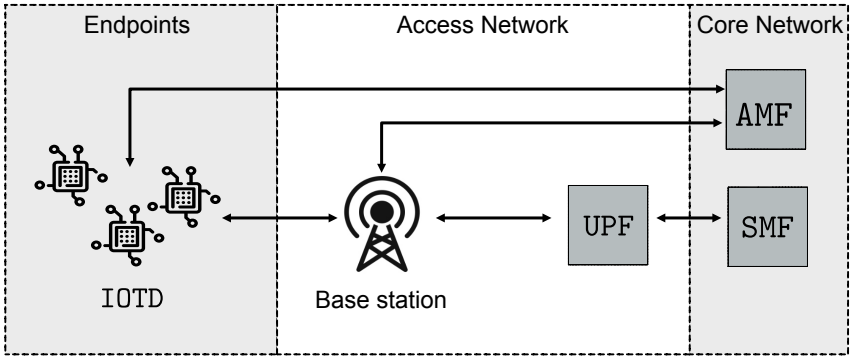

Figure 4.1: A simplified depiction of the 5G setup with IOTD.

The European Commission highlights the Future Radio Mobile Communication System (FRMCS) as a pivotal element in ERTMS, replacing GSM-R with 5G innovations [117]. Security challenges in 5G, including vulnerabilities to DDoS attacks, underscore the importance of robust encryption and authentication protocols. Common methods include encrypt-then-MAC and signcryption, addressing both encryption and authenticity. However, issues like ephemeral key leakage in untrusted devices remain a concern [115, 106].

To address these security issues in a 5G context, particularly for NB-IoT, we propose a *multi-party authenticated encryption* protocol MPAE that enhances data transmission efficiency. The MPAE scheme is defined on top of the architecture presented in Fig. 4.1. Despite advances in symmetric cryptography, challenges with ephemeral value leakage persist. Our modified scheme aims to mitigate this leakage issue, recognizing the potential for devices to be targeted by side-channel attacks or malicious hardware manufacturers. Ensuring data flow integrity is of high importance. Additionally, our approach integrates certificateless cryptography, offering a solution to the scalability challenges of traditional PKI.

4.2.1 Security Requirements

The minimal security requirements are *multi-party authentication*, i.e., all devices connected to the network should be authenticated simultaneously, and *data integrity and confidentiality* for all nodes. To summarize:

- Multi-party authentication, i.e., all devices connected to the network should be authenticated simultaneously.
- Data integrity and confidentiality for all nodes.

- The scheme should be resistant to ephemeral key leakage, i.e., messages should not be decrypted even if an adversary has control over the ephemeral source.

4.2.2 Related Work

Many protocols have focused on the security in LTE networks and the 5G architecture, e.g. the lack of proper identity protection allowing DDoS attacks [156]. Authenticated encryption is a way to both encrypt a message and at the same time provide authenticity. A typical pattern is the *encrypt-then-MAC*, where a message authentication code is computed over the encrypted values sent to a decryptor. Another technique is *signcryption* where the produced ciphertext and authentication value is computed in one logical step instead of two. In a multi-party setup where each node P_i need to signcrypt a message m_i and later forward it to a single verifier, each signcryption ciphertext can be *aggregated* into a single ciphertext which is later decrypted and authenticated into $m = \{m_1, ..., m_n\}$. Bit leakage was first addressed by Chari et al. [37], continued by Goubin and Patarin [67] and Alwen et al. [13]. Also, the problem of bit leakage from cryptographic keys was analyzed by Canetti et al. [34]. The problem with ephemeral key leakage in untrusted devices has been analyzed further, e.g. in [94]. For NB-IoT usage in a 5G architecture, a multi-party authenticated encryption algorithm with improved data transmission efficiency has been proposed [165]. However, despite the use of symmetric cryptography, data security is still compromised by the leakage of ephemeral values [115, 106].

4.2.3 Threat Model

The proposed authenticated encryption for IoT devices is based on a multi-party model. For instance, during the compromised execution of the message-relaying protocol, an `IOTD` device has to encrypt a message m. The `IOTD` device generates its secret key usk before registration. It also obtains a second key psk after registration. To perform the functionality described above, an ephemeral key v is used by the `IOTD` device when authenticating a message. Our stronger model considers the following attacks:

Ephemeral injection in HSM: In the first attack scenario the adversary gets hold of usk, stored within the device's HSM, and injects the ephemeral session key $\bar{v}$. Given these values, the adversary tries to impersonate the device without the missing psk.

Dual ephemeral injection in HSM: A second *dual* attack is also considered. Similarly to the previous scenario, the adversary can get psk and controls $\bar{v}$. Given these values, the adversary tries to impersonate the device and authenticate a fresh message without the missing usk.

Note that a typical construction like [165], do not consider the leakage of ephemeral values v at all. Knowledge of v allows the adversaries to break the system without the additional secret keys. Our proposed constructions are immune against the considered attacks.

4.2.4 Proposed Scheme

Architecture and Devices

The IOTD devices has two different types of secret keys. The user key usk generated locally in the device by a procedure PKeyGen, and psk obtained during the registration phase, where a third party of a key generation centre runs the procedure IPKeyGen. A registered device can produce signcrypted messages with usk and psk, together with a session-specific ephemeral key (randomness) and the recipients public key, running procedure AEn, as depicted in Fig. 4.2.

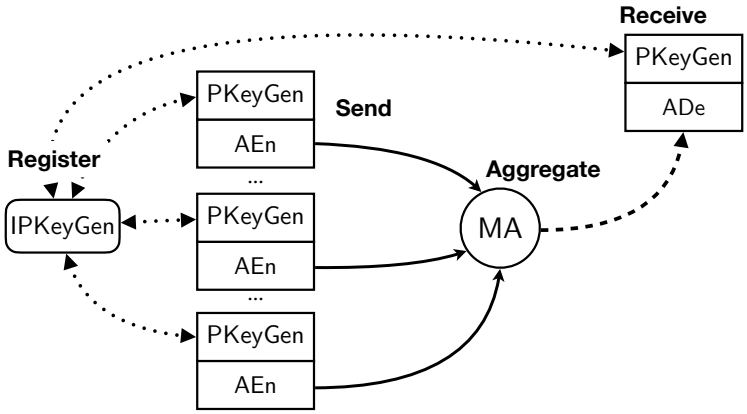

Figure 4.2: The functionality of the MPAE scheme with IOTD devices, a register node, an aggregation node and a receiver node.

The aggregation of authentication values (via the MA procedure) can be implemented as a specific UPF or AMF sub-function in access or core layers, or as a dedicated function in one of the endpoint devices. The security of authentication requires that both secret keys are used, and the lack of at least one key prevents the adversarial forgery. This implies that secret

keys are stored securely in separate HSMs, preferably from different vendors. Typically this should immune against forgery, even if one of the vendors are malicious, or an HSM is compromised due to production errors. However, this methodology assumes that the randomness is thoroughly protected since the security is broken if the adversary controls any randomness used in the protocol. Thus we also require the scheme to be *resistant to ephemeral key leakage*, i.e. the adversary should not impersonate the legitimate device even if the adversary controls the randomness.

We consider architecture based upon two distinct HSMs from different vendors, for each secret key. The scheme should be still secure even one of the two HSMs is compromised, and additionally if the ephemeral value in less restricted code area is controlled by the attacker. The HSMs should validate inputs from unsafe areas in the device due to attacks on invalid curve points. The device architecture for our proposed schemes is depicted in Fig 4.3. HSM1, HSM2, realizing minimal functionalities **f1**, **f2** of exponent with secret keys. Note, the scheme construction from [165] is inherently vulnerable to ephemeral leakage.

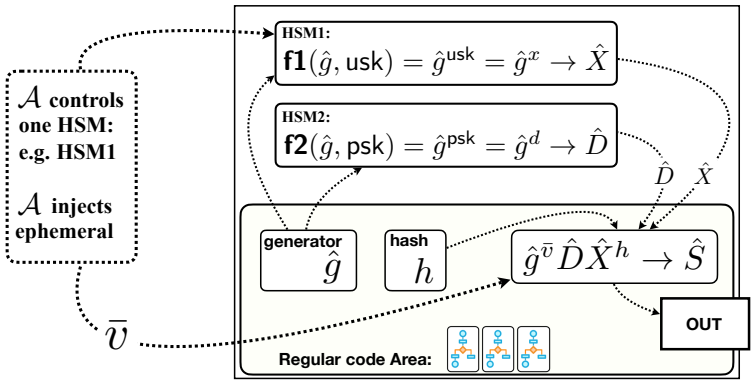

Figure 4.3: Device architecture for AEn procedure.

Multi-Party Authenticated Encryption

The original scheme [165] involves interaction between three types of parties: the Key Generate Center (KGC), IOTD devices and AMF. IOTD devices are indexed. The subset of indexes of IOTD devices that sends data is denoted as $\mathsf{ID} = \{i\}$, while the index of the receiver device is denoted by k. The KGC uses the Setup functions to generate basic parameters necessary for setting up a 5G communication channel between IOTD and AMF. It is also a trusted party of the protocol. AMF is the receiver party that verifies incoming data and decrypts the messages.

Definition 23. *A certificateless Multi-Party Authentication Encryption scheme* (MPAE) *consists of seven algorithms:* Setup, UKeyGen, IPKeyGen, PKeyGen, AEn, MA, *and* ADe.

$\mathsf{Setup}\left(1^{\lambda}\right) \rightarrow (\mathsf{par}, \mathsf{msk})$: *This algorithm is run by the* KGC. *Upon receiving a security parameter* λ, *it returns system public parameters* par *and a master secret key* msk. *The* par *are default parameters to all procedures of the scheme, thus we skip their explicit invocation.*

$\mathsf{UKeyGen}(\mathsf{ID}) \rightarrow (\mathsf{upk}_{\mathsf{ID}}, \mathsf{usk}_{\mathsf{ID}})$: *The user-side key generation algorithm is run by the user itself. For a user (i.e.* IOTD, AMF*) with identity* ID *this algorithm generates a user side public key* $\mathsf{upk}_{\mathsf{ID}}$ *and a user-side secret key* $\mathsf{usk}_{\mathsf{ID}}$.

$\mathsf{IPKeyGen}(\mathsf{msk}, \mathsf{ID}, \mathsf{upk}_{\mathsf{ID}}) \rightarrow (\mathsf{ippk}_{\mathsf{ID}}, \mathsf{ipsk}_{\mathsf{ID}})$ *: This algorithm is run by the* KGC *to generate initial-partial keys for a user with identity* ID. *On input* $\mathsf{msk}, \mathsf{ID}$ *and* $\mathsf{upk}_{\mathsf{ID}}$, *the algorithm returns an initial-partial public key* $\mathsf{ippk}_{\mathsf{ID}}$ *and the corresponding initial-partial secret key* $\mathsf{ipsk}_{\mathsf{ID}}$.

$\mathsf{PKeyGen}(\mathsf{upk}_{\mathsf{ID}}, \mathsf{ippk}_{\mathsf{ID}}, \mathsf{ipsk}_{\mathsf{ID}}) \rightarrow (\mathsf{ppk}_{\mathsf{ID}}, \mathsf{psk}_{\mathsf{ID}})$: *This algorithm is run by the user to transform initial partial keys to partial keys. On input* $\mathsf{upk}_{\mathsf{ID}}, \mathsf{ippk}_{\mathsf{ID}}$ *and* $\mathsf{ipsk}_{\mathsf{ID}}$, *the algorithm returns a partial public key and the corresponding partial secret key.*

$\mathsf{AEn}(\mathsf{PK}_{\mathsf{ID}}, \mathsf{SK}_{\mathsf{ID}}, \mathsf{PK}_k, m_{\mathsf{ID}}) \rightarrow \mathtt{ACT}_{\mathsf{ID}}$: *The authenticated aggregate encryption algorithm is performed by* $\{\mathtt{IOTD}_{\mathsf{ID}}\}_{i \in \mathsf{ID}}$. *Suppose* $\mathtt{IOTD}_{\mathsf{ID}}$ *takes as inputs* par, *its public key* $\mathsf{PK}_{\mathsf{ID}} = (\mathsf{upk}_{\mathsf{ID}}, \mathsf{ppk}_{\mathsf{ID}})$ *and corresponding secret key* $\mathsf{SK}_{\mathsf{ID}} = (\mathsf{usk}_{\mathsf{ID}}, \mathsf{psk}_{\mathsf{ID}})$, *the public key* PK_k *of a target, and a message* m_{ID}, *the algorithm generates an authenticated encryption ciphertext* $\mathtt{ACT}_{\mathsf{ID}}$

$\mathsf{MA}(\{\mathtt{ACT}_{\mathsf{ID}}\}_{i \in \mathsf{ID}}) \rightarrow \mathtt{ACT}$: *The multi-authenticated encryption algorithm is performed by* AMF. *It inputs a set of authenticated encryption ciphertexts* $\{\mathtt{ACT}_{\mathsf{ID}}\}_{i \in \mathsf{ID}}$, *and returns a multi-party aggregate authenticated encryption ciphertext* ACT.

$\mathsf{ADe}(\mathsf{SK}_k, \{\mathsf{PK}_{\mathsf{ID}}\}_{\mathsf{ID} \in \mathsf{ID}}, \mathtt{ACT}) \rightarrow m$ *or* $\perp$: *The multi-party aggregate authenticated decryption algorithm is run by* AMF. *It takes as inputs* $\mathsf{SK}_k, \{\mathsf{PK}_{\mathsf{ID}}\}_{i \in \mathsf{ID}}$ *and* ACT, *then outputs* $m = \{m_{\mathsf{ID}}\}_{i \in \mathsf{ID}}$, *if it was a valid aggregated multi-party authenticated encryption ciphertext, otherwise it outputs* $\perp$.

We require the scheme to be *correct*, i.e. messages authenticated via secret keys (of senders) and encrypted via a public key (of a recipient) should be positively verified and successfully decrypted.

Definition 24 (Correctness)**.** MPAE *scheme is correct if for any set of indices* $\Omega \subset \mathbb{Z}_p$*, any subset* $\mathsf{ID} \subset \Omega$*, any index of a recipient* $k \in \Omega$*, and any messages* $\{m_{\mathsf{ID}}\}_{i \in \mathsf{ID}} \subset \{0,1\}^l$*:*

$$\Pr\left[\begin{array}{l}(\mathsf{par}, \mathsf{msk}) \leftarrow \mathsf{Setup}(\lambda),\\ \textit{For each } i \in \Omega:\\ \quad (\mathsf{upk}_i, \mathsf{usk}_i) \leftarrow \mathsf{UKeyGen}(i),\\ \quad (\mathsf{ippk}_i, \mathsf{ipsk}_i) \leftarrow \mathsf{IPKeyGen}(\mathsf{msk}, i, \mathsf{upk}_i),\\ \quad (\mathsf{SK}_i, \mathsf{PK}_i) \leftarrow \mathsf{PKeyGen}(\mathsf{upk}_i, \mathsf{ippk}_i, \mathsf{ipsk}_i),\\ \textit{For each } i \in \mathsf{ID}:\\ \quad \mathtt{ACT}_i \leftarrow \mathsf{AEn}(\mathsf{PK}_i, \mathsf{SK}_i, \mathsf{PK}_k, m_i),\\ \mathtt{ACT} \leftarrow \mathsf{MA}(\{\mathtt{ACT}_i\}_{i \in \mathsf{ID}}),\\ \{m_{\mathsf{ID}}\}_{i \in \mathsf{ID}} \leftarrow \mathsf{ADe}(\mathsf{SK}_k, \{\mathsf{PK}_i\}_{i \in \mathsf{ID}}, \mathtt{ACT})\end{array}\right] = 1.$$

The definition below models the secrecy of messages encrypted to the recipient whose private decryption key is unknown to the adversary. We do not tweak that definition, and recall it just for the paper completeness.

Original Scheme MPAE [165]	**Modified Schemes**: MPAE-1 and MPAE-2
$\mathsf{Setup}(\lambda)$: $(\mathbb{G}, g, q) \leftarrow \mathsf{Setup}(1^\lambda)$ $\mathcal{H} : \{0,1\}^* \to \mathbb{Z}_q^*$, $\mathcal{H}_l : \{0,1\}^* \to \{0,1\}^l$ $a \leftarrow_\$ \mathbb{Z}_q^*$, $A = g^a$ $\mathsf{par} = (\mathbb{G}, g, q, A, \mathcal{H})$ **return** $\mathsf{par}, \mathsf{msk} = a$	$\mathsf{Setup}(\lambda)$: $(\mathbb{G}, \mathbb{G}_T, g, g_T, q, \hat{e}) \leftarrow \mathsf{Setup}(1^\lambda)$ $\mathcal{H} : \{0,1\}^* \to \mathbb{Z}_q^*$, $\mathcal{H}_l : \{0,1\}^* \to \{0,1\}^l$, $\mathcal{H}_g : \{0,1\}^* \to \mathbb{G}$, $a \leftarrow_\$ \mathbb{Z}_q^*$, $A = g^a$ $\mathsf{par} = (\mathbb{G}, \mathbb{G}_T, g, g_T, q, A, \hat{e}, \mathcal{H}, \mathcal{H}_g)$ **return** $\mathsf{par}, \mathsf{msk} = a$
$\mathsf{UKeyGen}(\mathsf{par}, \mathsf{ID})$: $x_{\mathsf{ID}} \leftarrow_\$ \mathbb{Z}_q^*$, store x_{ID} in HSM1 of i, compute $X_{\mathsf{ID}} = \mathbf{f1}(g) = g^{x_{\mathsf{ID}}}$ **return** $\mathsf{upk}_{\mathsf{ID}} = X_{\mathsf{ID}}, \mathsf{usk}_{\mathsf{ID}} = x_{\mathsf{ID}}$	
$\mathsf{IPKeyGen}(\mathsf{par}, a, \mathsf{ID}, \mathsf{upk}_{\mathsf{ID}})$: $r_{\mathsf{ID}} \leftarrow_\$ \mathbb{Z}_q^*, R_{\mathsf{ID}} = g^{r_{\mathsf{ID}}}$, $X_{\mathsf{ID}} \leftarrow \mathsf{upk}_{\mathsf{ID}}$, $\mathsf{n}_{\mathsf{ID}} \leftarrow_\$ \{0,1\}^l$, $h_{1,\mathsf{ID}} = \mathcal{H}(\mathsf{n}_{\mathsf{ID}}, X_{\mathsf{ID}}, R_{\mathsf{ID}})$, $u_{\mathsf{ID}} = r_{\mathsf{ID}} + a \cdot h_{1,\mathsf{ID}} + \mathcal{H}((X_{\mathsf{ID}})^a)$ **return** $\mathsf{ippk}_{\mathsf{ID}} = (\mathsf{n}_{\mathsf{ID}}, R_{\mathsf{ID}}), \mathsf{ipsk}_{\mathsf{ID}} = u_{\mathsf{ID}}$	
$\mathsf{PKeyGen}(\mathsf{par}, \mathsf{upk}_{\mathsf{ID}}, \mathsf{ippk}_{\mathsf{ID}}, \mathsf{ipsk}_{\mathsf{ID}})$: $u_{\mathsf{ID}} \leftarrow \mathsf{ipsk}_{\mathsf{ID}}$, $d_{\mathsf{ID}} = u_{\mathsf{ID}} - \mathcal{H}(\mathbf{f1}_i(A)) = u_{\mathsf{ID}} - \mathcal{H}(A^{x_{\mathsf{ID}}})$, $\mathsf{ppk}_{\mathsf{ID}} = (\mathsf{n}_{\mathsf{ID}}, R_{\mathsf{ID}}), \mathsf{psk}_{\mathsf{ID}} = d_{\mathsf{ID}}$ **return** $\mathsf{SK}_{\mathsf{ID}} = (\mathsf{usk}_{\mathsf{ID}}, \mathsf{psk}_{\mathsf{ID}})$, $\mathsf{PK}_{\mathsf{ID}} = (\mathsf{upk}_{\mathsf{ID}}, \mathsf{ppk}_{\mathsf{ID}})$	

Table 4.5: The modified schemes are on the right.

Definition 25 (Indistinguishability under adaptive chosen ciphertext attack)**.** *Let* $\mathsf{MPAE} = (\mathsf{Setup}, \mathsf{UKeyGen}, \mathsf{IPKeyGen}, \mathsf{PKeyGen}, \mathsf{AEn}, \mathsf{MA}, \mathsf{ADe})$ *be a certificateless Multi-Party Authentication Encryption scheme. The following security experiment* $\mathtt{Exp}_{\mathsf{ESS}}^{\lambda,\ell_1,\ell_2}(\mathcal{A}, \mathsf{MPAE})$ *is defined as:*

Original Scheme MPAE [165]	**Modified Scheme** MPAE-1	**Modified Scheme** MPAE-2
$\mathsf{AEn}(\mathsf{par}, \mathsf{PK}_{\mathsf{ID}}, \mathsf{SK}_{\mathsf{ID}}, \mathsf{PK}_k, m_{\mathsf{ID}})$:		
$v_{\mathsf{ID}} \leftarrow_{\$} \mathbb{Z}_q^*, V_{\mathsf{ID}} = g^{v_{\mathsf{ID}}}$, $(x_{\mathsf{ID}}, d_{\mathsf{ID}}) \leftarrow \mathsf{SK}_{\mathsf{ID}}$, $(X_{\mathsf{ID}}, R_{\mathsf{ID}}, \mathsf{n}_{\mathsf{ID}}) \leftarrow \mathsf{PK}_{\mathsf{ID}}$, $(X_k, R_k, \mathsf{n}_k) \leftarrow \mathsf{PK}_k$ $h_{1,k} = \mathcal{H}(\mathsf{n}_k, X_k, R_k)$, $Z_{\mathsf{ID}} = (X_k \cdot R_k \cdot A^{h_{1,k}})^{v_{\mathsf{ID}}}$, $h_{2,\mathsf{ID}} = \mathcal{H}_l(\mathsf{n}_k, V_{\mathsf{ID}}, Z_{\mathsf{ID}})$, $C_{\mathsf{ID}} = h_{2,i} \oplus m_{\mathsf{ID}}$, $h_{3,\mathsf{ID}} = \mathcal{H}(V_{\mathsf{ID}}, \mathsf{n}_{\mathsf{ID}}, C_{\mathsf{ID}}, X_{\mathsf{ID}}, R_{\mathsf{ID}}, Z_{\mathsf{ID}})$		
$s_{\mathsf{ID}} = v_{\mathsf{ID}} + d_{\mathsf{ID}} + x_{\mathsf{ID}} \cdot h_{3,\mathsf{ID}}$ **return** $\mathtt{ACT}_{\mathsf{ID}} = (s_{\mathsf{ID}}, V_{\mathsf{ID}}, C_{\mathsf{ID}})$	$\hat{g}_{\mathsf{ID}} = \mathcal{H}_g(V_{\mathsf{ID}}, \mathsf{n}_{\mathsf{ID}}, C_{\mathsf{ID}}, X_{\mathsf{ID}}, R_{\mathsf{ID}}, Z_{\mathsf{ID}})$ $\hat{S}_{\mathsf{ID}} = \hat{g}_{\mathsf{ID}}^{v_{\mathsf{ID}}} \mathbf{f2}_i(\hat{g}_i)\mathbf{f1}_i(\hat{g}_i)^{h_{3,\mathsf{ID}}} = \hat{g}_{\mathsf{ID}}^{v_{\mathsf{ID}}+d_{\mathsf{ID}}+x_{\mathsf{ID}} \cdot h_{3,\mathsf{ID}}}$ **return** $\mathtt{ACT}_{\mathsf{ID}} = (\hat{S}_{\mathsf{ID}}, V_{\mathsf{ID}}, C_{\mathsf{ID}})$	$\hat{g} = \mathcal{H}_g(com)$ $\hat{S_{\mathsf{ID}}} = \hat{g}^{v_{\mathsf{ID}}} \mathbf{f2}_i(\hat{g})\mathbf{f1}_i(\hat{g})^{h_{3,\mathsf{ID}}} = \hat{g}^{v_{\mathsf{ID}}+d_{\mathsf{ID}}+x_{\mathsf{ID}} \cdot h_{3,\mathsf{ID}}}$ **return** $\mathtt{ACT}_{\mathsf{ID}} = (\hat{S}_{\mathsf{ID}}, V_{\mathsf{ID}}, C_{\mathsf{ID}})$
$\mathsf{MA}(\mathsf{par}, \{\mathtt{ACT}_i\}_{i \in \mathsf{ID}})$: $s_{\mathsf{ID}} \leftarrow \mathtt{ACT}_{\mathsf{ID}}$, $s = \sum_{i \in \mathsf{ID}} s_i$ **return** $\mathtt{ACT} = \{s, \{V_{\mathsf{ID}}, C_i\}_{i \in \mathsf{ID}}\}$	$\mathsf{MA}(\mathsf{par}, \{\mathtt{ACT}_i\}_{i \in \mathsf{ID}})$: $\hat{S}_{\mathsf{ID}} \leftarrow \mathtt{ACT}_{\mathsf{ID}}$, $S = \prod_{i \in \mathsf{ID}} \hat{S}_{\mathsf{ID}}$ **return** $\mathtt{ACT} = \{S, \{V_{\mathsf{ID}}, C_i\}_{i \in \mathsf{ID}}\}$	
$\mathsf{ADe}\ (\mathsf{par}, \mathsf{SK}_k, \{\mathsf{PK}_i\}_{i \in \mathsf{ID}}, \mathtt{ACT})$: $(x_k, d_k) \leftarrow \mathsf{SK}_k$ **For each** $i \in \mathsf{ID}$ $(X_{\mathsf{ID}}, R_{\mathsf{ID}}, \mathsf{n}_{\mathsf{ID}}) \leftarrow \mathsf{PK}_{\mathsf{ID}}$, $Z_{\mathsf{ID}} = V_{\mathsf{ID}}^{(x_k+d_k)}$, $h_{1,\mathsf{ID}} = \mathcal{H}(\mathsf{n}_{\mathsf{ID}}, X_{\mathsf{ID}}, R_{\mathsf{ID}})$, $h_{3,\mathsf{ID}} = \mathcal{H}(V_{\mathsf{ID}}, \mathsf{n}_{\mathsf{ID}}, C_{\mathsf{ID}}, X_{\mathsf{ID}}, R_{\mathsf{ID}}, Z_{\mathsf{ID}})$. $h_1 = \sum_{i \in \mathsf{ID}} h_{1,i}$ $V = \prod_{i \in \mathsf{ID}} V_i$, $R = \prod_{i \in \mathsf{ID}} R_i$ **Accept iff** $g^s == V \cdot R \cdot A^{h1} \cdot \prod_{i \in \mathsf{ID}} X_i^{h_{3,i}}$ **For each** $i \in \mathsf{ID}$ $h_{2,\mathsf{ID}} = \mathcal{H}_l(\mathsf{n}_k, V_{\mathsf{ID}}, Z_{\mathsf{ID}})$, $m_{\mathsf{ID}} = h_{2,\mathsf{ID}} \oplus C_{\mathsf{ID}}$. **return** $\{m_{\mathsf{ID}}\}_{i \in \mathsf{ID}}$	$\mathsf{ADe}\ (\mathsf{par}, \mathsf{SK}_k, \{\mathsf{PK}_i\}_{i \in \mathsf{ID}}, \mathtt{ACT})$: $(x_k, d_k) \leftarrow \mathsf{SK}_k$ **For each** $i \in \mathsf{ID}$ $(X_{\mathsf{ID}}, R_{\mathsf{ID}}, \mathsf{n}_{\mathsf{ID}}) \leftarrow \mathsf{PK}_{\mathsf{ID}}$, $Z_{\mathsf{ID}} = V_{\mathsf{ID}}^{(x_k+d_k)}$, $h_{1,\mathsf{ID}} = \mathcal{H}(\mathsf{n}_{\mathsf{ID}}, X_{\mathsf{ID}}, R_{\mathsf{ID}})$, $h_{3,\mathsf{ID}} = \mathcal{H}(V_{\mathsf{ID}}, \mathsf{n}_{\mathsf{ID}}, C_{\mathsf{ID}}, X_{\mathsf{ID}}, R_{\mathsf{ID}}, Z_{\mathsf{ID}})$, $\hat{g}_{\mathsf{ID}} = \mathcal{H}_g(V_{\mathsf{ID}}, \mathsf{n}_{\mathsf{ID}}, C_{\mathsf{ID}}, X_{\mathsf{ID}}, R_{\mathsf{ID}}, Z_{\mathsf{ID}})$. **Accept iff** $\hat{e}(S, g) ==$ $== \prod_{i \in \mathsf{ID}} \hat{e}(\hat{g}_{\mathsf{ID}}, V_{\mathsf{ID}} \cdot R_{\mathsf{ID}} \cdot A^{h_{1,\mathsf{ID}}} \cdot X_i^{h_{3,i}})$ **For each** $i \in \mathsf{ID}$ $h_{2,\mathsf{ID}} = \mathcal{H}_l(\mathsf{n}_k, V_{\mathsf{ID}}, Z_{\mathsf{ID}})$, $m_{\mathsf{ID}} = h_{2,\mathsf{ID}} \oplus C_{\mathsf{ID}}$. **return** $\{m_{\mathsf{ID}}\}_{i \in \mathsf{ID}}$	$\mathsf{ADe}\ (\mathsf{par}, \mathsf{SK}_k, \{\mathsf{PK}_i\}_{i \in \mathsf{ID}}, \mathtt{ACT})$: $(x_k, d_k) \leftarrow \mathsf{SK}_k$ **For each** $i \in \mathsf{ID}$ $(X_{\mathsf{ID}}, R_{\mathsf{ID}}, \mathsf{n}_{\mathsf{ID}}) \leftarrow \mathsf{PK}_{\mathsf{ID}}$, $Z_{\mathsf{ID}} = V_{\mathsf{ID}}^{(x_k+d_k)}$, $h_{1,\mathsf{ID}} = \mathcal{H}(\mathsf{n}_{\mathsf{ID}}, X_{\mathsf{ID}}, R_{\mathsf{ID}})$, $h_{3,\mathsf{ID}} = \mathcal{H}(V_{\mathsf{ID}}, \mathsf{n}_{\mathsf{ID}}, C_{\mathsf{ID}}, X_{\mathsf{ID}}, R_{\mathsf{ID}}, Z_{\mathsf{ID}})$. $h_1 = \sum_{i \in \mathsf{ID}} h_{1,i}$, $\hat{g} = \mathcal{H}_g(com)$ $V = \prod_{i \in \mathsf{ID}} V_i$, $R = \prod_{i \in \mathsf{ID}} R_i$ **Accept iff** $\hat{e}(S, g) ==$ $== \hat{e}(\hat{g}, V \cdot R \cdot A^{h_1} \cdot \prod_{i \in \mathsf{ID}} X_i^{h_{3,i}})$ **For each** $i \in \mathsf{ID}$ $h_{2,\mathsf{ID}} = \mathcal{H}_l(\mathsf{n}_k, V_{\mathsf{ID}}, Z_{\mathsf{ID}})$, $m_{\mathsf{ID}} = h_{2,\mathsf{ID}} \oplus C_{\mathsf{ID}}$. **return** $\{m_{\mathsf{ID}}\}_{i \in \mathsf{ID}}$

Table 4.6: The modified schemes are in the middle and the right column and we note that the modified scheme MPAE-2 uses a unique bit string *com* during the aggregated encrypion process.

`Init Stage` : *The challenger generates common parameters* $\mathsf{par} \leftarrow \mathsf{Setup}(\lambda)$, *including the device indexes* Ω. *For each* $i \in \Omega$ *it generates keys:*

$(\mathsf{upk}_i, \mathsf{usk}_i) \leftarrow \mathsf{UKeyGen}(\mathsf{par}, i)$,
$(\mathsf{ippk}_i, \mathsf{ipsk}_i) \leftarrow \mathsf{IPKeyGen}(\mathsf{msk}, i, \mathsf{upk}_i)$,
$(\mathsf{SK}_i, \mathsf{PK}_i) \leftarrow \mathsf{PKeyGen}(\mathsf{upk}_i, \mathsf{ippk}_i, \mathsf{ipsk}_i)$.

$\mathcal{A}$ *chooses a set of indexes of sending devices* $\mathsf{ID} \subset \Omega$ *and the receiving device index* $k \in \Omega, k \notin \mathsf{ID}$. $\mathcal{A}$ *is given all generated data, except the master secret* msk *and secret keys of the receiving device* k: ipsk_k, $\mathsf{SK}_k = (\mathsf{usk}_k, \mathsf{psk}_k)$.

`Serving` $\mathcal{O}_{\mathsf{AEn}}$ `Oracle` : *The oracle* $\mathcal{O}_{\mathsf{AEn}}$ *accepts parameters* par, *public key* PK_i, *public key of the recipient,* PK_k, *and a message* m_i, *and outputs* $\mathtt{ACT}_i = (s_i, V_i, C_i)$ *which is verifiable and decryptable. Note that having the secret keys of all sending devices* $\mathcal{A}$ *can produce* $\mathtt{ACT}_i = (s_i, V_i, C_i)$ *itself.*

`Serving` $\mathcal{O}_{\mathsf{ADe}}$ `Oracle` : *The oracle* $\mathcal{O}_{\mathsf{ADe}}$ *accepts parameters* par, *index of*

the recipient k, the public keys $\{\mathsf{PK}_i\}_{i\in\mathsf{ID}}$ of senders, the aggregated and authenticated ciphertext $\mathtt{ACT} = \mathsf{MA}(\{\mathtt{ACT}_i\}_{i\in\mathsf{ID}})$. It outputs verified and decrypted messages $\{m_{\mathsf{ID}}\}_{i\in\mathsf{ID}}$.

`Query Stage 1` : *The adversary may issue ℓ_1 queries to oracles with inputs of its choice.*

`Challenge Stage` : *$\mathcal{A}$ outputs two messages $m_{i,0}$, $m_{i,1}$. The challenger chooses $b \in_{\$} \{0,1\}$ at random and computes a challenge as follows: $\mathtt{ACT}_{i,b} \leftarrow \mathsf{AEn}(\mathsf{PK}_{\mathsf{ID}}, \mathsf{SK}_{\mathsf{ID}}, \mathsf{PK}_k, m_{\mathsf{ID}})$.*

`Query Stage 2` : *The adversary may issue ℓ_2 queries to oracles with inputs of its choice, provided that $\{\mathtt{ACT}_i\}_{i\in\mathsf{ID}}$ for $\mathcal{O}_{\mathsf{ADe}}$ do not contain $\mathtt{ACT}_{i,b}$.*

`Response Stage` : *The adversary outputs a bit b' indicating which message was encrypted in the challenge stage.*

The adversary advantage in the experiment

$$\mathbf{Adv}(\mathtt{Exp}^{\lambda,\ell_1,\ell_2}_{\mathsf{ESS}}(\mathcal{A}, \mathsf{MPAE})) = |\Pr[b' = b] - 1/2|$$

is the probability of the event that $\mathcal{A}$ correctly guess which message was encrypted in the query stage. We say the scheme MPAE *provide message secrecy if the advantage of the adversary is negligible: $\mathbf{Adv}(\mathtt{Exp}^{\lambda,\ell_1,\ell_2}_{\mathsf{ESS}}(\mathcal{A}, \mathsf{MPAE})) \leq \epsilon(\lambda, \ell_1, \ell_2)$.*

A Stronger Authentication Model for MPAE

We now present a stronger security model where an impersonation attack is mounted given a malicious setup of the randomness and key leakage of one of the private keys.

Definition 26 (Ephemeral Setup Impersonation (ESI))**.** *Let* $\mathsf{MPAE} = (\mathsf{Setup}, \mathsf{UKeyGen}, \mathsf{IPKeyGen}, \mathsf{PKeyGen}, \mathsf{AEn}, \mathsf{MA}, \mathsf{ADe})$ *be a Multi-Party Authentication Encryption scheme. We define the following security experiment $\mathtt{Exp}^{\lambda,\ell}_{\mathsf{SR}}(\mathcal{A}, \mathsf{MPAE})$:*

`Init Stage` : *The challenger generates common parameters $\mathsf{par} \leftarrow \mathsf{Setup}(\lambda)$, including the device indexes Ω. For each $i \in \Omega$ it generates keys:*

$(\mathsf{upk}_i, \mathsf{usk}_i) \leftarrow \mathsf{UKeyGen}(i)$,
$(\mathsf{ippk}_i, \mathsf{ipsk}_i) \leftarrow \mathsf{IPKeyGen}(\mathsf{msk}, i, \mathsf{upk}_i)$,
$(\mathsf{SK}_i, \mathsf{PK}_i) \leftarrow \mathsf{PKeyGen}(\mathsf{upk}_i, \mathsf{ippk}_i, \mathsf{ipsk}_i)$.

It chooses a set of indexes of sending devices $\mathsf{ID} \subset \Omega$ and the receiving device k. The adversary $\mathcal{A}$ chooses one index $\bar{\imath} \in \mathsf{ID}$. Let $\bar{\mathsf{ID}} = \mathsf{ID} \setminus \{\bar{\imath}\}$.

$\mathcal{A}$ *is given all generated public data, including the public keys of the devices.*

`Serving` $\mathcal{O}_{\mathsf{AEn}}$ `Oracle:` *The oracle* $\mathcal{O}^{\bar{v}}_{\mathsf{AEn}}$ *accepts an ephemeral value* $\bar{v}$*, parameters* par*, public key* PK_i*, public key of the recipient,* PK_k*, and a message* m_i*, then outputs* $\mathtt{ACT}_i$ *which is verifiable and decryptable. The adversary issues* ℓ *number of queries to the oracle and each time setting the ephemeral* $\bar{v}$ *the oracle uses to produce* $\mathtt{ACT}_i$*. Let* $\mathcal{M} = \{m_i\}$ *denote the set of* ℓ *messages the oracles process.*

`Forgery Game I` : $\mathcal{A}$ *is given* $\mathsf{usk}_{\bar{\mathsf{I}}}$ *and return tuple:*
$\{\mathtt{ACT}^*_{\mathsf{ID}}\}_{i\in\mathsf{ID}} \leftarrow \mathcal{A}^{\mathcal{O}^{\bar{v}}_{\mathsf{AEn}}}(\mathsf{usk}_{\bar{\mathsf{I}}}, \{\mathsf{PK}_{\mathsf{ID}}\}_{i\in\mathsf{ID}}, \mathsf{PK}_k)$.

The advantage $\mathbf{Adv}^1(\mathtt{Exp}^{\lambda,\ell}_{\mathsf{SR}}(\mathcal{A}, \mathsf{MPAE}))$ *is defined as the probability of the event that* $\mathcal{A}^{\mathcal{O}^{\bar{v}}_{\mathsf{AEn}}}(\mathsf{usk}_{\bar{\mathsf{I}}}, \{\mathsf{PK}_{\mathsf{ID}}\}_{i\in\mathsf{ID}}, \mathsf{PK}_k)$ *outputs a verifiable aggregated ciphertext* $\mathtt{ACT}^*_{\bar{\mathsf{I}}}$ *decryptable to some* $m^*_{\bar{\mathsf{I}}}$ *which was not queried to* $\mathcal{O}^{\bar{v}}_{\mathsf{AEn}}$*:*

$$\Pr\left[\begin{array}{l} \{\mathtt{ACT}^*_{\mathsf{ID}}\}_{i\in\mathsf{ID}} \leftarrow \mathcal{A}^{\mathcal{O}^{\bar{v}}_{\mathsf{AEn}}}(\mathsf{usk}_{\bar{\mathsf{I}}}, \{\mathsf{PK}_{\mathsf{ID}}\}_{i\in\bar{\mathsf{ID}}}, \mathsf{PK}_k), \\ \mathtt{ACT}^* \leftarrow \mathsf{MA}(\{\mathtt{ACT}^*_i\}_{i\in\mathsf{ID}}), \\ \{m^*_{\mathsf{ID}}\}_{i\in\mathsf{ID}} \leftarrow \mathsf{ADe}(\mathsf{SK}_k, \{\mathsf{PK}_i\}_{i\in\mathsf{ID}}, \mathtt{ACT}^*), \\ m^*_{\bar{\mathsf{I}}} \in \{m^*_{\mathsf{ID}}\}_{i\in\mathsf{ID}}, \;\; m^*_{\bar{\mathsf{I}}} \notin \mathcal{M}. \end{array}\right].$$

`Forgery Game II` : $\mathcal{A}$ *is given* $\mathsf{psk}_{\bar{\mathsf{I}}}$ *and return tuple:*
$\{\mathtt{ACT}^*_{\mathsf{ID}}\}_{i\in\mathsf{ID}} \leftarrow \mathcal{A}^{\mathcal{O}^{\bar{v}}_{\mathsf{AEn}}}(\mathsf{psk}_{\bar{\mathsf{I}}}, \{\mathsf{PK}_{\mathsf{ID}}\}_{i\in\mathsf{ID}}, \mathsf{PK}_k)$.

The advantage $\mathbf{Adv}^2(\mathtt{Exp}^{\lambda,\ell}_{\mathsf{SR}}(\mathcal{A}, \mathsf{MPAE}))$ *is defined as the probability of the event that* $\mathcal{A}^{\mathcal{O}^{\bar{v}}_{\mathsf{AEn}}}(\mathsf{psk}_{\bar{\mathsf{I}}}, \{\mathsf{PK}_{\mathsf{ID}}\}_{i\in\mathsf{ID}}, \mathsf{PK}_k)$ *outputs a verifiable aggregated ciphertext* $\mathtt{ACT}^*$ *decryptable to some* $m^*_{\bar{\mathsf{I}}}$ *which was not queried to* $\mathcal{O}^{\bar{v}}_{\mathsf{AEn}}$*:*

$$\Pr\left[\begin{array}{l} \{\mathtt{ACT}^*_{\mathsf{ID}}\}_{i\in\mathsf{ID}} \leftarrow \mathcal{A}^{\mathcal{O}^{\bar{v}}_{\mathsf{AEn}}}(\mathsf{psk}_{\bar{\mathsf{I}}}, \{\mathsf{PK}_{\mathsf{ID}}\}_{i\in\bar{\mathsf{ID}}}, \mathsf{PK}_k), \\ \mathtt{ACT}^* \leftarrow \mathsf{MA}(\{\mathtt{ACT}^*_i\}_{i\in\mathsf{ID}}), \\ \{m^*_{\mathsf{ID}}\}_{i\in\mathsf{ID}} \leftarrow \mathsf{ADe}(\mathsf{SK}_k, \{\mathsf{PK}_i\}_{i\in\mathsf{ID}}, \mathtt{ACT}^*), \\ m^*_{\bar{\mathsf{I}}} \in \{m^*_{\mathsf{ID}}\}_{i\in\mathsf{ID}}, \;\; m^*_{\bar{\mathsf{I}}} \notin \mathcal{M}. \end{array}\right].$$

We say that the scheme MPAE *is secure if the advantages of the adversary in both forgery games are negligible:*

$$\mathbf{Adv}^1(\mathtt{Exp}^{\lambda,\ell}_{\mathsf{SR}}(\mathcal{A}, \mathsf{MPAE})) \leq \epsilon(\lambda, \ell),$$
$$\mathbf{Adv}^2(\mathtt{Exp}^{\lambda,\ell}_{\mathsf{SR}}(\mathcal{A}, \mathsf{MPAE})) \leq \epsilon(\lambda, \ell).$$

Improved MPAE Schemes Secure Against ESI

The original scheme is recalled in the first (left) column of Tables 4.5 and 4.6. The proposed modified versions are presented in the second (middle) and

third (right) columns. The devices use distinct HSMs: $\mathbf{f1}_i(\hat{g}) = \hat{g}_{\mathsf{ID}}^{x_i} = \hat{X}_i$, and $\mathbf{f2}_i(\hat{g}) = \hat{g}_{\mathsf{ID}}^{d_i} = \hat{D}_i$ to exponent with secret keys. In our first proposition, i.e. MPAE-1, the value $s_{\mathsf{ID}} = v_{\mathsf{ID}} + d_{\mathsf{ID}} + x_{\mathsf{ID}} \cdot h_{3,\mathsf{ID}}$ from the AEn procedure is moved to the exponent of a fresh generator of a secure group. Thus we have $\hat{S}_i = \hat{g}_{\mathsf{ID}}^{v_{\mathsf{ID}}} \mathbf{f2}_i(\hat{g}) \mathbf{f1}_i(\hat{g})^{h_{3,\mathsf{ID}}} = \hat{g}_{\mathsf{ID}}^{v_{\mathsf{ID}}} \hat{D}_i \hat{X}_i^{h_{3,\mathsf{ID}}} = \hat{g}_{\mathsf{ID}}^{v_{\mathsf{ID}}+d_{\mathsf{ID}}+x_{\mathsf{ID}} \cdot h_{3,\mathsf{ID}}}$. The verification in the ADe procedure uses a pairing function $\hat{e}$ to verify the equality of exponents. The first proposed construction MPAE-1 is asynchronous as the original scheme, so it requires $n+1$ pairing operations during verification. Since the pairing function is the most time-consuming operation, we propose a synchronous MPAE-2 construction. In this version, all messages are sent in *intervals* defined by thresholded timestamps (e.g. per whole minutes) or by challenges broadcasted from the aggregation device. We denote those timeframes via unique *com* bit sequences. It is used as the input for the hash $\mathcal{H}_g(com)$ to get the same fresh generator $\hat{g}$ for each sending device in the interval related to *com*. In MPAE-2 we only allow for one signing per device in each *com*. Otherwise a simple repetition attack could be mounted. The adversary gets two valid $\hat{S}_{\mathsf{ID}} = \hat{g}^{v_{\mathsf{ID}}+d_{\mathsf{ID}}+x_{\mathsf{ID}} \cdot h_{3,\mathsf{ID}}}$ and $\hat{S}'_{\mathsf{ID}} = \hat{g}^{v'_{\mathsf{ID}}+d_{\mathsf{ID}}+x_{\mathsf{ID}} \cdot h'_{3,\mathsf{ID}}}$ for two messages m, m' in a single *com*. It then computes $\hat{S}_{\mathsf{ID}} / \hat{S}_{\mathsf{ID}} = \hat{g}^{(v_{\mathsf{ID}}-v'_{\mathsf{ID}})+x_{\mathsf{ID}}(h_{3,\mathsf{ID}}-h'_{3,\mathsf{ID}})}$ and $\hat{g}^{x_i} = ((\hat{S}_{\mathsf{ID}} / \hat{S}_{\mathsf{ID}}) / (\hat{g}^{(v_{\mathsf{ID}}-v'_{\mathsf{ID}})}))^{(1/(h_{3,\mathsf{ID}}-h'_{3,\mathsf{ID}}))}$. Next, it computes $\hat{g}^{d_i} = \hat{S}_i / g^{v_i} (g^{x_i})^{h_{3,\mathsf{ID}}}$. Now the attacker could produce a fresh $\bar{v}_i$ and use it to encrypt a fresh m_i^* for the recipient k, as in the AEn procedure resulting with C_i^*. It uses that value in $\bar{h}_{3,\mathsf{ID}} = \mathcal{H}(V_{\mathsf{ID}}, \mathsf{n}_{\mathsf{ID}}, C_i^*, X_{\mathsf{ID}}, R_{\mathsf{ID}}, Z_{\mathsf{ID}})$. Now it can produce a verifiable forgery $\hat{S}^*_{\mathsf{ID}} = \hat{g}^{\bar{v}_i} \hat{g}^{d_i} (\hat{g}^{x_i})^{\bar{h}_{3,\mathsf{ID}}}$ for the message m_i^* encrypted to C_i^*.

Ephemeral Leakage Attack on the Original scheme

In Thm. 9 we state that the original construction from [165] is not secure in our model.

Theorem 9. *The original scheme* MPAE *in the left column of Tab. 4.6 is not secure in our new stronger model, as of Def. 26.*

Proof. After the system is initialized, the adversary $\mathcal{A}$ selects two ephemerals $v_{\bar{\imath}}, v'_{\bar{\imath}} \leftarrow_{\$} \mathbb{Z}_q$ and use them to query $\mathcal{O}^{\bar{v}}_{\mathsf{AEn}}$ twice for arbitrary messages m, m' and a recipient device k, obtaining:

$$\begin{aligned} (s_{\bar{\imath}}, V_{\bar{\imath}}, C_{\bar{\imath}}) &= \mathtt{ACT}_{\bar{\imath}} \leftarrow \mathcal{O}^{\bar{v}}_{\mathsf{AEn}}(\mathsf{PK}_{\bar{\imath}}, \mathsf{PK}_k, m), \\ (s'_{\bar{\imath}}, V'_{\bar{\imath}}, C'_{\bar{\imath}}) &= \mathtt{ACT}'_{\bar{\imath}} \leftarrow \mathcal{O}^{\bar{v}'}_{\mathsf{AEn}}(\mathsf{PK}_{\bar{\imath}}, \mathsf{PK}_k, m'). \end{aligned}$$

Next, it computes: $h_{1,k} = \mathcal{H}(\mathsf{n}_k, X_k, R_k)$, and values:

$$\begin{aligned} Z_{\bar{\imath}} &= (X_k \cdot R_k \cdot A^{h_{1,k}})^{v_{\bar{\imath}}}, \quad h_{3,\bar{\imath}} = \mathcal{H}(V_{\bar{\imath}}, \mathsf{n}_{\bar{\imath}}, C_{\bar{\imath}}, X_{\bar{\imath}}, R_{\mathsf{ID}}, Z_{\bar{\imath}}), \\ Z'_{\bar{\imath}} &= (X_k \cdot R_k \cdot A^{h_{1,k}})^{v'_{\bar{\imath}}}, \quad h'_{3,\bar{\imath}} = \mathcal{H}(V'_{\mathsf{ID}}, \mathsf{n}_{\bar{\imath}}, C'_{\bar{\imath}}, X_{\bar{\imath}}, R_{\bar{\imath}}, Z'_{\bar{\imath}}). \end{aligned}$$

These allows to build a system of equations, namely $s_{\bar{\imath}} = v_{\bar{\imath}} + d_i + x_i \cdot h_{3,\bar{\imath}}$ and $s'_{\bar{\imath}} = v'_{\bar{\imath}} + d_{\bar{\imath}} + x_i \cdot h'_{3,\bar{\imath}}$. Solving this system the adversary obtains:

$$x_i = \left((s_{\bar{\imath}} - s'_{\bar{\imath}}) - (v_{\bar{\imath}} - v'_{\bar{\imath}})\right) / (h_{3,\bar{\imath}} - h'_{3,\bar{\imath}}).$$

Having $x_{\bar{\imath}}$, the adversary $\mathcal{A}$ can easily compute the value $d_{\bar{\imath}}$ since $d_{\bar{\imath}} = s_{\bar{\imath}} - v_{\bar{\imath}} - x_{\bar{\imath}} \cdot h_{3,\bar{\imath}}$, which is the secret of $\bar{\imath}$-th device, allowing impersonation. □

4.2.5 Security Analysis

Correctness of The Proposed Schemes

Theorem 10. *The scheme* MPAE-1 *proposed in the middle column of Tab. 4.6 is correct.*

Proof. The value of Z_i computed by the verifier k, equals Z_i computed by the i-th sender. Indeed:

$$Z_i = V_i^{x_k + d_k} = g^{v_i \cdot (x_k + r_k + a \cdot h_{1,k})} = (X_k \cdot R_k \cdot A^{h_{1,k}})^{v_i}.$$

In the verification we have: $h_{3,\mathsf{ID}} = \mathcal{H}(V_{\mathsf{ID}}, \mathsf{n}_{\mathsf{ID}}, C_{\mathsf{ID}}, X_{\mathsf{ID}}, R_{\mathsf{ID}}, Z_{\mathsf{ID}})$, and therefore:

$$\begin{aligned}
\hat{e}(S, g) &= \hat{e}(\prod_{i \in \mathsf{ID}} \hat{S}_{\mathsf{ID}}, g) = \prod_{i \in \mathsf{ID}} \hat{e}(\hat{g}_{\mathsf{ID}}^{v_{\mathsf{ID}} + d_{\mathsf{ID}} + x_{\mathsf{ID}} \cdot h_{3,\mathsf{ID}}}, g) \\
&= \prod_{i \in \mathsf{ID}} \hat{e}(\hat{g}_{\mathsf{ID}}, g^{v_{\mathsf{ID}} + d_{\mathsf{ID}} + x_{\mathsf{ID}} \cdot h_{3,\mathsf{ID}}}) \\
&= \prod_{i \in \mathsf{ID}} \hat{e}(\hat{g}_{\mathsf{ID}}, V_i \cdot R_i \cdot A^{h_{1,i}} \cdot X_i^{h_{3,i}}).
\end{aligned}$$ □

Theorem 11. *The scheme* MPAE-2 *proposed in the right column of Tab. 4.6 is correct.*

Proof. Again, as in the previous proof, the value of Z_i computed by the verifier k equals Z_i computed by the i-th sender. Indeed:

$$Z_i = V_i^{x_k + d_k} = g^{v_i \cdot (x_k + r_k + a \cdot h_{1,k})} = (X_k \cdot R_k \cdot A^{h_{1,k}})^{v_i}.$$

We have $h_{3,\mathsf{ID}} = \mathcal{H}(V_{\mathsf{ID}}, \mathsf{n}_{\mathsf{ID}}, C_{\mathsf{ID}}, X_{\mathsf{ID}}, R_{\mathsf{ID}}, Z_{\mathsf{ID}})$, and therefore:

$$\begin{aligned}
\hat{e}(S, g) &= \hat{e}(\prod_{i \in \mathsf{ID}} \hat{S}_{\mathsf{ID}}, g) = \hat{e}(\hat{g}^{\sum_{i \in \mathsf{ID}} v_{\mathsf{ID}} + d_{\mathsf{ID}} + x_{\mathsf{ID}} h_{3,\mathsf{ID}}}, g) \\
&= \hat{e}(\hat{g}_{\mathsf{ID}}, g^{\sum_{i \in \mathsf{ID}} v_{\mathsf{ID}} + d_{\mathsf{ID}} + x_{\mathsf{ID}} h_{3,\mathsf{ID}}}) \\
&= \hat{e}(\hat{g}_{\mathsf{ID}}, \prod_{i \in \mathsf{ID}} V_i \cdot \prod_{i \in \mathsf{ID}} R_i \cdot A^{\sum_{i \in \mathsf{ID}} h_{1,i}} \cdot \prod_{i \in \mathsf{ID}} X_i^{h_{3,i}}) \\
&= \hat{e}(\hat{g}_{\mathsf{ID}}, V \cdot R \cdot A^{h_1} \cdot \prod_{i \in \mathsf{ID}} X_i^{h_{3,i}}).
\end{aligned}$$ □

Unforgeability of The New Schemes Under Impersonation Attacks

Theorem 12. *The scheme proposed in the middle column of Tab. 4.6 is secure in the sense of Definition 26.*

Sketch of the proof. `Init` : Let (g, g^α, g^β) be an instance of the GDH problem in $\mathsf{par} = (\mathbb{G}, \mathbb{G}_T, g, g_T, q, \hat{e})$.

`Forgery I Setup` : We generate all keys except for the device $\bar{\imath}$, for which we setup: $\mathsf{upk}_{\bar{\imath}} = X_{\bar{\imath}} = g^{x_{\bar{\imath}}} = g^\alpha$. Thus the unknown secret key $\mathsf{usk}_{\bar{\imath}} = x_{\bar{\imath}}$ equals the unknown α. We provide the adversary $\mathcal{A}$ access to $\mathcal{O}^{\bar{v}}_{\mathsf{AEn}}$, and hash $\mathcal{O}_{\{\mathcal{H}\}}$ oracles, and all public values, as in the security game.

`Forgery II Setup` : We setup the system s.t. $\mathsf{mpk} = A = g^a = g^\alpha$. Thus the unknown $\mathsf{msk} = a$ equals the unknown α. We generate all secret keys $\mathsf{usk}_{\mathsf{ID}} = x_{\mathsf{ID}}$ at random, and compute $\mathsf{upk}_{\mathsf{ID}} = X_{\mathsf{ID}}$. We compute $r_{\mathsf{ID}} \leftarrow_\$ \mathbb{Z}^*_q, R_{\mathsf{ID}} = g^{r_{\mathsf{ID}}}$, $\mathsf{n}_{\mathsf{ID}} \leftarrow_\$ \{0,1\}^l$, $h_{1,\mathsf{ID}} = \mathcal{H}(\mathsf{n}_{\mathsf{ID}}, X_{\mathsf{ID}}, R_{\mathsf{ID}})$ as in the $\mathsf{IPKeyGen}$ procedure. However, we do not compute u_i as we do not know the secret $a = \alpha$. Also all $\mathsf{psk}_{\mathsf{ID}} = d_{\mathsf{ID}}$ are unknown. We provide the adversary $\mathcal{A}$ with access to $\mathcal{O}^{\bar{v}}_{\mathsf{AEn}}$, and hash $\mathcal{O}_{\{\mathcal{H}\}}$ oracles, and all public values, as in the security game.

`Serving` $\mathcal{O}_{\mathcal{H}_g}$ `Oracle` : We allow ℓ fresh inputs to the $\mathcal{O}_{\mathcal{H}_g}$ oracle. We choose the random index $j \leftarrow_\$ \{1, \ldots, \ell\}$, which denotes the j-th invocation of $\mathcal{O}_{\mathcal{H}_g}$, for which we assume the forgery will happen.

- On the i-th, $(i \neq j)$ fresh input $V, \mathsf{n}, C, X, R, Z$, we compute $\omega \leftarrow_\$ \mathbb{Z}^*_q$, and register the value $\hat{g} = \mathcal{H}_g(V, \mathsf{n}, C, X, R, Z)$ in the ROM table for $\mathcal{H}_g$. We return $\hat{g}$ as the output.
- On the j-th fresh input $V, \mathsf{n}, C, X, R, Z$, we set $\hat{g} = (g^\beta)$ and register that value in the ROM table for $\mathcal{H}_g$. We return $\hat{g}$ as the output.

`Serving` $\mathcal{O}^{\bar{v}}_{\mathsf{AEn}}$ `Oracle` : On input $\bar{v}$ we compute $\bar{V}_i = g^{\bar{v}}$, $h_{1,k} = \mathcal{H}(\mathsf{n}_k, X_k, R_k)$, $Z_{\mathsf{ID}} = (X_k R_k A^{h_{1,k}})^{\bar{v}}$, $h_{2,\mathsf{ID}} = \mathcal{H}_l(\mathsf{n}_k, V_{\mathsf{ID}}, Z_{\mathsf{ID}})$, and $C_{\mathsf{ID}} = h_{2,i} \oplus m_{\mathsf{ID}}$, $h_{3,\mathsf{ID}} = \mathcal{H}(V_{\mathsf{ID}}, \mathsf{n}_{\mathsf{ID}}, C_{\mathsf{ID}}, X_{\mathsf{ID}}, R_{\mathsf{ID}}, Z_{\mathsf{ID}})$. Next, we serve a call to the hash function $\mathcal{H}_g(V_i, \mathsf{n}_i, C_i, X_i, R_i, Z_i)$, and if $(V_i, \mathsf{n}_i, C_i, X_i, R_i, Z_i)$ was not j-th fresh input to $\mathcal{O}_{\mathcal{H}_g}$, we return $\hat{g}_i = g^{\omega_i}$ for some ω_i registered in the ROM table. We compute $\hat{S}_{\mathsf{ID}} = \hat{g}_{\mathsf{ID}}^{v_{\mathsf{ID}} + d_{\mathsf{ID}} + x_{\mathsf{ID}} \cdot h_{3,\mathsf{ID}}}$ as $(V_{\mathsf{ID}} R_{\mathsf{ID}} A^{h_{1,\mathsf{ID}}} X_i^{h_{3,i}})^{\omega_i}$, and return $\mathtt{ACT}_{\mathsf{ID}} = (\hat{S}_{\mathsf{ID}}, V_{\mathsf{ID}}, C_{\mathsf{ID}})$. The verification in ADe holds as: $\hat{e}(S, g) = \hat{e}(\prod_{i \in \mathsf{ID}} \hat{S}_i, g) = \prod_{i \in \mathsf{ID}} \hat{e}(g^{\omega_i (v_{\mathsf{ID}} + d_{\mathsf{ID}} + x_{\mathsf{ID}} h_{3,\mathsf{ID}})}, g)$ which equals $\prod_{i \in \mathsf{ID}} \hat{e}(g^{\omega_i}, g^{(v_{\mathsf{ID}} + d_{\mathsf{ID}} + x_{\mathsf{ID}} h_{3,\mathsf{ID}})})$, which equals $\prod_{i \in \mathsf{ID}} \hat{e}(\hat{g}_{\mathsf{ID}}, V_{\mathsf{ID}} R_{\mathsf{ID}} A^{h_{1,\mathsf{ID}}} X_i^{h_{3,i}})$.

`Processing Forgery Game I` : According to the *Forking Lemma* for the hash $h_{3,\bar{\imath}} = \mathcal{H}(V_{\bar{\imath}}, \mathsf{n}_{\bar{\imath}}, C_{\bar{\imath}}, X_{\bar{\imath}}, R_{\bar{\imath}}, Z_{\bar{\imath}})$ the attacker returns two valid forgeries $\mathtt{ACT}^{*}_{\bar{\imath}}, \mathtt{ACT}^{*'}_{\bar{\imath}}$ for the respective tuples $(v_{\bar{\imath}}, h_{3,\bar{\imath}}, \hat{S}_{\bar{\imath}})$, $(v_{\bar{\imath}}, h'_{3,\bar{\imath}}, \hat{S}'_{\bar{\imath}})$ with the same randomness $v_{\bar{\imath}}$. We compute $\hat{S}_{\bar{\imath}}/\hat{S}'_{\bar{\imath}} = \hat{g}_{\bar{\imath}}^{v_{\bar{\imath}}+d_{\bar{\imath}}+x_{\bar{\imath}}h_{3,\mathsf{ID}}}/\hat{g}_{\bar{\imath}}^{v_{\bar{\imath}}+d_{\bar{\imath}}+x_{\bar{\imath}}h'_{3,\bar{\imath}}}$ equal to $\hat{g}_{\mathsf{ID}}^{x_{\bar{\imath}}(h_{3,\bar{\imath}}-h'_{3,\bar{\imath}})}$. With the non-negligible probability $1/\ell$, related to j-th fresh input to $\mathcal{O}_{\mathcal{H}_g}$, then $\hat{g}_{\bar{\imath}}$ equals g^{β} in both tuples. Thus we have $\hat{S}_{\bar{\imath}}/\hat{S}'_{\bar{\imath}} = g^{\beta x_{\bar{\imath}}(h_{3,\bar{\imath}}-h'_{3,\bar{\imath}})}$. Since we set $X_{\bar{\imath}} = g^{x_{\bar{\imath}}} = g^{\alpha}$, we can compute $g^{\alpha\beta} = (\hat{S}_{\bar{\imath}}/\hat{S}'_{\bar{\imath}})^{(1/(h_{3,\bar{\imath}}-h'_{3,\bar{\imath}}))}$, breaking the given instance of GDH.

`Processing Forgery Game II` : The successful attacker returns a verifiable and decryptable $\mathtt{ACT}^{*}_{\bar{\imath}} = (\hat{S}_{\bar{\imath}}, V_{\bar{\imath}}, C_{\bar{\imath}})$. With the non-negligible probability $1/\ell$, it is related to j-th fresh input to $\mathcal{O}_{\mathcal{H}_g}$ - so $\hat{g}_{\bar{\imath}}$ equals g^{β}. Thus we have $\hat{S}_{\bar{\imath}} = \hat{g}_{\bar{\imath}}^{v_{\bar{\imath}}+d_{\bar{\imath}}+x_{\bar{\imath}}h_{3,\mathsf{ID}}} = (g^{\beta})^{(v_{\bar{\imath}}+(r_{\bar{\imath}}+a\cdot h_{1,\bar{\imath}})+x_{\bar{\imath}}h_{3,\mathsf{ID}})}$. Since we set $\mathsf{msk} = A = g^{a} = g^{\alpha}$ we can compute $g^{\alpha\beta} = (\hat{S}_{\bar{\imath}}/(g^{\beta})^{(v_{\bar{\imath}}+r_{\bar{\imath}}+x_{\bar{\imath}}h_{3,\mathsf{ID}})})^{1/h_{1,\bar{\imath}}}$ for $h_{1,\bar{\imath}} = \mathcal{H}(\mathsf{n}_{\bar{\imath}}, X_{\bar{\imath}}, R_{\bar{\imath}})$, breaking the given instance of GDH. □

Theorem 13. *The scheme proposed in the right column of Tab. 4.6 is secure in the sense of Definition 26.*

Sketch of the proof. Essentially similar to the proof of Thm. 12, changing the definition of the oracle $\mathcal{O}_{\mathcal{H}_g}$, which with input *com* is unique for a single time frame, returning the same $\hat{g}$ for all devices. And devices are allowed to sign and authenticate only one message per single frame *com*. □

4.2.6 Performance Analysis

Our proposed schemes are based on symmetric pairings. However, a scheme based on asymmetric pairings would still be secure under a similar security analysis. All testing was performed on Ubuntu 18.04.5 LTS with an Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz. The implementation was done in Python using the Charm crypto library [7] with the SS512 curve for pairings. Tab. 4.7 shows the number of operations of each procedure in brackets, i.e. [`G:mul`, `G:exp`, `G:hashTo`, `Pairing`] where n is the number of `IOTD` senders. Tab. 4.8 shows the total number of fundamental operations for each scheme.

4.2.7 Conclusion

We have considered a multi-party authenticated encryption scheme that fits well for NB-IoT systems over 5G architectures, e.g. connected vehicle and railway infrastructures. We analyze the previous proposed scheme [165] in the new stronger security model with ephemeral key leakage and prove it

Table 4.7: Procedure performance and scalability analysis, all timings are measured in milliseconds.

Procedure	MPAE	MPAE − 1	MPAE − 2
Setup	[0, 1, 0, 0] 3.9131	[0, 1, 0, 0] 6.1523	[0, 1, 0, 0] 6.1680
UKeyGen	[0, n, 0, 0] 1.2198	[0, n, 0, 0] 1.2215	[0, n, 0, 0] 1.2302
IPKeyGen	[0, n, 0, 0] 2.4871	[0, n, 0, 0] 2.3537	[0, n, 0, 0] 2.3547
PKeyGen	[0, n, 0, 0] 1.1860	[0, n, 0, 0] 1.1177	[0, n, 0, 0] 1.1242
AEn	[2, $3n$, 0, 0] 8.6258	[2, $4n$, 1, 0] 11.7533	[2, $4n$, 1, 0] 11.7580
MA, $n = 1$	[0, 0, 0, 0] 0.0012	[n, 0, 0, 0] 0.0013	[n, 0, 0, 0] 0.0013
ADe, $n = 1$	[$2 + 3n$, $2n$, 0, 0] 0.0096	[$3n$, $3n$, n, $n + 1$] 0.0114	[$3n + 2$, $3n$, 1, 2] 0.0109
MA, $n = 10$	[0, 0, 0, 0] 0.0037	[n, 0, 0, 0] 0.0512	[n, 0, 0, 0] 0.0512
ADe, $n = 10$	[$2 + 3n$, $2n$, 0, 0] 0.0543	[$3n$, $3n$, n, $n + 1$] 0.1042	[$3n + 2$, $3n$, 1, 2] 0.0496
MA, $n = 100$	[0, 0, 0, 0] 0.0204	[n, 0, 0, 0] 0.5563	[n, 0, 0, 0] 0.5564
ADe, $n = 100$	[$2 + 3n$, $2n$, 0, 0] 0.4566	[$3n$, $3n$, n, $n + 1$] 1.1097	[$3n + 2$, $3n$, 1, 2] 0.4562

Table 4.8: Comparison of overall complexity.

Operation	Time (ms)	MPAE	MPAE − 1	MPAE − 2
`G:mul`	0.0033	$3n + 4$	$4n + 2$	$3n + 4$
`G:exp`	1.1089	$8n + 1$	$10n + 1$	$10n + 1$
`G:hashTo`	2.5223	0	$n + 1$	2
`Pairing`	0.6855	0	n	2

to be insecure. Subsequently we propose two modifications of that scheme, synchronized and asynchronized, which are provable secure in our stronger model. Our proposed model consider potential malicious ephemeral injections during both the setup phase and the execution of the complete protocol. Our proof of concept implementation shows the feasibility and scalability of the schemes in terms of participating nodes; the increased security only impact the performance marginally.

We conclude that our proposed schemes thus contributes to address the security challenges Sec1-Sec3 and Sec5; both schemes provides secure transmission of messages in intelligent transportation systems (Sec1), utilizes a `KGC` central for certificateless crypto systems (Sec2), and mitigates impersonation attacks (Sec3) in the stronger ephemeral leakage model (Sec5).

4.3 Scenario $\mathsf{P}_{\mathsf{III}}$: Circuit-Breaking Environments with Secure Signatures

The following section is based on the published paper $\mathsf{P}_{\mathsf{III}}$[3].

System and ENV setup: Our system model contain devices equipped with dual HSMs, that share the same source of randomness. Secret keys are split into parts $\mathsf{sk} = \mathsf{sk}_1 + \mathsf{sk}_2$ and stored in those HSMs. Before computing a signature, each key inside the HSMs are updated according to the random sample ξ, i.e. $\mathsf{sk}_1 = \mathsf{sk}_1 + \xi$, and $\mathsf{sk}_2 = \mathsf{sk}_2 - \xi$. Close-by devices can sample the same environment, seeding their internal pseduorandom generators with the same values. This feature will allow *circuit breaking functions* which in turn creates a *circuit breaking environment* (CBE), where the system of devices will alert if any of the participants are removed and the psuedorandomness is desynchronized. We generalize the device model into two separate devices, each with just one HSM, that collaboratively produce multi-signatures. Obviously, if secrets are to be updated synchronically, the devices must share the same source of randomness. Otherwise the resulted multi-signature would not be verified positively. This enables the circuit breaker function for the collaborative computation, since any deviations in the seeding will alert that the system of signing nodes is desynchronised.

The scenario we consider in this environment is a railway transportation setting; we consider a train $\mathcal{T}$ with a set of carriages. The transportation follows a route which continuously collect environment randomness with or without the help of infrastructure sensory devices. We fix one carriage $\mathcal{I}$ to be a *signer*, having HSM1 installed. Synchronized with $\mathcal{I}$ is the second signer $\mathcal{R}$ which is the cargo itself, with HSM2 installed. Signature creation and verification occurs continuously. The functionalities of these HSMs, denoted as **f1** and **f2**, performs a synchrounous computation of a signature. It could be later verified with the help of the partial aggregated public keys. From the circuit breaker feature, any changes in the shared environment resulting in a desynchronization, the devices and will then disable the correct verification of the signature. Overall, this creates a secure function of signature creation and verification within a CBE. Hence, it would solve the challenge of partial key leakage from collaborative devices relying on synchronization. We note that the randomness used in this model differs from previous scenarios where the

[3]Published at IEEE International Conference on Trust, Security and Privacy in Computing and Communications conference and is under copyright © 2022 IEEE.

ephemeral values are leaked (as in the Schnorr scheme); in this environment the randomness is used for seeding the devices and refreshing the secret keys instead. An overview of the system model is shown in Fig. 4.4.

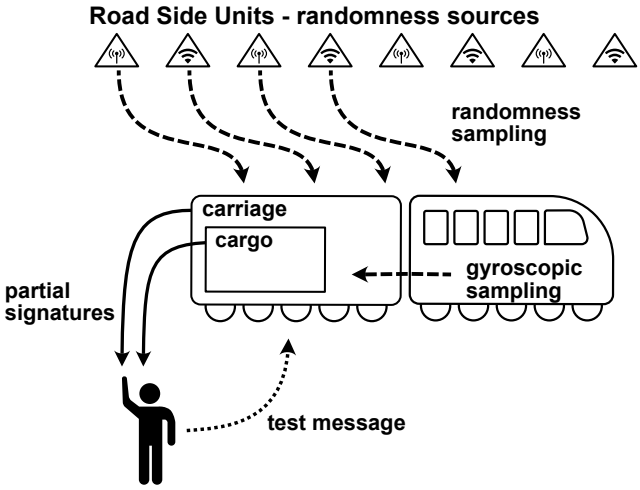

Figure 4.4: A carriage $\mathcal{I}$, and a cargo $\mathcal{R}$ sample randomness from gyroscopic data, and from track-side units along the route.

4.3.1 Security Requirements

We formulate the following security requirements for the system of generating and verifying messages:

- Short signatures over connectivity status messages between a cargo device and carriage device must be verfied and secure against impersonation attacks.
- Since different devices may have different vendors, and stationary cargo devices may be open for side channel attacks, the signature protocol must be secure against ephemeral leakage.
- The protocol must be usable for a CBE with devices using separate HSM areas, handle the private keys, where the randomness is synchronized for establish the secure CBE.

Our protocol is using the efficient BLS signature scheme, combined with our key split and refresh method during each signature creation. This scenario thus addresses challenges Sec1, Sec3 and Sec5, in a dual-party setting with close-by devices, synchronized in a CBE.

4.3.2 Related Work

The threat of memory compromising in the form of a *bounded memory-leakage* model was introduced in [6] for public key settings and for symmetric encryption in [52, 47]. In the model, an adversary have access to f, which is an efficiently computable leakage function that takes a secret key sk as input. $f(\mathsf{sk})$ then partially output bits from the key, up to some fixed leakage parameter λ - thus referred to as bounded. The model is under the condition that regardless of how much of computations occur, the leakage is still bounded in a fixed size, corresponding to λ. Another model is the *continuous* (unbounded) leakage model, introduced by Brakerski et. al [30] and Dodis et al [48]. In the continuous memory-leakage model, no restrictions are set of neither time or memory. When the adversary calls the leakage function, it may only receive at most a specified fraction of the total bits from the internal state of the attacked memory, which consists of the secret key and the entropy source [30]. The internal state updates with fresh randomness in certain time periods. The adversary is only allowed to serve calls to the leakage function f during these time periods. However, the model allows the adversary to use f continuously over all elapsed time periods, each ending with an update of the secret key. Multi-signature schemes can use different building blocks, e.g. Schnorr signatures [17, 128] or BLS signatures [21]. Several key split and refresh mechanisms have been proposed [163, 145, 88, 63, 129].

4.3.3 Threat Model

Let $\mathcal{A}$ be an active polynomial-time adversary being able to capture messages between a signer and verifer. We then consider three main threat scenarios:

Partial key leakage in dual HSM architectures : We consider an attacker $\mathcal{A}$ that is allowed to access a partial key share from HSM1 or HSM2 within a signing device. However, $\mathcal{A}$ is limited to only access one of the HSMs at the time during a signing session. This threat models the fact that key leakage can occur due to a compromised HSM manufactured by an untrusted vendor, or pre-shipping tampering of the HSM by the adversary.

Partial key leakage in collaborative signatures : In this scenario two devices collaboratively computes a multi-signature using their own synchronized HSMs, using pseudo randomness seeded via a CBE. We then consider an attacker model in which we allow $\mathcal{A}$ to access partial secret keys. We assume $\mathcal{A}$ can learn secrets from one HSM of a chosen device per signing session, but is still able to compromise a different HSM of

another device in different sessions - in one session $\mathcal{A}$ accesses sk_1 of $\mathcal{I}$ and in another session sk_2 of $\mathcal{R}$. This reflects an additional scenario of using an HSM from an untrusted vendor, or with implementation errors, where two devices collaboratively produces a multi-signature where a malicious device manufacturer allows leakages from one HSM per session.

Transportation in CBE: We consider in particular a threat model for the sensitive cargo scenario described in the introduction. Given a mode of transportation, e.g. a train $\mathcal{T}$ running on a connected railway with trackside units, we consider a certain cargo $\mathcal{I}$ that is registered together with a specific train carriage $\mathcal{R}$ which holds the cargo. Each part have a HSM and the carriage is the boundary of the CBE. An attacker may then try to disconnect or remove the cargo from the carriage, or compromise the HSMs.

We briefly note that rough public-key attacks, i.e. where an active adversary is able to generate its own public keys and register them for some unsuspecting signing party, are not possible in CBE scenarios. When a party is registering a key-pair to be used it must be in synchronization with the corresponding co-signing party, within the (secure) CBE setup. Therefore, no rough public keys can be inserted due to the controlled environment.

4.3.4 Preliminaries

We recall that *minimal functionality* of a module, such as an HSM, is the function $\mathbf{f}$ on module inputs with sk, stored inside the HSM. To illustrate this, assume the device implements a BLS signature scheme set in a group $\langle g \rangle = \mathbb{G}$ where the DLP holds. The HSM takes, through its input interface, the group element $h = \mathcal{H}(m) \in \mathbb{G}$, performs an exponentiation with sk and outputs h^{sk}. We denote that operation as $\mathbf{f}(h) = h^{\mathsf{sk}}$, where the exponentiation is the minimal functionality $\mathbf{f}$ depicted in Fig. 4.5.

Our constructions are based on BLS signatures [21], which rely on the computational co-Diffie-Hellman assumption ($\mathsf{co}-\mathsf{CDH}$). However, our contribution reduce the security of BLS with the key splitting and refreshment to unforgeability of regular BLS, and pseudorandomness PRNG function. We denote negligible values as ϵ. Let PRNG be a pseudorandom number generator that is initialized with a secret seed $\xi_0 \in \mathbb{Z}_q$. On each i-th call it is updated with a fresh input $\xi_i \in \mathbb{Z}_q$, and as a result it outputs elements from $\mathbb{Z}_q$. This reflects the function that can sample "environmental" randomness of such a high entropy that can be mapped to $\mathbb{Z}_q$, to update its internal

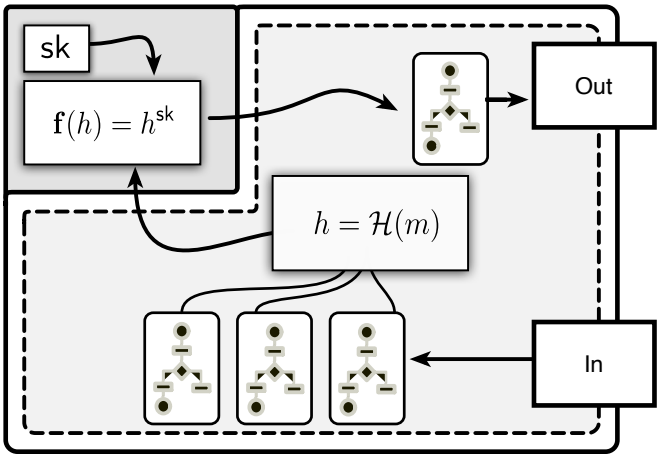

Figure 4.5: Device with HSM of $\mathbf{f}(h) = h^{\mathsf{sk}}$ functionality.

states, and to finally output pseudorandom results. Obviously the i-th result is determined via the $\xi_0, \ldots, \xi_i$ sequence. We denote such result related to the whole sequence by $\mathsf{PRNG}(\xi_i)$ for short.

Definition 27 (PRNG Pseudorandomness)**.** *Let $\bar{\xi}_k$ denote the uncomplete partial sequence $\{\xi_0, \ldots, \xi_i\} \setminus \{\xi_k\}$, where $k \in \{0, \ldots, i\}$. There is no efficient probabilistic algorithm $\mathcal{A}_{\mathsf{PRNG}}$ that given $\bar{\xi}_k$, for any $k \in \{0, \ldots, i\}$, distinguishes with a non-negligible probabilityfon between two distributions $D_1 = (\bar{\xi}_k, \mathsf{PRNG}(\xi_i))$ and $D_0 = (\bar{\xi}_k, r)$, where r is chosen at random from $\mathbb{Z}_q$. That is, for any efficient probabilistic algorithm $\mathcal{A}_{\mathsf{PRNG}}$ the advantage*

$$\mathsf{Adv}(\mathcal{A}_{\mathsf{PRNG}}) = |\Pr[\mathcal{A}_{\mathsf{PRNG}}(D_0) = 1] - \Pr[\mathcal{A}_{\mathsf{PRNG}}(D_1) = 1]| \leq \epsilon_{\mathsf{PRNG}}, \quad (4.11)$$

where ϵ_{PRNG} is negligible.

Comment: This type of PRNG function can be realized with an efficient hash function $\mathcal{H} : \{0,1\}^* \to \mathbb{Z}_q$, modeled as ROM. Namely, by recursion we get that $\mathsf{PRNG}(\xi_i) = \mathcal{H}(\mathsf{PRNG}(\xi_{i-1}), \xi_i)$, for $\mathsf{PRNG}(\xi_0) = \mathcal{H}(\xi_0)$.

4.3.5 Proposed Scheme

Separated HSM for Key Splitting

We propose a signature scheme with split BLS signatures with refresh updates, utilizing signature secret/public key pairs $(\mathsf{sk}, \mathsf{pk} = g^{\mathsf{sk}})$. The signing device is augmented with two separate HSMs. The secret key is split into

parts $\mathsf{sk} = \mathsf{sk}_1 + \mathsf{sk}_2$ stored in those HSMs. Before signing, the keys in the HSMs are updated according to the random sample ξ, i.e. $\mathsf{sk}_1 = \mathsf{sk}_1 + \xi$, and $\mathsf{sk}_2 = \mathsf{sk}_2 - \xi$. The device model is depicted in Fig. 4.6. Thus when

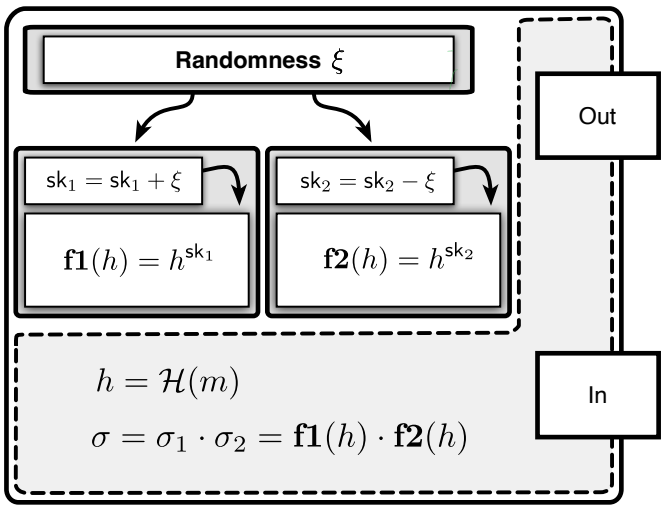

Figure 4.6: Signing device with two HSM for additive BLS split.

realized with a PRNG it gives shares that are refreshed in each session i in a synchronized way: $\mathsf{sk}_1^{(i)} = \mathsf{sk}_1^{(i-1)} + \mathsf{PRNG}(\xi_i)$, and $\mathsf{sk}_2^{(i)} = \mathsf{sk}_2^{(i-1)} - \mathsf{PRNG}(\xi_i)$.

Definition 28 (Signatures with key split and refresh (SIGSRF))**.** *Signatures with key split and refresh is defined as a tuple of procedures* SIGSRF = (ParGen, KeyGen, InitRF, RF, SignRF, Ver)*:*

ParGen$(\lambda) \to$ par*: takes security parameter λ and outputs parameters* par*. These are default parameters of the subsequent procedures in the scheme, therefore we omit them for simplicity of notation.*

KeyGen(par) $\to$ (sk, pk)*: takes parameters* par *and output a pair of secret and public keys* sk, pk *respectively.*

InitRF(sk) $\to$ (**f1**, **f2**)*: takes secret key* sk *splits it to parts* $\mathsf{sk}_1, \mathsf{sk}_2$ *and stores it securely in SM1, SM2 with functionalities* **f1**, **f2** *respectively.*

RF(**f1**, **f2**) $\to$ (**f1**, **f2**)*: takes modules* HSM*1,* HSM*2 with respective functionalities* **f1**, **f2***, and refreshes the partial keys* $\mathsf{sk}_1, \mathsf{sk}_2$ *stored in these modules.*

SignRF(m, **f1**, **f2**) $\to \sigma$*: Takes a message m and processes it with* HSM*1 and* HSM*2 via* **f1**, **f2** *respectively, outputting a signature σ.*

$\mathsf{Ver}(m, \sigma, \mathsf{pk}) \rightarrow 1/0$*: Returns* 1 *for "accept", or* 0 *for "reject"*

Definition 29 (SIGSRF correctness)**.** *Let* SIGSRF = (ParGen, KeyGen, InitRF, RF, SignRF, Ver) *is a signature scheme with key split and refresh.* SIGSRF *is correct if for any message* m *and any integer* ℓ*:*

$$\Pr\left[\begin{array}{l} \mathsf{ParGen}(\lambda) \rightarrow \mathsf{par}, \\ \mathsf{KeyGen}(\mathsf{par}) \rightarrow (\mathsf{sk}, \mathsf{pk}) \\ \mathsf{InitRF}(\mathsf{sk}) \rightarrow (\mathbf{f1}, \mathbf{f2}) \\ \text{for } i = 1 \textit{ to } \ell \text{ run } \mathsf{RF}(\mathbf{f1}, \mathbf{f2}) \\ \mathsf{SignRF}(m, \mathbf{f1}, \mathbf{f2}) \rightarrow \sigma \\ \mathsf{Ver}(m, \sigma, \mathsf{pk}) \rightarrow 1 \end{array}\right] = 1.$$

Additive BLS Key Splitting and Refresh Mechanisms

As shown in [145], an additive key splitting scheme with BLS extends into a signature split, with the long-term secret key divided between two units to mitigate forgery attacks, based on partial key leakages from one HSM. In our scenario we consider a more powerful adversary which can reveal partial secrets from both HSMs, provided that only one chosen HSM is attacked in a single signing session.

We define the following procedures:

Additive initialization $\mathsf{InitRF}(\mathsf{sk})$: Two HSMs are initialized in the following way. In the first stage of the initial session ($i = 0$) the secret key sk is split by computing at random $\mathsf{sk}_1^{(0)} \xleftarrow{\$} \mathbb{Z}_q$, then $\mathsf{sk}_2^{(0)} = \mathsf{sk} - \mathsf{sk}_1^{(0)}$. The sk_1 is stored in HSM1 and sk_2 in HSM2. The randomness ξ_0 is used to initialize PRNG of HSM1 and HSM2.

Additive refreshment $\mathsf{RF}(\mathbf{f1}, \mathbf{f2})$: The refreshed keys inside the HSMs in each session $i = 1, 2, \ldots$ are based on updating the corresponding values from previous session by the current output from $\mathsf{PRNG}(\xi_i)$, namely: $\mathsf{sk}_1^{(i)} = \mathsf{sk}_1^{(i-1)} + \mathsf{PRNG}(\xi_i)$, $\mathsf{sk}_2^{(i)} = \mathsf{sk}_2^{(i-1)} - \mathsf{PRNG}(\xi_i)$. Therefore in each session i the following relation holds: $\mathsf{sk} = \mathsf{sk}_1^{(i)} + \mathsf{sk}_2^{(i)}$.

Additive sign $\mathsf{SignRF}(m, \mathbf{f1}, \mathbf{f2})$: We denote $\mathbf{f1}(x) = x^{\mathsf{sk}_1^{(i)}}$, and $\mathbf{f2}(x) = x^{\mathsf{sk}_2^{(i)}}$ as minimal functions of the HSMs, which exponentiate the input x with the split secret key shares, updated in a synchronized way in every session. Thus we have $\mathbf{f1}(x) \cdot \mathbf{f2}(x) = x^{\mathsf{sk}_1^{(i)}} \cdot x^{\mathsf{sk}_2^{(i)}} = x^{\mathsf{sk}}$.

Generalization to multi-signatures with synchronized secrets refreshment

If we consider two separate distinct signing devices $\mathcal{I}$, $\mathcal{R}$ each equipped with just one HSM for storing secret keys sk_1, sk_2 respectively, but accessing the same external source of randomness ξ, then the above mention setup is equivalent to regular multi-signatures, as of [22], for which the refreshment procedure is applied. In this context, the procedures are as follows:

Additive initialization $\mathsf{InitRF}(\mathsf{sk})$: Each device is initialized separately with its own pair of keys, i.e. the first device $\mathcal{I}$ with $(\mathsf{sk}_1, \mathsf{pk}_1 = g_2^{\mathsf{sk}_1})$, and $\mathcal{R}$ with $(\mathsf{sk}_2, \mathsf{pk}_2 = g_2^{\mathsf{sk}_2})$. Then for the external verifier all devices perform a proof of knowledge of secret keys, just by signing a random message chosen by the verifier, to mitigate potential rogue public-key attack [22, 111]. The aggregated public key is computed $\mathsf{pk} = \mathsf{pk}_1 \cdot \mathsf{pk}_2$. The randomness ξ_0 is used to initialize PRNGs of $\mathcal{I}$ and $\mathcal{R}$.

Additive refreshment $\mathsf{RF}(\mathbf{f1}, \mathbf{f2})$: The keys inside the devices, in each session i, are updated by the current value from $\mathsf{PRNG}(\xi_i)$, namely: $\mathsf{sk}_1^{(i)} = \mathsf{sk}_1^{(i-1)} + \mathsf{PRNG}(\xi_i)$, $\mathsf{sk}_2^{(i)} = \mathsf{sk}_2^{(i-1)} - \mathsf{PRNG}(\xi_i)$. Therefore in each session i the following relation holds: $\mathsf{pk} = g_2^{\mathsf{sk}_1^{(i)}} \cdot g_2^{\mathsf{sk}_2^{(i)}}$.
Note 1. The key refreshment must occur synchronously in both devices per each signing session. This could be triggered just before signing, after obtaining a message m to sign, and after sampling the randomness in the same interval.
Note 2. Each key refresh would require sampling a fresh randomness from the source in a specific intervals of predefined granularity (at the beginning of a second/minute, etc.).
Note 3. Potential desynchronization of the randomness, e.g. due to removing the device from the system (removing a cargo from a carriage - breaking the boundary of the CBE) is regarded as the advantage in our scenarios. Synchronization is a plausible feature protecting against device removal from CBE.

Additive sign $\mathsf{SignRF}(m, \mathbf{f1}, \mathbf{f2})$: This is a multi-signature procedure. Each device produce a partial signature $\sigma_1 = \mathbf{f1}(x) = x^{\mathsf{sk}_1^{(i)}}$, and $\sigma_2 = \mathbf{f2}(x) = x^{\mathsf{sk}_2^{(i)}}$ as minimal functions of the HSMs, which exponentiate the input x with the split secret key shares, updated in a synchronized way in every session. Thus we have the multi-signature $\sigma = \sigma_1 \cdot \sigma_2 = x^{\mathsf{sk}_1^{(i)}} \cdot x^{\mathsf{sk}_2^{(i)}} = x^{\mathsf{sk}}$.

Comment: A similar refreshment procedure cannot be directly applied to the refined multi-signatures with a specially elaborated public-key aggregation [24], which mitigates the rogue public-key attack [22, 111]. It is an open problem for future research to find if (and how) the refreshment can be efficiently applied for such a public-key aggregation.

4.3.6 Security Analysis

In this section we provide a security analysis for our proposed scheme, using dual HSMs and a key split and refresh mechanism.

Theorem 14. *The additive BLS signature split realized via the* HSM *architecture (presented in Sec, 4.3.5) is correct in our proposed* SR *model, as of Definition 38.*

Proof. $\mathsf{InitRF}(\mathsf{sk}) \rightarrow (\mathbf{f1}, \mathbf{f2})$ initializes HSM1, HSM2 with $\mathsf{sk}_1^{(0)} \leftarrow \mathsf{PRNG}(0)$, $\mathsf{sk}_2^{(0)} = \mathsf{sk} - \mathsf{sk}_1^{(0)}$ respectively. Before signing in each session i, the key refresh is executed by $\mathsf{RF}(\mathbf{f1}, \mathbf{f2}) \rightarrow (\mathbf{f1}, \mathbf{f2})$. Therefore $\mathsf{sk}_1^{(i)} = \sum_i \mathsf{PRNG}(\xi_i)$ and $\mathsf{sk}_2^{(i)} = \mathsf{sk} - \sum_i \mathsf{PRNG}(\xi_i)$. Next, signing in any session i by $\mathsf{SignRF}(m, \mathbf{f1}, \mathbf{f2})$, in a parallel architecture $\mathbf{f1}(m) \cdot \mathbf{f2}(m)$, will result in $\sigma = \mathcal{H}(m)^{\mathsf{sk}_1^{(i)}} \cdot \mathcal{H}(m)^{\mathsf{sk}_2^{(i)}}$. The verification holds since:

$$\begin{aligned}\hat{e}(\sigma, g) &= \hat{e}(\mathcal{H}(m)^{\mathsf{sk}_1^{(i)}} \cdot \mathcal{H}(m)^{\mathsf{sk}_2^{(i)}}, g) = \hat{e}(\mathcal{H}(m)^{\mathsf{sk}_1^{(i)} + \mathsf{sk}_2^{(i)}}, g) \\ &= \hat{e}(\mathcal{H}(m)^{\sum_i \mathsf{PRNG}(\xi_i) + \mathsf{sk} - \sum_i \mathsf{PRNG}(\xi_i)}, g) \\ &= \hat{e}(\mathcal{H}(m)^{\mathsf{sk}}, g) = \hat{e}(\mathcal{H}(m), \mathsf{pk}).\end{aligned}$$

□

Corollary 1. *Unless the devices in a "multi-signature setup" are synchronized, the verification of the resulting multi-signature holds. If the circuit is broken by missing at least one randomness sample ξ_i from the equation, the verification would not hold, which would signal a potential attack (e.g. removing the cargo from the carriage).*

To address the scenario with partial secret key leakage, we propose a new, stronger security model for key split BLS schemes. In this model a malicious forger $\mathcal{F}$ has the ability to query an additional SignReveal oracle $\mathcal{O}_{\mathsf{SignRev}}$. The oracle is queried with a message m and the index (1 or 2) of one HSM for the key leakage. It returns the final signature, the partial secret stored in the indicated HSM, and the partial signature of that HSM. Such a leakage can happen only once per signing session.

Definition 30 (Sign Reveal Unforgeability (SR))**.** *Let* $\mathsf{SIGSRF} = (\mathsf{ParGen}, \mathsf{KeyGen}, \mathsf{InitRF}, \mathsf{RF}, \mathsf{SignRF}, \mathsf{Ver})$ *be a split signature scheme. We define security experiment* $\mathtt{Exp}^{\lambda,\ell}_{\mathsf{SR}}(\mathsf{SIGSRF})$*:*

`Init` : $\mathsf{par} \leftarrow \mathsf{ParGen}(\lambda)$ $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(\mathsf{par})$.

`SignRev Oracle` : *There are two potential calls:*

- $\mathcal{O}_{\mathsf{SignRev}}(m, 1) \rightarrow (\sigma_1, \mathsf{sk}_1, \sigma)$ *takes a message* m*, the* HSM *indicator* $i = 1$ *and outputs the following: a partial signature computed via* HSM*1 with minimal functionality* **f1***, the fresh partial secret key stored in* HSM*1, and a final signature* σ *generated with both* HSM*1 and* HSM*2, such that* $\mathsf{Ver}(\sigma, \mathsf{pk}, m) = 1$.
- $\mathcal{O}_{\mathsf{SignRev}}(m, 2) \rightarrow (\sigma_2, \mathsf{sk}_2, \sigma)$ *takes a message* m*, the* HSM *indicator* $i = 2$*, and outputs the following: a partial signature computed via* HSM*2 with minimal functionality* **f2***, the fresh partial secret key stored in* HSM*2, and a final signature* σ *generated with both* HSM*1 and* HSM*2, i.e.* $\mathcal{O}_{\mathsf{SignRev}}(m, 2) \rightarrow (\sigma_1, \mathsf{sk}_2, \sigma)$*, such that* $\mathsf{Ver}(\sigma, \mathsf{pk}, m) = 1$*,* $\sigma_1 = \mathbf{f1}(\mathcal{H}(m))$.

  The oracle models the device in which the partial secret key leakage happens only once and only from one HSM *per signing session.*

`Hash Oracle` : *The hash oracle* $\mathcal{O}_{\mathcal{H}}$ *is modeled in ROM.*

`Adversary` : *Let the adversary* $\mathcal{F}^{\mathcal{O}_{\mathsf{SignRev}}, \mathcal{O}_{\mathcal{H}}}(\mathsf{pk})$*, be a malicious algorithm initialized with the public key* pk*, having access to the oracles* $\mathcal{O}_{\mathsf{SignRev}}$ *and* $\mathcal{O}_{\mathcal{H}}$*. It issues* ℓ *number of queries to the oracles. Let* $\mathcal{M} = \{m_i\}_1^{\ell}$*, and* $\Omega = \{\sigma_i\}_1^{\ell}$ *denote the set of the messages, and the corresponding signatures the oracles process.*

`Forgery` : *The adversary generates a tuple:*
$(m^*, \sigma^*) \leftarrow \mathcal{F}^{\mathcal{O}_{\mathsf{SignRev}}, \mathcal{O}_{\mathcal{H}}}(\mathsf{pk})$ *for a new* $m^* \notin \mathcal{M}$*, which has never been queried to* $\mathcal{O}_{\mathsf{SignRev}}$ *oracle previously.*

We define the advantage of the forger $\mathcal{F}$ *in the experiment* $\mathtt{Exp}^{\lambda,\ell}_{\mathsf{SR}}$ *as the probability that the* $\mathcal{F}$ *produces a valid signature over the message not previously queried to signing oracles, i.e.:*

$$\mathsf{Adv}(\mathcal{F}, \mathtt{Exp}^{\lambda,\ell}_{\mathsf{SR}}) = \Pr\left[\begin{array}{l} \mathsf{ParGen}(\lambda) \rightarrow \mathsf{par}, \\ \mathsf{KeyGen}(\mathsf{par}) \rightarrow (\mathsf{sk}, \mathsf{pk}) \\ \mathsf{InitRF}(\mathsf{sk}) \rightarrow (\mathbf{f1}, \mathbf{f2}) \\ (m^*, \sigma^*) \leftarrow \mathcal{F}^{\mathcal{O}_{\mathsf{SignRev}}, \mathcal{O}_{\mathcal{H}}}(\mathsf{pk}) \\ \mathsf{Ver}(m^*, \sigma^*, \mathsf{pk}) \rightarrow 1 \\ m^* \notin \mathcal{M} \end{array}\right].$$

We say that the signature scheme is secure if the advantage of the adversary $\mathcal{F}$ negligible in parameters λ, ℓ, i.e.:

$$\mathsf{Adv}(\mathcal{F}, \mathtt{Exp}_{\mathsf{SR}}^{\lambda,\ell}) \leq \epsilon(\lambda, \ell).$$

Theorem 15. *The additive BLS signature split realised via the* HSM *architecture (presented in Sec, 4.3.5) is secure in our proposed* SR *model, as of Definition 39.*

Proof. ***This proof was part of the accepted paper, however due to space constraints it was removed before publishing.*** The proof forms a sequence of games, where an original experiment is modified in an indistinguishable way for the forger.

Game-0: This game is the original experiment $\mathtt{Exp}_{\mathsf{SR}}^{\lambda,\ell}(\mathsf{Sign})$. Let p_0 denote the probability that $\mathcal{F}$ returns a verifiable forgery (m^*, σ^*) in **Game-**0, i.e: $p_0 = \mathsf{Adv}(\mathcal{F}, \mathtt{Exp}_{\mathsf{SR}}^{\lambda,\ell})$.

Game-1: this game is a modified experiment in which the initial split and the refresh procedure in each i-th sign oracle query (for $i = 0, \ldots, \ell$) is done with a true random value r_i, instead of values from PRNG, namely: $r_i \xleftarrow{\$} \mathbb{Z}_q$, $\mathsf{sk}_1^{(i)} = \mathsf{sk}_1^{(i-1)} + r_i$, and $\mathsf{sk}_2^{(i)} = \mathsf{sk}_2^{(i-1)} - r_i$. Thus in each sign query i we have $\mathsf{sk}_1^{(i)} = \sum_{j=0}^{i} r_j$, and $\mathsf{sk}_2^{(i)} = \mathsf{sk} - \sum_{j=0}^{i} r_j$. Let p_1 denote the probability that $\mathcal{F}$ returns a verifiable forgery (m^*, σ^*) in Game-1, i.e: $p_1 = \mathsf{Adv}(\mathcal{F}, \textbf{Game-}1)$.

Game-2: this game is a modified experiment in which we redefine the refresh procedure in each i-th sign oracle query (for $i = 1, \ldots, \ell$) in the following way: $r_i \leftarrow_{\$} \mathbb{Z}_q$, $\mathsf{sk}_1^{(i)} = r_i$, and $\mathsf{sk}_2^{(i)} = \mathsf{sk} - r_i$. Let p_2 denote the probability that $\mathcal{F}$ returns a verifiable forgery (m^*, σ^*) in Game-2, i.e: $p_2 = \mathsf{Adv}(\mathcal{F}, \textbf{Game-}2)$.

Game-3: this game is a modified experiment in which we redefine the refresh procedure in each i-th sign oracle query $\mathcal{O}_{\mathsf{SignRev}}(m, 2)$ in the following way: $r_i \leftarrow_{\$} \mathbb{Z}_q$, $\mathsf{sk}_1^{(i)} = \mathsf{sk} - r_i$, and $\mathsf{sk}_2^{(i)} = r_i$. Let p_3 denote the probability that $\mathcal{F}$ returns a verifiable forgery (m^*, σ^*) in Game-3, i.e: $p_3 = \mathsf{Adv}(\mathcal{F}, \textbf{Game-}3)$.

Game-4: this game is a modified experiment in which we redefine the order of the procedures, namely we first wait for the i-th oracle query $\mathcal{O}_{\mathsf{SignRev}}(m, 1)$ or $\mathcal{O}_{\mathsf{SignRev}}(m, 2)$ and than we set the values of $\mathsf{sk}_1^{(i)}$ and $\mathsf{sk}_2^{(i)}$: accordingly as in **Game-**2, or as in **Game-**3. Let p_3 denote the probability that $\mathcal{F}$ returns a verifiable forgery (m^*, σ^*) in Game-4, i.e: $p_4 = \mathsf{Adv}(\mathcal{F}, \textbf{Game-}4)$.

We now state a set of claims and prove them in sequence.

Claim 1. $|p_0 - p_1| \leq \epsilon_{\mathsf{PRNG}}$.

Proof of Claim 1. Obviously any noticable difference between p_0 and p_1, would be used to treat the algorithm $\mathcal{F}$ as an effective distinguisher, breaking the security assumption of the PRNG.

Claim 2. $|p_1 - p_2| = 0$.

Proof of Claim 2. In this game in each sign query i we substitute $\sum_{j=0}^{i} r_j$ with $r_i \leftarrow_{\$} \mathbb{Z}_q$. Thus we replace $\mathsf{sk}_1^{(i)} = \sum_{j=0}^{i} r_j$ with $\mathsf{sk}_1^{(i)} = r_i$, and $\mathsf{sk}_2^{(i)} = \mathsf{sk} - \sum_{j=0}^{i} r_j$ with $\mathsf{sk}_2^{(i)} = \mathsf{sk} - r_i$. Notice that $\sum_{j=0}^{i} r_j$ computed modulo q for r_j, taken uniformly at randomly from $\{0, \ldots, q-1\}$, is itself a random variable of uniform distribution from $\{0, \ldots, q-1\}$. Therefore $(\mathsf{sk}_1^{(i)}, \mathsf{sk}_2^{(i)})$ for each sign query i in Game-2 have the same distribution as in previous Game-1.

Claim 3. $|p_2 - p_3| = 0$.

Proof of Claim 3. Notice that for any r, r', taken uniformly at randomly from $\{0, \ldots, q-1\}$, pairs: $(r, \mathsf{sk} - r)$, and $(\mathsf{sk} - r', r')$ have the same distributions, when computed modulo q. Therefore $(\mathsf{sk}_1^{(i)}, \mathsf{sk}_2^{(i)})$ for each sign query i in Game-3 have the same distribution as in previous Game-2.

Claim 4. $|p_3 - p_4| = 0$.

Proof of Claim 3. Notice that values of $\mathsf{sk}_1^{(i)}$ and $\mathsf{sk}_2^{(i)}$ are not revealed unless the appropriate oracle is called. The adversary cannot decide which alternative it deals with: $(\mathsf{sk}_1^{(i)} = r_i,\ \mathsf{sk}_2^{(i)} = \mathsf{sk} - r_i)$, or $(\mathsf{sk}_1^{(i)} = \mathsf{sk} - r_i,\ \mathsf{sk}_2^{(i)} = r_i)$ due to Claim 3. It also cannot tell if the alternative was chosen before the query was issued or after.

Claim 5. *The attacker $\mathcal{F}$ wins in Game-4 with probability no greater than the probability of a forgery by a forger $\mathcal{F}_{BLS}^{\mathcal{O}_{\mathsf{Sign}}, \mathcal{O}_{\mathcal{H}}}(\mathsf{pk})$ in the regular unforgeability experiment against regular BLS signature, i.e.*

$$p_3 = \mathsf{Adv}(\mathcal{F}, \textbf{Game-}4) \leq \epsilon_{BLS}(\lambda, \ell). \tag{4.12}$$

Proof of Claim 5. We build a wrapper around the unforgeability experiment for the regular BLS using the regular signing oracle $\mathcal{O}_{\mathsf{Sign}}$ of $\mathcal{F}_{BLS}$, and a regular hash random oracle $\mathcal{O}_{\mathcal{H}}$. When querying to $\mathcal{O}_{\mathsf{SignRev}}(m, 1)$ from $\mathcal{F}$ we first obtain the hash $h \leftarrow \mathcal{O}_{\mathcal{H}}(m)$, and the final signature $\sigma \leftarrow \mathcal{O}_{\mathsf{Sign}}(m)$. Subsequently we will generate a random r and output it as a partial secret stored in the queried module. Note that partial signatures for that module is h^r, and the partial signature for the other (unqueried module) is $\sigma/(h^r)$. Also note that the adversary cannot object to our answers as from its point of view (previous Claims 1-4) the reveal secret stored in the queried module after a refresh is a random variable from uniform distribution. Namely:

`Serving` $\mathcal{O}_{\mathcal{H}}$ `Oracle` : Any call to $\mathcal{O}_{\mathcal{H}}$ from $\mathcal{F}^{\mathcal{O}_{\mathsf{SignRev}},\mathcal{O}_{\mathcal{H}}}(\mathsf{pk})$ in Game-3 is forwarded to oracle $\mathcal{O}_{\mathcal{H}}$ of $\mathcal{F}_{BLS}$ and its corresponding answers are forwarded back to $\mathcal{F}^{\mathcal{O}_{\mathsf{SignRev}},\mathcal{O}_{\mathcal{H}}}(\mathsf{pk})$.

`Serving` $\mathcal{O}_{\mathsf{SignRev}}(\mathtt{m},\mathtt{1})$ `Oracle` : On i-th query first we call $h = \mathcal{O}_{\mathcal{H}}(m)$, then issue $\mathcal{O}_{\mathsf{Sign}}(m)$ of $\mathcal{F}_{BLS}$. We take its answer σ, generate $r_i \leftarrow_{\$} \mathbb{Z}_q$ and return $(\sigma_1 = h^{r_i},\, r_i, \sigma)$.

`Serving` $\mathcal{O}_{\mathsf{SignRev}}(\mathtt{m},\mathtt{1})$ `Oracle` : On i-th query first we call $h = \mathcal{O}_{\mathcal{H}}(m)$, then issue $\mathcal{O}_{\mathsf{Sign}}(m)$ of $\mathcal{F}_{BLS}$. We take its answer σ, generate $r_i \leftarrow_{\$} \mathbb{Z}_q$ and return $(\sigma_2 = h^{r_i},\, r_i, \sigma)$.

`Processing Forgery` : As $\mathcal{F}^{\mathcal{O}_{\mathsf{SignRev}},\mathcal{O}_{\mathcal{H}}}(\mathsf{pk})$ returns with (m^*, σ^*) we output it as the forgery to $\mathcal{F}_{BLS}$.

Corollary 2. *As an immediate conclusion from claims: Claim 1 to Claim 4 we have:*

$$p_0 = \mathsf{Adv}(\mathcal{F}, \mathtt{Exp}^{\lambda,\ell}_{\mathsf{SR}}) \leq \epsilon_{\mathsf{PRNG}} + \epsilon_{BLS}(\lambda, \ell). \tag{4.13}$$

□

BLS with key split and refresh (and regular BLS multi-signatures) could be analyzed in an even stronger unforegeability model, in which the adversary can query separate oracles, which would reveal separately: final signature, partial signatures indicated by HSM index, and partial secrets indicated by HSM index. However in this case the proof methodology would require the usage of the programmable ROM, and the explicit reduction to $\mathsf{co}-\mathsf{CDH}$ problem, as in the case of regular BLS. We leave that model analysis for future research.

4.3.7 Performance Analysis

In our proof of concept implementation we use the MCL-library [120] for pairings, WebAssembly (WASM) and JavaScript for the wrapping software. Our analysis consider fundamental operations performed such as multiplications, pairings etc. Timings are measured in milliseconds (ms) and is average over 1000 runs. We ran performance tests on the following devices: MacBook Pro 3.1 GHz Dual-Core Intel Core i7 (L), Apple iPad 9.7 A1893 (P), Apple iPhone XS (IH) and a Samsung Smart TV: The Frame (ST). Tab. 4.9 shows the mean running times and measured power consumption in mAh. We were not able to measure the power consumption on the Smart TV since the measurement software was not compatible with the device. We note that the timings for key splitting and refresh are very fast. This way we achieved a significant security improvement with low performance overhead.

Operation(s)	L	P	IH	ST	Δ L	Δ P	Δ IH
BLS sign	2.59	3.15	2.11	20.56	2.80	1.66	1.05
BLS verify	18.6	21.5	14.7	140.21	20.14	11.41	7.32
Key-split +	0.01	0.00	0.00	0.123	∗	∗	∗
Key-RF +	0.01	0.01	0.01	0.147	0.01	∗	∗
$\mathbb{G}_1$: Add	0.01	0.02	0.01	0.114	∗	∗	∗
$\mathbb{G}_1$: Mul	1.20	2.10	1.10	10.15	1.29	1.11	0.55
Hash to $\mathbb{G}_1$	1.30	2.20	1.20	11.40	1.40	1.16	0.61
Hash to $\mathbb{G}_2$	2.90	4.40	2.60	24.57	3.14	2.32	1.30
Hash	0.00	0.10	0.00	0.05	∗	∗	∗
Pairing	7.60	14.80	6.80	64.61	8.22	7.80	3.38

Table 4.9: Performance analysis in milliseconds (ms) and power consumption in milliampere-hours (mAh), denoted Δ. For running times less than 0.001 ms, we mark that with ∗.

4.3.8 Conclusion

In this paper we propose BLS signatures with an additive key split augmented with a refresh technique. We extrapolated this technique to distinct signing devices accessing the same external source of randomness ξ. This is suitable for CRI to achieve a CBE where a high sensitive cargo signing device is coupled with the train carriage signing device over the same source of randomness, to produce a single multi-signature. Due to this security feature, if a cargo removal attack is done between a key refresh, the system desychronize and positive verification is impossible. This protects against the cargo removal. From our performance analysis we conclude that the modifications will not increase computational complexity significantly.

Finally, we conclude that the proposed solution addresses challenge Sec1 since it provides authenticity and privacy of the messages sent between cargo and carriage. That privacy property is thus secure against impersonation attacks, addressing challenge Sec3, and the whole scheme handles partial key leakage of the HSMs, hence addressing challenge Sec5.

4.4 Scenario $\mathsf{P_{IV}}$: Source Hiding of Anonymous Signers

The following section is based on the published paper $\mathsf{P_{IV}}$[4].

ENV: We consider a VANET setup with connected nodes (vehicles), identified as $\mathsf{ID}_1, ..., \mathsf{ID}_n$ at the application layer. Each node ID_i has an additional (network layer) identity L_i, and we consider the OBU in each node as the primary computing device. The setup also consists of a proxy server ID_S which every node is able to communicate with via standard wireless VANET channels, e.g., IEEE 802.11p. The proxy server can be considered a RSU or any type of stationary or moving node in a VANET.

We recall the problem statement for this scenario: to ensure full anonymity of a node ID_i after producing a message m_i, and a corresponding ring signature σ, with a group of participating signers. This must be secured both on the application- and network layers as in the typical TCP/IP stack, i.e. by using a ring signature with the other participants to protect ID_i, but also an onion-like encryption using an interactive sub-protocol to protect the anonymity of the location or IP address L_i of ID_i. A proxy server will handle the unlinkability protocol, i.e. location anonymization, together with the signers. Moreover, to ensure even stronger security of a VANET system, the scheme should not rely on the proxy server to be a trusted third party (TTP) which facilitate location anonymization and location privacy, and any ring signature σ can successfully be verified by any external verfier holding the group's public keys.

4.4.1 Security Requirements

From a cryptographic perspective we formulate the following security requirements for the messages and signatures in a VANET:

- The requirement for *application layer anonymity* can be achieved when a message, originating from various signers and intended for a server, is signed anonymously within a group of potential signers. This can be accomplished through the use of ring signatures. Upon subsequent verification, the fact that the message originated from this group can be verified, while the identity of the specific sender remains hidden.

[4]Published at Information Security Practice and Experience conference and is under copyright © 2022 Springer.

- The requirement for *network layer anonymity* and *unlinkability* dictates that messages originating from a certain source and routed through an infrastructure network to a verification server should not be traceable back to the signer. This holds true even in scenarios where an adversary has access to the underlying communication links and nodes, underscoring the need for robust protections against potential eavesdropping.

This scenario addresses challenges Sec1 and Sec3, namely the authenticity and privacy of C-ITS generated data (the messages), and mitigation of impersonation attacks. Both of these in a multi-party setting with ad-hoc network environments.

4.4.2 Related Work

The concept of ring signatures, introduced by Rivest et al. [138], facilitates anonymous signing within a group of potential signers. While Kempf et al. incorporated ring signatures into the IPv6 secure neighbor discovery protocol for secure address proxying, they did not explore encryption for enhanced privacy and integrity. Although ring signatures have been examined within C-ITS and VANET contexts, prior work [107, 108, 29, 122] did not incorporate source hiding features. The concept of mixed networks, or mixnets [39], which ensure network communication anonymity and source hiding by cryptographically transforming and securely shuffling input, has been suggested. While various approaches to VANET location privacy exist, as summarized by Khan et al. [86], our work is the first to our knowledge to propose a location privacy solution for source hiding, which employs ring signatures and onion-like encryption without relying on a trusted third party.

4.4.3 Threat Model

We consider two main attacks in the given scenario: *path-tracing attacks* and *source address anonymity without a TTP*. These attacks works as follows:

Path-tracing attacks: Let $\{\mathsf{ID}_1, \ldots, \mathsf{ID}_n\}$ be the set of participating vehicles in a VANET, where each vehicle have a public/private signing key-pair $(\mathsf{sks}_i, \mathsf{pks}_i)$, and a logical location L_i associated to the node, e.g an IP address. While ring signatures offer anonymity of the messages on the application layer, these are still compiled into network packets and hence, can potentially be linked with the associated IP source addresses used. An adversary with control over the communication links

can attempt a path-tracing attack by examining IP source and destination addresses in the network layer, as well as the routing paths of such data packets. Thus, in addition to preserving cryptographic anonymity through a ring signature scheme at the application layer, it is crucial to protect L_i at the network layer as well.

Source address anonymity without a TTP: This attack is particularly important in the connected vehicle and infrastructure context, where both moving and stationary nodes are capable of sending high volumes of information with protected (and authenticated) data. It is critical to secure the source identity L_i of a moving vehicle in such ad-hoc groups without the reliance on a TTP. To elaborate, we aim not to depend on a TTP for an anonymization scheme that safeguards network layer source addresses through which all data packets with ring signatures traverse. This approach stands in contrast to solutions like ToR-routing [66] and mixnets [39]. The motivation behind this is to enhance the security of the VANET architecture by minimizing the dependence on one or more TTPs that are involved in cryptographic computations within the protocol.

4.4.4 Preliminaries

We consider two main node types in this scenario: a set of *participants* $\mathsf{ID}_1, ..., \mathsf{ID}_n$ and one *proxy server* ID_S. There is also an *verifier*, i.e. anyone with the participants' public keys, that can verify the produced ring signature σ_i over message m_i, originated from user ID_i. These nodes are confined in an ENV, realized as a VANET where the used communication channel is wireless short- or long range technology between the nodes, implying standard technology layers such as application-, network- and physical layers. We depict the simplified scenario setting in Fig. 4.7.

We set a 4-tuple $\mathsf{RS} = (\mathsf{ParGenS}, \mathsf{KeyGenS}, \mathsf{RingSign}, \mathsf{RingVerify})$ to be a ring signature scheme, and the tuple $\mathsf{ES} = (\mathsf{ParGenE}, \mathsf{KeyGenE}, E, D)$, to be an asymmetric encryption system. We recall the formal definition of the subroutines in ES:

$\mathsf{epar} \leftarrow \mathsf{ParGenE}(\lambda)$ takes the security parameter λ and outputs parameters $\mathsf{epar} = (\mathcal{K}, \mathcal{M}, \mathcal{C})$, where $\mathcal{K}$ is a key-pair space, $\mathcal{M}$ is a message space, $\mathcal{C}$ is a ciphertext space.

$(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(\mathsf{par})$ is a key-pair generation algorithm, which inputs par and outputs a key-pair $(\mathsf{sk}, \mathsf{pk}) \in \mathcal{K}$: a secret key sk and its corresponding public key pk .

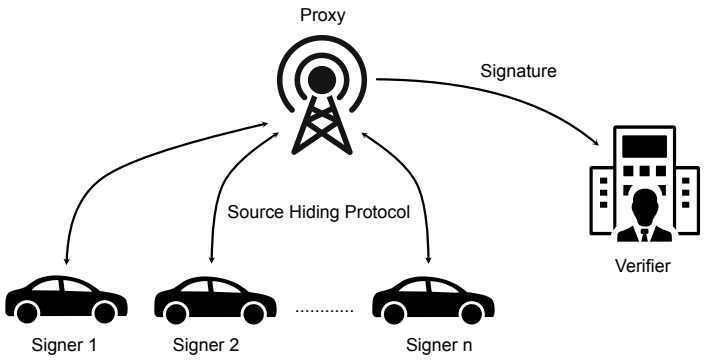

Figure 4.7: Simplified VANET setup with ring signature collaboration and a verifier outside of the VANET. The complete setting use a source hiding protocol with onion-like encryptions.

$c \leftarrow E(m, \mathsf{pk})$ is an encryption algorithm that inputs a message $m \in \mathcal{M}$, a public key pk, and outputs a ciphertext $c \in \mathcal{C}$.

$m \leftarrow D(c, \mathsf{sk})$ is a decryption algorithm that inputs a ciphertext $c \in \mathcal{C}$, a secret key sk, and outputs the corresponding plaintext m.

We consider the *Indistinguishability under Chosen Ciphertext Attack* (IND-CCA2) model for asymmetric-key algorithm cryptosystems from [84] that works as follows: an adversary with access to a decryption oracle should not be able to derive any useful information about the original plaintext from its corresponding ciphertext. More concretely, when an adversary generates two plaintexts of equal length and receives only one corresponding ciphertext, they should not be able to deduce which plaintext was encrypted. The IND-CCA2 model is defined by the following experiment:

Definition 31 (Indistingushability under CCA2)**.** *Given an asymmetric-key cryptosystem* $\mathsf{ES} = (\mathsf{ParGenE}, \mathsf{KeyGenE}, E, D)$ *we define the chosen ciphertext indistinguishability experiment* IND-CCA2*:*

`Init` : $\mathsf{epar} \leftarrow \mathsf{ParGenE}(\lambda)$, $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGenE}(\mathsf{par})$.

`Adversary` : *Let the adversary* $\mathcal{A}$*, be a malicious algorithm initialized with parameters* epar*, and the public key* pk*.*

`Decryption Oracle` : *Let a decryption oracle* $\mathcal{O}_D$*, be an algorithm initialized with parameters* par*, s.t. when queried with a ciphertext* $c \leftarrow E(m, \mathsf{pk})$*, it outputs the corresponding plaintext* m*, i.e.* $\mathcal{O}_D(c, \mathsf{pk}) \rightarrow m$*. The*

second argument pk *is just an indicator, i.e. which public key the ciphertext was computed from. Thus the oracle* $\mathcal{O}_D(c, \mathsf{pk})$ *is an equivalent to the entity holding the appropriate corresponding secret key* $\mathcal{O}_D(c, \mathsf{pk}) = D(c, \mathsf{sk})$.

`Guess Game` : *This game is described as in the following protocol:*

1. *The adversary can encrypt a number* q_E *of messages* $m \in \mathcal{M}$ *of its choice via* $E(m, \mathsf{pk})$. *The adversary can query a number* q_D *of ciphertext* $c \in \mathcal{C}$ *of its choice via* $\mathcal{O}_D(c, \mathsf{pk})$.
2. *The adversary generates two messages of its choice:* $(m_0, m_1) \leftarrow \mathcal{A}(\mathsf{par})$ *and sends them to a challenger.*
3. *The challanger generates a random bit* $b \leftarrow_{\$} \{0, 1\}$, *encrypts the message* m_b *to the ciphertext* $c_b = E(m_b, \mathsf{pk})$, *and sends* c_b *to* $\mathcal{A}$.
4. *The adversary can encrypt a number* ℓ_E *of messages* $m \in \mathcal{M}$ *of its choice via* $E(m, \mathsf{pk})$. *The adversary can query a number* ℓ_D *of ciphertext* $c \in \mathcal{C}$ *of its choice via* $\mathcal{O}_D(c, \mathsf{pk})$, *provided that* $c \neq c_b$.
5. *Let* $\mathcal{M}_E, \mathcal{C}_D$ *denote the messages encrypted, and ciphertexts queried to* $\mathcal{O}_D$, *in the steps 1 and 4 respectively. The adversary outputs its own bit*
   $\hat{b} \leftarrow \mathcal{A}(\mathsf{epar}, m_0, m_1, \mathsf{pk}, \mathcal{M}_E, \mathcal{C}_D)$.

We define the advantage of the adversary $\mathcal{A}$ *in the experiment as the probability that* $\mathcal{A}$ *outputs the correct bit* $\hat{b} = b$ *indicating the encrypted message* m_b, *i.e.:*

$$\mathbf{Adv}(\mathcal{A}, \mathsf{IND{-}CCA2}) = |\Pr\left[\hat{b} = b\right] - 1/2|. \tag{4.14}$$

Let ℓ *denotes the upper limit for the sum of all numbers of queries:* $q_E + q_D + \ell_E + \ell_D$ *in the* `Guess Game`. *We say that the encryption scheme is* $\mathsf{IND\text{-}CCA2}$ *secure if the advantage of the adversary* $\mathcal{A}$ *is negligible in parameters* λ, ℓ *i.e.:*

$$\mathbf{Adv}(\mathcal{A}, \mathsf{IND{-}CCA2})) \leq \epsilon(\lambda, \ell). \tag{4.15}$$

Next, we need to formally define RS along with the necessary security properties we aim to prove, i.e. *ring unforgeability* and *ring anonymity*. Again, we use the notion of security experiments.

Definition 32. *A 4-tuple* $\mathsf{RS} = (\mathsf{ParGenS}, \mathsf{KeyGenS}, \mathsf{RingSign}, \mathsf{RingVerify})$ *is a ring signature scheme defined as the following procedures:*

$\mathsf{spar} \leftarrow \mathsf{ParGenS}(\lambda)$ *takes the security parameter* λ *and produces parameters of the scheme* $\mathsf{spar} = (\mathcal{K}, \mathcal{M}, \mathcal{S})$, *where* $\mathcal{K}$ *is a key-pair space,* $\mathcal{M}$ *is a message space,* $\mathcal{S}$ *is a signature space.*

$(\mathsf{sks}, \mathsf{pks}) \leftarrow \mathsf{KeyGenS}(\mathsf{spar})$ *is a key-pair generation algorithm, which inputs* par *and outputs a key-pair* $(\mathsf{sks}, \mathsf{pks}) \in \mathcal{K}$*: a secret key* sks *and its corresponding public key* pks*.*

$\sigma \leftarrow \mathsf{RingSign}(m, \mathsf{sks}_j, \mathsf{PKS})$ *– signing procedure that takes a message* m*, the secret key* sks_j *and the set of public keys* $\mathsf{PKS} = \{\mathsf{pks}_1, \ldots, \mathsf{pks}_k\}$*,* $\mathsf{pks}_j \in \mathsf{PKS}$*. It returns a ring signature* σ*.*

$1/0 \leftarrow \mathsf{RingVerify}(\sigma, m, \mathsf{PKS})$ *a signature verification algorithm takes a signature* σ*, a message* m*, and the set of public keys* PKS*. It returns a bit (0 or 1) indicating whether the signature* σ *is valid, i.e., whether someone having a public key from the set* PKS *has signed* m*.*

We require that the signature scheme is correct, i.e. a signature created by signer $j \in \{1, \ldots, n\}$ from a set of n potential signers over any message $m \in \mathcal{M}$, is always positively verifiable:

$$\Pr\left[\begin{array}{l} \mathsf{ParGenS}(\lambda) \rightarrow \mathsf{spar}, \\ \mathsf{KeyGenS}(\mathsf{spar}) \rightarrow \{(\mathsf{sks}_i, \mathsf{pks}_i)\}_1^n \\ \forall_{(j,m)} : j \in \{1, \ldots, n\}, m \in \mathcal{M} \\ \left[\begin{array}{l} \mathsf{RingSign}(m, \mathsf{sks}_j, \{\mathsf{pks}_i\}_1^n) \rightarrow \sigma \\ \mathsf{RingVerify}(\sigma, m, \{\mathsf{pks}_i\}_1^n) \rightarrow 1 \end{array}\right] \end{array}\right] = 1. \tag{4.16}$$

Moreover, we assume that above schemes are *unforgeable* in the *chosen-message* scenario: suppose a forger's goal is to produce a verifiable signature σ for a message m which was not previously signed in the query stage. We then say that the forger succeeds, if it can forge σ for m with a non-negligible probability.

Definition 33 (Ring Unforgeability)). *Let* $\mathsf{RS} = (\mathsf{ParGen}, \mathsf{KeyGen}, \mathsf{RingSign}, \mathsf{RingVerify})$ *be a ring signature scheme. We define a security experiment:*

`Init` : $\mathsf{spar} \leftarrow \mathsf{ParGen}(\lambda)$*,* $\{(\mathsf{sks}_i, \mathsf{pks}_i)\}_1^n \leftarrow \mathsf{KeyGen}(\mathsf{spar})$*.*

`Ring Sign Oracle` : $\mathcal{O}_{\mathsf{RingSign}}(m, j, \mathsf{PKS}) \rightarrow \sigma$ *takes a message* m*, the signer indicator* $j \in \{1, \ldots, n\}$*, and the set of public keys* $\mathsf{PKS} = \{\mathsf{pks}_i\}_1^n$ *and outputs a ring signature* σ*, as if generated with the secret key* sks_j*, and the public keys* PKS*, s.t.* $\mathsf{RingVerify}(\sigma, m, \mathsf{PKS}) = 1$*.*

`Hash Oracle` : *The hash oracle* $\mathcal{O}_{\mathcal{H}}$ *is modeled in ROM.*

`Adversary` : *Let the adversary* $\mathcal{F}^{\mathcal{O}_{\mathsf{RingSign}}, \mathcal{O}_{\mathcal{H}}}(\mathsf{PKS})$*, be a malicious algorithm initialized with the public parameters* par *and public keys* PKS*, having access to the oracles* $\mathcal{O}_{\mathsf{RingSign}}$ *and* $\mathcal{O}_{\mathcal{H}}$*. It issues* ℓ *number of queries to the oracles. Let* $M = \{m_i\}_1^\ell$*, and* $\Omega = \{\sigma_i\}_1^\ell$ *denote the set of the messages, and the corresponding signatures the oracles process.*

`Forgery` : *The adversary generates a tuple:* $(m^*, \sigma^*) \leftarrow \mathcal{F}^{\mathcal{O}_{\mathsf{RingSign}}, \mathcal{O}_{\mathcal{H}}}(\mathsf{PKS})$ *for a new* $m^* \notin M$*, which was not queried to* $\mathcal{O}_{\mathsf{RingSign}}$ *oracle.*

We say that the signature scheme is secure if for each forgery type, the probability that the adversary produces a valid signature is negligible in parameters λ, ℓ*:*

$$\Pr\left[\begin{array}{l} \mathsf{ParGenS}(\lambda) \rightarrow \mathsf{spar}, \\ \mathsf{KeyGenS}(\mathsf{spar}) \rightarrow \{(\mathsf{sks}_i, \mathsf{pks}_i)\}_1^n \\ (m^*, \sigma^*) \leftarrow \mathcal{F}^{\mathcal{O}_{\mathsf{RingSign}}, \mathcal{O}_{\mathcal{H}}}(\{\mathsf{pks}_i\}_1^n) \\ \mathsf{RingVerify}(m^*, \sigma^*, \{\mathsf{pks}_i\}_1^n) \rightarrow 1 \\ m^* \notin M \end{array}\right] \leq \epsilon(\lambda, \ell). \tag{4.17}$$

Definition 34 (Ring Anonymity $\mathsf{RS}-\mathsf{A}$)**.** *Let* $\mathcal{D}$ *denote a distinguisher algorithm given public parameters* par *and a set of all keys* $\{(\mathsf{sks}_i, \mathsf{pks}_i)\}_1^n$*. It chooses a message* $m \in \mathcal{M}$*. A challenger chooses an index* $j \leftarrow_{\$} \{1, \ldots, n\}$ *uniformly at random and creates the signature* $\sigma \leftarrow \mathsf{RingSign}(m, \mathsf{sks}_j, \{\mathsf{pks}_i\}_1^n)$*. We say that the scheme is anonymous if the chance of* $\mathcal{D}$ *for guessing* j *is negligible different from* $1/n$*. We define the* $\mathsf{RS}-\mathsf{A}$ *experiment:*

`Init` : $\mathsf{spar} \leftarrow \mathsf{ParGenS}(\lambda)$, $\{(\mathsf{sks}_i, \mathsf{pks}_i)\}_1^n \leftarrow \mathsf{KeyGenS}(\mathsf{spar})$.

`Adversary` : *Let the adversary* $\mathcal{D}$*, be a malicious algorithm initialized with parameters* spar*, and the keys* $\{(\mathsf{sks}_i, \mathsf{pks}_i)\}_1^n$.

`Anonymity Game` : *This game is the following protocol:*

1. *The distinguisher generates a message:* $m \leftarrow \mathcal{D}(\mathsf{spar}, \{(\mathsf{sks}_i, \mathsf{pks}_i)\}_1^n)$ *and sends* m *to a challenger.*
2. *The challenger having access to* $\{(\mathsf{sks}_i, \mathsf{pks}_i)\}_1^n$ *generates a random index* $j \leftarrow_{\$} \{1, .., n\}$*, and signature* $\sigma \leftarrow \mathsf{RingSign}(m, \mathsf{sks}_j, \{\mathsf{pks}_i\}_1^n)$*, and sends* σ *to* $\mathcal{D}$*.*
3. *The distingusher outputs its own index* $\hat{j} \leftarrow \mathcal{D}(\sigma, m, \{(\mathsf{sks}_i, \mathsf{pks}_i)\}_1^n)$.

We define the advantage of the distinguisher $\mathcal{D}$ *in the experiment as the probability that* $\mathcal{D}$ *outputs the correct index* $\hat{j} = j$ *indicating the signer:*

$$\mathbf{Adv}(\mathcal{D}, \mathsf{RS}-\mathsf{A}) = |\Pr[\hat{j} = j] - 1/n|. \tag{4.18}$$

We say that the $\mathsf{RS}-\mathsf{A}$ *scheme is anonymous if the advantage of the distinguisher* $\mathcal{D}$ *is negligible in the parameter* λ *i.e.:*

$$\mathbf{Adv}(\mathcal{D}, \mathsf{RS}-\mathsf{A})) \leq \epsilon(\lambda). \tag{4.19}$$

4.4.5 Proposed Schemes

For node ID_i, let $(\mathsf{sks}_i, \mathsf{pks}_i)$ be a key-pair for a ring signature scheme RS, and $(\mathsf{ske}_i, \mathsf{pke}_i)$ a key-pair for encryption using encryption scheme ES. Let $\mathsf{PKS} = \{\mathsf{pks}_i\}_1^n$, $\mathsf{PKE} = \{\mathsf{pke}_i\}_1^n$ denote the sets of public keys for encryption and signing. Assuming an efficient ordering of any group of nodes by their encryption public keys, we form a sequence of these keys, $\langle \mathsf{PKE} \rangle$. Nodes in this sequence are aware of their position and those of others, with indexes assumed to be $\langle 1, \ldots, n \rangle$ without loss of generality. We also assume that each node have its own private random permutation function P_i. We propose two different schemes, π_{SHP} and π_{SHP2}. For each scheme, we assume a set of registered nodes for the ES and RS schemes i.e. having the necessary key-pairs, denoted by $\mathcal{U}(\{(\mathsf{sks}_i, \mathsf{ske}_i)\}_1^n)$ and a server having access to the public keys $\mathcal{S}(\{(\mathsf{pks}_i, \mathsf{pke}_i)\}_1^n)$. Assume that user ID_i creates a message m_i and signs it to $\sigma_i = \mathsf{RingSign}(m_i, \mathsf{sks}_i, \mathsf{PKS})$. If σ_i is send directly to the server an adversary that controls the underlying communication channel(s) can disclose the identity L_i even though the ring signature provides the anonymity property on the application layer.

The π_{SHP} Scheme

We propose an *onion* like source hiding scheme π_{SHP} that uses any suitable ES for route obfuscation of (m_i, σ_i) messages, send by ID_i with network layer identity L_i. The interactive scheme is depicted in Fig. 4.8 and works as follows:

1. The signer signs its message m_i anonymously into σ_i, using the ring signature scheme.
2. The signer sets $c_i^{(0)} = (m_i, \sigma_i)$.
3. The signer creates an onion ciphertext

$$c_i^{(n)} = E(\ldots(E(\ldots(E(c_i^{(0)}, \mathsf{pke}_1), \ldots), \mathsf{pke}_i)\ldots), \mathsf{pke}_n)$$

with n layers, first encrypting the $c_i^{(0)} = (m_i, \sigma_i)$ with the first public key pke_1 from $\langle \mathsf{PKE} \rangle$, and then encrypting the result with the next public key from $\langle \mathsf{PKE} \rangle$, including its own public key pke_i in the correct iteration. Since we assume there is a predetermined ordering of the public keys, this onion encryption process iterates according to the ordering. The result of each encryption is the input for the next encryption; hence the only way to retrieve $c_i^{(0)} = (m_i, \sigma_i)$ from the onion $c_i^{(n)}$ is by the group computation of collaboratively decrypt each group

member's layer with its own secret key. This process starts from outer layer n of the node decrypting it with ske_n, and continuous to the most inner ciphertext decrypted by the node possesing ske_1.

4. The signer send onion $c_i^{(n)}$ is to the server.

5. The server receives the set of all onion ciphertexts $\{c_1^{(n)}, \ldots, c_n^{(n)}\}$. These must of course be decrypted in the correct order by the appropriate key holders. Therefore, the server loops over the list of participants and sends $\{c_1^{(n)}, \ldots, c_n^{(n)}\}$ to each user according to the reverse order of $\langle \mathsf{PKE} \rangle$, i.e. from n down to 1.

6. For the i-th iteration of the loop, the sequence $\{c_1^{(i)}, \ldots, c_n^{(i)}\}$ are sent to user ID_i, which decrypts the outer i-th layer from all onions $c_i^{(i)} \in \{c_1^{(i)}, \ldots, c_n^{(i)}\}$ into $\{c_1^{(i-1)}, \ldots, c_n^{(i-1)}\}$, using the secret key ske_i. This sequence is shuffled into a random sequence using the node's private random permutation P_i. The randomized sequence is send back to the server. We note that each node ID_i is shuffling, as all onions are encrypted in i-th layer using pke_i. Therefore, to be able to decrypt the i-th layer, all onions must be sent to the i-th user with secret key ske_i.

7. The server checks that $\{c_1^{(0)}, \ldots, c_n^{(0)}\}$ equals to $\{(m_1, \sigma_1), \ldots, (m_n, \sigma_n)\}$.

8. The server outputs $\{(m_i, \sigma_i)\}_1^n$ and the protocol ends.

The π_{SHP2} Scheme

For our second scheme, we introduce a stronger adversary that is curious, and tries to manipulate onion ciphertexts such that the received messages and signatures to the server is changed. We therefore consider a malicious participant of the signing group $\mathcal{U}(\{\mathsf{sks}_i, \mathsf{ske}_i\}_1^n)$. Observe that in the π_{SHP} protocol any malicious user ID_i, that decrypts its layers $c_j^{(i)} \in \{c_1^{(i)}, \ldots, c_n^{(i)}\}$ via $D(c_j^{(i)}, \mathsf{ske}_i)$ into $c_j^{(i-1)}$, can easily substitute any of the decrypted results with a fresh ciphertext onion, e.g.: $\hat{c}_j^{(i-1)} = E(\ldots(E(\hat{c}_j^{(0)}, \mathsf{pke}_1), \ldots), \mathsf{pke}_{(i-1)})$. It just signs its own fresh message $\hat{m}$ anonymously with the ring signature into $\hat{\sigma}$ using public keys from $\{\mathsf{pke}_1, \ldots, \mathsf{pke}_{(i-1)}\}$, and sets $\hat{c}^{(0)} = (\hat{m}, \hat{\sigma})$. Such an onion will be correctly processed by subsequent users, and finally will be received in the decrypted form $(\hat{m}, \hat{\sigma})$ by the server. To mitigate this, we propose an enhanced version of π_{SHP}, denoted π_{SHP2}, described in Fig. 4.9. The modified scheme is executed in two main rounds as follows:

Setup $\mathsf{ES} = (\mathsf{ParGenE}, \mathsf{KeyGenE}, E, D)$ and $\mathsf{RS} = (\mathsf{ParGenS}, \mathsf{KeyGenS}, \mathsf{RingSign}, \mathsf{RingVerify})$ schemes.

Setup keys: for each user $i \in \{1, \ldots, n\}$ do $\{(\mathsf{sks}_i, \mathsf{pks}_i) \leftarrow \mathsf{KeyGenS}()$ and $(\mathsf{ske}_i, \mathsf{pke}_i) \leftarrow \mathsf{KeyGenE}()\}$.
$\mathsf{PKS} = \{\mathsf{pks}_i\}_1^n$, $\mathsf{PKE} = \{\mathsf{pke}_i\}_1^n$.

Source hiding protocol: $\pi_{\mathsf{SHP}}(\mathcal{U}(\{m_i, \mathsf{sks}_i, \mathsf{ske}_i\}_1^n), \mathcal{S}(\{\mathsf{pks}_i, \mathsf{pke}_i\}_1^n))$

A user the keys: $(\mathsf{ske}_i, \mathsf{pke}_i)$, $(\mathsf{sks}_i, \mathsf{pks}_i)$ **and the message** m_i	**Server**: $\mathcal{S}(\{\mathsf{pks}_i, \mathsf{pke}_i\}_1^n)$
1. Create ring signature $\sigma_i = \mathsf{RingSign}(m_i, \mathsf{sks}_i, \mathsf{PKS})$ 2. **Set:** $c_i^{(0)} = (m_i, \sigma_i)$, 3. Create an onion ciphertext $c_i^{(n)} = E(\ldots(E(\ldots(E(E(c_i^{(0)}, \mathsf{pke}_1), \mathsf{pke}_2), \ldots), \mathsf{pke}_i)\ldots), \mathsf{pke}_n)$. in the following way: **For each** $k \in \{1, \ldots, n\}$ $\quad c_i^{(k)} = E(c_i^{(k-1)}, \mathsf{pke}_i)$. 4. Send $c_i^{(n)}$ to server.	
	5. Wait for reception of $\{c_1^{(n)}, \ldots, c_n^{(n)}\}$ from all users. 6. **For** $i = n$ **down to** 1 $\quad$ send $\{c_1^{(i)}, \ldots, c_n^{(i)}\}$ to user i. $\quad \ldots$ $\quad \ldots$ $\quad \ldots$ $\quad \ldots$ $\quad \ldots$ $\quad \ldots$ $\quad \ldots$ $\quad$ receive $\{c_1^{(i-1)}, \ldots, c_n^{(i-1)}\}$ from user j. 7. Observe that: $\{c_1^{(0)}, \ldots, c_n^{(0)}\}$ equals to $\{(m_1, \sigma_1), \ldots, (m_n, \sigma_n)\}$
6.1. Receive $\{c_1^{(i)}, \ldots, c_n^{(i)}\}$ from the server. 6.2. **For each** $c_j^{(i)} \in \{c_1^{(i)}, \ldots, c_n^{(i)}\}$ $\quad c_j^{(i-1)} = D(c_j^{(i)}, \mathsf{ske}_i)$. 6.3. Shuffle randomly: $\{c_1^{(i-1)}, \ldots, c_n^{(i-1)}\} = P_i(\{c_1^{(i-1)}, \ldots, c_n^{(i-1)}\})$. 6.4. Send $\{c_1^{(i-1)}, \ldots, c_n^{(i-1)}\}$ to the server.	
	8. Output messages and ring signatures $\{(m_i, \sigma_i)\}_1^n$.

Figure 4.8: The proposed Source Hiding Protocol SHP with ring signatures.

1. In the first round (steps 1 and 2) each user i produces its message m_i and a commitment to that message $h_i = \mathcal{H}(m_i, r_i)$ for some random value r_i of appropriate size, where $\mathcal{H}$ is a secure hash function. Subsequently, users run protocol $\pi_{\mathsf{SHP}}(\mathcal{U}(\{(h_i, \mathsf{sks}_i, \mathsf{ske}_i)\}_1^n, \mathcal{S}(\{(\mathsf{pks}_i, \mathsf{pke}_i)\}_1^n)$. The server publishes the commitments (step 3), and each user checks if its commitment is published. A user which commitment is not published breaks (step 4).

2. The protocol $\pi_{\mathsf{SHP}}(\mathcal{U}(\{((m_i, r_i), \mathsf{sks}_i, \mathsf{ske}_i)\}_1^n), \mathcal{S}(\{(\mathsf{pks}_i, \mathsf{pke}_i)\}_1^n))$ is run in the second round (steps 5 to 7), where each user ring-signs its message consisting from two parts (m_i, r_i). Subsequently (step 6) the server publishes the resulting messages and signatures: $\{((m_i, r_i), \sigma_i)\}_1^n$. In the end (step 7) everybody can check if the messages complies with the commitments, i.e. if each commitment from the set $\mathsf{H} = \{h_i\}_1^n$ is opened with one published message m_i together with the corresponding randomness r_i.

4.4.6 Security Analysis

In this section we formally prove the security of π_{SHP} and π_{SHP2} given the previously described security properties. Our first analysis is within the

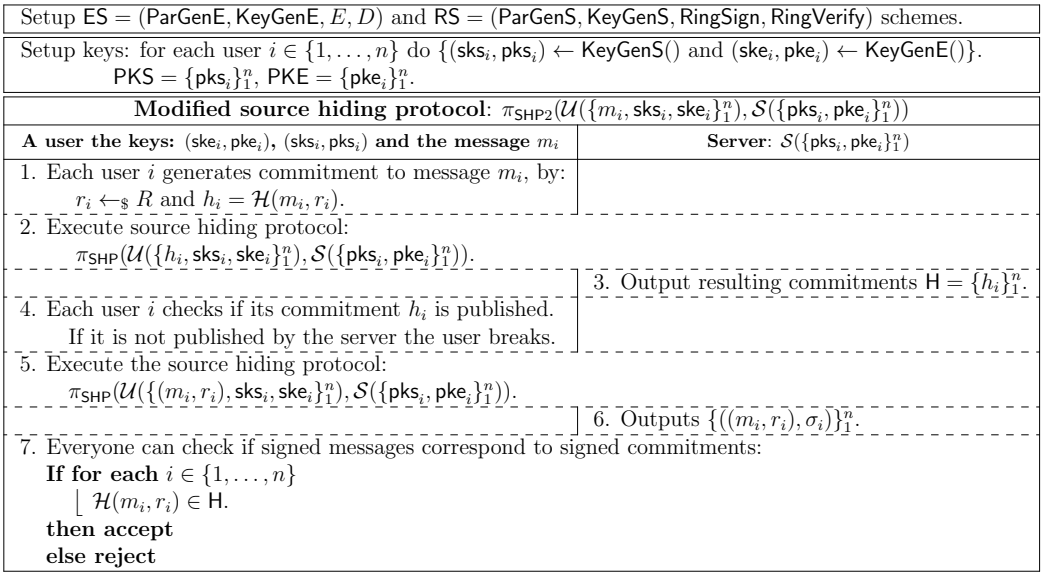

Figure 4.9: Modified version of SHP denoted π_{SHP2} that is immune against adversary type 2.

honest-but-curious adversary model where we assume that the adversary follows the rules of a given protocol π, but its goal is to deduce which node that produce message m_i with signature σ_i. The manifestation of the adversary is a curious server that wants to find where the message and signature from a chosen j-th user is located in the final list $\{(m_1, \sigma_1), \ldots, (m_n, \sigma_n)\}$, obtained in step 7 of π_{SHP}.

Definition 35 (ASHP - Anonymity Model of SHP)**.** *Let* ES, RS *be set with* $\lambda_{\mathsf{ES}}, \lambda_{\mathsf{RS}}$ *parameters. Assume that each user* i *holding* $\mathsf{sks}_i, \mathsf{ske}_i$ *is bounded to a unique location* L_i *and* $\mathcal{D}$ *knows all the routes for messages transported in the underlying network infrastructure. Let* $\mathcal{D}$ *denote a distinguisher algorithm given public parameters of the schemes:* $\mathsf{par} = (\mathsf{spar}, \mathsf{epar})$*, all* RS *keys:* $\{(\mathsf{sks}_i, \mathsf{pks}_i)\}_1^n$*, and all encryption keys:* $\{\mathsf{pke}_i\}_1^n$*. It chooses* n *messages* $\{m_i\}_1^n$*. A challenger randomly assigns those messages to users, i.e. each user* i *gets randomly one message* $m_{\hat{i}}$*. Next, the challenger chooses one index* $\hat{j}$ *indicating the original message* $m_{\hat{j}}$ *assign to user* j *and sends* $\hat{j}$ *to the adversary. Then the protocol* SHP *is executed. In the end the adversary outputs its index* k*. We say that the protocol* SHP *is anonymous and source hiding if the chance of* $\mathcal{D}$ *for outputting* k *equal to* j *that correctly indicates a user given* $m_{\hat{j}}$*, is negligible different from* $1/n$*. We define the* ASHP *experiment:*

`Init` : $\mathsf{par} \leftarrow \mathsf{ParGenS}(\lambda)$, $\{(\mathsf{sks}_i, \mathsf{pks}_i)\}_1^n \leftarrow \mathsf{KeyGenS}(\mathsf{par})$.

`Adversary` : *Let the adversary* $\mathcal{D}$*, be a malicious algorithm initialized with the parameters of* ES, RS *schemes, and the keys:* $\{(\mathsf{sks}_i, \mathsf{pks}_i)\}_1^n$, $\{\mathsf{pke}_i\}_1^n$.

`Source Hiding Game` : *It is the following protocol:*

1. *The distinguisher generates n messages of its choice:* $\{m_{\hat{i}}\}_1^n \leftarrow \mathcal{D}(\mathsf{par}, \{(\mathsf{sks}_i, \mathsf{pks}_i)\}_1^n, \{\mathsf{pke}_i\}_1^n)$ *and sends* $\{m_{\hat{i}}\}_1^n$ *to a challenger.*
2. *The challenger randomly permutes messages* $\{m_i\}_1^n = P(\{m_{\hat{i}}\}_1^n)$ *and assigned them to users, i.e. each user i gets randomly one message $\hat{i}$. Next the challenger generates a random index* $\hat{j} \leftarrow_{\$} \{1, \ldots, n\}$ *indicating a message before permutation, and sends $\hat{j}$ to $\mathcal{D}$.*
3. *The protocol* $\pi_{\mathsf{SHP}}(\mathcal{U}(\{m_i, \mathsf{sks}_i, \mathsf{ske}_i\}_1^n, \mathcal{D}(\{(\mathsf{sks}_i, \mathsf{pks}_i)\}_1^n, \{\mathsf{pke}_i\}_1^n))$ *executes.*
4. *The distingusher outputs its own index* $k \leftarrow \mathcal{D}(\mathsf{par}, \hat{j}, \{m_i\}_1^n, \{(\mathsf{sks}_i, \mathsf{pks}_i)\}_1^n, \{\mathsf{pke}_i\}_1^n)$ *indicating which user k was given the message $\hat{j}$ to sign and process in the protocol π_{SHP}.*

We define the advantage of the distinguisher $\mathcal{D}$ in the experiment as the probability that $\mathcal{D}$ outputs the correct index k equal to j indicating the user given $m_{\hat{j}}$ to process in π_{SHP}.

$$\mathbf{Adv}(\mathcal{D}, \mathsf{SHP}) = |\Pr[j = k] - 1/n|. \tag{4.20}$$

We say that the SHP *scheme is anonymous and source hiding if the advantage of the distinguisher $\mathcal{D}$ is negligible in the parameter $\lambda_{\mathsf{ES}}, \lambda_{\mathsf{RS}}$ i.e.:*

$$\mathbf{Adv}(\mathcal{D}, \mathsf{SHP})) \leq \epsilon(\lambda_{\mathsf{ES}}, \lambda_{\mathsf{RS}}). \tag{4.21}$$

Theorem 16. *The scheme π_{SHP} given in Fig. 4.8 is secure in the* ASHP *model as of Def. 35.*

Proof. To prove the theorem it suffices to show that the answer of the distinguisher $\mathcal{D}$ does not depend on the initial assignment of messages to users, and that its output k is equiprobable across all the initial setups. We use a *sequence-of-games* methodology iterating from game G0 to G5. G0 starts with the user i with message $m_{\hat{i}}$, and the user j with message $m_{\hat{j}}$. We modify the subsequent games, to finalize with G5 with the user i with message $m_{\hat{j}}$, and the user j with message $m_{\hat{i}}$. The adversary should not realize about the game changes.

Let **G0** denote the initial security game, where a message of index $\hat{j}$ was assign to user j, and some message $\hat{i}$ was given to another user i. We briefly

denote that state by:

$$\sigma_j = \mathsf{RingSign}(m_{\hat{\jmath}}, \mathsf{sks}_j, \mathsf{PKS}), c_j^{(0)} = (m_{\hat{\jmath}}, \sigma_j),$$
$$c_j^{(n)} = E(\ldots(E(c_j^{(0)}, \mathsf{pke}_1), \ldots), \mathsf{pke}_n),$$
$$\sigma_i = \mathsf{RingSign}(m_{\hat{\imath}}, \mathsf{sks}_i, \mathsf{PKS}),$$
$$c_i^{(0)} = (m_{\hat{\imath}}, \sigma_i), c_i^{(n)} = E(\ldots(E(c_i^{(0)}, \mathsf{pke}_1), \ldots), \mathsf{pke}_n).$$

Let p_0 denote the probability that $\mathcal{D}$ outputs index k with that setup in that game.

Let **G1** denote a modification of the previous game, where ring signatures are created with switched keys: a message of index $\hat{\jmath}$ is signed with sks_i, and the message $\hat{\imath}$ is signed with sks_j. We denote that state by:

$$\sigma'_j = \mathsf{RingSign}(m_{\hat{\jmath}}, \mathsf{sks}_i, \mathsf{PKS}), c_j^{(0)} = (m_{\hat{\jmath}}, \sigma'_j),$$
$$c_j^{(n)} = E(\ldots(E(c_j^{(0)}, \mathsf{pke}_1), \ldots), \mathsf{pke}_n),$$
$$\sigma'_i = \mathsf{RingSign}(m_{\hat{\imath}}, \mathsf{sks}_j, \mathsf{PKS}),$$
$$c_i^{(0)} = (m_{\hat{\imath}}, \sigma'_i), c_i^{(n)} = E(\ldots(E(c_i^{(0)}, \mathsf{pke}_1), \ldots), \mathsf{pke}_n).$$

Let p_1 denote the probability that $\mathcal{D}$ outputs index k with that setup.

Lemma 1. *$|p_0 - p_1| \leq \epsilon_{\mathsf{RS-A}}$, where $\epsilon_{\mathsf{RS-A}}$ is the advantage of breaking the anonymity of ring signature scheme* RS*.*

Proof of Lemma 1. It is straightforward. Any efficient algorithm $\mathcal{D}$ which outputs k with probability p_1 in G1 non-negligibly different than probability p_0 for outputting k in the game G0, could be used as a sub-procedure to the attacker algorithm against the anonymity of ring signature RS. □

Let **G2** denote a modification of the previous game, where the content of the inner onions $c_i^{(n)}$, is switched to some random values and $c_i^{(0)} = (m_{\hat{r}}, \sigma'_r)$, but in the final decrypted list the pair $c_i^{(0)} = (m_{\hat{\imath}}, \sigma'_i)$ appears. We briefly denote that state by:

$$\sigma'_j = \mathsf{RingSign}(m_{\hat{\jmath}}, \mathsf{sks}_i, \mathsf{PKS}),$$
$$c_j^{(0)} = (m_{\hat{\jmath}}, \sigma'_j), c_j^{(n)} = E(\ldots(E(c_j^{(0)}, \mathsf{pke}_1), \ldots), \mathsf{pke}_n),$$
$$\sigma'_i = \mathsf{RingSign}(m_{\hat{\imath}}, \mathsf{sks}_j, \mathsf{PKS}),$$
$$c_i^{(0)} = (m_{\hat{r}}, \sigma'_r), c_i^{(n)} = E(\ldots(E(c_i^{(0)}, \mathsf{pke}_1), \ldots), \mathsf{pke}_n)$$

Let p_2 denote the probability that $\mathcal{D}$ outputs index k with that setup.

Lemma 2. $|p_1 - p_2| \leq \epsilon_{\mathsf{IND-CCA2}}$, *where* $\epsilon_{\mathsf{IND-CCA2}}$ *is the advantage of breaking the security of encryption scheme* ES.

Proof of Lemma 2. Any efficient algorithm $\mathcal{D}$ which outputs k with probability p_2 in G2 non-negligibly different than probability p_2 for outputting k in the game G2, could be used as a sub-procedure to the attacker algorithm $\mathcal{A}$ against the security of encryption scheme ES. Assume that $\mathcal{A}$ plays a security experiment $\mathsf{IND\text{-}CCA2}$ against the key pk. This pk will be treated as a public key of user 1 in G2. $\mathcal{A}$ prepares the messages $m_0 = c_i^{(0)} = (m_{\hat{\imath}}, \sigma'_i)$, and $m_1 = c_i^{(0)} = (m_{\hat{r}}, \sigma'_r)$ for the experiment $\mathsf{IND\text{-}CCA2}$, respectively. After getting a challenge c_b, it simulates the rest of secret keys and public keys for the run of protocol π_{SHP} specifically with the onions: $c_j^{(n)} = E(\ldots(E((m_{\hat{\jmath}}, \sigma'_j), \mathsf{pk}), \ldots), \mathsf{pke}_n)$, $c_i^{(n)} = E(\ldots(c_b), \mathsf{pke}_n)$. Now if $\mathcal{D}$ returns k with probability p_1 it behaves like in game G1 and m_b encodes $m_0 = c_i^{(0)} = (m_{\hat{\imath}}, \sigma'_i)$, otherwise it behaves like in G2 and m_b encodes $m_1 = c_i^{(0)} = (m_{\hat{r}}, \sigma'_r)$ □

Let **G3** denote a modification of the previous game, where the content of the inner onions $c_j^{(n)}$, is switched to some random values and $c_j^{(0)} = (m_{\hat{r}'}, \sigma'_{r'})$, but the final decrypted list includes the pair $c_j^{(0)} = (m_{\hat{\jmath}}, \sigma'_j))$. We briefly denote that state by:

$$\begin{aligned}
&\sigma'_j = \mathsf{RingSign}(m_{\hat{\jmath}}, \mathsf{sks}_i, \mathsf{PKS}),\\
&c_j^{(0)} = (m_{\hat{r}'}, \sigma'_{r'}), c_j^{(n)} = E(\ldots(E(c_j^{(0)}, \mathsf{pke}_1), \ldots), \mathsf{pke}_n),\\
&\sigma'_i = \mathsf{RingSign}(m_{\hat{\imath}}, \mathsf{sks}_j, \mathsf{PKS}),\\
&c_i^{(0)} = (m_{\hat{r}}, \sigma'_r), c_i^{(n)} = E(\ldots(E(c_i^{(0)}, \mathsf{pke}_1), \ldots), \mathsf{pke}_n)
\end{aligned}$$

Let p_3 denote the probability that $\mathcal{D}$ outputs index k with that setup.

Lemma 3. $|p_2 - p_3| \leq \epsilon_{\mathsf{IND-CCA2}}$, *where* $\epsilon_{\mathsf{IND-CCA2}}$ *is the advantage of breaking the security of encryption scheme* ES.

Proof of Lemma 3. Essentially as the proof of Lemma 2. □

Let **G4** denote a modification of the previous game, where the content of the inner onions $c_i^{(n)}$, is switched to: $c_i^{(0)} = (m_{\hat{\jmath}}, \sigma'_j)$. We briefly denote that state by:

$$\begin{aligned}
&\sigma'_j = \mathsf{RingSign}(m_{\hat{\jmath}}, \mathsf{sks}_i, \mathsf{PKS}),\\
&c_j^{(0)} = (m_{\hat{r}'}, \sigma'_{r'}), c_j^{(n)} = E(\ldots(E(c_j^{(0)}, \mathsf{pke}_1), \ldots), \mathsf{pke}_n),\\
&\sigma'_i = \mathsf{RingSign}(m_{\hat{\imath}}, \mathsf{sks}_j, \mathsf{PKS}), c_i^{(0)} = (m_{\hat{\jmath}}, \sigma'_j),\\
&c_i^{(n)} = E(\ldots(E(c_i^{(0)}, \mathsf{pke}_1), \ldots), \mathsf{pke}_n).
\end{aligned}$$

Let p_3 denote the probability that $\mathcal{D}$ outputs index k with that setup.

Lemma 4. $|p_3 - p_4| \leq \epsilon_{\mathsf{IND-CCA2}}$, *where* $\epsilon_{\mathsf{IND-CCA2}}$ *is the advantage of breaking the security of encryption scheme* ES.

Proof of Lemma 4. Essentially as the proof of Lemma 2. □

Let **G5** denote a modification of the previous game, where the content of the inner onions $c_j^{(n)}$, is switched to $c_j^{(0)} = (m_{\hat{i}}, \sigma'_i)$. We briefly denote that state by:

$$\begin{aligned}
&\sigma'_j = \mathsf{RingSign}(m_{\hat{j}}, \mathsf{sks}_i, \mathsf{PKS}),\\
&c_j^{(0)} = (m_{\hat{i}}, \sigma'_i), c_j^{(n)} = E(\ldots(E(c_j^{(0)}, \mathsf{pke}_1), \ldots), \mathsf{pke}_n),\\
&\sigma'_i = \mathsf{RingSign}(m_{\hat{i}}, \mathsf{sks}_j, \mathsf{PKS}),\\
&c_i^{(0)} = (m_{\hat{j}}, \sigma'_j), c_i^{(n)} = E(\ldots(E(c_i^{(0)}, \mathsf{pke}_1), \ldots), \mathsf{pke}_n).
\end{aligned}$$

Let p_3 denote the probability that $\mathcal{D}$ outputs index k with that setup.

Lemma 5. $|p_4 - p_5| \leq \epsilon_{\mathsf{IND-CCA2}}$, *where* $\epsilon_{\mathsf{IND-CCA2}}$ *is the advantage of breaking the security of encryption scheme* ES.

Proof of Lemma 5. Essentially as the proof of Lemma 2. □

Now we have $|p_0 - p_5| \leq \epsilon_{\mathsf{RS-A}} + 4\epsilon_{\mathsf{IND-CCA2}}$, which is negligible. Note that p_0 is the probability of $\mathcal{D}$ outputting k in G0, where the user j was given and signed $m_{\hat{j}}$ with his secret key sks_j, and the user i was given and signed $m_{\hat{i}}$ with his secret key sks_i. However p_5 is the probability of $\mathcal{D}$ outputting k in G5, where the user j was given and signed $m_{\hat{i}}$ with his secret key sks_i, and the user i was given and signed $m_{\hat{j}}$ with his secret key sks_i. Thus $\mathcal{D}$ cannot distinguish between two setups G0 and G5 where the messages $m_{\hat{j}}$ and $m_{\hat{i}}$ were switched between users j and i. □

For our enhanced protocol π_{SHP2}. we addressed an even stronger adversary. It is curious, but would also like to manipulate the onion ciphertexts such that the received messages and signatures at the proxy server are changed without detection. In the enhanced protocol we are utilizing the commitment steps, and we can conclude the following corollaries:
Corollary 1*: Assuming none of the user breaks, all commitments were correctly processed via* π_{SHP} *and outputted in step 3 of* π_{SHP2}.
Corollary 2*: Assuming accept in step 7 of* π_{SHP2} *none of the messages and signatures were replaced in* π_{SHP} *run in step 5 of* π_{SHP2}.

4.4.7 Performance Analysis

From our proof of concept implementation, our benchmark analysis of the proposed construction is summarized in Tab. 4.10. The analysis of operations is over one participant, together with the total run of the SHP when run over n participants. Our experimental setup covers all cryptographic computations, excluding communication complexity. We used standard RSA encryption, BLS ring signatures [26] and SHA256 hash computations, implemented and run with Python. The benchmark was run on an Apple 2020 M1, 8 GB RAM. We note that for VANET equipment, the bottlenecks are usually in the communication part [119] rather than the computational part, and the protocol performance should still be feasible for significantly slower hardware as long as the networking hardware is sufficient.

n users	Per user	Per operation (ms)
Encryption_{RSA}	n	0.3790
Decryption_{RSA}	n	3.1679
$\mathsf{RingSign}_{BLS}$	1	1.6341
$\text{Commitment}_{SHA256}$	1	0.0231
π_{SHP} run with $n = 20$	-	72.5721
π_{SHP2} run with $n = 20$	-	73.0341

Table 4.10: Benchmark analysis for each procedure, measurements are in milliseconds (ms). Encryption and decryption uses RSA, the ring signature scheme use BLS signatures as basis, and the commitment execution uses SHA256 computations.

4.4.8 Conclusion

We have introduced two variations of the SHP protocol, offering source hiding abilities, thus effectively mitigating the risks of traceability attacks and de-anonymization of an individual signer within a group on application *and* network layers. We have demonstrated the security and effectiveness of our schemes and determine that, even in demanding environments like VANET:s that require near-instant responses, our schemes are performance-efficient and thus, practically implementable.

Clearly, π_{SHP} and π_{SHP2} provides both theoretical and practical results in terms of mitigating challenges Sec1 and Sec2. The security analysis shows that our schemes are provably secure against message authenticity and privacy (Sec1), but also against impersonation attacks due to the unforgeability (Sec3).

4.5 Scenario $\mathsf{P_V}$: Leakage-Resilient Authenticated Key Exchange for Short-Range Devices

The following section is based on the published paper $\mathsf{P_V}$[5]. We recall that the published paper $\mathsf{P_V}$ did not use a VANET or C-ITS scenario, but instead illustrated the idea using wearable devices that connect and pair during an authentication process. We will generalize the idea here and only use the term *device*, referring to any short-range device that can be connected and used in any IoT- or connected infrastructure environment.

System and ENV setup: We analyze scenarios for devices, which could be used for any type of short-range device coupling, e.g., a wearable device and a vehicle or two IoT devices paired up for synchronized monitoring and data collection of traffic or other continuously generated data. The data is to be encrypted via a fresh symmetric session key each time the devices are paired together. Therefore, prior data transmission, an Authenicated Key Exchange (AKE) protocol π is executed on both devices. Signing modules (SM) can be used for signing procedures. The aim of using SMs in a device is to provide secure storage and signing operations within otherwise potentially untrusted systems. Architectures of both devices include such security modules for storing long-term secret keys used for authentication in π. SM thus denote a tamper resistant module storing a secret key sk used for *signing functionality* $\mathbf{f}$. We consider devices equipped with two SMs, where the long-term key is split additive. The functionalities of these SMs, denoted as $\mathbf{f1}$ and $\mathbf{f2}$ perform an operation pipeline, realizing the authentication, e.g. a signature, verified with the help of the public key.

4.5.1 Security Requirements

We formulate the following security requirements, given the previously described ENV and two devices for authenticated key exchange:

- We require a secure protocol for establish connection between two devices that collaborate, that is secure against *impersonation attacks*.
- We require that the architecture and protocol construction is not vulnerable to *key leakage*, namely that an attacker is able to use side-

[5]Published at International Conference on Cryptology and Network Security conference and is under copyright © 2022 Springer.

channel attacks or other sophisticated methods for extracting partial or whole secret session keys.

- We require the devices to use their own security modules for managing the key parts when using a secure key split technique.

These requirements implies that the security challenges Sec3 (impersonation attacks) and Sec5 (leakage resilience).

4.5.2 Related Work

A *Leakage-Resilient Authenticated Key Exchange* (LR-AKE) scheme is secure against leakage of long-term secret keys and/or ephemeral values [14, 102]. For example, an attacker running a side-channel attack will not be able to extract any secret information. Several models for leakage resilience has been proposed for AKE schemes. In the *bounded memory-leakage* model the adversary have access to an efficiently computable leakage function f which takes the secret key sk as input. Then $f(\mathsf{sk})$ output bits from the key partially, up to some fixed leakage parameter λ, hence bounded. Several AKE schemes has been proven secure within the bounded model, e.g. [139, 14, 102]. Another model is the *continous* (unbounded) leakage model, introduced by Brakerski et. al [30] and Dodis et al [48]. In the continous memory-leakage model, no restrictions are set of neither time or memory. When the adversary calls the leakage function, it may only receive at most a specified fraction of the total bits from the internal state of the attacked memory, which consists of the secret key and the entropy source [30].

Key splitting and refresh mechanisms are found in different type of cryptographical schemes, not only additive key splitting is used as in [163, 145, 88, 63] but also more complicated variants; in puncturable encryption where the key is associated with a tag, the secret key sk is refreshed using exponentiations with a set of random values, proven secure with bilinear pairings [68]. We note that the practical implementation in [68] shows that the key refresh function is on average slower than the encryption function. Another type of key splitting is when using trusted third parties, e.g. in [33] where a key generation center issues keys and generates partial randomness used for key splitting and refreshing; again the scheme is secure under pairings. However, the security relies on trusted third parties which are not included in our scenario nor applicable for wearables and e-health devices.

In figure 4.10 we compare different kind of AKE protocols. The complexity is measured by the number of exponentiations on each party's side as well as the number of signatures (S), verifications (V), ring signatures (RS) and

Paper	Protocol	Complexity	Rounds
[102]	NAXOS	4	2
[134]	E-NAXOS	5	2
[154]	CMQV	3	2
[141]	SMQV	3	2
[79]	without-NAXOS	3	2
[104]	KEA+C	3	3
[93]	HMQV	3	3
[152]	BLS-HMQV	4	3
[36]	SIGMA ver. Σ_0	2 + S + V	3
[95]	mod Σ_0	2 + RS + RV	3
[74]	AMA	4	4
[75]	MRI	3	4

Figure 4.10: Protocol comparison.

ring verifications (RV). We use same notation from [36, 79, 152]. Note that authentication in [36, 79, 152] is provided by undeniable signature based schemes that is used for the mutual identification of parties. In [79, 152] the BLS signature has been used. It can be denoted as protocols "without NAXOS" and "BLS-HMQV" protocols, respectively.

4.5.3 Threat Model

The threat model covers attacks against authenticated key establishment (as of AKE protocols) between $\mathcal{I}$ and $\mathcal{R}$ parties. These involves impersonation attacks, and attacks against the session key secrecy, covering a typical *active attacker* $\mathcal{A}$ of the Canetti-Krawczyk (CK) model from [36] with the ability to interfere and block communication between $\mathcal{I}$ and $\mathcal{R}$. $\mathcal{A}$ can modify messages, replace, inject and redirect them towards parties $\mathcal{I}$ and $\mathcal{R}$.

Partial key leakage : We consider a *stronger attacker model* which allows $\mathcal{A}$ to access partial secret keys. We assume $\mathcal{A}$ can learn secrets from one SM of a chosen device per signing session, but is able to compromise different SMs of the target device in different sessions (see Fig. 4.11 - in one session $\mathcal{A}$ accesses sk_1 of $\mathcal{I}$ and sk_2 of $\mathcal{R}$). This reflects additional scenarios of using a SM from untrusted vendors, or with implementation errors, where a malicious device manufacturer allows leakages from one SM per session.

Impersonation : The impersonation attack against the scheme [163], shown in our paper, affects the first post requirement for AKE protocol execution, namely improper acceptance. Intuitively we demand that each party should use its secret key to perform the protocol and be accepted

by its peer party. In the impersonation attack we demonstrate how an active adversary can impersonate another party only with the knowledge of public parameters.

Session key secrecy compromising : In this type of attack the adversary tries to get hold of the session key used between two parties. It could be feasible due to numerous reasons, e.g. by key leakage or bad protocol design allowing for cryptanalysis of the key(s).

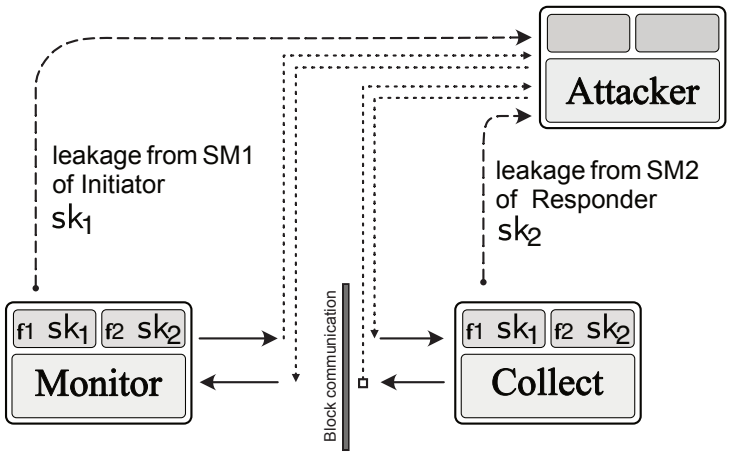

Figure 4.11: Two devices that are blocked during a run of an AKE protocol, where the attacker exploits key leakage for impersonation attacks.

To formalize the impersonation vulnerability we define the following notion:

Definition 36. *We say that AKE protocol π is* impersonation vulnerable *if there exists a polynomial-time adversary algorithm $\mathcal{A}$ such that at least one of the probabilities:*

$$\Pr[\pi(\mathcal{I}(\mathsf{sk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{R}}), \mathcal{A}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{R}})) \rightarrow (\mathcal{I} \texttt{ accept } \mathcal{A} \text{ as } \mathcal{R})],$$
$$\Pr[\pi(\mathcal{A}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{R}}), \mathcal{R}(\mathsf{sk}_{\mathcal{R}}, \mathsf{pk}_{\mathcal{I}})) \rightarrow (\mathcal{R} \texttt{ accept } \mathcal{A} \text{ as } \mathcal{I})]$$

is non-negligible. The first event $\mathcal{A}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{R}})$ denotes adversary $\mathcal{A}$ impersonating $\mathcal{R}$. Similarly, the second event $\mathcal{A}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{R}})$ is the impersonation of $\mathcal{I}$.

4.5.4 Preliminaries

Security Enchancing Techniques

A *secret key split* technique splits one initial secret key into several partial secrets stored independently in separate SMs, coming from different vendors. Having at least one trusted/fair vendor mitigates potential secret key compromising, even if a subset of SMs allow for partial secret leakages. A *partial secret refresh* technique changes the partial secrets synchronously, in a way that the final signing functionality performed collectively by SMs using partial secrets, can be verified with a single public key. The public key corresponds to the initial secret key (as it was before split). The goal of the continuous refreshing is to mitigate leakages that could occur from different SMs in different signing sessions, and to protect against side channel and timing attacks targeting operations involving secrets, that would potentially leave energy, voltage or radio frequency traces. This strategy is sometimes referred to as key blinding, or rather *key splitting* [145, 88, 63], where a subprotocol refreshes the key shares. We consider scenarios (after [163]) with a dual SM setup, with **f1** and **f2** signing functionalities respectively. We denote the relation between those SMs as a *pipeline*. It defines the order of execution of **f1** and **f2** in each SM, as well as the combined operation on their respective results. As in [163] an *additive* key splitting scheme is used where the secret key is split into two random shares $\mathsf{sk} = \mathsf{sk}_1^i + \mathsf{sk}_2^i$. The device is now able to erase sk and store $\mathsf{sk}_1^i, \mathsf{sk}_2^i$. These shares are refreshed synchronously, i.e. $\mathsf{sk}_1^{i+1} = \mathsf{sk}_1^i - r$ and $\mathsf{sk}_2^{i+1} = \mathsf{sk}_2^{i+1} + r$ for some random value $r \xleftarrow{\$} \mathbb{Z}_q^*$. An *OOB confirmation* technique is used for two devices with short range connectivity, e.g. IoT or wearables. It is not unusual to pair them using an optical OOB channel for initial authentication and security configurations. Typical scenarios cover pairing an IoT device with a smart monitor or hub, or a wearable device with the user's smartphone. During the pairing phase, the user may verify codes or authentication messages displayed on the devices. Therefore, the OOB confirmation involves human interaction and final accepting or rejecting the devices.

AKE protocols

We briefly explain the concept of an authenticated key exchange protocol. Let $\mathcal{I}$ denote an initiator device, and $\mathcal{R}$ a responder device. These two are the only peers in protocol π. The initiator is the party who initiates π and sends the first message. Each party have key-pairs of long-term secret/public keys, $(\mathsf{sk}_\mathcal{I}, \mathsf{pk}_\mathcal{I})$ and $(\mathsf{sk}_\mathcal{R}, \mathsf{pk}_\mathcal{R})$ respectively. Let $\pi(\mathcal{I}(\mathsf{sk}_\mathcal{I}, \mathsf{pk}_\mathcal{R}), \mathcal{R}(\mathsf{sk}_\mathcal{R}, \mathsf{pk}_\mathcal{I}))$ denote the protocol execution between $\mathcal{I}$ and $\mathcal{R}$, where each party have access

to the other's public key. For the protocol $\pi(\mathcal{I}(\mathsf{sk}_\mathcal{I}, \mathsf{pk}_\mathcal{R}), \mathcal{R}(\mathsf{sk}_\mathcal{R}, \mathsf{pk}_\mathcal{I}))$ we set the following *post execution protocol requirements*:

1. Let $\pi(\mathcal{I}(\mathsf{sk}_\mathcal{I}, \mathsf{pk}_\mathcal{R}), \mathcal{R}(\mathsf{sk}_\mathcal{R}, \mathsf{pk}_\mathcal{I})) \rightarrow (\mathcal{I}\ \texttt{accept}\ \mathcal{R})$ denote when $\mathcal{I}$ authenticates $\mathcal{R}$, i.e the holder of the secret $\mathsf{sk}_\mathcal{R}$ corresponds to the public key $\mathsf{pk}_\mathcal{R}$. Similarily we denote $\pi(\mathcal{I}(\mathsf{sk}_\mathcal{I}, \mathsf{pk}_\mathcal{R}), \mathcal{R}(\mathsf{sk}_\mathcal{R}, \mathsf{pk}_\mathcal{I})) \rightarrow (\mathcal{R}\ \texttt{accept}\ \mathcal{I})$ for the fact that $\mathcal{R}$ knows the identity of $\mathcal{I}$, i.e. the holder of the secret $\mathsf{sk}_\mathcal{I}$ corresponding to the public key $\mathsf{pk}_\mathcal{I}$. If both events occur, we say that $\mathcal{I}$ and $\mathcal{R}$ mutually authenticate each other.

2. $\mathcal{I}$ and $\mathcal{R}$ compute same session key K_s.

3. K_s is secret, i.e. it is only known to $\mathcal{I}$ and $\mathcal{R}$.

Assumptions

From the formal perspective we consider AKE protocols based on the Diffie-Hellman (DH) key exchange. We assume that corresponding computations on devices are done within a group $\mathbb{G} = \langle g \rangle$ of prime order q, where the chosen computational assumptions hold. Here we rely on the DLP,CDH, and DDH.

4.5.5 Cryptanalysis

Original Scheme

We make a remark here that the notation in the reminder of $\mathsf{P_V}$ is based on the original scheme's notation and does not follow our standard notation completely. This is due to better readability. The original scheme [163] use public parameters $\mathsf{par} = \{\mathbb{G}, g, q\}$, where $\langle g \rangle = \mathbb{G}$, and $|\mathbb{G}| = q$ is prime. Key-pairs for each party is generated as $\mathsf{sk}_\mathcal{I} \xleftarrow{\$} \mathbb{Z}_q^*, \mathsf{pk}_\mathcal{I} = g^{\mathsf{sk}_\mathcal{I}}$ and $\mathsf{sk}_\mathcal{R} \xleftarrow{\$} \mathbb{Z}_q^*, \mathsf{pk}_\mathcal{R} = g^{\mathsf{sk}_\mathcal{R}}$ respectively. Key splitting consists of two sub-protocols:

$\mathsf{Split}(\mathsf{sk}) \rightarrow (\mathsf{sk}_1^{(0)}, \mathsf{sk}_2^{(0)})$**:** takes a user's secret key sk as input, generates $\mathsf{sk}_1^{(0)} \xleftarrow{\$} \mathbb{Z}_q^*$ and computes $\mathsf{sk}_2^{(0)} = \mathsf{sk} - \mathsf{sk}_1^{(0)}$. Values $\mathsf{sk}_1^{(0)}$ and $\mathsf{sk}_2^{(0)}$ are stored securely and sk is removed from memory since $\mathsf{sk} = \mathsf{sk}_1^{(0)} + \mathsf{sk}_2^{(0)}$.

$\mathsf{RF}(\mathsf{sk}_1^{(i)}, \mathsf{sk}_2^{(i)}) \rightarrow (\mathsf{sk}_1^{(i+1)}, \mathsf{sk}_2^{(i+1)})$**:** every i-th time (for $i \geq 1$) the splitting values update as follows: $\mathsf{sk}_1^{(i+1)} = \mathsf{sk}_1^{(i)} + s_i$ and $\mathsf{sk}_2^{(i+1)} = \mathsf{sk}_2^{(i)} - s_i$, where $s_i \xleftarrow{\$} \mathbb{Z}_q^*$.

We recall the complete original scheme [163] in Fig. 4.12.

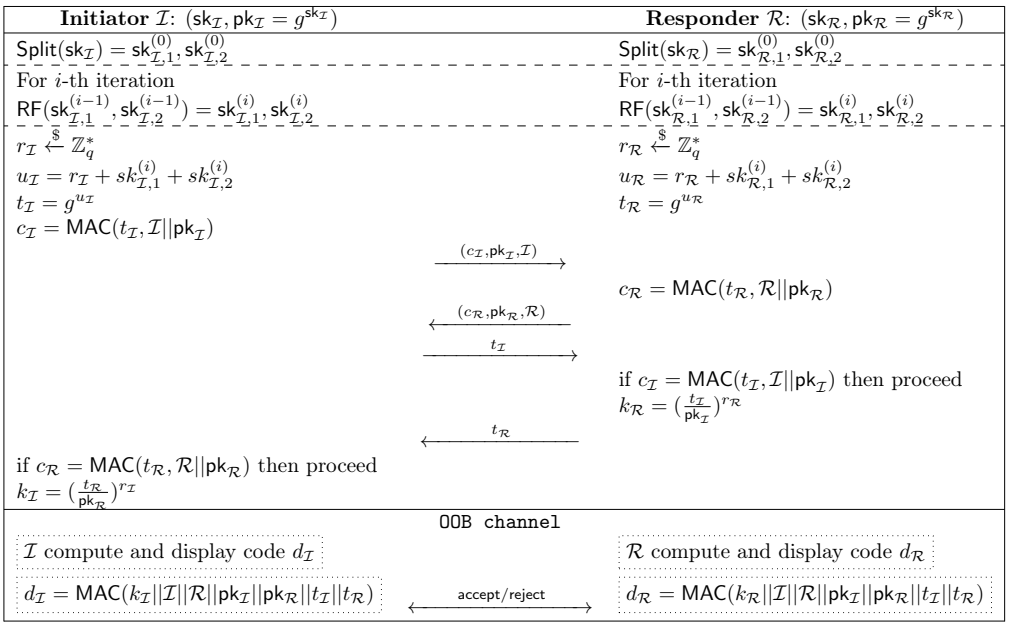

Figure 4.12: Original LR-AKE scheme with additive key splitting. OOB communication is used for verifying the display code on both devices, to finalize the pairing; here by the user checking that code $d_\mathcal{I}$ shows up and equals $d_\mathcal{R}$ on both displays.

Attack Description

Here we consider the attack in which $\mathcal{A}$ impersonate $\mathcal{R}$ in front of $\mathcal{I}$. Note that similar attack in which $\mathcal{A}$ impersonate $\mathcal{I}$ in front of $\mathcal{R}$ can be mounted in a *mutatis mutandis* manner. Let $\mathcal{I}$ and $\mathcal{R}$ setup the devices for the scheme in Fig. 4.12. When $\mathcal{I}$ starts the protocol towards $\mathcal{R}$, the adversary $\mathcal{A}$ captures $(c_\mathcal{I}, pk_\mathcal{I}, \mathcal{I})$ and subsequently impersonates $\mathcal{R}$ in front of $\mathcal{I}$, using its own fake device with its own choice of random values. We also note that the shared secret $k_\mathcal{I} = k_\mathcal{R}$ does not bind the private long-term key in any step (i.e. provide authentication); it rather cancels out via the public key.

$$k_\mathcal{I} = \left(\frac{t_\mathcal{R}}{\mathsf{pk}_\mathcal{R}}\right)^{r_\mathcal{I}} = \left(\frac{g^{u_\mathcal{R}}}{\mathsf{pk}_\mathcal{R}}\right)^{r_\mathcal{I}} = \left(\frac{g^{r_\mathcal{R} + \mathsf{sk}^{(i)}_{\mathcal{R},1} + sk^{(i)}_{\mathcal{R},2}}}{\mathsf{pk}_\mathcal{R}}\right)^{r_\mathcal{I}} \tag{4.22}$$

$$= \left(\frac{g^{r_\mathcal{R}} g^{\mathsf{sk}_\mathcal{R}}}{\mathsf{pk}_\mathcal{R}}\right)^{r_\mathcal{I}} = \left(\frac{g^{r_\mathcal{R}} \mathsf{pk}_\mathcal{R}}{\mathsf{pk}_\mathcal{R}}\right)^{r_\mathcal{I}} = g^{r_\mathcal{I} r_\mathcal{R}} = k_\mathcal{R}. \tag{4.23}$$

and similarly

$$k_\mathcal{R} = g^{r_\mathcal{I} r_\mathcal{R}}.$$

Now, the attack is mounted explicitly as follows:

1. $\mathcal{A}$ prepares a fake computing device using only the public parameters.
2. $\mathcal{I}$ sends to $\mathcal{R}$ a tuple $(c_{\mathcal{I}}, \mathsf{pk}_{\mathcal{I}}, \mathcal{I})$.
3. $\mathcal{A}$ intercepts that message. It generates $r_{\mathcal{A}} \xleftarrow{\$} \mathbb{Z}_q^*$, computes $t_{\mathcal{A}} = \mathsf{pk}_{\mathcal{R}} \cdot g^{r_{\mathcal{A}}}$, and $c_{\mathcal{A}} = \mathsf{MAC}(t_{\mathcal{A}}, \mathcal{R}||\mathsf{pk}_{\mathcal{R}})$. and sends to $\mathcal{I}$ the response message $(c_{\mathcal{A}}, \mathsf{pk}_{\mathcal{R}}, \mathcal{R})$, i.e. $\mathcal{A}$ tries to be $\mathcal{R}$ in front of $\mathcal{I}$.
4. $\mathcal{A}$ computes its session key $k_{\mathcal{A}} = \left(\frac{t_{\mathcal{I}}}{\mathsf{pk}_{\mathcal{I}}}\right)^{r_{\mathcal{A}}}$ which will be used instead of the session key of $\mathcal{R}$.
5. $\mathcal{A}$ displays $d_{\mathcal{A}} = \mathsf{MAC}(k_{\mathcal{A}}||\mathcal{I}||\mathcal{R}||\mathsf{pk}_{\mathcal{I}}||\mathsf{pk}_{\mathcal{R}}||t_{\mathcal{I}}||t_{\mathcal{A}})$ code on the fake device.
6. $\mathcal{I}$ computes its session key $k_{\mathcal{I}} = \left(\frac{t_{\mathcal{A}}}{\mathsf{pk}_{\mathcal{R}}}\right)^{r_{\mathcal{I}}}$.
7. $\mathcal{I}$ displays $d_{\mathcal{I}} = \mathsf{MAC}(k_{\mathcal{I}}||\mathcal{I}||\mathcal{R}||\mathsf{pk}_{\mathcal{I}}||\mathsf{pk}_{\mathcal{R}}||t_{\mathcal{I}}||t_{\mathcal{A}})$ code on the device.

The impersonation attack is possible despite the fact that the secret key is split, and the refresh procedure is triggered for each protocol execution. We note that $\mathcal{A}$ impersonates $\mathcal{R}$ in front of $\mathcal{I}$ since:

- In both devices, $\mathcal{I}$ and $\mathcal{A}$ compute the same session key $k_{\mathcal{I}} = k_{\mathcal{A}} = g^{r_{\mathcal{I}} r_{\mathcal{A}}}$, namely

$$k_{\mathcal{I}} = \left(\frac{t_{\mathcal{A}}}{\mathsf{pk}_{\mathcal{R}}}\right)^{r_{\mathcal{I}}} = \left(\frac{\mathsf{pk}_{\mathcal{R}} \cdot g^{r_{\mathcal{A}}}}{\mathsf{pk}_{\mathcal{R}}}\right)^{r_{\mathcal{I}}} = g^{r_{\mathcal{I}} r_{\mathcal{A}}}, \tag{4.24}$$

$$k_{\mathcal{A}} = \left(\frac{t_{\mathcal{I}}}{\mathsf{pk}_{\mathcal{I}}}\right)^{r_{\mathcal{A}}} = \left(\frac{g^{u_{\mathcal{I}}}}{\mathsf{pk}_{\mathcal{I}}}\right)^{r_{\mathcal{A}}} = \left(\frac{g^{r_{\mathcal{I}} + sk_{\mathcal{I},1}^{(i)} + sk_{\mathcal{I},2}^{(i)}}}{\mathsf{pk}_{\mathcal{I}}}\right)^{r_{\mathcal{A}}} \tag{4.25}$$

$$= \left(\frac{g^{r_{\mathcal{I}} + \mathsf{sk}_{\mathcal{I}}}}{\mathsf{pk}_{\mathcal{I}}}\right)^{r_{\mathcal{A}}} = \left(\frac{g^{r_{\mathcal{I}}} \cdot \mathsf{pk}_{\mathcal{I}}}{\mathsf{pk}_{\mathcal{I}}}\right)^{r_{\mathcal{A}}} = g^{r_{\mathcal{I}} r_{\mathcal{A}}}. \tag{4.26}$$

- The secret key $\mathsf{sk}_{\mathcal{R}}$ is not necessary when computing the session key $k_{\mathcal{A}}$, since $k_{\mathcal{I}} = k_{\mathcal{A}}$ and both devices will display the same message authentication code $d_{\mathcal{I}} = d_{\mathcal{A}}$.
- The device of $\mathcal{I}$ proceeds as in regular communication with $\mathcal{R}$, making all computations using the public key $\mathsf{pk}_{\mathcal{R}}$ and values received, it "thinks" the computed session key is shared with the device of $\mathcal{R}$, hence "certified" with the public key $\mathsf{pk}_{\mathcal{R}}$.

- A user who inspects the device visually, falsely concludes that they mutually authenticated themselves as the displayed values $d_{\mathcal{I}}$ and $d_{\mathcal{A}}$ are equal.

4.5.6 Proposed Scheme

Instead of designing a new protocol from scratch, we integrate new functionalities in existing solutions. Our methodology of improvement relies on provably secure cryptographic building blocks. The fundamental feature is the modular construction of SIGMA, based on signature scheme SIG, message authentication code MAC, and pseudorandom function PRF. The layered architecture enables implementation flexibility based on reuse of existing libraries, already tested in commercial and industrial environments. Alternatively, it is possible to replace the building blocks, with specialized counterparts to achieve additional functionality. This approach was used in several constructions; e.g. in [152] a BLS layer was added on the HMQV protocol to mitigate extended key compromise attacks (eKCI), or in [97], where the regular signature SIG in SIGMA was replaced by anonymous ring signature RSIG to achieve the deniability property. The methodology could be summarised in the following stages:

1. **Base AKE choice**: select an existing AKE scheme, preferably one that it is based on SIG and MAC components, which are provable secure in a commonly accepted formal security model.

2. **Long-term secret key split**: apply a split-modification on the SIG scheme, i.e. split the long-term signing secret key into two secret subkey shares.

3. **Distinct SMs**: store the shares of the split secret key in two separate SMs, each performing the signing functionality.

4. **Key refreshment in SMs**: apply a key refreshment technique in both SMs, each using preferably the same hardware-based source of randomness.

5. **OOB channel**: define a protocol step which utilizes an OOB channel.

6. **Initial security model adjustment**: adjust the initial security model of the chosen base AKE, to reflect the long-term secret key splitting and refresh menchanism. Adjust the adversary power to the ability of learning partial secrets in different protocol sessions. This would

address the risk of secret key-leakage e.g. produced by an untrusted manufacturer of the SMs.

7. **Formal proof**: prove the security of the construction in the adjusted model.

Secure AKE Construction with Signature Components

Let us briefly recall the 3-round version of the SIGMA protocol, denoted as Σ_0 in [36]. Two parties, the initiator $\mathcal{I}$ and responder $\mathcal{R}$, exchange messages using predefined secure building blocks such as a signature function SIG, a message authentication code function MAC, and a pseudorandom function PRF, to identify themselves and to establish a secret session key.

The SIGMA protocol is described in Fig. 4.13 by rows indicated as `a) original SIGMA`. We briefly recall the steps:

1. Initiator $\mathcal{I}$ generate a session identifier sid, pick at random $x \xleftarrow{\$} \mathbb{Z}_q$ and compute an ephemeral DH public key g^x. $\mathcal{I}$ then sends (sid, g^x) to $\mathcal{R}$.

2. $\mathcal{R}$ picks at random $y \xleftarrow{\$} \mathbb{Z}_q$ and computes an ephemeral DH public key g^y. Next, a key $(g^x)^y = g^{xy}$ is computed. $\mathcal{R}$ then derive keys $k_0 = \mathsf{PRF}_{g^{xy}}(0)$, and $k_1 = \mathsf{PRF}_{g^{xy}}(1)$. Subsequently, $\mathcal{R}$ erases y and g^{xy} from the device memory and computes $\mathsf{MAC}_{k_1}(\text{“1”}, \mathsf{sid}, ID_{\mathcal{R}})$, and $\mathsf{SIG}_{\mathsf{sk}_{\mathcal{R}}}(\text{“1”}, \mathsf{sid}, g^x, g^y)$. Finally, $\mathcal{R}$ sends sid, its own identifier $ID_{\mathcal{R}}$, the public key g^y, the signature, and the computed MAC in the response message to $\mathcal{I}$.

3. $\mathcal{I}$ computes the key $(g^y)^x = g^{xy}$ and derives $k_0 = \mathsf{PRF}_{g^{xy}}(0)$, and $k_1 = \mathsf{PRF}_{g^{xy}}(1)$. Subsequently, $\mathcal{I}$ erases x and g^{xy} from the device memory. Next, $\mathcal{I}$ verifies $\mathsf{MAC}_{k_1}(\text{“1”}, \mathsf{sid}, ID_{\mathcal{R}})$, retrieves the public key of the party identified by $ID_{\mathcal{R}}$ and verifies $\mathsf{SIG}_{\mathsf{sk}_{\mathcal{R}}}(\text{“1”}, \mathsf{sid}, g^x, g^y)$. If any of the verification procedures fail, $\mathcal{I}$ aborts the session and outputs “reject”. If verification is succesful, $\mathcal{I}$ computes $\mathsf{MAC}_{k_1}(\text{“0”}, s, ID_{\mathcal{I}})$, and $\mathsf{SIG}_{\mathsf{sk}_{\mathcal{I}}}(\text{“0”}, \mathsf{sid}, g^x, g^y)$. Finally, $\mathcal{I}$ sends sid, $ID_{\mathcal{I}}$, the signature, and the computed MAC to $\mathcal{R}$. $\mathcal{I}$ completes the session with public output $(ID_{\mathcal{I}}, \mathsf{sid}, ID_{\mathcal{R}})$ and the secret session key k_0.

4. $\mathcal{R}$ verifies $\mathsf{MAC}_{k_1}(\text{“0”}, \mathsf{sid}, ID_{\mathcal{I}})$, retrieves the public key $\mathsf{pk}_{\mathcal{I}}$ and verifies signature $\mathsf{SIG}_{\mathsf{sk}_{\mathcal{I}}}(\text{“0”}, \mathsf{sid}, g^x, g^y)$. If any of the verifications fail, $\mathcal{R}$ aborts the session and outputs “reject”, otherwise $\mathcal{R}$ completes the session with public output $(ID_{\mathcal{R}}, \mathsf{sid}, ID_{\mathcal{I}})$ and the session key k_0.

Improved LR-AKE using Split Schnorr Signatures with Key Refresh and OOB Optical Inspection

We propose a modified SIGMA with split Schnorr signatures [130], utilizing signature secret/public key pairs $(\mathsf{sk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{I}} = g^{\mathsf{sk}_{\mathcal{I}}})$ and $(\mathsf{sk}_{\mathcal{R}}, \mathsf{pk}_{\mathcal{R}} = g^{\mathsf{sk}_{\mathcal{R}}})$ for $\mathcal{I}$ and $\mathcal{R}$ respectively, set in an appropriate group $\mathbb{G}$ of computation. Each party's device is augmented with two separate SMs for Schnorr signature functionality, containing signing secret key split shares. These shares are refreshed in each session i in a synchronized way; namely the SM of initiator $\mathcal{I}$ in session i contain keys: $\mathsf{sk}_{\mathcal{I},1}^{(i)}, \mathsf{sk}_{\mathcal{I},2}^{(i)}$. Similarly the SM of responder $\mathcal{R}$ contains $\mathsf{sk}_{\mathcal{R},1}^{(i)}, \mathsf{sk}_{\mathcal{R},2}^{(i)}$. For the key refreshment procedure, each SM is equipped with the pseudo-random number generator PRNG, accessing the hardware based source of randomness $\xi_{\mathcal{I}}$, $\xi_{\mathcal{R}}$ for $\mathcal{I}$ and $\mathcal{R}$ devices respectively. We assume for the security and synchronization purposes, that both SMs access the same source of randomness.

Definition 37 (Signatures with key split and refresh (SIGSRF))**.** *Signatures with key split and refresh is defined as a tuple of procedures* SIGSRF = (ParGen, KeyGen, InitRF, RF, SignRF, Ver)*:*

$\mathsf{ParGen}(\lambda) \to \mathsf{par}$*: takes security parameter* λ *and outputs parameters* par*. These are default parameters of the subsequent procedures in the scheme, therefore we omit them for simplicity of notation.*

$\mathsf{KeyGen}(\mathsf{par}) \to (\mathsf{sk}, \mathsf{pk})$*: takes parameters* par *and output a pair of secret and public keys* sk, pk *respectively.*

$\mathsf{InitRF}(\mathsf{sk}) \to (\mathbf{f1}, \mathbf{f2})$*: takes secret key* sk *splits it to parts* $\mathsf{sk}_1, \mathsf{sk}_2$ *and stores it securely in SM1, SM2 with functionalities* $\mathbf{f1}, \mathbf{f2}$ *respectively.*

$\mathsf{RF}(\mathbf{f1}, \mathbf{f2}) \to (\mathbf{f1}, \mathbf{f2})$*: takes modules SM1, SM2 with respective functionalities* $\mathbf{f1}, \mathbf{f2}$*, and refreshes the partial keys* $\mathsf{sk}_1, \mathsf{sk}_2$ *stored in these modules.*

$\mathsf{SignRF}(m, \mathbf{f1}, \mathbf{f2}) \to \sigma$*: Takes a message* m *and processes it with SM1 and SM2 via* $\mathbf{f1}, \mathbf{f2}$ *respectively, outputting a signature* σ*.*

$\mathsf{Ver}_{\mathsf{pk}}(m, \sigma) \to 1/0$*: Returns* 1 *for "accept", or* 0 *for "reject"*

Definition 38 (SIGSRF correctness)**.** *Let* SIGSRF = (ParGen, KeyGen, InitRF, RF, SignRF, Ver) *is a signature scheme with key split and refresh.* SIGSRF *is*

correct if for any message m and any integer ℓ:

$$\Pr\left[\begin{array}{l}\mathsf{ParGen}(\lambda)\rightarrow\mathsf{par},\\ \mathsf{KeyGen}(\mathsf{par})\rightarrow(\mathsf{sk},\mathsf{pk})\\ \mathsf{InitRF}(\mathsf{sk})\rightarrow(\mathbf{f1},\mathbf{f2})\\ \text{for } i=1 \textit{ to } \ell \text{ run } \mathsf{RF}(\mathbf{f1},\mathbf{f2})\\ \mathsf{SignRF}(m,\mathbf{f1},\mathbf{f2})\rightarrow\sigma\\ \mathsf{Ver}(m,\sigma,\mathsf{pk})\rightarrow 1\end{array}\right]=1.$$

To address the scenario with partial secret key leakage, a new, stronger security model for key splitting and refresh is used. In this model a malicious forger $\mathcal{F}$ has the ability to query an additional $\mathsf{SignReveal}$ oracle $\mathcal{O}_{\mathsf{SignRev}}(m,j)$ which return the signature over m, all messages T exchanged between modules SM1, SM2, and actual partial secret key stored in module indicating by index j.

Definition 39 (Sign Reveal Unforgeability (SR))**.** *Let* (ParGen, KeyGen, InitRF, RF, SignRF, Ver) *be a split signature scheme. We define security experiment* $\mathtt{Exp}^{\lambda,\ell}_{\mathsf{SR}}$*:*

`Init` : $\mathsf{par}\leftarrow\mathsf{ParGen}(\lambda)$ $(\mathsf{sk},\mathsf{pk})\leftarrow\mathsf{KeyGen}(\mathsf{par})$.

`SignRev Oracle` : *There are two potential calls:*

- $\mathcal{O}_{\mathsf{SignRev}}(m,1)\rightarrow(T,\mathsf{sk}_1,\sigma)$ *takes a message m, the SM indicator $j=1$ and outputs a transcript T of messages exchanged between modules SM1, SM2, the fresh partial secret key stored in SM1, and the final signature σ generated with both SM1 and SM2, such that* $\mathsf{Ver}(\sigma,\mathsf{pk},m)=1$.
- $\mathcal{O}_{\mathsf{SignRev}}(m,2)\rightarrow(T,\mathsf{sk}_2,\sigma)$ *similarly as above, where sk_2 is the partial secret key stored in SM2.*

The oracle models the device in which partial secret key leakage can happen multiple times in different sessions, but only once and from one SM per signing session (i.e. never from both devices in single signing session).

`Hash Oracle` : *The hash oracle $\mathcal{O}_{\mathcal{H}}$ is modeled in ROM.*

`Adversary` : *Let the adversary $\mathcal{F}^{\mathcal{O}_{\mathsf{SignRev}},\mathcal{O}_{\mathcal{H}}}_{\mathsf{SR}}(\mathsf{pk})$, be a malicious algorithm initialized with the public key pk, having access to the oracles $\mathcal{O}_{\mathsf{SignRev}}$ and $\mathcal{O}_{\mathcal{H}}$. It issues ℓ number of queries to the oracles. Let $\mathcal{M}=\{m_i\}^{\ell}_1$, and $\Omega=\{\sigma_i\}^{\ell}_1$ denote the set of the messages, and the corresponding signatures the oracles process.*

`Forgery` : *The adversary generates a tuple:* $(m^*, \sigma^*) \leftarrow \mathcal{F}_{\mathsf{SR}}^{\mathcal{O}_{\mathsf{SignRev}}, \mathcal{O}_{\mathcal{H}}}(\mathsf{pk})$ *for a new* $m^* \notin \mathcal{M}$*, which was not queried to* $\mathcal{O}_{\mathsf{SignRev}}$ *oracle.*

We say that the signature scheme is secure if for each forgery type, the probability that the adversary produces a valid signature is negligible in parameters λ, ℓ*:*

$$\Pr\left[\begin{array}{l} \mathsf{ParGen}(\lambda) \rightarrow \mathsf{par}, \\ \mathsf{KeyGen}(\mathsf{par}) \rightarrow (\mathsf{sk}, \mathsf{pk}) \\ \mathsf{InitRF}(\mathsf{sk}) \rightarrow (\mathbf{f1}, \mathbf{f2}) \\ (m^*, \sigma^*) \leftarrow \mathcal{F}_{\mathsf{SR}}^{\mathcal{O}_{\mathsf{SignRev}}, \mathcal{O}_{\mathcal{H}}}(\mathsf{pk}) \\ \mathsf{Ver}(m^*, \sigma^*, \mathsf{pk}) \rightarrow 1 \\ m^* \notin \mathcal{M} \end{array}\right] \leq \epsilon(\lambda, \ell).$$

Schnorr Signatures with Additive Key Splitting and Refresh

We recall the modification of the Schnorr signature with secret key splitting [130]. The scheme assumes the parties involved in signing (i.e. SM1 and SM2 in our naming convention) share the same source of randomness, e.g. secure PRNG initiated with a hardware based seed. The scheme is defined as follows:

$\mathsf{ParGen}(\lambda) \rightarrow \mathsf{par}$: takes security parameter λ and outputs parameters $\mathsf{par} = (p, q, \langle g \rangle = \mathbb{G})$, where p, q are large primes chosen such that the DLP is assumed hard in a subgroup $\langle g \rangle$, of order q in $\mathbb{Z}_p$. The Schnorr based scheme uses a secure hash function $\mathcal{H} : \{0,1\}^* \rightarrow \mathbb{Z}_q$. The signing procedure itself forms a protocol between signing entities, where commitment to a randomness exchange is realized w.l.o.g via $\mathcal{H}$.

$\mathsf{KeyGen}(\mathsf{par}) \rightarrow (\mathsf{sk}, \mathsf{pk})$: takes parameters par and output a secret and public keys sk, pk respectively.

$\mathsf{InitRF}(\mathsf{sk}) \rightarrow (\mathbf{f1}, \mathbf{f2})$: two SMs are initialized in the following way. In the first stage ($i = 0$) sk is split by computing $\mathsf{sk}_1^{(0)} \xleftarrow{\$} \mathbb{Z}_q$, then $\mathsf{sk}_2^{(0)} = \mathsf{sk} - \mathsf{sk}_1^{(0)}$. The sk_1 is stored in SM1 and sk_2 in SM2.

$\mathsf{RF}(\mathbf{f1}, \mathbf{f2}) \rightarrow (\mathbf{f1}, \mathbf{f2})$: The hardware based randomness is used to initialize the PRNG of SM1 and SM2. The refreshed keys inside the SMs in each session $i = 1, 2, \ldots$ are based on updating the corresponding values from the previous session by the current output from $\mathsf{PRNG}(i)$, namely: $\mathsf{sk}_1^{(i)} = \mathsf{sk}_1^{(i-1)} + \mathsf{PRNG}(i)$, $\mathsf{sk}_2^{(i)} = \mathsf{sk}_2^{(i-1)} - \mathsf{PRNG}(i)$. Therefore, in each session i the relation $\mathsf{sk} = \mathsf{sk}_1^{(i)} + \mathsf{sk}_2^{(i)}$ holds.

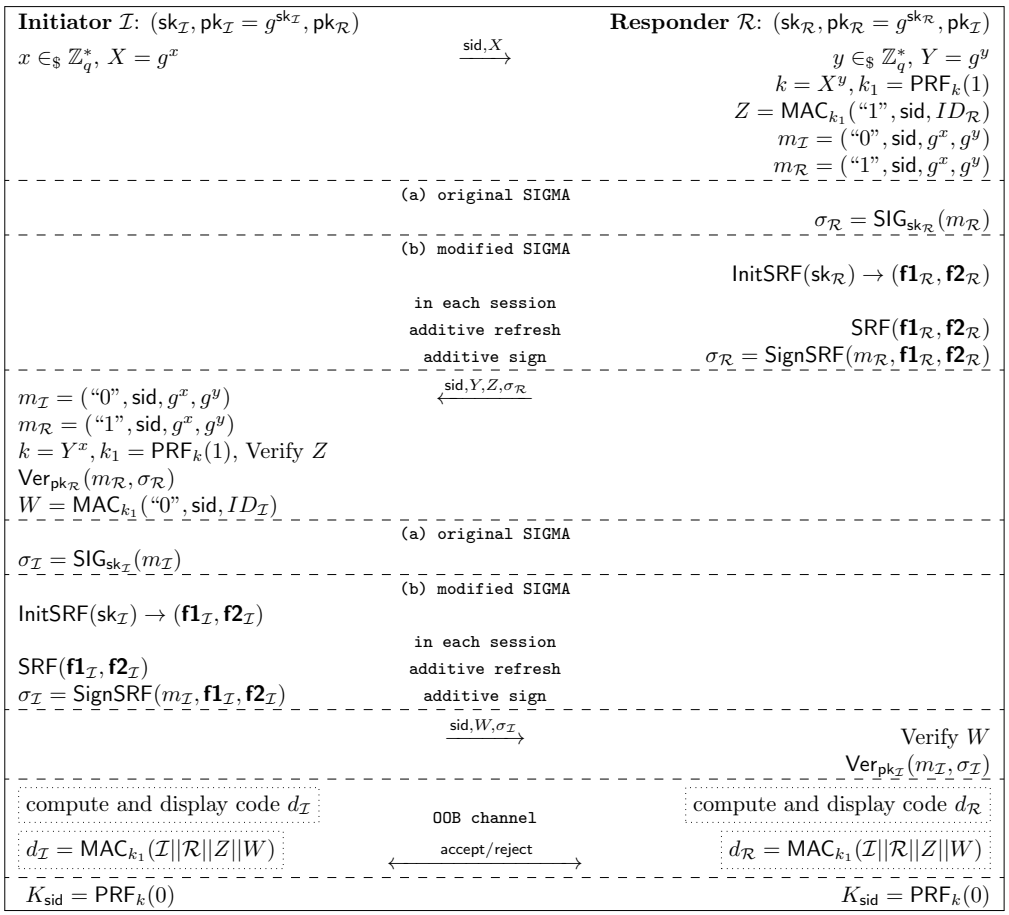

Figure 4.13: SIGMA with Schnorr signatures: a) original signature, b) additive split constituting protocol $\pi_{\mathsf{LR-AKE}}$. Keys are stored in two distinct SMs.

$\mathsf{SignRF}(m, \mathbf{f1}, \mathbf{f2}) \rightarrow \sigma = (s, r)$: in each i-th session, SM1 computes $k_1 \leftarrow_\$ \mathbb{Z}_q$ and $r_1 = g^{k_1} \bmod p$. Similarily, SM2 computes $k_2 \leftarrow_\$ \mathbb{Z}_q$ and $r_2 = g^{k_2} \bmod p$. Next, SM2 sends the commitment hash $\mathcal{H}(r_2)$ to SM1 which responds with $\mathcal{H}(r_2), r_1$. SM2 verifies the correctness of r_1 by checking if $r_1^q \bmod p$ equals to 1. If successful, SM2 computes $r = r_1 r_2 \bmod p$, $s_2 = k_2 + \mathsf{sk}_2^{(i)} \mathcal{H}(m, r) \bmod q$. SM2 then sends r_1, r_2 and s_2 to SM1. SM1 verifies $\mathcal{H}(r_2)$ and analogously compute $s_1 = k_1 + \mathsf{sk}_1^{(i)} \mathcal{H}(m, r) \bmod q$. In the final step one of the entities outputs the signature $\sigma = (s, r)$, where $s = s_1 + s_2 \bmod q$.

$\mathsf{Ver}_{\mathsf{pk}}(m, \sigma) \rightarrow 1/0$: Returns 1 for "accept", or 0 for "reject" respectively, where the verification procedure by checking that $g^s \stackrel{?}{=} r\mathsf{pk}^{\mathcal{H}(m,r)}$. Note

that $r\mathsf{pk}^{\mathcal{H}(m,r)} = g^{(k_1+k_2)} g^{(\mathsf{sk}_1^{(i)}+\mathsf{sk}_2^{(i)})\mathcal{H}(m,r)} \bmod p$.

In [130] the scheme is proved unforgeable against an adversary being able to get partial secrets from different modules in different signing sessions, provided that leakage of partial secret occurs after the key refresh, i.e. the scheme is unforgeable in the sense of Def. 39. We utilize this stronger unforgeability in the modified SIGMA with split Schnorr signatures proposed in Fig. 4.13, and we call it $\pi_{\mathsf{LR-AKE}}$.

4.5.7 Security Analysis

We provide the formal security analysis with corresponding proofs for our proposed improved scheme, using dual SMs and a key splitting pipeline methodology. As described in [130], an attacker may compromise both SM1 and SM2 given that honest key refreshes are made in-between sessions, however only one SM at the time can be compromised per signature session. We formulate an extension to the Canetti-Krawczyk key exchange model in which we prove impersonation and session key secrecy for our proposed schemes. Note that the regular SIGMA security is based on the unforgeabilty of the underlying signature used. Therefore by replacing the regular signature scheme, with the enhanced scheme secure in the stronger model, we obtain the stronger AKE protocol.

Modification Of the Canetti-Krawczyk Security Model

To reflect the possibility of the leakage from one SM we introduce the additional oracle "Partial-Key-Reveal" to the regular CK model.

Partial-Key-Reveal$(\mathcal{P}, session, signing\_module)$ This can be used only once in the session sid. If party $\mathcal{P}$ processed computation through SMs pipeline then partial results computations of SMs are returned, and the partial secret key stored in SM indicated by index *signing_module* is revealed. If $\mathcal{P}$ did not process computation through SMs pipeline it returns $\perp$.

Obviously this query is a wrapper for `SignRev Oracle` $\mathcal{O}_{\mathsf{SignRev}}(m, i)$, from our Sign Reveal Unforgeability (SR) model, which could be used for modular AKE protocols which use signature components. With this new query we define that a session sid is **exposed**, not only if the adversary $\mathcal{A}$ issues a *Session-Key-Reveal*, *State-Reveal* or *Corrupt* query for one of the session's parties, but also if $\mathcal{A}$ makes two partial reveal queries for one party in the same session sid, namely: **Partial-Key-Reveal**$(party, \mathsf{sid}, 1)$, **Partial-Key-Reveal**$(\mathcal{P}, \mathsf{sid}, 2)$ which would leak a complete secret key from that party.

	$\mathcal{A}$ against mutual identification	$\mathcal{A}$ against session key secrecy
$\mathcal{A}$ goal	Impersonating one party in front of the other party, without the full secret keys (only partial secrets can be leaked). This means impersonating: $\mathcal{I}$ in front of $\mathcal{R}$, and $\mathcal{R}$ in front of $\mathcal{I}$.	Learning the secret session key established in the protocol execution from either the device or communication part of protocol π.
$\mathcal{A}$ power	Active blocking adversary that can: block parties and intercept messages in a Man-in-the-Middle type of attack and extract partial secrets from an SM of parties except both partial secrets of one party in a single seession sid.	For not fully corrupted parties the adversary can exploit partial long term secret key leakage as in: active adversary in the protocol, and passive (observing) adversary
$\mathcal{A}$ oracles	Apart from the regular CK model oracles for an unexposed session s with uncorrupted parties, the active adversary can issue the following combination of queries: **Partial-Key-Reveal**$(\mathcal{I}, \mathsf{sid}, 1)$, **Partial-Key-Reveal**$(\mathcal{R}, \mathsf{sid}, 2)$, or **Partial-Key-Reveal**$(\mathcal{I}, \mathsf{sid}, 2)$, **Partial-Key-Reveal**$(\mathcal{R}, \mathsf{sid}, 1)$.	Adversary can execute oracles as above. The session key is protected by the secrecy of ephemeral values coined by the parties. Executing the *Real-or-Random*-game, issuing the **Test** oracle.
Forbidden combination of queries	Apart from the forbidden CK model oracles for an unexposed test session sid with uncorrupted parties, the active adversary cannot issue the following combination of queries: **Partial-Key-Reveal**$(\mathcal{I}, \mathsf{sid}, 1)$, **Partial-Key-Reveal**$(\mathcal{I}, \mathsf{sid}, 2)$, or **Partial-Key-Reveal**$(\mathcal{R}, \mathsf{sid}, 2)$, **Partial-Key-Reveal**$(\mathcal{R}, \mathsf{sid}, 1)$. These combination would simply reveal the whole secret of parties (as in the case of **Corrupt**$(\mathcal{I}, \mathsf{sid})$, **Corrupt**$(\mathcal{R}, \mathsf{sid})$ queries from CK model which reveal long term secrets) and make the attack trivial.	Adversary cannot execute any oracles that would reveal the ephemeral values, which are used to compute the session key. As in the regular CK model the adversary cannot reveal the *state* of the party - as this would reveal the ephemeral randomness od the parties device.

Figure 4.14: Aadversary's goal, power, and equivalent oracles in mCK model.

Definition 40. *A protocol π provides the session key security if for all adversaries $\mathcal{A}$ the following properties P1 and P2 holds:*

P1: if two uncorrupted parties $\mathcal{I}$ and $\mathcal{R}$, for which the queries were not issued: ***Partial-Key-Reveal***$(\mathcal{I}, \mathsf{sid}, 1)$, ***Partial-Key-Reveal***$(\mathcal{I}, \mathsf{sid}, 2)$, ***Partial-Key-Reveal***$(\mathcal{R}, \mathsf{sid}, 1)$, *and* ***Partial-Key-Reveal***$(\mathcal{R}, \mathsf{sid}, 2)$, *then complete a matching session* sid, *and correctly identifies peer party in that session, i.e. as $(\mathcal{I}, \mathsf{sid}, \mathcal{R})$ and $(\mathcal{R}, \mathsf{sid}, \mathcal{I})$ respectively, then the session key K output in these sessions is the same for $\mathcal{I}$ and $\mathcal{R}$ except for a negligible probability.*

P2: an adversary $\mathcal{A}$ succeeds in distinguishing the output from its test query with probability no more than $\frac{1}{2}$ plus a negligible fraction.

Security Against Impersonation and Session Key Secrecy

The security of the proposed protocols could be proved analogous to the proof of the original SIGMA protocol.

Theorem 17. *Under the the* DDH *assumption in* $\mathbb{G}$*, assuming the security of chosen cryptographic components* $\mathsf{PRF}_k(x)$*,* MAC_k*, and unforgeability of the underlying Schnorr split signature* $\mathsf{SignRF}(\mathsf{sk}, m)$*, with refresh function* RF *in each session realized via distinct* **f1** *and* **f2** *modules, the proposed modified 3-round SIGMA protocol from section 4.5.6 is secure in the sense of Def 40.*

Proof. The proof is analogous to the original proof from [36]. It consist of distinct proofs for property P1 and P2 respectively. As we modified the SIGMA scheme by substituting regular signatures with the split signatures with refresh mechanisms, only the proof for P1 requires appropriate adjustment and comments.

Proof of property P1. We strictly follow the security proof from [36]. Let SignRF denote the split signing with key refreshing. Let $\mathcal{A}$ denote the attacker having access to the allowed combination of oracles which also includes **Partial-Key-Reveal**. Let $\mathcal{I}$ and $\mathcal{R}$ denote two uncorrupted entities that complete matching sessions ($\mathcal{I}$, $\mathcal{R}$, sid, *initiator*) and ($\mathcal{I}, \mathcal{R}, \mathsf{sid}, \mathit{responder}$). In order to prove that $\mathcal{I}$ and $\mathcal{R}$ compute the same session key k_0 it suffices to show that both compute the same DH key g^{xy}, from which k_0 is derived. Let $u_\mathcal{I} = g^x$ denote the ephemeral key which $\mathcal{I}$ sends in the first message, and $v_P = g^y$ the ephemeral key which $\mathcal{I}$ receives in the second message. Similarly, we denote $u_\mathcal{R} = g^x$ and $v_\mathcal{R} = g^y$ as the keys which party $\mathcal{R}$ receives and sends, respectively. Now the signature produced by $\mathcal{R}$ is $\mathsf{SignRF}_{\mathsf{sk}_\mathcal{R}}(\mathsf{sid}, u_\mathcal{R}, v_\mathcal{R})$ and the signature verified by $\mathcal{I}$ is $\mathsf{SignRF}_{\mathsf{sk}_\mathcal{I}}(\mathsf{sid}, u_\mathcal{I}, v_\mathcal{I})$. As $\mathsf{SignRF}_{\mathsf{sk}_\mathcal{R}}(\mathsf{sid}, u_\mathcal{R}, v_\mathcal{R})$ is the only signature made by $\mathcal{R}$ over sid, so arguments to $\mathsf{SignRF}_{\mathsf{sk}_\mathcal{R}}(\mathsf{sid}, u_\mathcal{R}, v_\mathcal{R})$ and $\mathsf{SignRF}_{\mathsf{sk}_\mathcal{I}}(\mathsf{sid}, u_\mathcal{I}, v_\mathcal{I})$ have to be the same, or a valid signature over a different pair $u_\mathcal{I}, v_\mathcal{I}$ was forged by the adversary. In the latter case the adversary can be used to construct an effective forger $\mathcal{F}_{\mathsf{SR}}$ for SignRF in the SR model with partial key leakage (as of Def. 39). Thus, if the probability that $\mathcal{A}$ forges a valid a signature for the SignRF is negligible, then we have that $u_\mathcal{I} = u_\mathcal{R}$ and $v_\mathcal{I} = v_\mathcal{R}$ except for a negligible probability. Similarly, the signature produced by $\mathcal{I}$ is $\mathsf{SignRF}_{\mathsf{sk}_\mathcal{I}}(\mathsf{sid}, u_\mathcal{I}, v_\mathcal{I})$, and the signature verified by $\mathcal{R}$ is $\mathsf{SignRF}_{\mathsf{sk}_\mathcal{I}}(\mathsf{sid}, u_\mathcal{R}, v_\mathcal{R})$. As $\mathsf{SignRF}_{\mathsf{sk}_\mathcal{I}}(\mathsf{sid}, u_\mathcal{I}, v_\mathcal{I})$ is the only signature made by $\mathcal{I}$ over sid, so arguments to these signatures have to be the same,

or a valid signature over a different pair $u_{\mathcal{R}}$, $v_{\mathcal{R}}$ was forged by the adversary $\mathcal{F}_{\mathsf{SR}}$. Thus again, if $\mathcal{A}$ cannot forge a valid a signature for the SignRF in the SR model, then we have that $u_I = u_{\mathcal{R}}$ and $v_{\mathcal{I}} = v_{\mathcal{R}}$ except for a negligible probability. So, the DH value computed by $\mathcal{I}$ is $v_{\mathcal{I}}^x = v_{\mathcal{R}}^y = (g^y)^x$ and the DH value computed by $\mathcal{R}$ is $u_{\mathcal{R}}^y = u_{\mathcal{I}}^y = (g^x)^y$. Thus, the session keys for session sid computed by $\mathcal{I}$ and $\mathcal{R}$ are the same. □

Proof of property P2. Essentially the same as to proof of the property P2 (*Real-or-Random* indistinguishability) from [36]. □

These proofs summarize the overall proof for Thm. 17. □

Corollary 1: *Unless an adversary learns the both partial secrets of one party in a single session (which would enable reconstruction of the long term secret of that party), the proposed modified SIGMA protocol is immune against impersonation attacks.*
Corollary 2: *In order to hijack the session, impersonating some party of chosen identity, and make the peer party to complete the protocol in the accepting state, the active attacker should forge the underlying signature of the impersonated party, or extract both partial key from both SM in one session - which is the equivalent to the whole long term key leakage of that party.*

4.5.8 Conclusion

From our analysis we conclude that the proposed key splitting improvements will not increase computational complexity significantly. This indicates that minor algorithmic adjustments, given dual SM properties of the devices, is well justified for increased security against impersonation attacks analyzed in this paper. Our modified unforgeability and CK models allows formal security analysis against impersonation attacks. We argue that same approach could be applied to other AKE protocols with modular architectures, like BLS-HMQV [152].

As we note, $\pi_{\mathsf{LR-AKE}}$ provides security such that we properly address security challenges Sec3 and Sec5 since the impersonation attacks are addressed properly, and then leakage vulnerability is covered in the security model.

Chapter 5

Implementation

5.1 Lab Environment and Hardware

Specifically for this thesis, our proof of concept implementations were primarily developed for the STA research labs for information and communication technology, called *Trafikverket Labs* [153]. These facilities have their own network setup with sandbox and test environments and provide both hardware and computational capacity if needed to developers and researchers; its primary function is to *"...a research lab that exists to promote innovation and collaboration between researchers and companies."* [153]. In several papers produced during the research, we did experiments on different devices, ranging from modern laptops to smartphone devices. However, all relevant schemes have also been implemented and evaluated using the laboratory hardware. For comparison, our specifications differ from other IoT- and embedded hardware performance evaluations of pairing-based schemes, that has been published; although these were using much older hardware [133]. Their evaluation uses Raspberry Pie 1 and 2, and Intel ® Edison-based devices when running pairing-based crypto computations, with significantly slower hardware. Without revealing the manufacturers and other potentially sensitive data, we summarize the lab hardware used in Tab. 5.1. We note that the stated manufacturing year for D_{LAB} is 2021, but earlier versions exist and has been deployed in earlier projects, however not for cryptography purposes nor cybersecurity. The used lab equipment in this thesis is thus the latest available hardware laboratory equipment from STA.

For development of pairing-based schemes, there exists several programming libraries for a variety of programming languages. We have chosen Python as the primary language, built on a core crypto library written in C, namely MCL [120], with the binding library `mcl-python` [150]. The reason

ID	Type	Processor	RAM	Year
D_{LAB}	Lab hardware	1.0 GHz Quad-Core	4 GB	2021
D_{LT}	Laptop	3.2 Ghz Octa-Core	8 GB	2020

Table 5.1: Hardware specifications for the laboratory IoT device and computer equipment.

for choosing Python is due to its simplicity and good compatibility with the lab equipment. The lab devices had pre-installed Linux operating systems with Python 2.7 and Python 3. Also, several of our papers have used the same development setup, but for other type of devices. In our research we have explored different programming technologies such as WASM and python wrappers for MCL, but in order to align the performance analysis and have a better summary on the lab equipment, we have chosen to re-implement all schemes using the same library as mentioned above. The underlying MCL library provides the core cryptographic primitives such as optimal Ate pairings over Barreto-Naehrig and BLS12-381 curves. MCL is supported on several architectures, including x86 (32 and 64 bit) Windows and Linux, ARM, Android and iOS.

5.2 Complexity and Performance Analysis

5.2.1 Curve Selection

Curve selection in pairings is of paramount importance for several reasons. The security of pairing-based schemes heavily relies on the hardness of certain mathematical problems on the chosen curve, e.g., the BDHP. Some curves are susceptible to specific attacks that can compromise the hardness of these problems, making some of the schemes insecure. For instance, some curves such as the so called *supersingular* or *anomalous* curves, are problematic. The ECDLP on supersingular curves can be solved fast by employing the Menezes–Okamoto–Vanstone (MOV) attack [118]. This particular attack transforms the ECDLP on the elliptic curve into a more tractable DLP in a finite field. Therefore, if an elliptic curve's parameters are not chosen with care, ECDLP could be compromised.

The computational efficiency of pairing operations can vary depending on the curve. Some curves allow for faster arithmetic which would be crucial for real-world applications, especially in connected infrastructures and IoT-environments. The curve's *embedding degree* plays a significant role as it determines the size of the finite field $\mathbb{F}_q$ over which the pairing is com-

puted. This factor affects both the security and efficiency of the pairing, and a suitable curve must balance computational efficiency with the degree of security. As pairing-based cryptography drives towards more real-world applications, standardization of curves becomes essential. Using standardized curves ensures that systems can interoperate and that the security of the curve undergoes scrutiny by a broad community. Standardized curves such as NIST P-256, Curve25519, and secp256k1 have a widespread adoption and are generally considered secure against prevalent attacks. Nevertheless, curve selection must be approached with caution.

In our research we use the `BLS12 381` curve provided in the MCL library, and is part of the IETF Internet drafts for pairing-friendly curved [80]. The curve provides 128 bits of security and is provided by many other crypto libraires such as RELIC and MIRACL, and is also used in real-world applications such as Ethereum and zCash [80]. In our implentation we initialize the curve from the `mcl_init` function from the wrapper; this function also allows for choosing several other curve types.

```
1    import mcl
2    ml.mcl_init(mcl.CurveType.MCL_BLS12_381)
```

Listing 5.1: Initialization of curves in MCL.

5.2.2 Hash-to-Group Computations

Hashing an input to a secure group element, particularly in the context of elliptic curve cryptography, involves several complex aspects. The process is computationally intensive and directly impacts the performance of cryptographic protocols. The hashing function must ensure deterministic and uniform mapping, meaning it consistently produces the same output for a given input while ideally distributing these inputs across the group elements uniformly. Additionally, the implementation must be resistant to side-channel attacks, mitigating vulnerabilities that could be exploited from external observations like power consumption patterns. The `hashAndMapTo` function provided in the MCL-library is used for any secure hash function $\mathcal{H}$. The function `hashAndMapTo` maps its output to a group element in $\mathbb{G}$ (either $\mathbb{G}_1$ or $\mathbb{G}_2$). For example, given a message m_1 that is a vector of values b_1, v_1, it would be hashed into a group element as shown in Lst. 5.2:

```
1    m1 = [b_1, v_1]
2    m1hash = G1.hashAndMapToG1(bytes(m1))
```

Listing 5.2: Hash-to-Group example in MCL.

For asymmetric pairings, some optimizations in hashing into groups can be realized, and much is dependent on the selected curves [64]. The MCL library also supports the hash to curve implementation as described in IETF draft *Hashing to Elliptic Curves* [62].

5.2.3 Performance Analysis

In this section we will provide the results from all proof of concept implementations, executed on the D_{LAB} and D_{LT} devices. However, we will give selected code snippets here along with the performance results to better discuss the experiments. Since the proposed schemes are different in how they work it is not feasible to make a one-to-one comparison between them. Instead, we measure the different sub protocols for each scheme. The collected measurements in our experiments are shown in Tab. 5.2, and fundamental operations such as exponentiation, multiplication, hashing and more are summarized in Tab. 5.3. All performance measurements are run in iterations of 1000 rounds, where the mean execution time is noted in Tab. 5.2 and 5.3. The reason is to get as accurate measurements as possible since the underlying operating system may affect the computations due to running background processes. A typical performance test of an operation is executed as shown in Lst. 5.3. The `time()` function return seconds, hence we do not need to divide with 1000 after each run since we want to express the measure in milliseconds (which would require a multiplication of 1000).

```
1    time_start = time.time()
2        for i in range(1000):
3            g1 * x
4    time_stop = time.time()
5    print("G1 mul: ", f'{(time_stop-time_start):.10f}')
```

Listing 5.3: Performance testing of multiplication in group $\mathbb{G}_1$.

We note that addition and multiplication are within groups $\mathbb{G}_1$ and $\mathbb{G}_2$; with the multiplicative notation it means that multiplication in $\mathbb{G}$ means g^x for $g \in \mathbb{G}$ and $x \in \mathbb{Z}_q^*$. Exponents are in $\mathbb{G}_T$, i.e., the measure of `Exponentiation` is on value y^x for $y \in \mathbb{G}_T$ (after a pairing operation) and $x \in \mathbb{Z}_q^*$. We also provide a difference calculation for each operation, $\Delta = \frac{\mathsf{D}_{LAB}}{\mathsf{D}_{LT}}$ which is the approximate number of times slower the execution is on the lab equipment compared to a typical (2020) laptop.

Other implementation considerations to mention are the symmetric and asymmetric encryption algorithms used. In paper $\mathsf{P_I}$ and $\mathsf{P_{IV}}$ we are using encryption/decryption as primitives, where our choice of implementation is 2048-bit RSA and 256-bit AES symmetric encryption in the tests. In the

proof of concepts we used the Python `cryptography.hazmat` library for both algorithms. When generating the symmetric AES keys we used the operating systems underlying (secure) randomness source, namely the `dev/urandom` interface since both D_{LAB} and D_{LT} are running on Linux operating systems. This is also used when generating random values in $\mathbb{Z}_q^*$ in MCL using the `setByCSPRNG` since the library reads directly from `dev/urandom`. These operations are shown in the below Python code:

```
aes_key = os.urandom(32) # generate 32*8 bits
rand = Fr()
rand.setByCSPRNG()       # generate 120*8 bits by default
```

Listing 5.4: Generating randomness from dev/urandom in both native Python and the MCL library.

5.2.4 Feasibility Analysis

In a feasibility analysis of cryptographic schemes, computational efficiency is fundamental, involving benchmarking of key operations like pairings, hashing, encryption, decryption, and the ones mentioned in Tab. 5.3. Implementation challenges must also be addressed, whereas scalability is also a vital factor, determining how well the schemes can handle an increase of nodes (or devices). Moreover, cost implications, related to both software development and maintenance expenses, are analyzed to understand the financial viability of implementing our schemes. In this section we thus elaborate on these dimensions of feasibility, arguing how well our proof of concepts align with future real-world implementations.

Performance

From Tab. 5.3 we see that the difference of execution time is very large between the lab equipment and a modern laptop. This is to be expected but we note that the performance on D_{LAB} still lies within fast execution (fractions of milliseconds). For example, a pairing which is one of the most costly operations is still in 5.755 ms which is very fast in the context of C-ITS; the maximum sending time between vehicles are 500 ms [157] which implies that running a secure protocol with computations taking up to < 60 ms (up to the equivalent of 10 pairing operations) still gives 440 ms marginal in sending the message. Haidar et al. provide experiments of computing and exchanging pseudonyms between running vehicles, and the complete exchange protocol following the ETSI standard with TG3 profile (secure communication profile) was executing just above 600 ms [71]. Hence, we conclude that the running

Scheme	**Complexity**	D_{LAB} (ms)	D_{LT} (ms)
π_{PSC}			
Id	$\mathsf{R}+2\mathsf{M}+\mathsf{H}$	2.258	0.002
Ver	$2\mathsf{P}+\mathsf{A}+\mathsf{M}+\mathsf{H}$	12.498	0.012
SC	$\mathsf{R}+\mathsf{A}+4\mathsf{M}+\mathsf{H}+\mathsf{E}_{\mathsf{AES}}$	3.848	0.003
USC	$3\mathsf{P}+2\mathsf{A}+4\mathsf{M}+2\mathsf{H}+\mathsf{D}_{\mathsf{AES}}$	24.767	0.024
π_{MPAE}			
UKeyGen	$\mathsf{R}+\mathsf{M}$	0.637	6.33×10^{-4}
IPKeyGen	$\mathsf{R}+\mathsf{M}$	0.755	7.92×10^{-4}
PKeyGen	$2\mathsf{M}$	0.669	7.14×10^{-4}
AEn ($\pi_{\mathsf{MPAE}-1}$)	$\mathsf{n}(\mathsf{R}+4\mathsf{A}+7\mathsf{M}+\mathsf{H}+\mathsf{E}_{\mathsf{XOR}})$	4.355	0.005
AEn ($\pi_{\mathsf{MPAE}-2}$)	$\mathsf{n}(\mathsf{R}+4\mathsf{A}+7\mathsf{M}+\mathsf{H}+\mathsf{E}_{\mathsf{XOR}})$	4.271	0.005
MA	nA	0.009	0.06×10^{-4}
$\mathsf{MA}_{n=100}$	”	1.234	8.62×10^{-4}
$\mathsf{MA}_{n=1000}$	”	12.389	0.008
$\mathsf{MA}_{n=10000}$	”	124.723	0.085
ADe ($\pi_{\mathsf{MPAE}-1}$)	$\mathsf{P}(\mathsf{n}+1)+3\mathsf{nA}+3\mathsf{nM}+\mathsf{nH}+\mathsf{nD}_{\mathsf{XOR}}$	14.002	0.014
$\mathsf{ADe}_{n=100}$	”	829.508	0.891
$\mathsf{ADe}_{n=1000}$	”	8228.272	8.803
$\mathsf{ADe}_{n=10000}$	”	82221.243	87.583
ADe ($\pi_{\mathsf{MPAE}-2}$)	$2\mathsf{P}+\mathsf{A}(3+3\mathsf{n})+2\mathsf{nM}+\mathsf{H}+\mathsf{nD}_{\mathsf{XOR}}$	13.959	0.014
$\mathsf{ADe}_{n=100}$	”	251.522	0.302
$\mathsf{ADe}_{n=1000}$	”	2388.629	2.902
$\mathsf{ADe}_{n=10000}$	”	23746.130	28.737
π_{KeySplit}			
InitRF	$\mathsf{R}+\mathsf{A}$	0.014	0.07×10^{-4}
RF	$2\mathsf{R}$	0.020	0.10×10^{-4}
SignRF	$\mathsf{A}+2\mathsf{M}+2\mathsf{H}$	2.283	0.002
Ver	$2\mathsf{P}+\mathsf{H}$	12.372	0.012
π_{SHP}			
RingSign	$\mathsf{n}(\mathsf{A}+\mathsf{M}+\mathsf{H})$	1.659	0.173
$\mathsf{RingSign}_{n=100}$	”	166.092	16.494
$\mathsf{RingSign}_{n=1000}$	”	1663.893	165.314
$\mathsf{RingSign}_{n=10000}$	”	16612.242	1651.473
RingVerify	$\mathsf{n}(\mathsf{P}+\mathsf{A}+\mathsf{H})+\mathsf{P}$	12.560	1.228
$\mathsf{RingVerify}_{n=100}$	”	688.685	66.927
$\mathsf{RingVerify}_{n=1000}$	”	6815.613	663.708
$\mathsf{RingVerify}_{n=10000}$	”	68109.302	6637.584
$\mathsf{SourceHiding}_{\pi_{\mathsf{SHP}}}$	$\mathsf{n}(\mathsf{E}_{\mathsf{RSA}}+\mathsf{D}_{\mathsf{RSA}})+\mathsf{R}$	8.138	1.285
$\mathsf{SourceHiding}_{\pi_{\mathsf{SHP}},n=100}$	”	802.689	111.301
$\mathsf{SourceHiding}_{\pi_{\mathsf{SHP}},n=1000}$	”	8000.202	1109.805
$\mathsf{SourceHiding}_{\pi_{\mathsf{SHP}},n=10000}$	”	79969.867	11130.922
$\pi_{\mathsf{LR-AKE}}$			
Sign	$2(\mathsf{R}+\mathsf{A}+\mathsf{M}+\mathsf{H})$	2.355	0.002
Ver	$2\mathsf{P}+\mathsf{H}$	12.119	0.012

Table 5.2: Summary of the proposed schemes' running times. R is generating a number in $\mathbb{Z}_q$, P is pairing, A, M and H are addition, multiplication (denoted g^x for $g \in \mathbb{G}$ and $x \in \mathbb{Z}_q$) and hashing in group $\mathbb{G}$, and $\mathsf{E}_i, \mathsf{D}_i$. for encryption and decryption using scheme i.

Operation	**Run-time D_{LAB}**	**Run-time D_{LT}**	Δ
`Addition` ($\mathbb{G}_1$)	0.008	6.678 $\times 10^{-4}$	12
`Addition` ($\mathbb{G}_2$)	0.012	9.951 $\times 10^{-4}$	12
`Multiplication` ($\mathbb{G}_1$)	0.351	0.061	6
`Multiplication` ($\mathbb{G}_2$)	1.152	0.108	11
`Exponentiation` ($\mathbb{G}_T$)	0.194	0.026	8
`Hash` ($\mathbb{Z}_q^*$)	0.006	2.512 $\times 10^{-4}$	24
`HashToGroup` ($\mathbb{G}_1$)	0.868	0.090	10
`HashToGroup` ($\mathbb{G}_2$)	1.793	0.189	9
`Pairing`	5.755	0.563	10
`Rand` ($\mathbb{Z}_q^*$)	0.006	4.599 $\times 10^{-4}$	13
`BLS` (sign)	1.501	0.151	10
`BLS` (verify)	12.372	1.212	10
`AES Encryption` (256 bit)	0.290	0.054	5
`AES Decryption` (256 bit)	0.163	0.015	11
`RSA Encryption` (2048 bit)	0.271	0.038	7
`RSA Decryption` (2048 bit)	6.094	1.077	6

Table 5.3: Summary of the fundamental operations used in our proposed schemes, all tests are measured in milliseconds (ms). Δ shows the approximate number of times faster D_{LT} is compared to D_{LAB}, each Δ is computed as $\frac{\mathsf{D}_{LAB}}{\mathsf{D}_{LT}}$.

times of our protocols lies significantly within reasonable performance on the provided lab equipment.

Scalability

We see clearly that protocols π_{MPAE} and π_{SHP} scales linearly in the sub-procedures when adding nodes. When computing on $n = 100$ participating nodes we are well within reasonable timings for the scaled sub-procedures MA and ADe: 1.234+14.002 = 18.357 ms for $\pi_{\mathsf{MPAE}^{-1}}$ and 1.234+13.959 = 15.193 ms for $\pi_{\mathsf{MPAE}^{-1}}$. Given the remaining sub-procedures to be run on one device the complete running times are still around 25 ms. In the case of π_{SHP} we argue that the protocol should in practice never handle more than 100 participants since the privacy requirement of anonymity is connected to the server, in the vehicle case that would be a RSU. This implies that there is a geographically limited area where the server (RSU) operates, typically a traffic light conjunction, a tunnel or road segment. In such areas it is unreasonable to have several thousand vehicles in a short-range communication

setup. Regardless, the bottleneck of the protocol is more likely to be the PKI management and using the RSA certificates. From that perspective the protocol would not scale optimaly, but again, operating in an environment with less than 100 vehicles should not be problematic given the computational efficiency.

Ease of Implementation

Since the additional implementation of the protocols were in Python there was a clear benefit in terms of ease of implementation, compared to C programming which require a higher degree of complexity; studies has shown that Python clearly has benefits over C in terms of learning and adopting the programming language [15, 158], and that Python in particular is suitable for learning algorithms [41]. We draw the conclusion that such benefits in learning Python over C, makes the programmer to be able to focus on the algorithmic parts of the protocols instead of having to handle explicit memory management and other complex tasks as in C/C++. This in turn could imply that the implementer of such secure protocols does not need to be highly specialized, which would be cost efficient and less riskier (personal dependency of a few selected experts) for the company. On the other hand, one could argue that deep knowledge of the underlying cryptography is necessary for secure and stable implementation in terms of trouble shooting or knowing that each line of code is really reflecting part of the protocol. However, this could be mitigated in the same manner as the MCL library is providing fundamental pairing operations, namely if we build specific protocol libraries that programmers can use. Building such library (with the built-in dependency of MCL), there would be yet another level of easiness of implementing the protocols. In that case, such library could provide procedures in the API, e.g., RingSign or RingVerify.

Chapter 6

Conclusion

6.1 Future Work and Improvements

Our research has provided a framework for enhancing the security of cryptographic schemes, particularly in the context of weak devices in connected infrastructures. Moving forward, several areas of interest are subject for deeper exploration of our work. Identifying and addressing vulnerabilities in additional cryptographic schemes susceptible to ephemeral leakage, would be a natural next step. Especially regarding already published research that contains vulnerable schemes suited for C-ITS or similar environments.

We have provided security proofs in the asymmetric configuration of pairings. We suggest further expansion on the security analysis, finding more security models in other type of environments, e.g., maritime and unmanned aerial vehicle (UAV) transportation scenarios. That expansion would also ensure the applicability of the utilization of our security enhancements across a wider spectrum of security models. Furthermore, the scalability and adaptability of our schemes would require up-scaled testing across various low-powered devices beyond the provided laboratory equipment. Specifically, implementing these schemes in prototype or on-site VANET communication systems, would offer a good opportunity to assess their performance in a more real-world environment. By evaluating bandwidth latency and communication complexity in moving nodes, we would be able to expand our research further to meet the specific demands of mobile communication networks.

6.2 Conclusions

In this final part of the thesis, we will conclude the aggregated result of the research and outline the resolution of RQ1 and RQ2 along with the hypotheses

HT1, HT2. Moreover, we will also make final concluding remarks that shows how goals GL1-GL3 are fulfilled.

6.2.1 Fulfillment of Goals and Hypothesis

Each proposed solution, as in papers $\mathsf{P_I}$-$\mathsf{P_V}$ has provided both a formal security analysis (showing the theoretical security) and a proof of concept implementation (showing the industrial feasibility). Even further, we provided cryptanalysis in three cases - two under a stronger security model, and one in the current model - illustrating the vulnerabilities in previous schemes. Our results thus indicate that the theoretical research, complemented by the practical proofs of performance elaborated in Sec. 5.2.4, closes the gap between an academic view and the industrial perspective of the security challenges Sec1-Sec5; the security analysis contributes to the scientific community by presenting enhanced cryptographic schemes, and the implementation part contributes to the industry by providing proof of concept results that can be used for further development at testing sites or laboratories.

We recall the formulated goals for this thesis:

GL1: *Propose how to mitigate and improve secure schemes that are vulnerable to weak devices and impersonation attacks, and can be used in the domain of connected infrastructure and cooperative intelligent transportation systems.*

GL2: *Implement and carefully evaluate the proposed schemes from a performance perspective, aligned with relevant hardware for the connected infrastructure and cooperative intelligent transportation systems domain.*

GL3: *Address the major security challenges identified in this thesis with applicable and theoretically sound solutions. The specific security challenges are denoted* Sec1-Sec5 *and further described in Sec. 2.1.5.*

We have provided cryptanalysis of three schemes (in papers $\mathsf{P_I}$, $\mathsf{P_{II}}$ and $\mathsf{P_V}$) and shown how to mitigate the found vulnerabilities. Moreover, each proposed scheme in all papers $\mathsf{P_I}$-$\mathsf{P_V}$ are proven secure in stronger security models considering weak devices and impersonation attacks. Therefore:

Conclusion 1. *We have successfully reached* GL1 *since our proposed schemes mitigate found weaknesses in previously published schemes, and are proven secure to impersonation attacks in weak devices using stronger security models, particularly suitable for C-ITS environments.*

Next, we have provided proof of concept implementations to all proposed schemes along with performance evaluations in both laboratory equipment used for C-ITS purposes, and in research computers for comparison. Therefore:

Conclusion 2. *We have successfully reached* GL2 *due to the provided proof of concept implementations and performance analysis.*

The security challenges are addressed accordingly as specified in Tab. 2.1, shown in each scenario described in chapter 4. Not only does the provided research cover each challenge, but the papers overlap multiple challenges as well. Therefore:

Conclusion 3. *We have successfully reached* GL3 *due to the published research, provided by the scenarios corresponding to* $\mathsf{P_I}$-$\mathsf{P_V}$.

All three goals are therefore reached, hence we have provided both theoretical and practical improvements, namely formal security analysis and implementations. Not only did the research output contain theoretical analysis proving the security and proposing new, stronger models, but also implementing these results in two different devices for practical analysis. This analysis make us draw the following conclusion:

Conclusion 4. *We have bridged the gap between theory and practice in our research due to the proof of concept implementations, hence we accept* HT1.

The feasibility analysis alone in Sec. 5.2.4 make us draw the following conclusion:

Conclusion 5. *We have shown the feasibility of the proposed schemes in terms of security and efficiency, hence we accept* HT2.

6.2.2 Answering the Research Questions

Finally, we recall the research questions formulated for this thesis:

RQ1: *How can we mitigate and improve vulnerable signature and authentication schemes subject to weak devices (low-powered and vulnerable against ephemeral injection/leakage), to be used in the connected infrastructure domain?*

RQ2: *What level of feasibility and performance will the improved and proposed schemes have, i.e. are they suitable for real-world implementations?*

Given our analysis up til this point, summarized in the previous conclusions above, we formulate two separate answers in the subsequent sections.

Answer to RQ1

As shown in the cryptanalysis, we were able to break previously published schemes. At the same time we also tweaked and proposed new schemes in stronger security models where an attacker is able to exploit the device by injecting or stealing ephemeral values. Moreover, all scenarios described plausible applications for these schemes in connected infrastructure systems, especially in the context of C-ITS. Hence, our proposed schemes (and architectures) using the security enhancing techniques such as exponentiation, key split and key refresh mechanisms, provides academia and industry with stronger and secure signature and authentication schemes. This proposition is strengthen as a result of accepting HT1, namely that the gap between theory and practice in our context is bridged.

However, we note that these theoretical results are within the formulated security models, suitable for connected infrastructures with weak devices, hence there might be environments where other security models apply. Therefore, there is no guarantee of generalization in the sense of using these schemes in contexts where the security models use different assumptions or requirements. Although our security models are strong in the sense of handling leakage of ephemeral session keys and randomness, there might be other type of attacks with sophisticated ways of breaking underlying assumptions.

Answer to RQ2

From the feasibility analysis and proof of concept evaluations we see clearly that modern hardware, even for weak devices, have the ability to run our proposed schemes. We even show the scalability for the schemes where applicable. Since the tests are run on STA lab equipment we can only conclude that future hardware will be equally good or better, hence real-world deployment of the proposed solutions is possible. We note that although the performance is within good marginals, there might be unforeseen issues with future standardization bodies that might limit what type of cryptography primitives will be allowed. Moreover, adopting technology, especially in the context of C-ITS on the European market, involves many stakeholders such as governments and the vehicle industry. Therefore, accepting new standards or investing in new technology is also a political and financial issue where the complexity even increases due to combinations of national policy and European commission guidelines. Regardless of future directions of the standardization bodies, our results can be helpful for any technology readiness level assessment necessary.

Bibliography

[1] IEEE Standard for Information technology– Local and metropolitan area networks– Specific requirements– Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 6: Wireless Access in Vehicular Environments. IEEE Std 802.11p-2010 (Amendment to IEEE Std 802.11-2007 as amended by IEEE Std 802.11k-2008, IEEE Std 802.11r-2008, IEEE Std 802.11y-2008, IEEE Std 802.11n-2009, and IEEE Std 802.11w-2009) pp. 1–51 (2010). https://doi.org/10.1109/IEEESTD.2010.5514475

[2] Cyber Security and Resilience of Intelligent Public Transport Good practices and recommendations. `www.enisa.europa.eu/publications/good-practices-recommendations` (2015), accessed: 2023-04-10

[3] IEEE Guide for Wireless Access in Vehicular Environments (WAVE) Architecture. IEEE Std 1609.0-2019 (Revision of IEEE Std 1609.0-2013) pp. 1–106 (2019). https://doi.org/10.1109/IEEESTD.2019.8686445

[4] Swedish Transport Administration: ITS. `https://bransch.trafikverket.se/en/startpage/operations/Operations-road/ITS/` (2022), accessed: 2022-05-10

[5] Swedish Transport Administration. `https://bransch.trafikverket.se/en/startpage/` (2023), accessed: 2023-03-03

[6] Akavia, A., Goldwasser, S., Vaikuntanathan, V.: Simultaneous hardcore bits and cryptography against memory attacks. In: Reingold, O. (ed.) Theory of Cryptography. pp. 474–495. Springer Berlin Heidelberg, Berlin, Heidelberg (2009)

[7] Akinyele, J.A., Garman, C., Miers, I., Pagano, M.W., Rushanan, M., Green, M., Rubin, A.D.: Charm: a framework for rapidly prototyping cryptosystems. Journal of Cryptographic Engineering **3**(2), 111–128 (2013)

[8] Al-Riyami, S.S., Paterson, K.G., et al.: Certificateless public key cryptography. In: Asiacrypt. vol. 2894, pp. 452–473. Springer (2003)

[9] Al-Zubi, M., Abu-Shareha, A.A.: Efficient Signcryption Scheme Based on El-Gamal and Schnorr. Multimedia Tools Appl. **78**(9), 11091?11104 (May 2019). https://doi.org/10.1007/s11042-018-6636-7, https://doi.org/10.1007/s11042-018-6636-7

[10] Ali, I., Hassan, A., Li, F.: Authentication and privacy schemes for vehicular ad hoc networks (vanets): A survey. Vehicular Communications **16**, 45–61 (2019). https://doi.org/https://doi.org/10.1016/j.vehcom.2019.02.002, https://www.sciencedirect.com/science/article/pii/S2214209618303192

[11] Alimohammadi, M., Pouyan, A.A.: Sybil attack detection using a low cost short group signature in vanet. In: 2015 12th International Iranian Society of Cryptology Conference on Information Security and Cryptology (ISCISC). pp. 23–28. IEEE (2015)

[12] Alwen, J., Dodis, Y., Wichs, D.: Leakage-Resilient Public-Key Cryptography in the Bounded-Retrieval Model. In: Proceedings of the 29th Annual International Cryptology Conference on Advances in Cryptology. p. 36?54. CRYPTO '09, Springer-Verlag, Berlin, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03356-8_3, https://doi.org/10.1007/978-3-642-03356-8_3

[13] Alwen, J., Dodis, Y., Wichs, D.: Leakage-resilient public-key cryptography in the bounded-retrieval model. In: Halevi, S. (ed.) Advances in Cryptology - CRYPTO 2009. pp. 36–54. Springer Berlin Heidelberg, Berlin, Heidelberg (2009)

[14] Alwen, J., Dodis, Y., Wichs, D.: Leakage-Resilient Public-Key Cryptography in the Bounded-Retrieval Model. In: Halevi, S. (ed.) Advances in Cryptology - CRYPTO 2009. pp. 36–54. Springer Berlin Heidelberg, Berlin, Heidelberg (2009)

[15] Balreira, D.G., Silveira, T.L.d., Wickboldt, J.A.: Investigating the impact of adopting python and c languages for introductory engineering programming courses. Computer Applications in Engineering Education **31**(1), 47–62 (2023)

[16] Bellare, M., Fischlin, M., Goldwasser, S., Micali, S.: Identification Protocols Secure Against Reset Attacks (2000), http://eprint.iacr.or

g/2000/015, extended abstract appeared in proceedings of Eurocrypt 2001. This is the full version. mihir@cs.ucsd.edu 11585 received 28 Apr 2000, last revised 20 Sep 2001

[17] Bellare, M., Neven, G.: Multi-signatures in the plain public-key model and a general forking lemma. In: Proceedings of the 13th ACM Conference on Computer and Communications Security. p. 390–399. CCS '06, Association for Computing Machinery, New York, NY, USA (2006). https://doi.org/10.1145/1180405.1180453, https://doi.org/10.1145/1180405.1180453

[18] Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: Proceedings of the 1st ACM Conference on Computer and Communications Security. pp. 62–73 (1993)

[19] Bellare, M., Rogaway, P.: Introduction to modern cryptography (2005)

[20] Bhatia, V., Jaglan, V., Kumawat, S., Siwach, V., Sehrawat, H.: Intelligent transportation system applications: A traffic management perspective. In: Raj, J.S., Palanisamy, R., Perikos, I., Shi, Y. (eds.) Intelligent Sustainable Systems. pp. 419–433. Springer Singapore, Singapore (2022)

[21] Boldyreva, A.: Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme. In: Desmedt, Y.G. (ed.) Public Key Cryptography — PKC 2003. pp. 31–46. Springer Berlin Heidelberg, Berlin, Heidelberg (2002)

[22] Boldyreva, A.: Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme. In: Public Key Cryptography - PKC 2003, 6th International Workshop on Theory and Practice in Public Key Cryptography, Miami, FL, USA, January 6-8, 2003, Proceedings. pp. 31–46 (2003). https://doi.org/10.1007/3-540-36288-6_3, https://doi.org/10.1007/3-540-36288-6_3

[23] Boldyreva, A., Palacio, A., Warinschi, B.: Secure Proxy Signature Schemes for Delegation of Signing Rights. Journal of Cryptology **25** (06 2003). https://doi.org/10.1007/s00145-010-9082-x

[24] Boneh, D., Drijvers, M., Neven, G.: Compact multi-signatures for smaller blockchains. In: Advances in Cryptology - ASIACRYPT

2018 - 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2-6, 2018, Proceedings, Part II. pp. 435–464 (2018). https://doi.org/10.1007/978-3-030-03329-3_15, https://doi.org/10.1007/978-3-030-03329-3_15

[25] Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. In: Kilian, J. (ed.) Advances in Cryptology — CRYPTO 2001. pp. 213–229. Springer Berlin Heidelberg, Berlin, Heidelberg (2001)

[26] Boneh, D., Gentry, C., Lynn, B., Shacham, H.: Aggregate and verifiably encrypted signatures from bilinear maps. In: Biham, E. (ed.) Advances in Cryptology — EUROCRYPT 2003. pp. 416–432. Springer Berlin Heidelberg, Berlin, Heidelberg (2003)

[27] Boneh, D., Lynn, B., Shacham, H.: Short signatures from the weil pairing. In: Boyd, C. (ed.) Advances in Cryptology — ASIACRYPT 2001. pp. 514–532. Springer Berlin Heidelberg, Berlin, Heidelberg (2001)

[28] Boneh, D., Shoup, V.: A graduate course in applied cryptography. Draft 0.5 (2020)

[29] Bouakkaz, S., Semchedine, F.: A certificateless ring signature scheme with batch verification for applications in vanet. Journal of Information Security and Applications **55**, 102669 (2020). https://doi.org/https://doi.org/10.1016/j.jisa.2020.102669, https://www.sciencedirect.com/science/article/pii/S2214212620308218

[30] Brakerski, Z., Kalai, Y.T., Katz, J., Vaikuntanathan, V.: Overcoming the Hole in the Bucket: Public-Key Cryptography Resilient to Continual Memory Leakage. In: 2010 IEEE 51st Annual Symposium on Foundations of Computer Science. pp. 501–510 (2010). https://doi.org/10.1109/FOCS.2010.55

[31] C-Roads: C-roads - the platform of harmonised c-its deployment in europe. https://www.c-roads.eu (2022), accessed: 2022-05-14

[32] Cahyadi, E.F., Hwang, M.S.: A comprehensive survey on certificateless aggregate signature in vehicular ad hoc networks. IETE Technical Review **39**(6), 1265–1276 (2022). https://doi.org/10.1080/02564602.2021.2017800, https://doi.org/10.1080/02564602.2021.2017800

[33] Camenisch, J., Kohlweiss, M., Rial, A., Sheedy, C.: Blind and anonymous identity-based encryption and authorised private searches on public key encrypted data. In: Jarecki, S., Tsudik, G. (eds.) Public Key Cryptography – PKC 2009. pp. 196–214. Springer Berlin Heidelberg, Berlin, Heidelberg (2009)

[34] Canetti, R., Dodis, Y., Halevi, S., Kushilevitz, E., Sahai, A.: Exposure-resilient functions and all-or-nothing transforms. In: EUROCRYPT (2000)

[35] Canetti, R., Krawczyk, H.: Analysis of key-exchange protocols and their use for building secure channels. In: Advances in Cryptology—EUROCRYPT 2001: International Conference on the Theory and Application of Cryptographic Techniques Innsbruck, Austria, May 6–10, 2001 Proceedings 20. pp. 453–474. Springer (2001)

[36] Canetti, R., Krawczyk, H.: Security analysis of ike's signature-based key-exchange protocol. Cryptology ePrint Archive, Report 2002/120 (2002)

[37] Chari, S., Jutla, C.S., Rao, J.R., Rohatgi, P.: Towards sound approaches to counteract power-analysis attacks. In: Wiener, M. (ed.) Advances in Cryptology — CRYPTO' 99. pp. 398–412. Springer Berlin Heidelberg, Berlin, Heidelberg (1999)

[38] Charpentier, V., Slamnik-Krijestorac, N., Marquez-Barja, J.: Latency-aware c-its application for improving the road safety with cam messages on the smart highway testbed. In: IEEE INFOCOM 2022-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS). pp. 1–6. IEEE (2022)

[39] Chaum, D.L.: Untraceable electronic mail, return addresses, and digital pseudonyms. Commun. ACM **24**(2), 84–90 (feb 1981). https://doi.org/10.1145/358549.358563, `https://doi.org/10.1145/358549.358563`

[40] Chen, S., Hu, J., Shi, Y., Peng, Y., Fang, J., Zhao, R., Zhao, L.: Vehicle-to-everything (v2x) services supported by lte-based systems and 5g. IEEE Communications Standards Magazine **1**, 70–76 (01 2017). https://doi.org/10.1109/MCOMSTD.2017.1700015

[41] Chou, P.H.: Algorithm education in python. Proceedings of Python **10**, 177–185 (2002)

[42] Commission, I.E., et al.: Iec 62443: Industrial communication networks—network and system security. IEC Central Office: Geneva, Switzerland (2010)

[43] Conference of European Directors of Roads: CEDR Transnational Road Research Programme Call 2022: Data. https://www.cedr.eu/docs/view/628de7c042f66-en (2019), accessed: 2023-01-11

[44] Council of European Union: Council regulation (EU) no 2010/40 (2010)

[45] Dai, C., Xu, Z.: Pairing-free certificateless aggregate signcryption scheme for vehicular sensor networks. IEEE Internet of Things Journal **10**(6), 5063–5072 (2022)

[46] Dent, A.W., Zheng, Y. (eds.): Practical Signcryption. Information Security and Cryptography, Springer (2010). https://doi.org/10.1007/978-3-540-89411-7, https://doi.org/10.1007/978-3-540-89411-7

[47] Di Crescenzo, G., Lipton, R., Walfish, S.: Perfectly secure password protocols in the bounded retrieval model. In: Halevi, S., Rabin, T. (eds.) Theory of Cryptography. pp. 225–244. Springer Berlin Heidelberg, Berlin, Heidelberg (2006)

[48] Dodis, Y., Haralambiev, K., Lopez-Alt, A., Wichs, D.: Cryptography against Continuous Memory Attacks. In: 2010 IEEE 51st Annual Symposium on Foundations of Computer Science. pp. 511–520 (2010). https://doi.org/10.1109/FOCS.2010.56

[49] Dodis, Y., Haralambiev, K., López-Alt, A., Wichs, D.: Efficient public-key cryptography in the presence of key leakage. In: Abe, M. (ed.) Advances in Cryptology - ASIACRYPT 2010. pp. 613–631. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)

[50] Dodis, Y., Pointcheval, D., Ruhault, S., Vergniaud, D., Wichs, D.: Security analysis of pseudo-random number generators with input: /dev/random is not robust. In: Proceedings of the 2013 ACM SIGSAC Conference on Computer Communications Security. p. 647–658. CCS '13, Association for Computing Machinery, New York, NY, USA (2013). https://doi.org/10.1145/2508859.2516653, https://doi.org/10.1145/2508859.2516653

[51] Dong, S., Su, H., Xia, Y., Zhu, F., Hu, X., Wang, B.: A comprehensive survey on authentication and attack detection schemes that threaten it in vehicular ad-hoc networks. IEEE Transactions on Intelligent Transportation Systems pp. 1–30 (2023). https://doi.org/10.1109/TITS.2023.3297527

[52] Dziembowski, S.: Intrusion-resilience via the bounded-storage model. In: Halevi, S., Rabin, T. (eds.) Theory of Cryptography. pp. 207–224. Springer Berlin Heidelberg, Berlin, Heidelberg (2006)

[53] Elkamchouchi, H.M., Elkhair, E.F.A., Abouelseoud, Y.: An Efficient Proxy Signcryption Scheme Based on the Discrete Logarithm Problem. International Journal of Information Technology, Modeling and Computing **1**(2), 7–19 (2013). https://doi.org/10.5121/ijitmc.2013.1202

[54] Elshobaky, A., Rasslan, M., Guirguis, S.: Implementation of schnorr signcryption algorithm on dsp **9**, 217–230 (01 2015). https://doi.org/10.14257/ijsia.2015.9.11.21

[55] European Commission: Road safety statistics 2023. `https://transport.ec.europa.eu/background/road-safety-statistics-2023_en` (2023), accessed: 2024-10-13

[56] European Parliament: Directive (EU) 2022/2555 of the European Parliament and of the Council of 14 December 2022 on measures for a high common level of cybersecurity across the Union, amending Regulation (EU) No 910/2014 and Directive (EU) 2018/1972, and repealing Directive (EU) 2016/1148 (NIS 2 Directive). `https://www.etsi.org/standards` (2022), accessed: 2023-03-03

[57] European Telecommunications Standards Institute: ETSI EN 302 663 V1.3.1: Intelligent Transport Systems (ITS); ITS-G5 Access layer specification for Intelligent Transport Systems operating in the 5 GHz frequency band. `https://www.etsi.org/standards` (2020), accessed: 2023-03-18

[58] European Telecommunications Standards Institute: ETSI TS 102 940 V2.1.1: Intelligent Transport Systems (ITS); Security; ITS communications security architecture and security management; Release 2. `https://www.etsi.org/standards` (2021), accessed: 2022-11-18

[59] European Union: European Rail Traffic Management System: European rail traffic management system. `https://www.ertms.net` (2023), accessed: 2023-03-12

[60] European Union: Shift2Rail: Shift2rail. https://projects.shift2rail.org (2023), accessed: 2023-04-01

[61] Eurostat: Railway safety statistics in the eu. https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Railway_safety_statistics_in_the_EU (2023), accessed: 2024-10-13

[62] Faz-Hernandez, A., Scott, S., Sullivan, N., Wahby, R., Wood, C.: Hashing to elliptic curves. https://datatracker.ietf.org/doc/html/draft-irtf-cfrg-hash-to-curve-09 (2023), accessed: 2023-11-26

[63] Ferreira, L.C., Dahab, R.: Blinded-key signatures: Securing private keys embedded in mobile agents. In: Proceedings of the 2002 ACM Symposium on Applied Computing. p. 82?86. SAC '02, Association for Computing Machinery, New York, NY, USA (2002). https://doi.org/10.1145/508791.508808, https://doi.org/10.1145/508791.508808

[64] Fuentes-Castañeda, L., Knapp, E., Rodríguez-Henríquez, F.: Faster hashing to ${\mathbb{F}}_2$. $In : Miri, A., Vaudenay, S.(eds.)SelectedAreasinCryptography.pp.412 - -430.SpringerBerlinHeidelberg, Berlin, Heidelberg(2012)$

[65] Goldreich, O.: Computational complexity: A conceptual perspective. SIGACT News **39**, 35–39 (01 2008). https://doi.org/10.1017/CBO9780511804106

[66] Goldschlag, D.M., Reed, M.G., Syverson, P.F.: Hiding routing information. In: Proceedings of the First International Workshop on Information Hiding. p. 137–150. Springer-Verlag, Berlin, Heidelberg (1996)

[67] Goubin, L., Patarin, J.: DES and Differential Power Analysis The "Duplication" Method. In: Koç, Ç.K., Paar, C. (eds.) Cryptographic Hardware and Embedded Systems. pp. 158–172. Springer Berlin Heidelberg, Berlin, Heidelberg (1999)

[68] Green, M.D., Miers, I.: Forward Secure Asynchronous Messaging from Puncturable Encryption. In: 2015 IEEE Symposium on Security and Privacy. pp. 305–320 (2015). https://doi.org/10.1109/SP.2015.26

[69] Guo, R., Xu, L., Li, X., Zhang, Y., Li, X.: An efficient certificateless ring signcryption scheme with conditional privacy-preserving in vanets. Journal of Systems Architecture **129**, 102633 (2022)

[70] Hahn, D., Munir, A., Behzadan, V.: Security and privacy issues in intelligent transportation systems: Classification and challenges. IEEE Intelligent Transportation Systems Magazine **13**(1), 181–196 (2021). https://doi.org/10.1109/MITS.2019.2898973

[71] Haidar, F., Kaiser, A., Lonc, B., Urien, P.: C-its pki protocol: Performance evaluation in a real environment. In: 2019 15th Annual Conference on Wireless On-demand Network Systems and Services (WONS). pp. 52–55 (2019). https://doi.org/10.23919/WONS.2019.8795479

[72] Hammi, B., Monteuuis, J.P., Petit, J.: Pkis in c-its: Security functions, architectures and projects: A survey. Vehicular Communications **38**, 100531 (2022). https://doi.org/https://doi.org/10.1016/j.vehcom.2022.100531, https://www.sciencedirect.com/science/article/pii/S2214209622000078X

[73] Han, L., Cao, S., Yang, X., Zhang, Z.: Privacy protection of vanet based on traceable ring signature on ideal lattice. IEEE Access **8**, 206581–206591 (2020)

[74] Hanzlik, L., Kluczniak, K., Krzywiecki, L., Kutylowski, M.: Mutual chip authentication. In: 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom 2013 / 11th IEEE International Symposium on Parallel and Distributed Processing with Applications, ISPA-13 / 12th IEEE International Conference on Ubiquitous Computing and Communications, IUCC-2013, Melbourne, Australia, July 16-18, 2013. pp. 1683–1689 (2013). https://doi.org/10.1109/TrustCom.2013.209, http://dx.doi.org/10.1109/TrustCom.2013.209

[75] Hanzlik, L., Kluczniak, K., Kutylowski, M., Krzywiecki, L.: Mutual restricted identification. In: Public Key Infrastructures, Services and Applications - 10th European Workshop, EuroPKI 2013, Egham, UK, September 12-13, 2013, Revised Selected Papers. pp. 119–133 (2013). https://doi.org/10.1007/978-3-642-53997-8_8, http://dx.doi.org/10.1007/978-3-642-53997-8_8

[76] Harvey, J., Kumar, S.: A survey of intelligent transportation systems security: Challenges and solutions. In: 2020 IEEE 6th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing, (HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS).

pp. 263–268 (2020). https://doi.org/10.1109/BigDataSecurity-HPSC-IDS49724.2020.00055

[77] Hoffstein, J.: Additional Topics in Cryptography, pp. 1–23. Springer New York, New York, NY (2008). https://doi.org/10.1007/978-0-387-77993-5_8, https://doi.org/10.1007/978-0-387-77993-5_8

[78] Hoffstein, J.: An Introduction to Cryptography, pp. 1–58. Springer New York, New York, NY (2008). https://doi.org/10.1007/978-0-387-77993-5_1, https://doi.org/10.1007/978-0-387-77993-5_1

[79] Huang, H.: Provable Security: 5th International Conference, ProvSec 2011, Xi'an, China, October 16-18, 2011. Proceedings, chap. Strongly Secure One Round Authenticated Key Exchange Protocol with Perfect Forward Security, pp. 389–397. Springer Berlin Heidelberg, Berlin, Heidelberg (2011). https://doi.org/10.1007/978-3-642-24316-5_28

[80] IETF: Pairing-friendly curves. Internet-Draft draft-irtf-cfrg-pairing-friendly-curves-11, Internet Engineering Task Force (2021)

[81] International Union of Railways: International union of railways: Frmcs. https://uic.org/rail-system/frmcs/, note = Accessed: 2023-03-23 (2023)

[82] Cooperative intelligent transport systems (C-ITS) — Guidelines on the usage of standards — Part 3: Security. Standard, International Organization for Standardization, Geneva, CH (Feb 2021)

[83] Jiang, Y., Ge, S., Shen, X.: Aaas: An anonymous authentication scheme based on group signature in vanets. IEEE Access **8**, 98986–98998 (2020)

[84] Katz, J., Lindell, Y.: Introduction to Modern Cryptography, Second Edition. Chapman Hall/CRC, 2nd edn. (2014)

[85] Khan, S., Luo, F., Zhang, Z., Rahim, M.A., Ahmad, M., Wu, K.: Survey on issues and recent advances in vehicular public-key infrastructure (vpki). IEEE Communications Surveys Tutorials **24**(3), 1574–1601 (2022). https://doi.org/10.1109/COMST.2022.3178081

[86] Khan, S., Sharma, I., Aslam, M., Khan, M.Z., Khan, S.: Security challenges of location privacy in vanets and state-of-the-art solutions: A survey. Future Internet **13**(4) (2021). https://doi.org/10.3390/fi13040096, https://www.mdpi.com/1999-5903/13/4/96

[87] Kietzmann, P., Schmidt, T.C., Wählisch, M.: A guideline on pseudorandom number generation (prng) in the iot. ACM Comput. Surv. **54**(6) (jul 2021). https://doi.org/10.1145/3453159, https://doi.org/10.1145/3453159

[88] Kiltz, E., Pietrzak, K.: Leakage Resilient ElGamal Encryption. In: Abe, M. (ed.) Advances in Cryptology - ASIACRYPT 2010. pp. 595–612. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)

[89] Ko, J., Jang, J., Oh, C.: Assessing the safety benefits of in-vehicle warning information by vehicle interaction analysis in c-its environments. Journal of Korean Society of Transportation **39**(1), 1–13 (2021)

[90] Kotsi, A., Mitsakis, E., Tzanis, D.: Overview of c-its deployment projects in europe and usa. In: 2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC). pp. 1–6 (2020). https://doi.org/10.1109/ITSC45102.2020.9294441

[91] Koziel, P., Krzywiecki, L., Stygar, D.: Identity-based Conditional Privacy-Preserving Authentication Scheme Resistant to Malicious Subliminal Setting of Ephemeral Secret. In: Obaidat, M.S., Samarati, P. (eds.) Proceedings of the 16th International Joint Conference on e-Business and Telecommunications, ICETE 2019 - Volume 2: SECRYPT, Prague, Czech Republic, July 26-28, 2019. pp. 492–497. SciTePress (2019). https://doi.org/10.5220/0007954204920497, https://doi.org/10.5220/0007954204920497

[92] Koziel, P., Krzywiecki, L., Stygar, D., Obaidat, M., Samarati, P.: Identity-based conditional privacy-preserving authentication scheme resistant to malicious subliminal setting of ephemeral secret. In: ICETE (2). pp. 492–497 (2019)

[93] Krawczyk, H.: Hmqv: A high-performance secure diffie-hellman protocol. In: CRYPTO. pp. 546–566 (2005)

[94] Łukasz Krzywiecki, Bobowski, A., Słowik, M., Słowik, M., Kozieł, P.: Schnorr-like identification scheme resistant to malicious subliminal setting of ephemeral secret. Computer Networks **179**, 107346 (2020)

[95] Krzywiecki, L.: Deniable version of SIGMA key exchange protocol resilient to ephemeral key leakage. In: Provable Security - 8th International Conference, ProvSec 2014, Hong Kong, China, October 9-10,

2014. Proceedings. pp. 334–341 (2014). https://doi.org/10.1007/978-3-319-12475-9_25, http://dx.doi.org/10.1007/978-3-319-12475-9_25

[96] Krzywiecki, Ł.: Schnorr-like identification scheme resistant to malicious subliminal setting of ephemeral secret. In: Bica, I., Reyhanitabar, R. (eds.) Innovative Security Solutions for Information Technology and Communications. pp. 137–148. Springer International Publishing, Cham (2016)

[97] Krzywiecki, L., Kluczniak, K., Koziel, P., Panwar, N.: Privacy-oriented dependency via deniable SIGMA protocol. Comput. Secur. **79**, 53–67 (2018). https://doi.org/10.1016/j.cose.2018.08.002, https://doi.org/10.1016/j.cose.2018.08.002

[98] Krzywiecki, L., Koziel, P., Panwar, N.: Signature Based Authentication for Ephemeral Setup Attacks in Vehicular Sensor Networks. In: Gkoulalas-Divanis, A., Marchetti, M., Avresky, D.R. (eds.) 18th IEEE International Symposium on Network Computing and Applications, NCA 2019, Cambridge, MA, USA, September 26-28, 2019. pp. 1–4. IEEE (2019). https://doi.org/10.1109/NCA.2019.8935058, https://doi.org/10.1109/NCA.2019.8935058

[99] Krzywiecki, Ł., Słowik, M., Szala, M.: Identity-based signature scheme secure in ephemeral setup and leakage scenarios. In: Heng, S.H., Lopez, J. (eds.) Information Security Practice and Experience. pp. 310–324. Springer International Publishing, Cham (2019)

[100] Krzywiecki, L., Slowik, M., Szala, M.: Identity-Based Signature Scheme Secure in Ephemeral Setup and Leakage Scenarios. In: Heng, S., López, J. (eds.) Information Security Practice and Experience - 15th International Conference, ISPEC 2019, Kuala Lumpur, Malaysia, November 26-28, 2019, Proceedings. Lecture Notes in Computer Science, vol. 11879, pp. 310–324. Springer (2019). https://doi.org/10.1007/978-3-030-34339-2_17, https://doi.org/10.1007/978-3-030-34339-2_17

[101] LaMacchia, B., Lauter, K., Mityagin, A.: Stronger security of authenticated key exchange. In: Susilo, W., Liu, J.K., Mu, Y. (eds.) Provable Security. pp. 1–16. Springer Berlin Heidelberg, Berlin, Heidelberg (2007)

[102] LaMacchia, B.A., Lauter, K.E., Mityagin, A.: Stronger security of authenticated key exchange. vol. 2006, p. 73 (2006), http://eprint.iacr.org/2006/073

[103] Lamssaggad, A., Benamar, N., Hafid, A.S., Msahli, M.: A survey on the current security landscape of intelligent transportation systems. IEEE Access **9**, 9180–9208 (2021). https://doi.org/10.1109/ACCESS.2021.3050038

[104] Lauter, K.E., Mityagin, A.: Security analysis of KEA authenticated key exchange protocol. vol. 2005, p. 265 (2005), http://eprint.iacr.org/2005/265

[105] Lim, K., Liu, W., Wang, X., Joung, J.: Sskm: Scalable and secure key management scheme for group signature based authentication and crl in vanet. Electronics **8**(11) (2019). https://doi.org/10.3390/electronics8111330, https://www.mdpi.com/2079-9292/8/11/1330

[106] Liu, C.L., Tsai, W.J., Chang, T.Y., Liu, T.M.: Ephemeral-Secret-Leakage Secure ID-Based Three-Party Authenticated Key Agreement Protocol for Mobile Distributed Computing Environments (2018)

[107] Liu, F., Wang, Q.: Ibrs: An efficient identity-based batch verification scheme for vanets based on ring signature. In: 2019 IEEE Vehicular Networking Conference (VNC). pp. 1–8 (2019). https://doi.org/10.1109/VNC48660.2019.9062800

[108] Liu, L., Wang, Y., Zhang, J., Yang, Q.: Efficient proxy ring signature for vanet. The Journal of Engineering **2019**(9), 5449–5454 (2019)

[109] Liu, X., Yang, Y., Xu, E., Jia, Z.: An authentication scheme in vanets based on group signature. In: Intelligent Computing Theories and Application: 15th International Conference, ICIC 2019, Nanchang, China, August 3–6, 2019, Proceedings, Part I 15. pp. 346–355. Springer (2019)

[110] Liu, Y., Wang, L., Chen, H.H.: Message authentication using proxy vehicles in vehicular ad hoc networks. IEEE Transactions on Vehicular Technology **64**(8), 3697–3710 (2015). https://doi.org/10.1109/TVT.2014.2358633

[111] Lu, S., Ostrovsky, R., Sahai, A., Shacham, H., Waters, B.: Sequential aggregate signatures and multisignatures without random oracles. In:

Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings. pp. 465–485 (2006). https://doi.org/10.1007/11761679_28, https://doi.org/10.1007/11761679_28

[112] Maaloul, S., Aniss, H., Kassab, M., Berbineau, M.: Classification of c-its services in vehicular environments. IEEE Access **9**, 117868–117879 (2021). https://doi.org/10.1109/ACCESS.2021.3105815

[113] Malone-Lee, J.: Signcryption with Non-interactive Non-repudiation. Designs, Codes and Cryptography **37**(1), 81–109 (2005). https://doi.org/10.1007/s10623-004-3806-6, https://doi.org/10.1007/s10623-004-3806-6

[114] Manivannan, D., Moni, S.S., Zeadally, S.: Secure authentication and privacy-preserving techniques in vehicular ad-hoc networks (vanets). Vehicular Communications **25**, 100247 (2020). https://doi.org/https://doi.org/10.1016/j.vehcom.2020.100247, https://www.sciencedirect.com/science/article/pii/S2214209620300188

[115] Manulis, M., Suzuki, K.: Modeling Leakage of Ephemeral Secrets in Tripartite/Group Key Exchange (2019)

[116] Masson, É., Gransart, C.: Cyber security for railways – a huge challenge – shift2rail perspective. In: Pirovano, A., Berbineau, M., Vinel, A., Guerber, C., Roque, D., Mendizabal, J., Bonneville, H., Aniss, H., Ducourthial, B. (eds.) Communication Technologies for Vehicles. pp. 97–104. Springer International Publishing, Cham (2017)

[117] Matthias Ruete, E.C.: ERTMS: First Work Plan of the European Coordinator (2020)

[118] Menezes, A., Okamoto, T., Vanstone, S.A.: Reducing elliptic curve logarithms to logarithms in a finite field. IEEE Transactions on Information Theory **39**(5), 1639–1646 (1993)

[119] Mishra, R., Wang, R., Zhang, X., Xu, Z., Zhao, X., Li, X.: Research on performance and function testing of v2x in a closed test field. Journal of Advanced Transportation **2021**, 9970978 (2021). https://doi.org/10.1155/2021/9970978, https://doi.org/10.1155/2021/9970978

[120] Mitsunari, S.: "MCL cryptolibrary". https://github.com/herumi/mcl (2019)

[121] Molina-Masegosa, R., Gozalvez, J., Sepulcre, M.: Comparison of ieee 802.11p and lte-v2x: An evaluation with periodic and aperiodic messages of constant and variable size. IEEE Access **PP**, 1–1 (07 2020). https://doi.org/10.1109/ACCESS.2020.3007115

[122] Mundhe, P., Yadav, V.K., Singh, A., Verma, S., Venkatesan, S.: Ring signature-based conditional privacy-preserving authentication in vanets. Wireless Personal Communications **114**(1), 853–881 (2020). https://doi.org/10.1007/s11277-020-07396-x, https://doi.org/10.1007/s11277-020-07396-x

[123] Nath, H.J., Choudhury, H.: Privacy-preserving authentication protocols in vanet. SN Computer Science **4**(5), 589 (2023)

[124] National Access Point Coordination Organisation for Europe: National access point coordination organisation for europe. https://napcore.eu, note = Accessed: 2023-03-23 (2023)

[125] National Transport Commission, Australia: National transport commission, policy paper, regulating government access to c-its and automated vehicle data. https://www.ntc.gov.au (2019), accessed: 2023-03-23

[126] National Transport Commission, policy paper: Regulating government access to C-ITS and automated vehicle data. https://www.ntc.gov.au (2019), accessed: 2022-05-19

[127] NGMN Alliance: V2X Application Requirements by NGMN Alliance. https://www.ngmn.org/fileadmin/ngmn/content/downloads/Technical/2018/\\V2X_white_paper_v1_0.pdf (2018)

[128] Nick, J., Ruffing, T., Seurin, Y., Wuille, P.: MuSig-DN: Schnorr Multi-Signatures with Verifiably Deterministic Nonces, p. 1717–1731. Association for Computing Machinery, New York, NY, USA (2020), https://doi.org/10.1145/3372297.3417236

[129] Nicolosi, A., Krohn, M., Dodis, Y., Mazières, D.: Proactive two-party signatures for user authentication. In: In Proceedings of the 10 th ISOC Network and Distributed System Security Symposium. pp. 233–248 (2003)

[130] Nicolosi, A., Krohn, M.N., Dodis, Y., Mazières, D.: Proactive two-party signatures for user authentication. In: NDSS (2003)

[131] Niu, S., Shao, H., Hu, Y., Zhou, S., Wang, C.: Privacy-preserving mutual heterogeneous signcryption schemes based on 5g network slicing. IEEE Internet of Things Journal **9**(19), 19086–19100 (2022)

[132] NordicWay 3: Nordicway 2 and nordicway 3. `https://www.nordicway.net` (2022), accessed: 2022-05-14

[133] Ometov, A., Masek, P., Malina, L., Florea, R., Hosek, J., Andreev, S., Hajny, J., Niutanen, J., Koucheryavy, Y.: Feasibility characterization of cryptographic primitives for constrained (wearable) iot devices. In: 2016 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops). pp. 1–6 (2016). https://doi.org/10.1109/PERCOMW.2016.7457161

[134] Park, J.H., Chen, H., Atiquzzaman, M., Lee, C., Kim, T., Yeo, S. (eds.): Advances in Information Security and Assurance, Third International Conference and Workshops, ISA 2009, Seoul, Korea, June 25-27, 2009. Proceedings, Lecture Notes in Computer Science, vol. 5576. Springer (2009). https://doi.org/10.1007/978-3-642-02617-1, `https://doi.org/10.1007/978-3-642-02617-1`

[135] Perera, M.N.S., Nakamura, T., Hashimoto, M., Yokoyama, H., Cheng, C.M., Sakurai, K.: A survey on group signatures and ring signatures: Traceability vs. anonymity. Cryptography **6**(1) (2022). https://doi.org/10.3390/cryptography6010003, `https://www.mdpi.com/2410-387X/6/1/3`

[136] Pointcheval, D., Stern, J.: Security proofs for signature schemes. In: Maurer, U. (ed.) Advances in Cryptology — EUROCRYPT '96. pp. 387–398. Springer Berlin Heidelberg, Berlin, Heidelberg (1996)

[137] Purwanto, Y., Ruriawan, M.F., Alamsyah, A., Wijaya, F.P., Husna, D.N., Kridanto, A., Nugroho, F., Fakhrudin, A., Itqon, M., Febrianta, M.Y., et al.: Security architecture for secure train control and monitoring system. Sensors **23**(3), 1341 (2023)

[138] Rivest, R.L., Shamir, A., Tauman, Y.: How to leak a secret. In: Boyd, C. (ed.) Advances in Cryptology — ASIACRYPT 2001. pp. 552–565. Springer Berlin Heidelberg, Berlin, Heidelberg (2001)

[139] Ruan, O., Zhang, Y., Zhang, M., Zhou, J., Harn, L.: After-the-Fact Leakage-Resilient Identity-Based Authenticated Key Exchange. IEEE Systems Journal **12**(2), 2017–2026 (2018). https://doi.org/10.1109/JSYST.2017.2685524

[140] Salin, H., Lundgren, M.: A gap analysis of the adoption maturity of certificateless cryptography in cooperative intelligent transportation systems. Journal of Cybersecurity and Privacy **3**(3), 591–609 (2023)

[141] Sarr, A.P., Elbaz-Vincent, P., Bajard, J.C.: A new security model for authenticated key agreement. In: SCN. pp. 219–234 (2010)

[142] Sars, J., Matteoni, I., Roos, A.R.: Omvärldsbevakning its 2022: sammanställning av projekt, aktiviteter och tjänster inom det svenska its-området (2022)

[143] Schnorr, C.P.: Efficient signature generation by smart cards. Journal of cryptology **4**, 161–174 (1991)

[144] Sedar, R., Kalalas, C., Vázquez-Gallego, F., Alonso, L., Alonso-Zarate, J.: A comprehensive survey of v2x cybersecurity mechanisms and future research paths. IEEE Open Journal of the Communications Society **4**, 325–391 (2023). https://doi.org/10.1109/OJCOMS.2023.3239115

[145] Selvi, S.S.D., Paul, A., Rangan, C.P., Dirisala, S., Basu, S.: Splitting and Aggregating Signatures in Cryptocurrency Protocols. In: 2019 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPCON). pp. 100–108 (2019). https://doi.org/10.1109/DAPPCON.2019.00021

[146] e Silva, E.d.B., Hadiwardoyo, S.A., Costa, C.E., Marquez-Barja, J.M.: Reinforcing traffic safety by using cam to verify velocity accuracy. In: 2021 IEEE/ACM 25th International Symposium on Distributed Simulation and Real Time Applications (DS-RT). pp. 1–8. IEEE (2021)

[147] Sripathi Venkata Naga, S.K., Yesuraj, R., Munuswamy, S., Arputharaj, K.: A comprehensive survey on certificate-less authentication schemes for vehicular ad hoc networks in intelligent transportation systems. Sensors **23**(5) (2023). https://doi.org/10.3390/s23052682, `https://www.mdpi.com/1424-8220/23/5/2682`

[148] Sripathi Venkata Naga, S.K., Yesuraj, R., Munuswamy, S., Arputharaj, K.: A comprehensive survey on certificate-less authentication schemes

for vehicular ad hoc networks in intelligent transportation systems. Sensors **23**(5), 2682 (2023)

[149] Sudhakar, T., Ramalingam, P., Jagatheswari, S.: An improved proxy-vehicle based authentication scheme for vehicular ad-hoc networks. International Journal of Information Technology **14**(5), 2441–2449 (2022)

[150] Szyma, P.: MCL-Python: Python wrapper for the mcl library. `https://github.com/piotrszyma/mcl-python` (2020), accessed: 2024-01-06

[151] Tang, Q., Chen, L.: Extended KCI attack against two-party key establishment protocols. Information processing letters **111**(15), 744–747 (2011)

[152] Tang, Q., Chen, L.: Extended kci attack against two-party key establishment protocols. Inf. Process. Lett. **111**(15), 744–747 (2011)

[153] Trafikverket: Trafikverket labs. `https://www.trafikverket.se/om-oss/jobba-hos-oss/mot-vara-medarbetare2/niclas--labbkoordinator/` (2023), accessed: 2024-01-06

[154] Ustaoglu, B.: Obtaining a secure and efficient key agreement protocol from (h)mqv and naxos. Des. Codes Cryptography **46**(3), 329–342 (2008)

[155] Vercauteren, F., et al.: Final report on main computational assumptions in cryptography. European Network of Excellence in Cryptography II **11** (2013)

[156] Vidal, J.M., Orozco, A.L.S., Villalba, L.J.G.: Mitigation of DDoS Attacks in 5G Networks: a Bio-inspired Approach. National Academies of Sciences, Engineering, and Medicine (2017)

[157] Vinel, A., Lan, L., Lyamin, N.: Vehicle-to-vehicle communication in c-acc/platooning scenarios. Communications Magazine, IEEE **53**, 192–197 (08 2015). https://doi.org/10.1109/MCOM.2015.7180527

[158] Wainer, J., Xavier, E.C.: A controlled experiment on python vs c for an introductory programming course: Students' outcomes. ACM Transactions on Computing Education (TOCE) **18**(3), 1–16 (2018)

[159] Wolf, M., Gendrullis, T.: Design, implementation, and evaluation of a vehicular hardware security module. In: Information Security and Cryptology-ICISC 2011: 14th International Conference, Seoul, Korea,

November 30-December 2, 2011. Revised Selected Papers 14. pp. 302–318. Springer (2012)

[160] Wolf, M., Gendrullis, T.: Design, implementation, and evaluation of a vehicular hardware security module. In: Kim, H. (ed.) Information Security and Cryptology - ICISC 2011. pp. 302–318. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)

[161] Xie, Z., Chen, Y., Ali, I., Pan, C., Li, F., He, W.: Efficient and secure certificateless signcryption without pairing for edge computing-based internet of vehicles. IEEE Transactions on Vehicular Technology (2022)

[162] Xu, J., Wang, L., Wen, M., Long, Y., Chen, K.: Dpbma: Low-latency message authentication scheme based on distributed verification and priority in vehicular ad hoc network. IEEE Transactions on Vehicular Technology **72**(4), 5152–5166 (2023). https://doi.org/10.1109/TVT.2022.3225204

[163] Zeng, W., Zhang, J.: Leakage-Resilient and Lightweight Authenticated Key Exchange for E-Health. In: 2020 6th International Conference on Information Management (ICIM). pp. 162–166 (2020). https://doi.org/10.1109/ICIM49319.2020.244691

[164] Zhang, C., Xue, X., Feng, L., Zeng, X., Ma, J.: Group-signature and group session key combined safety message authentication protocol for vanets. IEEE Access **7**, 178310–178320 (2019). https://doi.org/10.1109/ACCESS.2019.2958356

[165] Zhang, Y., Ren, F., Wu, A., Zhang, T., Cao, J., Zheng, D.: Certificateless Multi-Party Authenticated Encryption for NB-IoT Terminals in 5G Networks (2019)

[166] Zheng, Y.: Digital signcryption or how to achieve cost (signature & encryption) cost (signature)+ cost (encryption). In: Advances in Cryptology—CRYPTO'97: 17th Annual International Cryptology Conference Santa Barbara, California, USA August 17–21, 1997 Proceedings 17. pp. 165–179. Springer (1997)