UNIVERSITY OF CALIFORNIA
RIVERSIDE

Lightweight Cryptographic Mechanisms for Internet of Things and Embedded Systems

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

by

Abdulrahman Abdulaziz Bin Rabiah

March 2023

Dissertation Committee:

Professor Nael Abu-Ghazaleh, Co-Chairperson
Professor Silas Richelson, Co-Chairperson
Professor Jiasi Chen
Professor Michalis Faloutsos
Professor Walid Najjar

The Dissertation of Abdulrahman Abdulaziz Bin Rabiah is approved:

Committee Co-Chairperson

Committee Co-Chairperson

University of California, Riverside

Acknowledgments

I am deeply thankful and grateful to my advisors Prof. Nael Abu-Ghazaleh and Prof. Silas Richelson for giving me their endless time, help, support and guidance. I truly thank you for being open and encouraging to my new ideas and for being very nice, understanding and reasonable. Because of you, I was able to get the most out of my PhD and have become a solid researcher. I also thank the committee Prof. Jiasi Chen, Prof. Walid Najjar and Prof. Michalis Faloutsos for their valuable comments and feedback.

To my father (may his soul rest in peace) and my mother, all of my successes are undoubtedly because of you. You have made me a strong independent person from the first day of my life. You have always provided me with smart and prudent guidance as well as unlimited help and support. This PhD is just a result of your great nurturing. I appreciate your patience towards me being far away for a long time.

To my lovely wife and friend, Reem, it is hard to find the words to express my profound appreciation and gratitude to you. You were always with me in every bit of this journey; you were very supportive during my ups as well as enduring and soothing during my downs. You were a great listener and a wise advisor. I value our discussions and your insights. Thank you for traveling with me away from family, believing in me and making my PhD easier!

To my little beautiful daughter Norah, you are the most blessing that I have ever had in my life. Although you arrived in late stages of my PhD, you had a significant impact on me and always granted me with love and hope.

To my brother Ahmad, thank you for being their for me whenever I needed you and for your immense help and support. Our insightful discussions helped me make intelligent moves and decisions during my PhD, which I already started reaping its fruit.

To my collaborators Yugarshi Shashwat, Anas Alsoliman, Fatemah Alharbi, Ahmad Bin Rabiah, Elizabeth Liri and Prateek Singh, your invaluable discussions and effort mean a lot to me and I am so happy we made quality and impactful research.

I also thank my friends Hisham Alhulayyil, Abdulrahman Alorini, Abdulmalek Alhathloul for the nice and memorable time and that we had together at UC Riverside.

I also would like to thank my lab mates for the good times that we spent together Ali Mohammadkhan, Esmaeil Koruyeh, Shafiur Rahman, Hoda Naghibijouybari, Khaled Khasawneh, Ahmed Abdo, Saquib Razak, Jason Zellmer, Fatemah Alharbi, Hodjat Esfeden, Elizabeth Liri, Shahryar Afzal, Aditya Dhakal, and Mohammad Jahanian.

Many thanks go to Mrs. Vanda Yamaguch for facilitating the administrative work at the department of Computer Science and Engineering at UC Riverside.

I also thank my other brothers Mohammad, Abdullah, Faisal and Khalid for always being there for me as well as my sisters Asma, Hind, Rabah, Sumayah and Munirah for their continuous love and support.

This PhD is dedicated to my lovely father, who passed away during the pandemic COVID and I never had a chance to say goodbye. I know you would be very proud!

ABSTRACT OF THE DISSERTATION

Lightweight Cryptographic Mechanisms for Internet of Things and Embedded Systems

by

Abdulrahman Abdulaziz Bin Rabiah

Doctor of Philosophy, Graduate Program in Computer Science
University of California, Riverside, March 2023
Professor Nael Abu-Ghazaleh, Co-Chairperson
Professor Silas Richelson, Co-Chairperson

Today, IoT devices such as health monitors and surveillance cameras are widespread. As the industry matures, IoT systems are becoming pervasive. This revolution necessitates further research in network security, as IoT systems impose constraints on network design due to the use of lightweight, computationally weak devices with limited power and network connectivity being used for varying and unique applications. Thus, specialized secure protocols which can tolerate these constraints are needed. This dissertation examines three problems in the constrained IoT setting: 1) Key exchange, 2) Authentication and 3) Key management.

First, IoT devices often gather critical information that needs to be communicated in a secure manner. Authentication and secure communication in an IoT environment can be difficult because of constraints, in computing power, memory, energy and network connectivity. For secure communication with the rest of the network, an IoT device needs to trust the gateway through which it communicates, often over a wireless link. An IoT device needs a way of authenticating the gateway and vice-versa, to set up that secure channel. We introduce a lightweight authentication and key exchange system for IoT environments that is tailored to handle the IoT-imposed constraints. In our system, the gateway and IoT device communicate over an encrypted channel that uses a shared symmetric session key which changes periodically (every session) in order to ensure perfect forward secrecy. We combine both symmetric-key and public-key cryptography based authentication and key exchange, thus reducing the overhead of manual configuration. We study our proposed system, called Haiku, where keys are never exchanged over the network. We show that Haiku is lightweight and provides authentication, key exchange, confidentiality, and message integrity. Haiku does not need to contact a Trusted Third Party (TTP), works in disconnected IoT environments, provides perfect forward secrecy, and is efficient in compute, memory and energy usage. Haiku achieves 5x faster key exchange and at least 10x energy consumption reductions.

Second, signature-based authentication is a core cryptographic primitive essential for most secure networking protocols. We introduce a new signature scheme, MSS, that allows a client to efficiently authenticate herself to a server. We model our new scheme in an offline/online model where client online time is premium. The offline component derives basis signatures that are then composed based on the data being signed to provide signatures efficiently and securely during run-time. MSS requires the server to maintain state and is suitable for applications where a device has long-term associations with the server. MSS allows direct comparison to hash chain-based authentication schemes used in similar settings, and is relevant to resource-constrained devices *e.g.,* IoT. We derive MSS instantiations for

two cryptographic families, assuming the hardness of RSA and decisional Diffie-Hellman (DDH) respectively, demonstrating the generality of the idea. We then use our new scheme to design an efficient time-based one-time password (TOTP) system. Specifically, we implement two TOTP authentication systems from our RSA and DDH instantiations. We evaluate the TOTP implementations on Raspberry Pis which demonstrate appealing gains: MSS reduces authentication latency and energy consumption by a factor of ∼82 and 792, respectively, compared to a recent hash chain-based TOTP system.

Finally, we examine an important sub-component of the massive IoT technology, namely connected vehicles (CV)/Internet of Vehicles (IoV). In the US alone, the US department of transportation approximates the number of vehicles to be around 350 million. Connected vehicles is an emerging technology, which has the potential to improve the safety and efficiency of the transportation system. To maintain the security and privacy of CVs, all vehicle-to-vehicle (V2V) communications are typically established on top of pseudonym certificates (PCs) which are maintained by a vehicular public key infrastructure (VPKI). However, the state-of-the-art VPKIs (including SCMS; the US VPKI standard for CV) often overlooked the reliability constraint of wireless networks (which eventually degrades the VPKI security) that exists in high-mobility environments such as CV networks. This constraint stems from the short coverage time between an on-board unit (OBU) inside a fast moving vehicle and a stationary road-side unit (RSU). In this work, we present TVSS, a novel VPKI design that pushes critical VPKI operations to the edge of the network; the RSU, while maintaining all security and privacy assumptions in the state-of-the-art VPKIs. Our real-life testbed shows a reduced PC generation latency by 28.5x compared to

recent VPKIs. Furthermore, our novel local pseudonym certificate revocation lists (PCRLs) achieves 13x reduction in total communication overhead for downloading them compared to delta PCRLs.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Cryptography is a valuable mathematical tool that can help provide many security and privacy properties including confidentiality, integrity, non-repudiation, anonymity and unlinkability through the use of special cryptographic systems and keys (which need to be carefully managed). These properties can be achieved using either symmetric key cryptography (aka secret key cryptography) or asymmetric key cryptography (aka public key cryptography). In symmetric key cryptography, a pair of users share the same cryptographic key sk to carry out a cryptograpihc operation such as encrypting data to keep it private from non-authorized users or hashing a challenge using sk to achieve authentication. While this mechanism does not require much computation, it requires pairwise sharing of keys making it unscalable when used for authentication in large scale systems as the total number of distinct keys in the system becomes $n^2 \cdot n/2$ for n users. In asymmetric key cryptography however, each user maintains a pair of cryptographically bounded keys (sk, pk), where sk is kept private and pk is published to other users in the system; for example, many digital signature constructions utilize this type of cryptography to achieve identity and message authentication, where data signed using the secret key sk can be verified using the public key pk. A remarkable advantage of this mechanism is that it scales well as the total number of keys in the system becomes $2n$; this makes it common for daily use in large scale systems. However, it requires computationally expensive cryptographic operations as it relies on complex mathematical operations. Public key infrastructure (PKI) is a traditional key management system based on public key cryptography wherein users are equipped with public-private key pairs as well as public key bound digital certificates that are signed by trusted third parties (TTPs). PKI allows for smooth key management such as adding users to the system, revoking users from the system and managing user credentials.

With the increasing deployment of resource-limited devices (*e.g.,* sensors, connected vehicles and Internet of Things devices (IoTs)) [8], designing secure systems with low computational and network overhead has become a critical issue. When devices have limited computational power, memory, network connectivity and/or energy reserves, security often takes a back seat to reducing protocol latency, reducing CPU and memory footprint, and lowering energy consumption. Several high profile attacks in recent years (*e.g.,* Mirai botnet [54] and BrickerBot [188]) highlight the need for better security for these devices [172].

Traditional solutions offering the aforementioned security and privacy properties often rely on expensive cryptographic mechanisms and require frequent network communication, high bandwidth and network latency. Such solutions can be prohibitive in the context of constrained environments due to the limitations in device capabilities, intermittent connectivity as well as not being able to scale to the large number of IoT devices expected to be fully deployed in the near future. Therefore, it is essential to provide lightweight cryptogrphic primitives that allow building lightweight secure and privacy preserving communication systems for IoTs and embedded systems.

1.1 Motivation

This dissertation is motivated by three security aspects that are critical for any IoT system: 1) Key exchange, 2) Authentication and 3) Key management.

1.1.1 Key Exchange

Secret key cryptography is the traditional solution utilized to encrypt massive session data exchanged over the network because of its superior performance over public key cryptography. It is recommended that encryption keys are to be short-lived and rotated periodically (*e.g.,* each session) so as to limit cryptanalysis as well as the amount of data exposed in case of key compromise. Previous mechanisms suggest deriving these session keys from long-term keys using cryptographic key derivation functions (KDF); however, the fact that IoT devices might be easily reachable by surrounding/curious entities indicates potential exposure of long-term keys (and consequently session keys) and confidential data of previous sessions. Perfect forward secrecy (PFS) is a strong security property that guarantees that all keys of past sessions cannot be recovered in case of device compromise (*i.e.,* long-term sk compromise) [76, 53]. Many schemes have been introduced in the literature that attain the PFS property using a PKC-based key agreement protocol, namely authenticated Diffie-Hellman key exchange (*e.g.,* [81, 1, 99]). However, repeated use of public key cryptography places a burden on limited-resource devices for its expensive computation, rendering it impractical. More recently, some approaches [170, 77, 22, 43, 103, 56] have been proposed to completely decouple session key updates from expensive public key operations while achieving PFS; in these approaches however when the key is compromised, the previous key does not maintain its security (it becomes distinguishable from a random value). Therefore, these work do not achieve the ideal PFS, where the loss of a key does not compromise the security of any previous key, making them susceptible to pre-computation attacks. A lightweight authenticated key exchange protocol that can attain the ideal PFS model is important as critical data such as military or self identifying information may be communicated in the IoT environment.

1.1.2 Authentication

Authentication is a main challenge for secure communication in IoT environments. Quick authentication of data can be critical to saving lives and businesses; consider a patient with irregular heart beats or blood pressure, and thus her doctor must instantly be warned for quick response in case of emergency. Symmetric key cryptography based authentication offers a computationally lightweight solution but imposes key-management issues and introduces security vulnerabilities, For instance, SKC requires the server to store IoT authentication keys, which makes the IoT devices subject to impersonation attacks if the server is compromised and the keys are stolen. As a result, digital signatures have been the traditional solution for authentication since it overcomes SKC limitations. However, it imposes considerable overhead on resource-constrained IoT devices. This involves significant computational overhead to generate keys and signatures. Digital signatures additionally require storage of large keys and communication of signatures that are at least several hundreds of bits. This makes it impractical especially for situations in which signers are of limited resources and/or signatures need to be generated extensively and fast (*e.g.,* smart cards or wireless sensor networks (WSN)). Alternatively, hash chains are a cryptographic primitive that can be used to build an authentication system with appealing features compared to digital signatures. A hash chain is a list of vertices $\{v_0, \dots, v_N\}$ labeled by the corresponding list of strings $\{x_0, \dots, x_N\}$ such that $x_{i+1} = H(x_i)$ for all i, and H is a hash function. Traversing the chain in the forwards direction can be done efficiently whereas it is computationally infeasible in the backwards direction due to the hardness of inverting H. A user uses her chain head x_0 to generate x_{N-1} to authenticate herself to a server holding the chain tail x_N. This scheme reduces the bandwidth requirements as it requires only half the signature size. However, hash chains have a limited lifespan and require the user to have a $\mathcal{O}(N)$ time×space complexity for authentication. Therefore, new public-key authentication solutions that are efficient in the amount of computation and energy usage are needed for IoT.

1.1.3 Key Management

Connected vehicles (CVs) is an important part of the emerging IoT technology. CVs allows vehicles and the infrastructure to communicate and collaborate to improve the safety and efficiency of the overall transportation system. CVs and the infrastructure roadside units (RSUs) utilize a direct communication model since low latency is paramount for such real-time systems; they often rely on the direct wireless communication technology called dedicated short range communication (DSRC). Authenticated communication in CV networks is essential as such real-time systems need to make critical decisions based on the information they receive from other vehicles and the infrastructure. Thus, CVs must utilize a vehicular PKI (VPKI) key management system and use their certificates to authenticate and broadcast messages in the CV network. CVs additionally imposes privacy requirements on the VPKI system so as to protect vehicles participating in the CV network from potential vehicle driving activity tracking. In order to prevent other vehicles from linking messages of a single vehicle for a long time (and thus tracking its driving activities), CVs have to periodically refresh their pseudonym certificates (PCs) with new certificates essentially with new pseudonym identities. This property is called unlinkability. In order to also prevent the certificate authority from tracking vehicles, a standard solution is to separate the duties of renewing the pseudonym certificates on two non-colluding authorities such that no one authority is able to map the messages originating from the same vehicle to its long term identity. This property is called anonymity. A major concern (and often overlooked) in the design of VPKIs is the high-mobility nature of CV networks. In essence, a fast vehicle passing by an RSU would have a very brief time of network connectivity which could be less than a second in highway scenarios. Therefore, there is a high probability that a CV request of a PC from the cloud through an RSU would fail (the vehicle might leave the RSU coverage before receiving the PC). Furthermore, the innate channel uncertainty of wireless networks tend to intensify in highly mobile environments [79]. This natural fluctuation of reliability in CV networks would eventually degrade the security level of a given VPKI (*e.g.,* a vehicle might be forced to use the same PC for a long time which would increase the likelihood of a linkability attack). State of the art vehicular PKI systems require highly mobile vehicles to continuously interact with the cloud in order to refresh their pseudonym certificates, which imposes high latency leading to high certificate renewal failures and eventually degradation of the privacy of CVs.

Furthermore, the current revocation mechanism used in VPKIs is certificate revocation lists (CRLs), in which revoked vehicles are compiled in a single list that is then distributed to the whole CV network. This is because mechanisms that rely on the network (such as Online Certificate Status Protocol, OCSP [161]) are not feasible due to the intermittent connectivity of CV networks. Because revocation is more likely in a CV environment (*e.g.,* due to malicious or non-malicious events), this list is expected to grow large overtime, which requires high bandwidth on the vehicles to download it. For instance, consider the cyber attack in 2015 which allowed hackers to disable vehicles and triggered Chrysler to recall 1.4 million vehicles [2]. To mitigate this issue, the US standard VPKI, Security Credential Management System, SCMS, suggests that the CRL is to hold at most 10,000 entries (*i.e.,* ∼400KB); however, this opens an even more dangerous vulnerability as now legitimate entities will not be able to have a holistic knowledge of revoked vehicles, which can lead to malicious attacks. In order to reduce the bandwidth requirements of downloading a CRL, VPKIs additionally consider utilizing delta CRLs, where the CRL is incrementally updated so that the vehicle only downloads the newly revoked vehicles (i.e., delta CRL) when it gets network connectivity with the RSU/backend (e.g., weekly/daily). Nonetheless, the high scale of CVs and the limited contact time that a CV has with the infrastructure make it hard for CVs to successfully download such delta CRLs. VPKI systems also suggest dividing the revoked certificates on different CRLs based on some common factor, such as region of revocation (*e.g.,* a state) [40], so that a vehicle only downloads and keeps a relevant CRL to it. Notice that this approach is not secure because it can still allow a revoked vehicle to maliciously participate in the system outside the region of revocation without being detected as long as the PCs are valid (*e.g.,* SCMS PCs are valid for a week); this can degrade the safety and efficiency of the vehicular environment. Thus, a privacy preserving and scalable VPKI system that can address the aforementioned limitations is of great importance so as to improve road safety and efficiency.

1.2 Dissertation Contributions

This dissertation introduces lightweight cryptographic mechanisms that allow building efficient and scalable IoT systems with strong security and privacy properties.

1.2.1 Key Exchange

In the first work, we construct a secure authentication and key-exchange protocol which achieves ideal PFS, and, after an initial setup phase, requires only *lightweight* private key operations. Specifically, our scheme returns to the model where the master key is fixed once and for all, and where each subsequent session key is computed from the previous, using a hash function. At the core of our new technique is we use the entropy inherent in the messages exchanged during a session in the update procedure. Each entity relies on the session key and an apriori agreed upon set of random messages exchanged during the session (using the previous session key) to update the session key K_s to K_s' using symmetric key cryptography (SKC) and a cryptographic hash function. We make sure no secrets are shared over the channel. As a result, the long-term master keys play a minimal role in our protocol, which allows us to remove the reliance on it. Our protocol guarantees that if the master key or a session key is compromised then all previous session keys retain their security in the sense that they remain indistinguishable from random. Thus it achieves the ideal PFS achieved by [81] (but not by [56]), while still being as lightweight as [56]. We call our protocol **Haiku**, to reflect simplicity and the lightweight nature of the authentication and key-exchange protocol.

Haiku makes use of public-key cryptography only during an initialization phase, where it relies on Elliptic Curve Digital Signature Algorithm (ECDSA) for authentication and the Elliptic Curve Diffie-Hellman Ephemeral (ECDHE) algorithm for key exchange. For normal operations, it uses lightweight symmetric-key mechanisms: a symmetric key cryptosystem, cryptographic hash function and Hashed Message Authentication Code (HMAC)-based Key Derivation Function (HKDF) for confidentiality, message integrity, and authentication. Haiku minimizes the number of messages as well as total bytes exchanged for authentication and key exchange to save energy [118, 146, 42]. Additionally, it does not depend on a central, trusted third party, thus allowing the IoT device and gateway to securely exchange information in a disconnected environment. The protocol minimizes human intervention by not requiring any input from the user for initial setup. Finally, Haiku achieves performance and memory improvements that are compelling, achieving around 5 and 4 times reduction in latency and memory usage, respectively, for initial setup as well as session key updates compared with using public-key cryptography and a TTP. Our experiments also show that IoT devices can reduce energy and CPU cycle consumption by 26 and 20 times for authentication and key exchange, respectively, and reduce the total bytes exchanged over the channel by 6 times. This allows IoT devices to achieve significant energy savings, which is critical since they often depend on limited battery power [185].

Haiku's design is intuitive and straightforward to understand. Despite the fact that it combines elements of public-key and private-key cryptography, it is quite simple, since the composition is modular. This makes it easy to reason about the various parts of the protocol in isolation which keeps our security analysis clean. We also provide a formal proof of security for Haiku and show that it is secure against a series of attacks. Moreover, although we have implemented Haiku using an ECDHE-based authenticated key-agreement (AKE) protocol, any secure AKE would suffice. Thus, if a faster AKE protocol were developed, we could replace this module in our scheme to improve performance. This work is published at USENIX NDSS DISS 2018 [30] and at IEEE ICDCN 2021 [31].

1.2.2 Authentication

The starting point for this work is the observation that hash chain based authentication can be improved by using specific hash functions which support faster traversal from the head label to the (preimage of the) tail label; we call these hash functions *mergeable hash functions*. In essence, they allow the generation of any point in a hash chain of length N using a cost of $\mathcal{O}(\log N)$, an exponential improvement. The offline cost of the scheme is the cost of generating a set of basis component hash functions, leaving the online cost to be that of merging these (which mathematically consists only of multiplication operations). The scheme requires additional storage space to store the pre-computed basis hashes, but the total time×space required by the client is $\mathsf{polylog}(N)$. This achieves a significant improvement.

Our work applies this idea of improving hash chain authentication via mergeable hash functions over to the realm of signature-based authentication. Specifically, we show that if a signature scheme supports a special "merge" procedure (which, roughly speaking, allows logarithmically many signatures to be merged to produce a linear number of signatures), then it can be used to drastically improve the efficiency of signature-based authentication. We present our new MSS signature scheme abstractly and show how to obtain from it an efficient stateful signature scheme (which directly yields an efficient authentication scheme). We then instantiate our construction from RSA *and*, for the first time from discrete-log (such as ElGamal [59]) signatures, by showing that RSA *and* BLS [37] signatures support such a merge procedure. We find it surprising that our techniques extend to work in discrete-log groups since discrete-log-based hash functions do not support the same merge-friendliness exhibited by RSA hash functions. To demonstrate the utility of MSS and its constructions, we describe a concrete application next.

We note that similar ideas have appeared in prior work [39, 133], but in some other contexts, for example to speed up the runtime of algorithms which repeatedly apply the RSA function. Some subtleties need to be dealt with to turn this idea into a secure authentication scheme. We are unaware of any prior work in using mergeable signatures for ElGamal signatures.

Application – Time-based One Time Passwords. We use MSS to implement a time-based one-time password (TOTP) authentication system. TOTP systems allow a user to authenticate herself to a server using a ”one-time” password that is valid for a short, fixed time. After the time expires, the user will have to authenticate herself again using another password.

Finally, we implement and compare our scheme to other authentication implementations in the context of a TOTP system in a number of scenarios involving resource-constrained devices. The experimental results on Raspberry Pis show that the new MSS based TOTP systems provide appealing efficiency gains. Our RSA-based TOTP system cuts down authentication latency and energy consumption by 6 and 10 times, respectively, compared to a recent TOTP system based on one-dimensional hash chains *(with client storage of first hash of each week)* [100]. Additionally, our elliptic curve ElGamal (ECElGamal) based TOTP system reduces authentication latency and energy consumption by ∼82 and 792 times, respectively, compared to [100] (*with client hash storage as in the previous case*). We believe our proposed scheme provides an excellent option for verification for energy-constrained devices. This work is published at IEEE ICDCS 2021 [148].

1.2.3 Key Management

Our contributions in this work are the following:

Edge-based VPKI: We propose Token-based Vehicular Security System (TVSS), a new system architecture for VPKI with properties which are essential for a large scale mobile PKI system. The core novel feature of is that it takes advantage of the compute power of the network of roadside units (rather than using RSUs simply as a network of proxies connecting vehicles to the backend servers). We find that computationally able RSUs fit seamlessly into VPKI, yielding improvements across the board. Our system has the following benefits, which are listed and discussed below to show their importance:

1. **Low latency PC generation:** TVSS offers a lightweight PC generation protocol consisting essentially of just a handshake between the vehicle and an RSU. In particular, PC generation requires *no online involvement from the back-end*. This allows for new use cases which were unsupported by previous systems (*e.g.*, high speed PC generation that is similar to the electronic toll road collection gates such as the E-ZPass toll collection gates in the US).
2. **Improved Revocation:** The revocation procedure in TVSS requires drastically less total communication than the central PCRL-based solution.
3. **Simple architecture:** Passing computation to the RSUs greatly simplifies the system as a whole. The overall footprint of TVSS is much smaller than that of the US standard, SCMS. Thus, if implemented, TVSS would be much cheaper to maintain. We stress that *this comes with no loss of security.*

Formalizing VPKI Security: In order to foster future work on VPKI, we give formal game-based security definitions for unlinkability and anonymity, the two main security requirements of VPKI. Additionally, we consider a new type of attack on a VPKI scheme we call *clone attacks* and show how to neutralize them in TVSS. Clone attacks fall under the broad umbrella of *sybil attacks* [55], which occur when a single vehicle obtains several different copies of valid credentials in order to pretend to be several different vehicles. A clone attack is a variant which occurs when an authorized vehicle shares its CA-validated credentials with an unauthorized vehicle in an attempt for both cars to obtain valid PCs from different RSUs. Similar attacks have been considered in cryptocurrencies [45, 202], transportation toll collection [159] and network authorization [140] schemes. We show how to handle clone attacks against TVSS; no discussion or defense to clone attacks is given in other VPKI systems.

General OBU and RSU Testbed: We build and assemble a real testbed of OBUs and RSUs that have technical specifications similar to commercial OBUs and RSUs. Specifically, we set up the networking standard specifically designed for connected vehicles, IEEE 802.11p/dedicated short range communication (DSRC). We believe these OBUs and RSUs can be very useful to other research, enabling general vehicle to infrastructure (V2I) and V2V applications as they are open source and re-programmable.

VPKI Implementation and Field Experiments: We implement TVSS and other VPKI systems on real OBUs and RSUs, conduct a series of highway and in-city street experiments at different velocities ranging from 25mph to 85mph. TVSS achieves 28.5x reductions in its PC generation latency compared to a recent VPKI. We also observe a ∼13x reduction in total communication for our optimized version of PCRLs compared to delta PCRLs (updated daily). At extreme speeds, TVSS achieves 3.85x improvements in success ratio of OBUs refreshing PCs compared to a recent VPKI system while SCMS is unable to refresh PCs. At extreme speeds also, local PCRLs are 6.5x more likely to be successfully downloaded by OBUs compared to the best alternative delta PCRLs.

1.3 Dissertation Outline

In Chapter 2, we provide a literature review and discuss the state-of-the-art mechanisms of key exchange, authentication, key management and certificate revocation. We then present our novel efficient authenticated key exchange protocol in Chapter 3. In Chapter 4, we introduce our novel online/offline signature scheme that yields an efficient authentication protocol. In Chapter 5, we introduce a novel privacy preserving, scalable and low latency vehicular pubic key infrastructure system for connected vehicles. We finally conclude and present the future work in Chapter 6.

Chapter 2

Related Work

We divide the related work into six groups: 1) Authenticated Key Exchange, 2) Online/offline signature based authentication, 3) Hash chain based authentication, 4) Second-factor authentication, 5) Vehicular public key infrastructure based key management and 6) Digital certificate revocation.

2.1 Authenticated Key Exchange

Traditional authentication and key exchange protocols might not be suitable for IoT environments due to making heavy use of PKC, which is heavy for environments with resource-constrained devices; for example, Transport Layer Security (TLS) [154] and Datagram Transport Layer Security (DTLS) [155]. Some traditional solutions might be infeasible in disconnected IoT environments since they rely on a TTP, such as Kerberos [101]. Other solutions do not achieve the strong security property, PFS, like Wi-Fi Protected Access 2-Pre Shared Key (WPA2-PSK) [4].

Some IoT-specific protocols use PKC repeatedly for authentication and key establishment [121, 147, 181, 99, 196], which is expensive. Other protocols rely mainly on a central trusted third party (*e.g.,* Certificate Authority, CA) for authentication and key exchange [21, 153, 145, 171, 88], which is infeasible in disconnected environments. Other solutions do not achieve PFS, (*e.g.,* [68, 89]). Some approaches [170, 77, 22, 43, 103] rely on weaker security models (susceptible to dictionary attacks) than ours, to achieve PFS. Haiku achieves PFS using a stronger security model (via random salts) while using lightweight key updates. Another approach introduced relies mainly on the hardware capability for introducing randomness for authentication and key exchange using Physically Unclonable Function (PUF) [15]. The PUF approach seems helpful, but it is still not widely deployed in devices. The approach in [201] requires the authenticator (*e.g.,* gateway) to move towards the IoT back and forth or do some physical motions while the IoT is sending random packets. The authenticating device then matches the IoT Received Signal Strength (RSS)-trace with an apriori RSS-variation. However, human presence is needed or the authenticator needs to be able to do the motions by itself. Other context-based authentication and key exchange schemes for IoT are also introduced in [109, 195, 129, 158, 95, 92].

The authors in [191] take advantage of the randomness in wireless channels to update the session key. This approach relies mainly on the assumption that the wireless channel is not perfect (loss free). One downside of their protocol is that if the adversary has access to a perfect channel, the protocol becomes vulnerable to both passive (*e.g.,* eavesdropping) and active attacks (*e.g.,* hijacking). Another downside is that their protocol does not provide authentication. Since each pair of nodes starts the first session with a publicly fixed initial session key, their protocol is susceptible to impersonation attacks, which are problematic for machine to machine communication.

The protocol proposed in [30] improves the work in [191] by providing lightweight authentication and not requiring adversaries have an imperfect wireless channel. Haiku improves on the work in [30] by making changes to the protocol, thus making it support lossy links, more scalabel, efficient and secure. Particularly, [30] achieves PFS as long as an attacker, who has captured all encrypted messages of all sessions, is able to find *only one secret key*, either the long term key or session key, during a session. Haiku improves security and achieves PFS even if the attacker is able to find *all secret keys* during a session. We prevent such an attacker from updating the session key given that he/she acquires all secret keys during a session along with all encrypted messages of all sessions. We also provide a formal proof of security for the proposed protocol. We finally provide implementation results: latency under various network conditions, number of bytes exchanged over the network, memory footprints and energy consumption.

2.2 Online/Offline Signature based Authentication

There are a number of applications in which signers are of limited resources and/or signatures need to be generated extensively (*e.g.,* smart card or wireless sensor network (WSN) settings). The notion of online/offline signatures was first introduced in [63] to help signers avoid expensive operations at the (critical) time of signing and generate signatures quickly. The authors in [63] proposed a hybrid scheme that relies on a Lamport-like one-time signature scheme for efficient online message signing and on a traditional signature scheme (*e.g.,* RSA) for offline signing. While this scheme speeds up online signing since it depends on lightweight hashing operations, it imposes long signatures, which are expensive in resource-constrained environments. Other schemes improve the signature phase at the expense of imposing extra overhead on the signature verification phase (*e.g.,* [167, 94, 197, 71, 193, 104, 163, 166]), which is infeasible if the signature verifier is constrained (*e.g.,* a smart lock). Moreover, [39, 166, 194, 67, 106] incur extra expensive computations during offline signing in order to make online signing more efficient. MSS achieves efficient online signing without incurring extra overhead at offline signing, or verification and signature length. Similar ideas have also appeared in other contexts [133, 132, 124].

2.3 Hash Chain based Authentication

Lamport proposed an OTP scheme in which a client is authenticated using uniquely changing values rather than static passwords, without a server keeping secrets [107]. This scheme was implemented as S/Key [80] using one-dimensional hash chains. This primitive can achieve OTP authentication using one-way communication. S/Key is prone to phishing attacks because the next password can be valid for a long time. T/Key [100] is a recent hash chain-based OTP system that improves S/Key and makes phishing attacks harder by restricting each password to a small authentication window. However, infrequent authentications make its verification too expensive. Furthermore, hash chains are used to build cost-effective micropayment schemes; for instance, PayWord [156] is a credit-based scheme which relies on hash chains and requires minimal use of heavyweight signatures (namely one signature on chain tail) as apposed to traditional micropayment schemes (*e.g.,* [49, 128]). Other hash chain based micropayment schemes and applications are also introduced in [119, 17, 141, 82, 91, 20, 175, 203, 177, 75]. In addition, hash chains appear in group key establishment [29], Winternitz one-time signatures [41], broadcast authentication [9, 142] and Merkle-Damgard construction [127]. Moreover, Nguyen [133] introduced a MDHC construction using RSA commutative hash functions to improve the performance of PayWord. Using their MDHC construction, we implemented a MDHC-TOTP system and show that our TOTP systems outperform it. Ehdaie et al. [58] proposed a two-dimensional hash chain scheme for key agreement in WSN. Commutative hash functions are the basis for hash trees used in broadcast encryption and group key agreement schemes [65, 168].

2.4 Second-Factor Authentication

Shortcomings of static passwords are discussed in [87]. An intensive formal analysis of a suite of second factor authentication schemes with different security, usability and deployability is introduced in [169]. Online second-factor authentication relies on challenge-response protocols between the second-factor device and server, requiring bidirectional communication [50, 69, 120, 174, 51]. Popular schemes include YubiKey [198] and Duo [164]. Such systems, however, use OTP systems when the second-factor device is offline. The HMAC-based OTP (HOTP) system [116] requires the client and server to share a symmetric secret key k and an incremental counter c that they both use to generate an OTP using HMAC(k, c). However, the next password is valid for a long time (similar to S/Key), which makes it subject to phishing attacks. The TOTP system [130] makes this attack less effective by factoring in a timestamp t when generating one-time passwords. A remarkable advantage of these two systems is that they can produce small one time passwords, which allow for better system usability (*e.g.,* OTPs of 20 bits that can be represented as 6 digits). Both schemes, however, share the secret k with the server, rendering them susceptible to server compromise attacks. The attack against the well-known TOTP-based RSA SecurID [113] is expected to have affected more than 40 million people [200]. Other OTP schemes have also been proposed in the literature (*e.g.,* [192, 184]). Our TOTP systems are best suited for offline second factor authentication; they do not share secrets and are more efficient than others.

2.5 Vehicular Public Key Infrastructure based Key Management

Since the development of the first generation of cellular networks, the research towards building CV networks for civilian use, as well as addressing the security and privacy issues of such networks, have evolved into different directions. Ultimately, many research efforts have been consolidated under the term vehicular PKI (VPKI) which is coined after the traditional PKI that secures the Internet.

Early VPKI Designs: In its early designs [189, 136, 32], the VPKI primarily consisted of a CA that issues enrollment certificates (ECs) to vehicles and later issues PCs for V2V communication. While this protects a CV from tracking from other CVs, the CA can still track all CVs.

Separation of Duties in a VPKI: At later iterations, the design of the VPKI separated the PC issuance tasks from CA and assigned them to a different VPKI entity named

Pseudonym Certification Authority (PCA). The main motivation of this separation is to prevent a single entity from linking an EC to its corresponding PC. The US department of transportation (US DoT) has mandated a VPKI system called Security Credential Management System (SCMS) [40, 134]. Similarly, the European Union (EU) has mandated a VPKI system called cooperative intelligent transport systems (C-ITS) certificate management system (EU CMSS) [46, 61, 62]. In these systems, vehicles get fresh PCs from the PCA only after PC requests are validated by the CA using vehicles' ECs. The separation of duties helps prevent the PCA from revealing vehicles' ECs and the CA from revealing PCs information (e.g., public/verification keys). Furthermore, the authors in [10, 72, 33, 98] have proposed a *ticketing scheme* where a vehicle requests a *PC-ticket* from the CA then uses this ticket to acquire a PC from a PCA. The authors in [162] further improved the concept of tickets and introduced what they call a V-Token, an encrypted PC-EC linkage information which is embedded inside every PC and can only be decrypted by the collaboration of several VPKI entities using a threshold encryption scheme. V-Token also utilizes a blind signature scheme to prevent the CA from peering into the content of tokens but at the expense of revealing (hence discarding) some of the tokens in order to provide a probabilistic authentication for the remaining tokens.

Fault-Tolerance in a VPKI: As a different improvement, the authors in [96] have proposed SECMACE, a VPKI architecture that *splits* the CA into multiple CAs where each CA is responsible of a manageable set of vehicles (usually bounded by the same geographical region). The authors in [179] have proposed an additional improvement over SECMACE by introducing IOTA-VPKI, a Distributed Ledger Technology (DLT) implementation based on Direct Acyclic Graph (DAG). The main purpose of this improvement is to prevent a *single-point-of-failure* whenever one of the CAs goes down. The authors of SECMACE have further improved their VPKI design by implementing it as a *cloud-based* solution named VPKI as a Service (VPKIaaS) [97] making use of many cloud-based paradigms such as server migration and resource expansion.

Drawbacks: All of the previously mentioned VPKI designs heavily rely on backend services that are located behind a network infrastructure. Furthermore, every PC request needs at least two separate roundtrips across the network (vehicle-CA and vehicle-PCA roundtrips).

Hierarchical VPKI Certification: As a notable work, the authors in [117] have proposed a VPKI design that is similar to ours. Instead of obtaining PCs, a vehicle obtains Long-Term PCs (LPC) from a PCA that can be valid up to months. The vehicle then obtains the regular short-termed PCs using an LPC from another new entity in the VPKI named Road Authority (RA) which is responsible of issuing PCs for a specific geographical region. A single RA typically controls multiple RSUs that cover the region of the controlling RA.

Drawbacks: With such a hierarchical design, there is a high probability that a vehicle would request PCs from the same RA using the same LPC because of 1) the long-term validity time of LPC entails a bigger exposure of its signature, and 2) the repetitive routine of the daily commutes of drivers which make them interact with the same RA on a daily basis (e.g. from home to work and vice versa). This situation would enable an RA to link multiple PCs to the same vehicle.

2.6 Digital Certificate Revocation

It is important for Intelligent Transportation System (ITS) that misbehaving entities are to be revoked from the system in order to ensure safety on the roads as well as road flow efficiency [139]. There is a consensus to use vehicular PKI (VPKI) to secure ITS. In VPKI, certificate revocation using compilation and distribution of certificate revocation lists (CRLs [47]) that contain certificate serial numbers is the most common technique [70, 137, 152]. Particularly, the standard system in the US, Security Credential Management System (SCMS), mandates such a technique as the main vehicle revocation technique [40].

Previous work has suggested CRL distribution using road-side units (RSUs) [138] and car-to-car epidemic [138, 78, 105]. However, a CRL grows very large overtime and thus it becomes prohibitively bandwidth consuming to download it. To reduce the overhead of distributing the CRL on the VPKI, the CRL can be divided into pieces and each piece is sent to the rest of the VPKI system (*e.g.,* RSUs, .etc). An RSU can then deliver such CRL pieces to vehicles [138], which meanwhile can contribute to disseminating those CRL pieces [105] to other vehicles till the whole system is in synch; however, this solution is vulnerable to pollution attacks as fake CRL pieces can be maliciously injected, which eventually prevents recovering the whole CRL (or even a part of it). To overcome this issue, the backend system could individually sign each CRL piece and disseminate it so that it can be individually verified by recipients; however, this introduces a significant computational burden on both the backend system (*i.e.,* linear number of singing operations) and the recipients (*i.e.,* linear number of signature verification operations). On the other hand, revocation techniques that require vehicles to have constant network communication with the certificate authorities are not realistic to be used in the connected vehicle environment due to network latency and intermittency. An example of these protocols includes the online certificate status protocol (OCSP).

Chapter 3

Haiku: Efficient Authenticated Key Agreement with Strong Security Guarantees for IoT

Today, IoT devices such as health monitors and surveillance cameras are widespread. As the industry matures, IoT systems are becoming pervasive. This revolution necessitates further research in network security, as IoT systems impose constraints on network design due to the use of lightweight, computationally weak devices with limited power being used for varying applications. Thus, specialized secure protocols which can tolerate these constraints are needed.

In this work, we examine the problem of secure authentication and key-exchange in an IoT setting. This problem is fundamental and arises whenever an IoT device wishes to communicate privately with other nodes in its network. Traditional solutions either involve

Table 3.1: Latency (in μs) on Arduino Uno running at 16 MHz.

Operation	Public Key Cryptography (PKC)		Operation	Symmetric Key Cryptography (SKC)	
	EdDSA	ECDHE		AES256	SHA256
Key generation	3,763,668	3,769,856	Key generation	206.27	-
Sign/Key exchange	6,111,812	3,763,952	Encryption/Hash (per byte)	49.66	167
Verify	9,717,781	-	Decryption (per byte)	95.95	-

making heavy use of public-key cryptography (PKC), or relying on a trusted third party (TTP), e.g., [99, 181, 171]. Unfortunately, neither of these solutions is ideal for a number of IoT settings. PKC imposes a computational overhead because of the need for choosing large random prime numbers and computing modular exponentiations. When devices are resource constrained, this cost represents a computational bottleneck and care must be taken during protocol design to avoid incurring these costs too often. Table 3.1 demonstrates the computational latency of standard cryptographic schemes on a constrained IoT device. On the other hand, TTP-based solutions are not ideal for IoT devices either, as use-case constraints might require IoT devices to operate while offline or with only intermittent connectivity with the rest of the network, including the TTP. In this work, we describe a protocol for secure key exchange and authentication which makes minimal use of expensive PKC primitives and achieves strong security without relying on TTP.

Perfect Forward Secrecy. PFS is a strong security notion for communication protocols which persist over time. Roughly speaking, a protocol with PFS segments time into sessions and guarantees that even if a long-term secret key is compromised during a session, previous sessions retain their security [76, 53].

Having a secure, authenticated communication framework between an IoT device and gateway that provides PFS is highly desirable, since critical and private data (e.g., medical, or personal identifying information) may be exchanged in the IoT environment. It is important to ensure that the data is not compromised even if an attacker records the data in the hope of subsequently performing cryptanalysis to derive the secret key and decrypt past information exchanges. PFS demands limiting the use of a fixed secret key to a single session (*i.e.,* a limited number of packet exchanges). Between sessions, the secret keys are updated and old keys discarded.

Prior Work on PFS. PFS is defined and implemented according to [81]. In this construction, two types of keys were maintained – a master key and a session key. The master key was fixed once and for all, while each session key was generated at the beginning of the session using a key-agreement protocol. Session key generation was independent of the communication across all prior sessions, and independent of the master key. So if the adversary compromised the master key, for example, all session keys maintained their security (in the sense that an adversary, given the master key, cannot distinguish the session key from a random string). A clear downside of [81] is that an expensive key agreement protocol must be run in every session. More recently, [56] gave a construction which requires only symmetric key operations. Roughly speaking, their construction breaks time into blocks of multiple sessions. At the beginning of each block, a master key $\mathsf{MK}' = \mathsf{H}_1(\mathsf{MK})$ is computed by applying a hash function to the previous master key. Likewise, at the beginning of each session, a session key $K_s' = \mathsf{H}_2(K_s)$ is computed by applying a hash to the previous session key (or to the master key for the first session in a block). The security guarantee of [56] is that if a master key is compromised then all session keys from previous blocks maintain their security. Note however that when the master key is compromised, the previous master key does not maintain its security (it becomes distinguishable from a random value). Thus, this work does not attain the ideal PFS, where the loss of a key does not compromise the security of any previous key.

Our Contribution. In this work, we construct a secure authentication and key-exchange protocol which achieves ideal PFS, and, after an initial setup phase, requires only *lightweight* symmetric key operations. Specifically, our scheme returns to the model where the master key is fixed once and for all, and where each subsequent session key is computed from the previous, using a hash function. At the core of our new technique is we use the entropy inherent in the messages exchanged during a session in the update procedure. Each entity relies on the session key and an a priori agreed upon set of random messages exchanged (using the previous session key) during the session to update the session key K_s to K_s' using SKC and a cryptographic hash function. We make sure no secrets are shared over the channel. As a result, the long-term master keys play a minimal role in our protocol, which allows us to remove the reliance on it. Our protocol guarantees that if the master key or a session key is compromised then all previous session keys retain their security in the sense that they remain indistinguishable from random. Thus it achieves the ideal PFS achieved by [81] (but not by [56]), while still being as lightweight as [56]. It additionally prevents a passive adversary who somehow possesses the master key or a session key from obtaining future session keys. We call our protocol **Haiku**, to reflect simplicity and the lightweight nature of the authentication and key-exchange protocol.

Haiku makes use of public-key cryptography only during an initialization phase, where it relies on Elliptic Curve Digital Signature Algorithm (ECDSA) for authentication and the Elliptic Curve Diffie-Hellman Ephemeral (ECDHE) algorithm for key exchange. For normal operations, it uses lightweight symmetric-key mechanisms: a symmetric key cryptosystem, cryptographic hash function and Hashed Message Authentication Code (HMAC [26])-based Key Derivation Function (HKDF) for confidentiality, message integrity, and authentication. Haiku minimizes the number of messages as well as total bytes exchanged for authentication and key exchange to save energy [118, 146, 42]. Additionally, it does not depend on a central, trusted third party, thus allowing the IoT device and gateway to securely exchange information in a disconnected environment. The protocol minimizes human intervention by not requiring any input from the user for initial setup. Finally, Haiku achieves performance and memory improvements that are compelling, achieving around 5 and 4 times reduction in latency and memory usage, respectively, for initial setup as well as session key updates compared with using public-key cryptography and a TTP. Our experiments also show that IoT devices can reduce energy and CPU cycle consumption by 26 and 20 times for authentication and key exchange, respectively, and reduce the total bytes exchanged over the channel by 6 times. This allows IoT devices to achieve significant energy savings, which is critical since they often depend on limited battery power [185].

Haiku's design is intuitive and straightforward to understand. Despite the fact that it combines elements of public-key and symmetric-key cryptography, it is quite simple, since the composition is modular. This makes it easy to reason about the various parts of the protocol in isolation which keeps our security analysis clean. We also provide a formal proof of security for Haiku and show that it is secure against a series of attacks. Moreover, although we have implemented Haiku using an ECDHE-based authenticated key-agreement (AKE) protocol, any secure AKE would suffice. Thus, if a faster AKE protocol were developed, we could replace this module in our scheme to improve performance.

3.1 IoT Environment Constraints and Requirements

As mentioned, IoT environments impose a number of constraints. IoT devices are often limited in terms of energy, memory and/or processing power [185]. Further, specialized use-cases might require IoT devices to operate while off-line or disconnected from the rest of the network [112], without access to a TTP. We next outline our network model and specify the attack scenarios considered in this work.

3.1.1 Network Model and Assumptions

The network topology considered is shown in Fig. 3.1. The network has multiple (potentially a large number of) IoT devices and a gateway with an intermittent connectivity with the cloud. The IoT devices communicate exclusively through the gateway. We assume that communication may be over a wireless network, where other parties may be able to sniff and capture the encrypted packets exchanged between the IoT device and the gateway. We assume the MAC layer protocol does not provide link-level reliability.

A new IoT device joins the network by performing a secure handshake with the gateway. We assume the IoT device and the gateway are equipped with a limited amount of non-volatile storage (*e.g.,* an EEPROM). We also assume IoT devices are equipped with

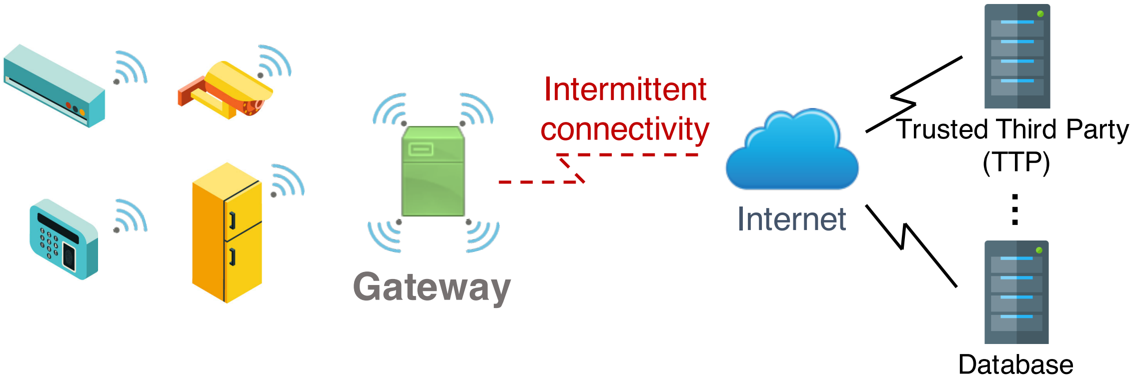

Figure 3.1: Network Topology.

certificates (*i.e.*, a private/signature key sk and a public/verification key vk) [48, 90] and a hardware security extension technology, like ARM TrustZone [144] and Intel SGX [3], providing trusted execution environment (TEE) to protect secret keys from intruders.

3.1.2 Attack Scenarios

We outline the possible attack scenarios that may be used by an adversary $\mathcal{A}$ to exploit vulnerabilities of an IoT protocol such as Haiku. We aim to ensure that the protocol we design is robust against these potential attacks.

- $\mathcal{A}$ may seek to sniff on the channel to find the secret keys from authentication and key exchange messages and also potentially change the content of data messages.
- $\mathcal{A}$ may try to cause disruptions by altering, fabricating or replaying authentication and key exchange messages.
- $\mathcal{A}$ may try to provide false data by replaying old data messages.
- $\mathcal{A}$ who determines a session key may also seek to determine keys of future or previous

sessions to gain access to confidential messages or alter data of future sessions.

- $\mathcal{A}$ who determines the long-term secret (*i.e.,* IoT/gateway signature key, sk) and who has recorded all the encrypted messages may seek to find previous session keys to decrypt those previous messages.

- A passive attacker who determines all the secret keys during a session may also seek to determine the session keys for subsequent sessions in order to continue to eavesdrop on the channel.

Haiku is designed to prevent all of these attacks, and a security analysis of the protocol is provided in Section 3.3 to verify this.

3.2 Haiku

3.2.1 Protocol Overview

Haiku consists of three algorithms: $(\mathsf{Init}, \mathsf{Update}, \mathsf{Comm})$. Roughly speaking, Init is used once at the beginning to set the first session key; Update is run at the end of each session to refresh the session key; Comm is used for communication during a session. Importantly, the first procedure, Init, is the only one which makes use of public key operations; Update is entirely symmetric key based. PFS demands that after Update is run, to refresh a session key and delete the old key. The communication of the old sessions are secure even if the adversary learns the new session key and the long term private key associated with the device. We begin with a high level discussion of each algorithm. They are described formally later in this section. We envision Haiku providing link layer security.

Parameters and Subroutines. Haiku is parameterized by a security parameter n and integer N which controls the length of each session. The communication subroutine Comm uses a symmetric-key encryption scheme and a secure MAC. We denote these encryption, decryption and signing procedures by $\mathsf{E}, \mathsf{D}, \mathsf{MAC}$, respectively. Also, Update uses a hash function H.

- $\mathsf{Init}(1^n)$**:** Two parties, Alice and Bob, use ECDHE to agree on a session key K_s and a set of frames $T_{\mathsf{frames}} \subset \{1, \ldots, N\}$ to be used during Update. Additionally, Init initializes $\mathsf{F.Data} = \emptyset$ and $i = 0$; $\mathsf{F.Data}$ will be populated and i incremented throughout the session; once $i = N$, Update is run.

- $\mathsf{Comm}(K_s, T_{\mathsf{frames}}, \mathsf{F.Data}, \mathsf{msg}, i)$**:** This is used for one party to securely send msg to the other party as long as $i < N$.

  - if $i \geq N$, both parties do nothing;
  - Alice sets $F_i = (\mathsf{msg}, \sigma)$ where σ is a MAC of msg and computes the ciphertext $\mathsf{ct} = \mathsf{E}_{K_s}(F_i)$ and sends ct to Bob; F_i is the payload of the $i-$th frame.
  - if $i \in T_{\mathsf{frames}}$, both parties set $\mathsf{F.Data} = \mathsf{F.Data} \cup \{(i, F_i)\}$; both parties increment i (Bob learns F_i by decrypting ct).

- $\mathsf{Update}(K_s, \mathsf{F.Data})$**:** computes a new key and frame set as $(K'_s, T'_{\mathsf{frames}}) = \mathsf{H}(K_s, \mathsf{F.Data})$. Re-initializes $\mathsf{F.Data} = \emptyset$ and $i = 0$.

Intuition. Fig. 3.2 shows an overview of Haiku's operation. In each session, a number of messages are exchanged (F_i's), the red envelopes correspond to the randomly selected subset T_{frames} whose contents are used during Update to generate the next session key.

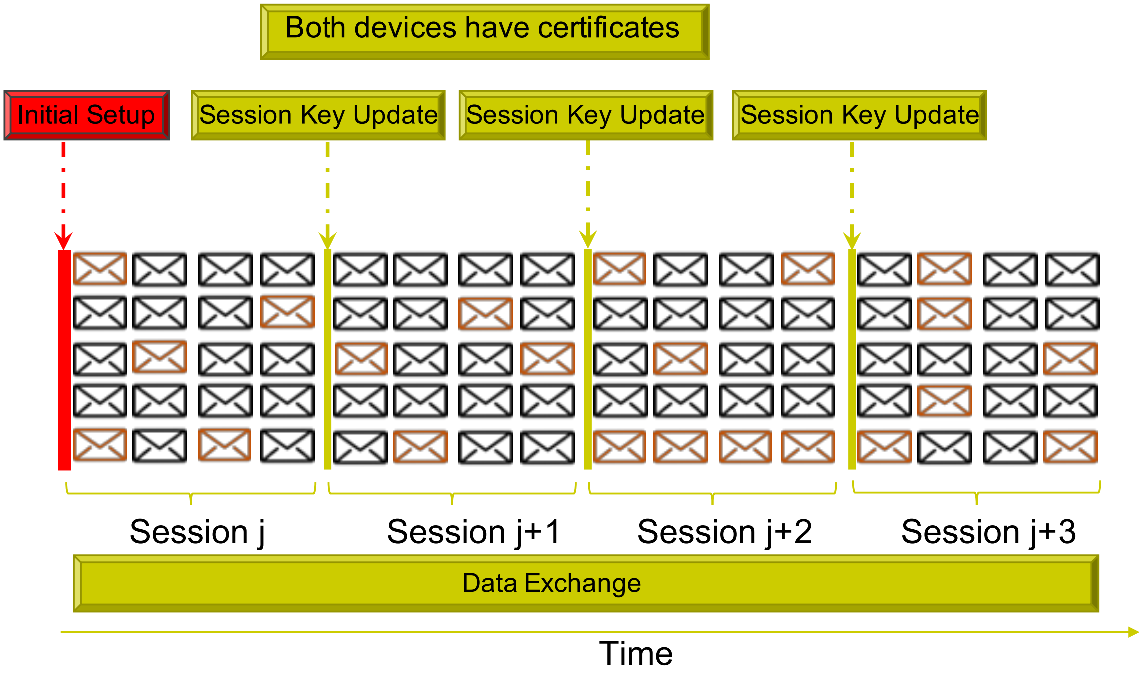

Figure 3.2: Overview of Haiku operation.

Implementation Details. Often IoT devices possess unique credentials signed by the manufacturer, and devices use these credentials to authenticate one another before any interaction takes place [180]. Our implementation includes this handshake as part of the Init subroutine. In our implementation, we utilize Authenticated Encryption with Associated Data (AEAD), namely the Counter with CBC-MAC (CCM) [187] block cipher mode, across all Haiku phases. CCM mode uses CBC-MAC to calculate a Message Authentication Code (MAC) for the whole frame (header, nonce and payload) using a secret key (*i.e.*, K_s), and it uses the Counter mode to encrypt the payload and the MAC using a nonce and K_s whereas the header fields (*e.g.*, MAC addresses) are left unencrypted to allow the receiver to process the frame properly. Using one key with CCM for confidentiality and integrity is provably secure [93] and saves memory. We choose CCM because it is provably secure, patent-free (unlike other modes like OCB [102]), requires small memory [176] and is faster than other modes like EAX [28] and GCM when no pre-computed memory is used [125]. When the number of data messages exchanged during a session reaches an a priori agreed upon threshold (N), Update is called, obtaining a new session key; the old key is deleted and a new session starts. The session keys are never sent over the network, even in encrypted form. As an optimization, our implementation computes the next session key data: $(K'_s, T'_{\mathsf{frames}})$ incrementally during the current session by initializing $(K'_s, T'_{\mathsf{frames}}) = (K_s, T_{\mathsf{frames}})$ and then each time $i \in T_{\mathsf{frames}}$ updating $(K'_s, T'_{\mathsf{frames}}) = \mathsf{H}(K_s||K'_s||F_i)$. In this way, the parties (who are limited in terms of space) are not responsible for holding a large fraction of all data sent during the session. Finally, if a synchronization failure occurs during Update the parties execute Init to negotiate a new session key via ECDHE key-exchange.

3.2.2 Setup/Reset Phase (Init)

A new IoT device added to the network completes an initial setup to authenticate the gateway and vice-versa, and establish a symmetric session key. Both nodes depend on both verification of certificates that have already been provisioned by the manufacturer and the other node's signature for authentication, and ECDHE key exchange for negotiating a random K_s, which will be used to encrypt and hash subsequent session data messages. We choose ECDHE because it helps achieve PFS and requires neither communicating secrets over the network nor using complicated commit protocols. ECDHE key exchange allows two entities to exchange some public parameters, including random temporary public keys (each entity generates and sends one), over the network, which allows each entity to use its own temporary ECDHE private key along with the other entity's temporary ECDHE

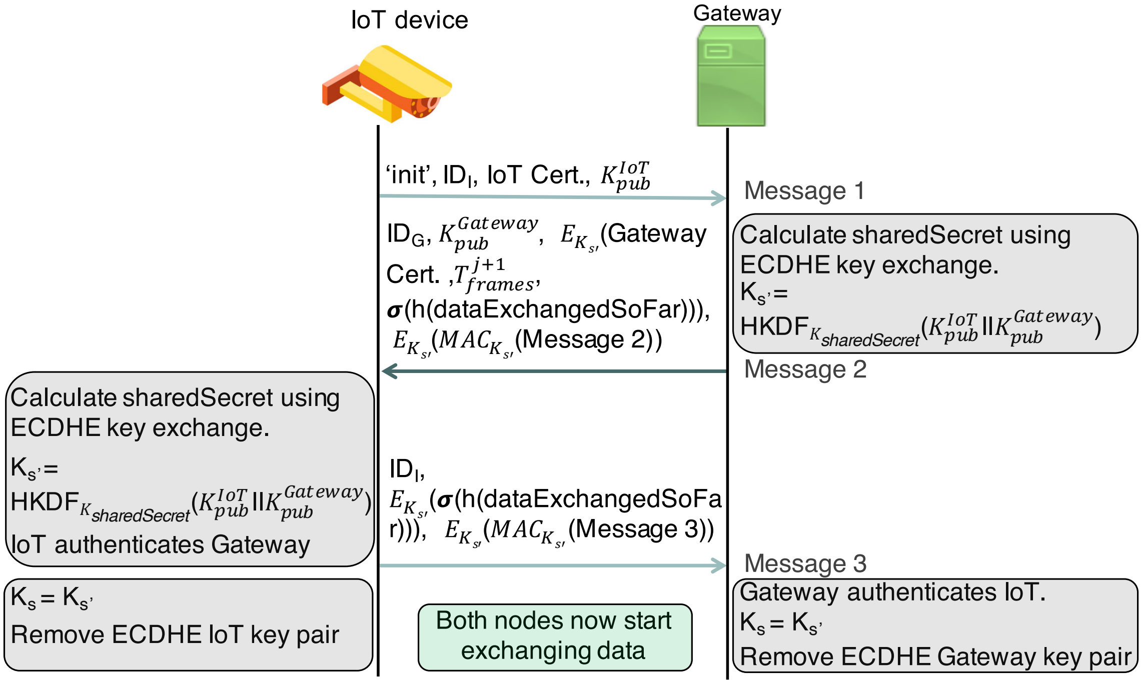
IoT device
Gateway
'init', ID$_I$, IoT Cert., K_{pub}^{IoT}
Message 1
ID$_G$, $K_{pub}^{Gateway}$, $E_{K_{s'}}$(Gateway Cert. ,T_{frames}^{j+1}, $\boldsymbol{\sigma}$(h(dataExchangedSoFar))), $E_{K_{s'}}(MAC_{K_{s'}}$(Message 2))
Calculate sharedSecret using ECDHE key exchange.
K$_{s'}$ = HKDF$_{K_{sharedSecret}}(K_{pub}^{IoT} || K_{pub}^{Gateway})$
Message 2
Calculate sharedSecret using ECDHE key exchange.
K$_{s'}$ = HKDF$_{K_{sharedSecret}}(K_{pub}^{IoT} || K_{pub}^{Gateway})$
IoT authenticates Gateway
ID$_I$, $E_{K_{s'}}$($\boldsymbol{\sigma}$(h(dataExchangedSoFar))), $E_{K_{s'}}(MAC_{K_{s'}}$(Message 3))
Message 3
K$_s$ = K$_{s'}$
Remove ECDHE IoT key pair
Both nodes now start exchanging data
Gateway authenticates IoT.
K$_s$ = K$_{s'}$
Remove ECDHE Gateway key pair

Figure 3.3: Setup/Reset (Init).

public key to derive the same symmetric secret (*i.e.,* K_s). This phase allows the gateway to make sure it is communicating with the legitimate IoT device and vice versa, and thus they can accept messages received from each other. We build this phase (Fig. 3.3) based on the authentication and key exchange process used in Transport Layer Security (TLS 1.3). This phase can also be used to securely reset a new K_s when either one's K_s is inconsistent with the other node for any reason (malicious or otherwise) or when either node suspects a potential attacker.

Message 1. The IoT device uses Message 1 to initiate secure communication with the gateway and provide the gateway with the IoT's certificate to learn and verify its verification key, vk that will then be used to verify data signed by the IoT device. The IoT device selects a random pair of ECDHE public-private keys, denoted by K_{pub}^{IoT} and K_{pri}^{IoT}, that will be used by ECDHE in order to negotiate a symmetric secret, namely $K_{s'}$. As part of Message 1, the IoT device also communicates K_{pub}^{IoT} to allow the gateway to securely derive the symmetric secret, $K_{s'}$, and to challenge the gateway with this random value to verify its identity and verify it is not a spoofing or replay attack. The IoT device sends an 'init', its ID_I, its IoT certificate and its K_{pub}^{IoT} to the gateway.

Message 2. When Message 1 is received, the gateway verifies the IoT certificate using vk of the signer (*e.g.,* manufacturer). The gateway also generates another random pair of ECDHE keys, denoted by $K_{pub}^{Gateway}$ and $K_{pri}^{Gateway}$, to complete the ECDHE key exchange process and derive the symmetric secret, $K_{s'}$. The gateway extracts a shared secret from its $K_{pri}^{Gateway}$ and the received K_{pub}^{IoT} using ECDHE key exchange algorithm. Because directly using the just extracted shared secret as the symmetric secret key might lead to subtle vulnerabilities [108], gateway uses HKDF to derive a new proposed value for the session key, $K_{s'}$, using the just extracted shared secret, used as a HKDF key, and K_{pub}^{IoT} and $K_{pub}^{Gateway}$, used as a salt input. Because adversaries might be willing to cause disruptions at the gateway by spoofing or replaying Message 1 to cause the gateway to change its K_s and end up having a different key as compared to the IoT device, K_s is not changed with the new proposed value, $K_{s'}$, until the gateway receives Message 3 and ensures it is not a spoofing or replay attack.

The gateway sends Message 2 to respond to the initialization request, provide the IoT device with the gateway certificate to learn and verify its vk and provide its ECDHE key share so that the IoT device can derive the same $K_{s'}$. The gateway also proposes a random set of sequence numbers of future frames, T_{frames}^{j+1}, to be considered when updating the session key next time. *The size of the set is also chosen randomly within the size of the session.* It also hashes all data exchanged between the two entities so far, including Message 2 parameters. The hash is then signed by the gateway, σ, to confirm all the data exchanged up to this point. The gateway σ confirms to the IoT device that the gateway has received Message 1 correctly and verifies Message 2 integrity and data authenticity. Because the gateway σ includes the gateway signing the IoT's challenge, K_{pub}^{IoT}, this proves to the IoT device the gateway has the correct sk associated with vk contained in the gateway certificate, and proves this is not a replay attack. Gateway σ also includes a signature on ECDHE keys, K_{pub}^{IoT} and $K_{pub}^{Gateway}$, so that their integrity is preserved and man-in-the-middle attacks are prevented. For example, if ECDHE keys exchanged over the network are not signed, a man-in-the-middle attack can use each node to authenticate itself to the other node while exchanging two different symmetric keys, one with the IoT device, $K_{s'}\_1$, and the other with the gateway, $K_{s'}\_2$. Then, when the authentication is over, confidential data will be forwarded by such an attacker for/to both sides. Gateway σ also includes a signature on the correct T_{frames}^{j+1} so that the IoT device is sure it has received the right set of frames that both ends will use in deriving the next K_s. This prevents malicious changes to this set of frames that could cause disruptions in generating the next K_s. The IoT device can thus authenticate the gateway. T_{frames}^{j+1}, the gateway's certificate, and σ are confidentially communicated to the IoT device using $K_{s'}$, in order to provide adversaries with as little information as possible. In order to verify Message 2 integrity and data authenticity, the gateway calculates the **MAC** of the whole message, including the gateway σ using $K_{s'}$ as a key. Encrypting and hashing data in Message 2 using $K_{s'}$ allows the gateway to prove its knowledge of the key to the IoT device. The gateway sends its ID_G, $K_{pub}^{Gateway}$ and $E_{K_{s'}}$(gateway certificate, T_{frames}^{j+1}, $\sigma, \mathbf{MAC}_{K_{s'}}$(Message 2)).

Message 3. When the IoT device receives Message 2, it extracts the same shared secret from its K_{pri}^{IoT} and the received $K_{pub}^{Gateway}$ using ECDHE. The IoT device also derives the corresponding random $K_{s'}$ by using the just extracted shared secret, as a HKDF key, and K_{pub}^{IoT} and $K_{pub}^{Gateway}$, as a salt input. Additionally it verifies the $\mathbf{MAC}_{K_{s'}}$(Message 2) and if valid, it knows Message 2 integrity is maintained and the gateway has the correct $K_{s'}$. The IoT device verifies the gateway σ with the hash of all dataExchangedSoFar, excluding the gateway σ, that it calculates. This is used along with verifying the gateway certificate to mark the gateway as authenticated. The IoT device uses Message 3 to prove to the gateway it is able to sign the received challenge, $K_{pub}^{Gateway}$, with sk associated with vk that it sent in Message 1 as part of its certificate, proving its identity and that it is not a replay attack. The IoT device also signs the ECDHE keys exchanged over the network indicating that it is deriving the new symmetric secret, $K_{s'}$, using these specific ECDHE keys, which prevents man-in-the-middle attacks. Message 3 also confirms arrival of the correct T_{frames}^{j+1} from Message 2. Moreover, the IoT device signs a hash of all Message 1-3 parameters, IoT σ, to confirm all data exchanged up to this point. By sending the IoT σ, the IoT device confirms to the gateway Message 1's content, correct receipt of Message 2 and the integrity and data authenticity of Message 3. The IoT device sends its ID_I, an encryption of its IoT σ and $\mathbf{MAC}_{K_{s'}}$(Message 3) using $K_{s'}$. It now sets its K_s to $K_{s'}$ and removes ECDHE keys. If the gateway successfully verifies Message 3 **MAC**, it knows that Message 3's integrity has been maintained and the IoT device has the correct $K_{s'}$. The gateway also verifies IoT σ with the hash of all dataExchangedSoFar, excluding the IoT σ, that it calculates. If verified, the gateway can mark the IoT device as authenticated, set its K_s to $K_{s'}$ and remove its ECDHE keys. Even if the attacker finds the long-term signature key of either node later (after this session), this initial K_s cannot be recovered because ECDHE keys are deleted.

3.2.3 Normal Communication Phase (Comm)

Both devices use the derived K_s, which is never exchanged on the wire, to send (encrypt) and receive (decrypt) data messages securely. For each packet, they also include a hash of the whole packet (including a nonce) calculated and encrypted using K_s in order to prevent malicious packet alterations and replay attacks. Fig. 3.4 shows session interaction. While exchanging data messages, both entities incrementally calculate the next session key, $K_{s'}$, using the selected messages based on the sequence number set T^{j+1}_{frames}. Incremental computation of the next session key allows both devices to avoid storing the content of the agreed upon data messages in memory till the end of the session. After exchange of the first session message F_i, $i \in T^{j+1}_{frames}$ both nodes derive a $K_{s'}$ using the current K_s, used as a HKDF key, and F_i, used as an information input. For each of the other messages $F_{i'}, i' > i$ and $i' \in T^{j+1}_{frames}$, both nodes continue to update the new $K_{s'}$ using the current K_s, used as a HKDF key, and $F_{i'}$ along with so-far-calculated $K_{s'}$, used as an information input (context and application specific information).

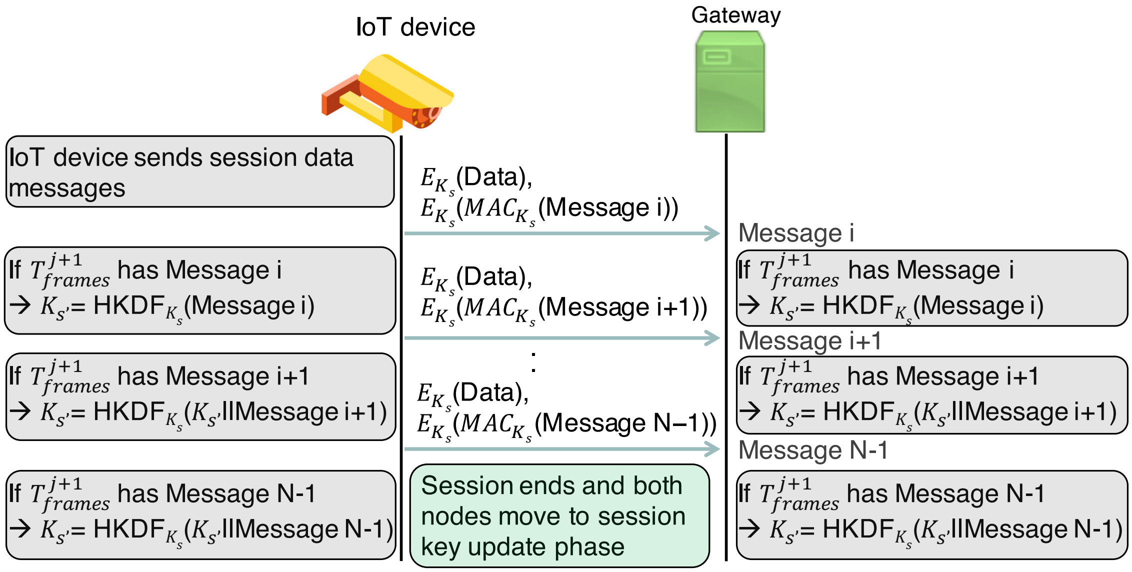
IoT device
Gateway
IoT device sends session data messages
E_{K_s}(Data),
$E_{K_s}(MAC_{K_s}$(Message i))
Message i
If T_{frames}^{j+1} has Message i
→ $K_{s'}$= HKDF$_{K_s}$(Message i)
If T_{frames}^{j+1} has Message i
→ $K_{s'}$= HKDF$_{K_s}$(Message i)
E_{K_s}(Data),
$E_{K_s}(MAC_{K_s}$(Message i+1))
Message i+1
:
If T_{frames}^{j+1} has Message i+1
→ $K_{s'}$= HKDF$_{K_s}$($K_{s'}$‖Message i+1)
E_{K_s}(Data),
$E_{K_s}(MAC_{K_s}$(Message N−1))
If T_{frames}^{j+1} has Message i+1
→ $K_{s'}$= HKDF$_{K_s}$($K_{s'}$‖Message i+1)
Message N-1
If T_{frames}^{j+1} has Message N-1
→ $K_{s'}$= HKDF$_{K_s}$($K_{s'}$‖Message N-1)
Session ends and both nodes move to session key update phase
If T_{frames}^{j+1} has Message N-1
→ $K_{s'}$= HKDF$_{K_s}$($K_{s'}$‖Message N-1)

Figure 3.4: Normal Communication (Comm).

An attacker has no knowledge of the confidentially negotiated T_{frames}^{j+1}, and might also not receive all data messages. Haiku can also be integrated with link layer protocols (e.g., IEEE 802.15.4, WiFi or Bluetooth) providing security capabilities. It can provide the link layer with K_s to protect confidentiality and integrity of exchanged data. Both nodes exchange messages till reaching the session threshold, N, after which they transition to K_s update.

3.2.4 Session Key Update using SKC (Update)

In order to limit possible cryptanalysis to derive the secret key, provide PFS, and limit exposure of confidential data if that secret key is discovered by an attacker for any reason, both nodes need to use frequently updated session keys. This phase allows both nodes to achieve this goal and switch from their previous K_s to the new proposed value,

$K_{s'}$, that has been calculated during the session in the Comm phase as in Section 3.2.3. They also negotiate a new random set of T_{frames}^{j+1} (*with a new random size*) to construct the next K_s for the subsequent session. T_{frames}^{j+1} helps produce random session keys at each update since it makes use of the randomness existing in the data frames and depends on such randomness to generate next K_s. For IoT applications where data frames might have repetition or low entropy, random data can be periodically injected during sessions to ensure data considered for key update has high entropy. We also enhance the approach proposed in [191] by letting both nodes confidentially agree on that random set of T_{frames}^{j+1} for each update of K_s, which prevents an attacker with a perfect channel from knowing the frames that will be used to construct next K_s. Because T_{frames}^{j+1} *is negotiated at the beginning of session j in an encrypted form using K_s of session $j-1$*, this prevents an attacker who finds K_s of session j and decrypts the session data messages from deriving K_s of session $j+1$ since *he/she cannot decrypt* T_{frames}^{j+1} and learn the subset of session j frames that needs to be used to construct K_s of session $j+1$. K_s update is based on TLS 1.3 and shown in Fig. 3.5. During this phase, nodes exchange messages encrypted and hashed via K_s.

Message 1. The IoT device sends Message 1 to let the gateway know it wants both sides to have a new K_s for this new session. This message helps the IoT device make sure it is communicating with the right gateway so that it can accept the new random set of T_{frames}^{j+1} and avoid possible disruptions if a fake T_{frames}^{j+1} were received; the IoT device challenges the gateway with a fresh nonce to authenticate it and prevent replay attacks. $Nonce_1$ is encrypted to limit the amount of information adversaries can see. The IoT device uses K_s to calculate the message **MAC** to verify its integrity and authenticity. The IoT device then

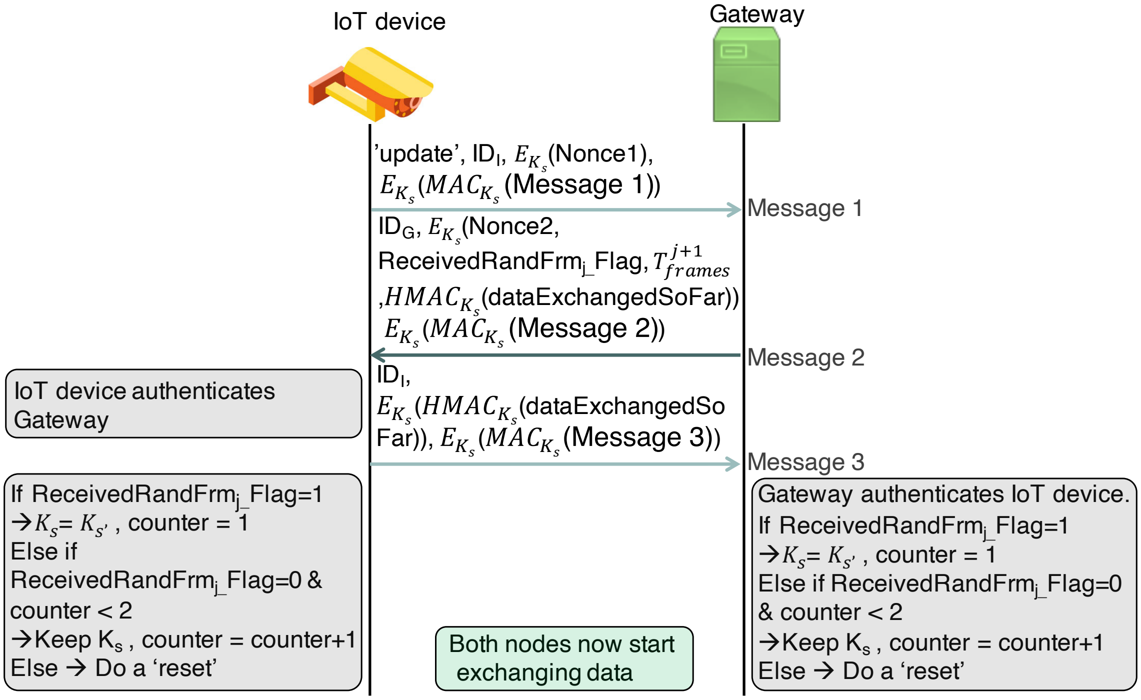
IoT device
Gateway
'update', $\mathrm{ID_I}$, E_{K_s}(Nonce1),
$E_{K_s}(MAC_{K_s}$(Message 1))
Message 1
$\mathrm{ID_G}$, E_{K_s}(Nonce2,
ReceivedRandFrm$_j$_Flag, T^{j+1}_{frames}
,$HMAC_{K_s}$(dataExchangedSoFar))
$E_{K_s}(MAC_{K_s}$(Message 2))
Message 2
IoT device authenticates Gateway
$\mathrm{ID_I}$,
$E_{K_s}(HMAC_{K_s}$(dataExchangedSoFar)), $E_{K_s}(MAC_{K_s}$(Message 3))
Message 3
If ReceivedRandFrm$_j$_Flag=1
→K_s= $K_{s'}$, counter = 1
Else if ReceivedRandFrm$_j$_Flag=0 & counter < 2
→Keep $\mathrm{K_s}$, counter = counter+1
Else → Do a 'reset'
Both nodes now start exchanging data
Gateway authenticates IoT device.
If ReceivedRandFrm$_j$_Flag=1
→K_s= $K_{s'}$, counter = 1
Else if ReceivedRandFrm$_j$_Flag=0 & counter < 2
→Keep $\mathrm{K_s}$, counter = counter+1
Else → Do a 'reset'

Figure 3.5: Session Key Update (Update).

sends the 'update' command, ID_I and $Nonce_1$, along with $\mathbf{MAC}_{K_s}$(Message 1) encrypted with K_s to the gateway.

Message 2. The gateway verifies Message 1 **MAC** using K_s. It also uses Message 2 to challenge the IoT device with $Nonce_2$ to verify its identity and prevent potential replay attacks, communicate a new random set of T^{j+1}_{frames}, with the set size also being random, for updating the next K_s. In order to enable Haiku to function with lossy links (*i.e.,* without link layer reliability or at least one having some residual loss), and allow incremental update of $K_{s'}$ at both nodes as shown in Section 3.2.3, the gateway communicates a flag, $ReceivedRandFrm_j\_Flag$, used to inform the IoT device whether the gateway has received all agreed upon frames based on T^{j}_{frames} to help them decide on this new K_s. One solution to solve the problem of enabling the protocol to work under lossy links is to make the IoT device share a hash of the content of the frames based on T^j_{frames} in this phase; however, we make sure that such a hash is never exchanged over the network and T^j_{frames} is confidentially exchanged exactly once over the network. This prevents passive attackers who have recorded all encrypted messages, and who find K_s in the middle of a session somehow, from being able to update the K_s. This is because they do not know the agreed upon frames based on T^j_{frames} exchanged at the beginning of previous session using a *previous* K_s which is no longer active or stored anywhere.

The gateway, similar to Section 3.2.2, calculates the **HMAC** of all data exchanged between both nodes so far using K_s, $\mathbf{HMAC}_{K_s}$(dataExchangedSoFar), to confirm all data exchanged in Message 1 and 2. The gateway $\mathbf{HMAC}_{K_s}$(dataExchangedSoFar) confirms correct receipt of Message 1 and verifies Message 2 integrity and data authenticity. The gateway $\mathbf{HMAC}_{K_s}$(dataExchangedSoFar) also includes the IoT challenge, $Nonce_1$, to prove the gateway identity and prevent replay attacks. The gateway also verifies integrity and authenticity of Message 2, including the gateway $\mathbf{HMAC}_{K_s}$(dataExchangedSoFar), by calculating the **MAC** using K_s. It then sends ID_G and encryption of $Nonce_2$, $ReceivedRandFrm_j\_Flag$, *random* T^{j+1}_{frames}, its $\mathbf{HMAC}_{K_s}$(dataExchangedSoFar) and $\mathbf{MAC}_{K_s}$(Message 2) using K_s to the IoT device.

Message 3. The IoT device decrypts Message 2 and computes the **MAC** using K_s so that it can be verified with the received **MAC**. If verified, the IoT device knows Message 2 integrity is maintained. It also computes the **HMAC** of all data exchanged so far, excluding the gateway $\mathbf{HMAC}_{K_s}$(dataExchangedSoFar), and checks if it matches the gateway $\mathbf{HMAC}_{K_s}$(dataExchangedSoFar). If verified, the IoT device is confident that the just received T_{frames}^{j+1} is correct. It also knows the gateway is aware of the previously sent $Nonce_1$ from Message 1 and right K_s, so it is not a replay attack. Therefore, the IoT device marks the gateway as authenticated. The IoT device uses Message 3 to prove its identity through $Nonce_2$ from Message 2 and that it is not a replay attack. Message 3 also confirms that the IoT device received correct T_{frames}^{j+1}. It also includes an **HMAC** of all Message 1-3 parameters using K_s, IoT $\mathbf{HMAC}_{K_s}$(dataExchangedSoFar), in order to confirm to the gateway the content of Message 1, the correct receipt of Message 2, including the gateway challenge $Nonce_2$, and the data authenticity and integrity of Message 3. Message 3 helps the IoT device tell gateway it now knows whether all agreed upon frames from previous session were received, and thus the IoT device is able to decide on new K_s for this new session too. The IoT device sends ID_I and encryption of its $\mathbf{HMAC}_{K_s}$(dataExchangedSoFar) and $\mathbf{MAC}_{K_s}$(Message 3) using K_s.

The gateway marks the IoT device as authenticated after successfully verifying Message 3 **MAC** as well as IoT $\mathbf{HMAC}_{K_s}$(dataExchangedSoFar). If the value of $ReceivedRandFrm_j\_Flag$ is set, both devices set K_s to the new session key, $K_{s'}$, that has already been calculated from the Comm phase. On the other hand, frames might get lost due to multiple reasons, and some of those lost ones might belong to the agreed upon random frames from last session based on T_{frames}^{j} that are needed for this session key update. We accommodate this situation in two ways. First, if the current K_s has been used for only one session, both devices use it also for only one more session in order to lower the probability of occurrence of such a situation and avoid frequent resets. Otherwise, if the current K_s has already been used for two consecutive sessions, both devices limit its use by initiating a reset and exchanging a new random K_s, which helps achieve PFS and limit amount of exposure in the worst case. To avoid resets, only the randomly selected frames corresponding to T^j_{frames} have to be correctly identified and captured rather than all frames in the entire session; the probability of losing a frame in T^j_{frames} can be decreased by keeping the size of T^j_{frames} relatively small to all frames exchanged in a single session. This ends the key update phase.

3.3 Security

3.3.1 Our Model

We model security as a game between a challenger C and an adversary $\mathcal{A}$. Recall the three algorithms of the protocol $(\mathsf{Init}, \mathsf{Update}, \mathsf{Comm})$. Also recall that each session consists of N communication messages, at which point Update is called and a new session begins. The game takes place in three stages:

1. Initialize phase: C runs $\mathsf{Init}(1^n)$ between two parties A and B and sends the public transcript to $\mathcal{A}$. C keeps secret the session key K_s and a set of frames T_{frames}. The game now moves to the query phase.

2. Query phase: $\mathcal{A}$ sends C a query; C returns a response. We expand below on the types of queries $\mathcal{A}$ can send. The game remains in the query phase until $\mathcal{A}$ decides to move on. At this point, $\mathcal{A}$ will have sent a query which makes C choose a random bit $b \in \{0, 1\}$ to generate its response.

3. Challenge phase: $\mathcal{A}$ sends C a challenge b' and wins if $b' = b$.

Queries. During the initialization phase, C sets a session counter $\mathsf{count} = 0$, initializes a set of old session keys to $\emptyset$, and initializes the current session info to $(K_s, T_{\mathsf{frames}}, 0, \emptyset)$. During the query phase $\mathcal{A}$ is allowed the following queries:

- $\big(\mathsf{communicate}, \mathsf{msg}, \mathsf{dir}\big)$; $\mathsf{dir} \in \{\mathsf{A\_to\_B}, \mathsf{B\_to\_A}\}$. When C receives this query it does the following:
  - C obtains $(K_s, T_{\mathsf{frames}}, i, \mathsf{F.Data})$ from the current session info set; if $i = N$, C does nothing;
  - C executes $\mathsf{Comm}(K_s, T_{\mathsf{frames}}, \mathsf{F.Data}, \mathsf{msg}, i)$ in the direction specified by $\mathcal{A}$'s query and sends the resulting transcript to $\mathcal{A}$; note this process includes C updating $\mathsf{F.Data}$ and incrementing i.
- (update). When C receives this query it does the following:
  - C retrieves $(K_s, T_{\mathsf{frames}}, i, \mathsf{F.Data})$ from the current session info set; if $i \neq N$, C does nothing;
  - C executes $\mathsf{Update}(K_s, \mathsf{F.Data})$ and sends the resulting transcript to $\mathcal{A}$;
  - C adds (count, K_s) to the set of old session keys, re-initializes the current session info to $(K'_s, T'_{\mathsf{frames}}, 0, \emptyset)$; C also increments count.
- $(\mathsf{update\_and\_reveal})$. This and the next query are challenge queries; $\mathcal{A}$ can ask at most one such query during the entirety of the query phase. When C receives this query it does the following:
  - C retrieves $(K_s, T_{\mathsf{frames}}, i, \mathsf{F.Data})$ from the current session info set; if $i \neq N$, C does nothing;

  - C executes $\mathsf{Update}(K_s, \mathsf{F.Data})$ and sends the resulting transcript to $\mathcal{A}$;
  - C re-initializes the current session info to $(K'_s, T'_{\mathsf{frames}}, 0, \emptyset)$ and sends K_s to $\mathcal{A}$;
  - C chooses a bit $b \in \{0, 1\}$ at random and sends K^*_s to $\mathcal{A}$ where $K^*_s = K'_s$ if $b = 0$ and K^*_s is a random string if $b = 1$.

- $(\mathsf{reveal\_and\_guess}, \mathsf{count}^*)$. This is also a challenge query. When C receives this query it does the following.
  - C collects all elements of the set of old session keys (count, K_s) such that $\mathsf{count} > \mathsf{count}^*$ and sends them to $\mathcal{A}$. C also sends the current session key K_s to $\mathcal{A}$;
  - C chooses a bit $b \in \{0, 1\}$ at random and sends K^*_s to $\mathcal{A}$ where $(\mathsf{count}^*, K^*_s)$ is in the set of old session keys if $b = 0$, and K^*_s is a random string if $b = 1$.

3.3.2 Discussion of Our Model

We now discuss the key features of our security model.

Perfect Forward Secrecy. This means that an adversary $\mathcal{A}$ cannot recover past session keys (or even distinguish past session keys from random) given the current session key. This is captured in our model by the inability of the adversary to win the game via the $(\mathsf{reveal\_and\_guess})$ query. One difference between our security definition and traditional PFS is that the long-term keys play a minimal role in our protocol, as they are only used during initialization (and not at all during Update). Most prior PFS schemes maintain a long-term key and a session key and require that past session keys remain secure even if the long-term key is compromised (but does not promise security in session $i - 1$ if the key of session i is compromised) [56]. Our scheme also guarantees that past session keys remain secure even if the long-term key is compromised.

Update Prediction Attacks. In these attacks, $\mathcal{A}$ somehow learns the current K_s and tries to predict the *next* K_s obtained after updating. These attacks are ruled out even if $\mathcal{A}$ learns K_s as long as it doesn't learn the agreed-upon random frames to use during Update. This is captured in our model by the inability of $\mathcal{A}$ to win the game via the ($\mathsf{update\_and\_reveal}$) query. Although theoretically $\mathcal{A}$ who somehow finds two K_s's of two consecutive sessions and all agreed-upon frames can update to next K_s, the probability that $\mathcal{A}$ captures all previous update frames (including T_{frames}), and $\mathcal{A}$ and gateway together capture all agreed-upon frames is negligible, especially in environments with imperfect eavesdropping and inevitable errors like wireless communication [191]; thus, this prevents $\mathcal{A}$ from updating to next K_s.

Man-in-the-Middle Attacks and Session Hijacking. We analyze an idealized model where the adversary cannot compromise the long-term private device keys. In our scheme these keys are stored in a trusted execution environment (*e.g.,* Intel SGX) and never exchanged over the network.

Multiple Users. Our simplified model considers only a single interaction between Alice and Bob. Security can be proved in a more general model where the adversary is allowed to spawn and control new users and engage in new protocol instances with the challenger. We omit this for simplicity.

MAC Forgeries or Semantic Security Breaks. In order to simplify matters, we assume Alice and Bob are connected by an ideal point-to-point channel. Such a channel is securely implemented assuming the semantic security and unforgeability of the symmetric-key encryption and authentication schemes used by our scheme. This prevents an adversary from injecting or modifying messages. Also, replay attacks are ruled out because each message has a MAC on both content and a unique nonce.

Forced Reset Attacks. These are attacks where the adversary injects bogus messages into one of the nodes in the system in order to force the parties to reset the protocol and run the Init procedure again to generate new session keys. This type of attack scenario only serves to disrupt the parties in the system and does not compromise data integrity or privacy.

3.3.3 Proof of Security

We show that for any polynomial time adversary $\mathcal{A}$, the probability that $\mathcal{A}$ wins the security game is at most $1/2+\epsilon$ for some negligible quantity ϵ. Our proof is by reduction to the security of the hash function H used during Update. Specifically, we reduce to (a version of) the following game for a hash function H, parameterized by an integer N, and played between a challenger C and adversary $\mathcal{A}$.

1. C chooses a random string K_s and a random set $T \subset [N]$;
2. $\mathcal{A}$ sends N messages $x_1, \ldots, x_N$ to C;
3. C computes $\mathsf{H}(K_s, \{x_i\}_{i \in T}) = (K'_s, T')$, where K'_s is another string and $T' \subset [N]$, another subset;
4. $\mathcal{A}$ sends either image or $\mathsf{preimage}$ to C;
5. C draws a random bit $b \in \{0, 1\}$;

- if $\mathcal{A}$ sent image, C sets $K_s^* = K_s'$ if $b = 0$, K_s^* random if $b = 1$ and returns (K_s, K_s^*, T') to $\mathcal{A}$;
- if $\mathcal{A}$ sent $\mathsf{preimage}$, C sets $K_s^* = K_s$ if $b = 0$, K_s^* random if $b = 1$ and returns (K_s^*, K_s', T') to $\mathcal{A}$;

6. $\mathcal{A}$ returns a bit b' and wins if $b' = b$.

Intuitively, this game captures the hardness of recovering K_s such that the following $\mathsf{H}(K_s, \{x_i\}_{i \in T}) = (K_s', T')$ given (K_s', T') and $\{x_i\}_{i \in [N]}$ but not T. In fact, it says more: it is hard even to distinguish the correct K_s from a random string. Note that K_s can be recovered in 2^N time by trying all possible $T \subset [N]$. We assume this game is hard to win for an efficient adversary. This is the case when H is modeled as a random oracle.

We reduce to a version of the above game where $\mathcal{A}$ chooses the N messages in step 2 adaptively, and each time he or she sends an x_i to C, C sends back an encryption of a related message F_i using the secret key K_s. Then the F_i are used to compute the hash in step 3, rather than the x_i. Moreover, we require C to generate (K_s, T) using an ECDHE key exchange protocol, and C begins by sending the public transcript of this protocol. For our reduction, we assume an adversary $\mathcal{A}$ plays against C in the above game and acts as the challenger against another adversary $\mathcal{A}'$ in the security game for Haiku. We show how $\mathcal{A}$ can use an adversary who wins the latter game to win the former. We handle separately the cases when $\mathcal{A}'$ enters the challenge phase of Haiku's security game by sending the $(\mathsf{update\_and\_reveal})$ query and the $(\mathsf{reveal\_and\_guess}, \mathsf{count}^*)$ query; we assume for simplicity that in the former case, $\mathcal{A}'$ does not invoke the (update) query at all, and in the latter case that $\mathsf{count}^* = 1$. These assumptions are essentially without loss of generality; the general case can be handled without difficulty in much the same way. We now proceed formally with our reduction.

Suppose $\mathcal{A}$ plays in the above game against C as follows:

1. $\mathcal{A}$ invokes $\mathcal{A}'$ who plays against $\mathcal{A}'$ in the security game for Haiku.

2. Upon receiving the transcript of the key exchange protocol used to generate (K_s, T) from C, $\mathcal{A}$ forwards the transcript to $\mathcal{A}'$;

3. Each time $\mathcal{A}'$ sends $\mathcal{A}$ a query of the form $(\mathsf{communicate}, \mathsf{msg}, \mathsf{dir})$, $\mathcal{A}$ sends msg to C and receives (y, ct), where ct is an encryption of y using K_s and where y includes msg and an authentication MAC, $\mathcal{A}$ forwards ct to $\mathcal{A}'$; the $i-$th time this occurs, $\mathcal{A}$ sets $x_i = \mathsf{msg}$.

4. **In case of** $\mathsf{reveal\_and\_guess}$**:**

   - The first time $\mathcal{A}'$ sends $\mathcal{A}$ the query (update), $\mathcal{A}$ sends $\mathsf{preimage}$ to C and receives (K_s^*, K_s', T') where (K_s', T') is the key information for the new session, and K_s^* is either K_s or a random string; $\mathcal{A}$ will have to guess which.

   - All subsequent times $\mathcal{A}'$ sends $\mathcal{A}$ the (update) query, $\mathcal{A}$ runs the Update procedure itself (now $\mathcal{A}$ knows (K_s', T')) and sends the resulting transcript to $\mathcal{A}'$; it stores the old session key.

   - When $\mathcal{A}'$ sends $(\mathsf{reveal\_and\_guess}, 1)$ to $\mathcal{A}$, $\mathcal{A}$ returns all session keys to $\mathcal{A}'$ along with K_s^*; when $\mathcal{A}'$ returns b', $\mathcal{A}$ forwards b' to C.

5. **In case of** $\mathsf{update\_and\_reveal}$**:**

- When $\mathcal{A}'$ sends ($\mathsf{update\_and\_reveal}$) to $\mathcal{A}$, $\mathcal{A}$ sends image to C and receives (K_s, K_s^*, T') from C; $\mathcal{A}$ forwards K_s, K_s^* to $\mathcal{A}'$; when $\mathcal{A}'$ responds with b', $\mathcal{A}$ forwards b' to C.

It is clear that $\mathcal{A}$ wins the hash function game for H whenever $\mathcal{A}'$ wins the security game for Haiku.

3.4 Implementation and Evaluation

We used Java to implement the following AKE protocols: 1) Haiku, 2) WPA3 personal [12], 3) a simplified version of WPA2 enterprise with EAP-TLS [4] and 4) TLS with raw public key (TLS-RPK) [190, 199]. In WPA2 enterprise, all nodes communicate with a certificate authority (CA) using the On-line Certificate Status Protocol (OCSP) [161]. In TLS-RPK, both nodes contact a TTP to get the other node's public keys and avoid the overhead of exchanging and verifying certificates. TLS-RPK is utilized to provide authentication and key establishment for link layer security [85]. Across all protocols, we used AES (SKC) for symmetric-key encryption with 256-bit keys, SHA-256 (SKC) for hashing, ECDSA (PKC) for signatures and ECDHE (PKC) for key exchange with 384-bit keys, CCM mode to encrypt and hash, MACs of 128 bits, X.509 certificates and the NIST P-384 elliptic curve. For experiments, nodes run a complete instance of each protocol.

3.4.1 Experimental Setup

We compare Haiku with IoT protocols that provide link-layer security and achieve PFS. Fig. 3.6 shows the two experimental setups: Haiku and WPA3 use setup 1 whereas

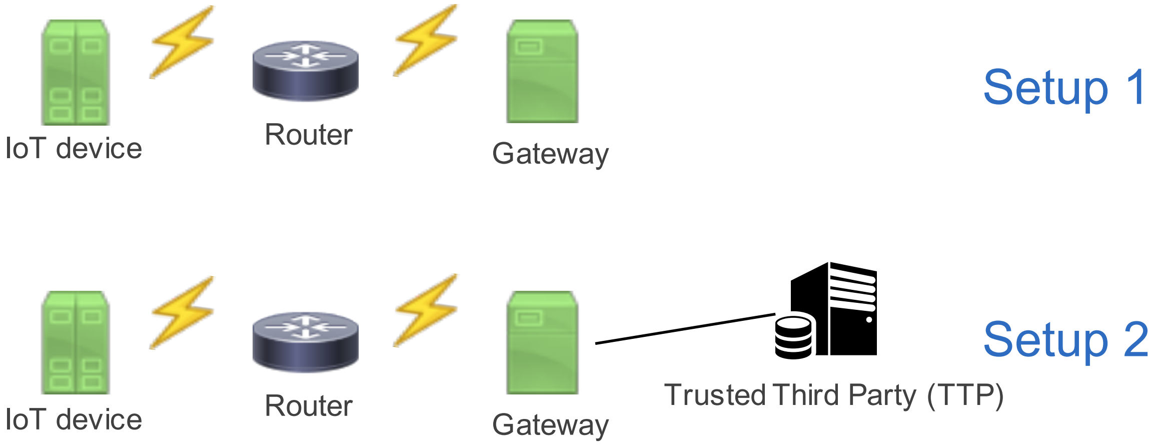

Figure 3.6: Experimental setups.

WPA2-EAP-TLS and TLS-RPK use setup 2. Two laptops communicate over a wireless channel through the wireless router. The two laptops, in setup 2, additionally contact a CA/TTP during authentication to either make sure the received certificates are not revoked or get the other node's public key. The third laptop is the CA/TTP, connected with the gateway over Ethernet. We run a network emulator called NetEm [86], a Linux built-in traffic controller (TC), at the NIC of the IoT device to emulate delay and packet loss in the network. We collected 100 data points for each experiment to get statistically reasonable results and calculate the mean and 95% confidence interval for each metric. For each message, we set a timeout value of 500ms. The maximum number of times a packet is transmitted is set to 2, to show the robustness of Haiku even with high residual loss.

3.4.2 Performance Analysis

We discuss the performance measurements for Haiku and the alternatives under various network conditions.

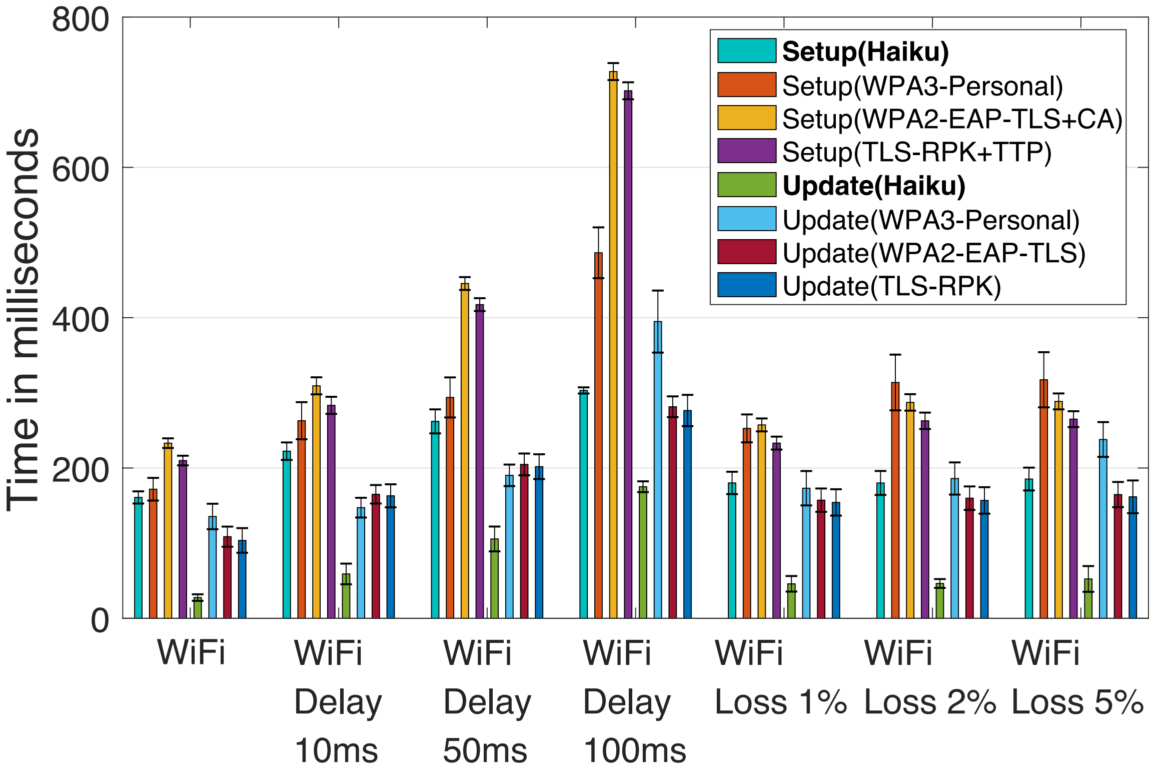

Figure 3.7: Latency for setup & key update phases.

To observe the latency for Haiku in an actual wireless network, we performed our experiments over WiFi. The Haiku update and setup/reset phases show ∼4-5 and 1.05-1.5 times latency reduction, respectively, over the alternatives as shown in Fig. 3.7. Fig. 3.7 also shows the latency of Haiku and its counterparts for networks with higher delay - we use an emulated delay of 10, 50 and 100ms. Protocols that require exchanging additional packets or contacting a CA/TTP add a significant latency especially when there is a large network delay (*e.g.,* 100ms). Haiku has 1.5-2.5 times lower latency than alternatives when the network delay is 100ms. Haiku update also achieves ∼1.8-3 times lower latency compared to the update based on PKC in all cases. Thus, Haiku demonstrates good performance even under varying network delays.

IoT networks are likely to experience frequent and possibly significant residual packet loss. Thus, we also test the performance of Haiku in networks with 1%, 2% and %5 packet loss. Fig. 3.7 shows as packet loss increases in the network, PKC key update and setup when using extra protocol packets or a CA/TTP have worse latency, by up to 4.5 and 1.7 times, respectively, compared to Haiku. Thus, Haiku provides better performance in such networks.

In Fig. 3.8 we further breakdown the latency of the setup/reset based on PKC with/without contacting a CA/TTP. The breakdown is across 7 sub-tasks for the setup: (a) certificate verification using ECDSA, (b) key exchange using ECDHE, (c) network time, (d) calculation of σ for authentication which includes signing and verifying using ECDSA/MAC, (e) contacting a CA/TTP, (f) encryption and decryption using AES, and (g) other processing which includes generating nonces and T_{frames}. Fig. 3.8 shows that the use of PKC constitutes around 33% in the WiFi experiments and around 37% when also using the CA/TTP. Because WiFi results in extra latency, this causes the network time to increase, and thus protocols using extra messages incur a significant additional penalty.

For poor WiFi networks with a 100ms delay, the network time dominates the total latency as expected, as shown in the case of protocols using extra messages like WPA3 or others contacting CA/TTP; however, the use of PKC still accounts for around 20% of the latency. When contacting a CA/TTP under this network condition, it adds a significant burden, and this along with the use of PKC constitutes almost 37% of the total latency. For WiFi with 5% packet loss, setup is also impacted from the use of PKC and contacting a CA/TTP, with almost 31% for the use of PKC and 37% when combined with contacting

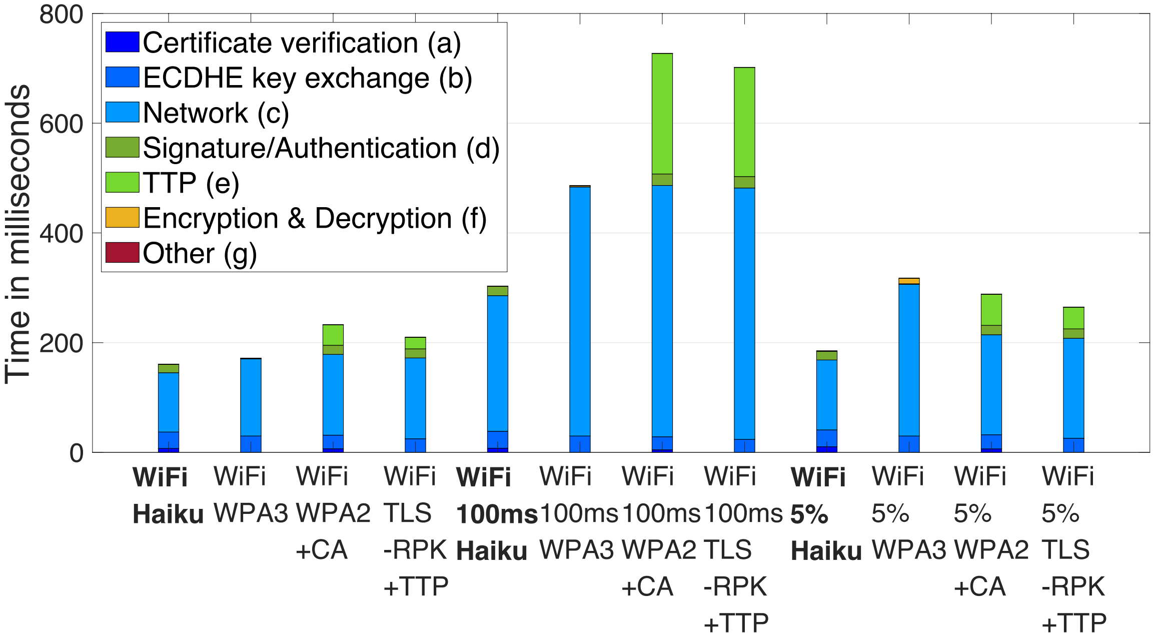

Figure 3.8: Breakdown of setup/reset phase latency.

a CA/TTP. WPA3's network delays increase (due to increased retransmissions) as it exchanges ∼3 times more messages than Haiku. Fig. 3.8 indicates that infrequent use of PKC and fewer message exchanges are better as these require significant processing and network time (expensive in IoT environments).

3.4.3 Overhead Analysis

We evaluated the overhead associated with Haiku and alternatives. Table 3.2 shows Haiku needs at most 3 messages in all phases. The Haiku update and setup exchange up to ∼6 and 1.5 times fewer bytes over the network compared to alternatives. Byte exchange savings at update are due to reducing the number of protocol messages and eliminating usage of PKC (signatures and ECDHE key materials) which require exchanging more bytes compared to SKC; at setup, savings come from omitting exchange of extra messages. This

Table 3.2: Number of messages and overhead (in bytes).

Phase	Number of Messages	Message 1	Message 2	Message 3	Message 4	Message 5	Message 6	Message 7	Message 8	Total Bytes
Initial Setup/Reset (Haiku)	**3**	**680**	**828**	**151**	*****	*****	*****	*****	*****	**1659**
Initial Setup/Reset (WPA3-Personal)	8	194	194	130	130	162	349	402	162	1723
Initial Setup/Reset (WPA2-EAP-TLS with CA)	5	729	885	152	103	665	*	*	*	2534
Initial Setup/Reset (TLS-RPK with TTP)	5	344	500	152	216	711	*	*	*	1923
Update (Haiku)	**3**	**73**	**94**	**77**	*****	*****	*****	*****	*****	**244**
Update (WPA3-Personal)	6	194	194	162	349	402	162	*	*	1463
Update (WPA2-EAP-TLS)	3	168	297	152	*	*	*	*	*	617
Update (TLS-RPK)	3	168	297	152	*	*	*	*	*	617

makes the protocol simpler, faster, and helps IoT's save battery [42]. This is important for resource constrained IoT devices.

Fig. 3.9 shows Haiku update and setup reduce CPU cycle consumption by up to ∼*20 and 1.5* times compared to alternatives, except for WPA3 setup; reducing CPU cycles at the update is more important since it is the constantly recurring phase, as opposed to the setup which occurs only once. Since energy can be scarce in IoT settings, we also use a power meter that logs power consumption with millisecond precision to allow us to make an accurate comparison of energy consumed across different phases. Each phase is run 100 times across each device with power being logged each millisecond to get statistically accurate results. Difference of device baseline power (device power when idle) and logged power is calculated and then averaged to finally calculate energy as follows: $Energy(Joule) = Power(Watt) \cdot Duration(Second)$. Fig. 3.9 also shows Haiku update and setup reduce energy consumption by up to ∼26 and 1.7 times over alternatives. Reductions in Haiku's CPU cycle and energy consumption are because it mainly relies on lightweight SKC which reduces the amount of processing significantly and it exchanges fewer protocol messages (less effort and fewer bytes sent on the link, which saves battery [42]).

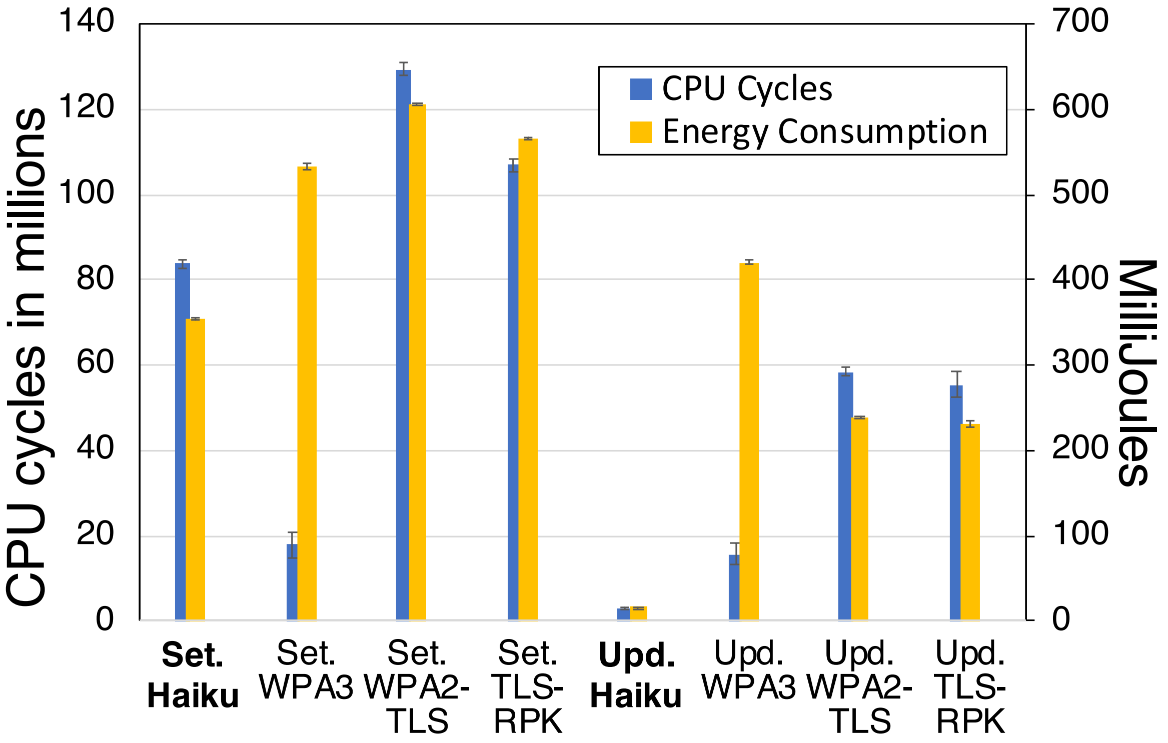

Figure 3.9: Computational costs.

Fig. 3.10 shows memory needed by Haiku and its alternatives. The code size forms most of the memory used in each phase, which can be reduced by code optimization ..etc. The other category involves memory used for other components when protocol is running (*e.g.*, global/local variables ..etc). We emphasize the other category as it does not necessarily change if code size changes. Haiku update and setup reduce memory needed when protocol is running by ∼4 and 1.5 times compared to alternatives. This is because Haiku update removes space overhead imposed by PKC (*e.g.*, longer ECDHE key materials) as opposed to alternatives, and its setup removes parameters needed for extra protocol messages. Our prototype shows Haiku code size is ∼1.3 times less than alternatives.

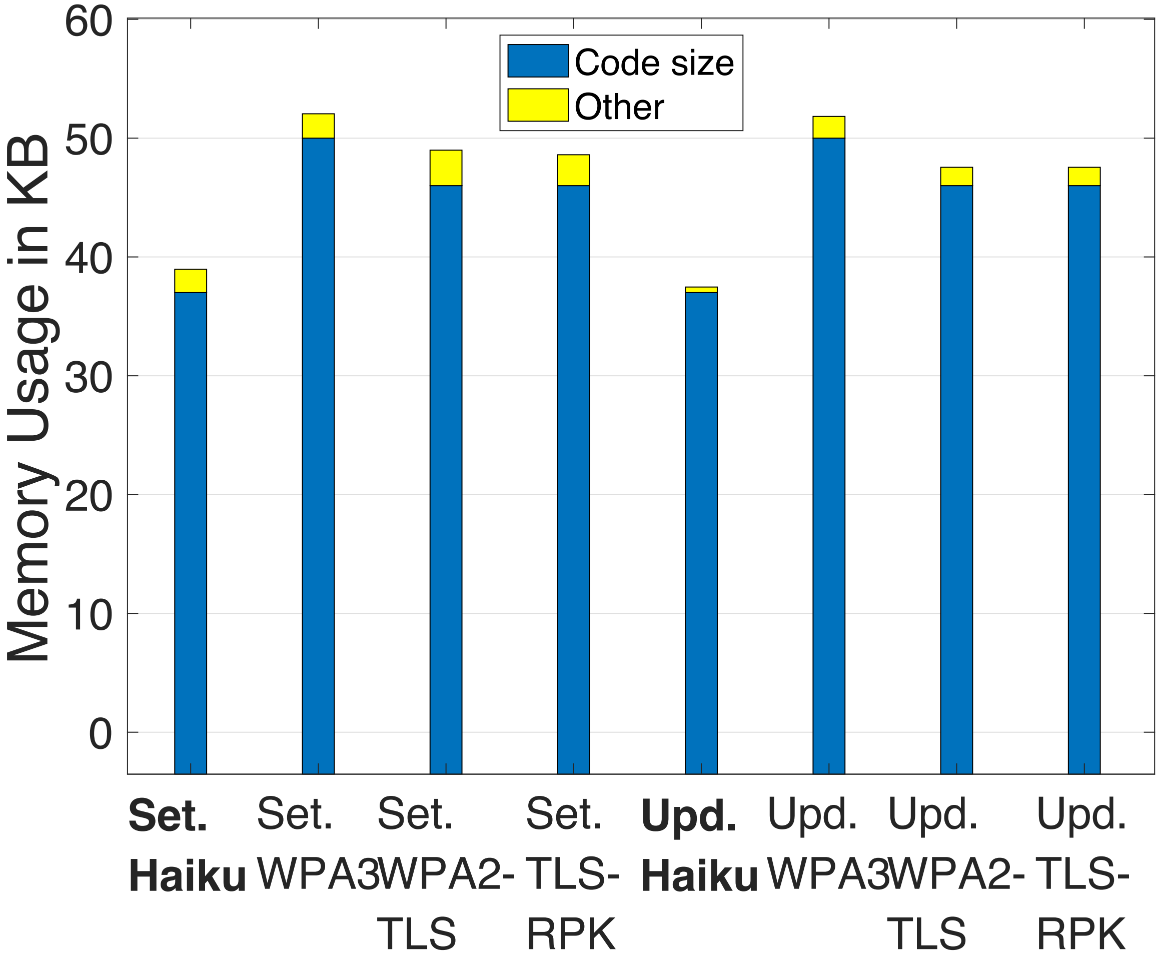

Figure 3.10: memory footprints.

Chapter 4

MSS: Lightweight network authentication for resource constrained devices via Mergeable Stateful Signatures

With the increasing deployment of resource-limited devices (*e.g.,* sensors and Internet of Things devices (IoTs)) [8], designing secure systems with low computational overhead has become a critical issue. When devices have limited computational power, memory and/or energy reserves, security often takes a back seat to reducing protocol latency, reducing CPU and memory footprint, and lowering energy consumption. Several high profile attacks in recent years (*e.g.,* Mirai botnet [54] and BrickerBot [188]) highlight the need for better security for these devices [172] since their high scale can cause serious consequences

Table 4.1: Profile of typical IoT energy consumption and PKC computational latency on Arduino Uno running at 16 MHz.

Operation	Energy Consumption	Operation	Latency (seconds)	
			RSA	EdDSA
PKC	51.81%	Sign/Decrypt	731.52	6.11
Sensing	48.15%	Verify/Encrypt	8.26	9.72
Communication	0.03%	-	-	-

if they are maliciously controlled by adversaries. Such devices include, but not limited to, smart home, smart cities, ..etc.

Authentication is a central challenge in secure protocol design for edge devices, where digital signatures – the traditional solution from cryptography – are too costly. Unlike other settings, the IoT environment often has a special system model in which IoT devices frequently communicate a small amount of authenticated data to a single server (*e.g.,* a gateway/sink). For instance, the Message Queue Telemetry Transport (MQTT [23]), a popular IoT networking protocol, uses a publish/subscribe paradigm where IoT devices periodically publish authenticated data to the same MQTT broker. Quick authentication of such data can be critical to saving lives and businesses; consider a patient with irregular heart beats or blood pressure, and thus her doctor must instantly be warned for quick response in case of emergency. Moreover, IoT devices are often powered by limited rechargeable batteries, so the authentication solution must not consume high energy. Traditional public-key cryptography (PKC) authentication is a computational bottleneck in IoT applications due to the performance constraints of the IoT device [11]; Table 4.1 shows the high computational latency of two digital signature standards on an IoT board along with the high percentage of typical IoT device battery that PKC uses. Often symmetric key cryptography (SKC) is used instead, which imposes key-management issues and introduces new security vulnerabilities; for instance, SKC requires the server to store IoT authentication keys, which makes the IoT devices subject to impersonation attacks if the server is compromised and the keys are stolen. Authentication based on hash chains overcomes traditional SKC shortcomings, but it has a lifespan and requires expensive computation (please see Section 4.4 for details). Therefore, new public-key authentication solutions that are efficient in the amount of computation and energy usage are needed for IoT.

In this work, we design a novel signature scheme which yields an authentication protocol with low overhead. Our scheme, which we call (MSS), operates in a model where the verifier is assumed to always be the same party (*e.g.,* an MQTT broker). This allows state to be maintained across multiple signatures, which in turn allows for efficiency improvements over standard signatures. We analyze the security of MSS in the offline/online model of Even, Goldreich and Micali [63], where the signing algorithm is split into two parts. The first part is costlier and can be performed offline, but importantly, *before the message to sign is known.* The second part is online, and can make use of the result of the offline computation to provide low cost signature. The efficiency of our online phase is tied to the length of the message being signed. Our scheme is most efficient for short messages,

Table 4.2: Example IoT applications for MSS and efficiency gains.

Application	Number of Messages (per Day)	Message Size (bits)	MSS-RSA vs. base RSA		MSS-ECELGamal vs. base ECELGamal	
			Speedup	Extra Battery Lifetime	Speedup	Extra Battery Lifetime
Heart rate monitor [18, 160]	144-1,440	8	32x	2x	-	-
Continuous glucose monitor [66, 126]	288	9	28x	2x	-	-
Temperature sensor [122, 165]	1,440	12	21x	2x	-	-
Soil moisture sensor [135, 52]	1,234	14	18x	2x	-	-
Vehicle tracker [115, 123]	2,880	57	4x	1.8x	-	-
Humidity sensor [178, 182]	1,440	14	18x	2x	-	-
Smart electricity meter [186]	24-1,440	40	6x	1.8x	-	-
People counting sensor [84, 16]	96	17	15x	1.9x	-	-
Water level sensor [111, 83]	96	7	36x	2x	-	-
Smart lock [111, 83]	368	21	-	-	2x	1.54x
Drone command and control (1hr use) [111, 83]	36,000	4	-	-	2.1x	1.56x

and efficiency degrades as the message length grows. For messages which are 256 bits or longer (in which case, we would sign a 256-bit *hash* of the message), the online phase is no faster than a standard public-key signature. Thus, our intended use case is where the IoT device is frequently sending small amounts of data to a single server. It is likely that this pattern will be pervasive in IoT and other cyberphysical systems; for example, Table 4.2 shows some recent applications that have an IoT client or a sensor communicating with a single server, and the efficiency improvements introduced by MSS. Furthermore, MSS can also be utilized to reduce the signature verification cost when the client-server roles are switched and the IoT device becomes the server/verifier, which makes our scheme versatile and useful in other applications (*e.g.,* last two applications in Table 4.2).

We present MSS abstractly in Section 4.1 and implement it twice, within two digital signature standards: RSA signatures and elliptic curve ElGamal signatures. In

Section 4.2 we demonstrate a concrete application; specifically, we use MSS to implement a time-based one-time password (TOTP) authentication protocol. TOTP systems allow a user to authenticate herself to a server using a "one-time" password that is valid for a short, fixed time. After the time expires, the user will have to authenticate herself again using another password.

Our experiments (implemented on Raspberry Pis) show that the new MSS-based TOTP systems provide appealing efficiency gains (in Section 5.6). Our RSA-based system cuts down authentication latency and energy consumption by 12 and 20 times, respectively, compared to a traditional RSA-based system. Additionally, our elliptic curve ElGamal (ECElGamal) based system reduces authentication latency and energy consumption by 2 and 3 times, respectively, compared to traditional ECElGamal/ECDSA/EC-Schnorr-based system. Our ECElGamal-based system also reduces authentication latency and energy consumption by ~82 and 792 times, respectively, compared to a recent TOTP system based on hash chains *(with client storage of first hash of each week)* [100] but requires double the password size.

We also present an asymptotic analysis of MSS's efficiency in Section 4.4. We show that MSS can be used in place of hash chains and reduces their online time×space complexity from $\mathcal{O}(N)$ to $\mathcal{O}(\mathsf{polylog}(N))$ where N is the number of signatures. We further show that our ECElGamal implementation allows us to reduce the offline time to $\mathcal{O}(\mathsf{polylog}(N))$ using specifics of the ECElGamal signature scheme. Section 5.4 presents a formal proof of the security of MSS, by defining an incremental forgery game and showing that the attacker cannot win the game (break the system) provided the underlying scheme is secure.

4.1 MSS via Mergeable Signatures

In this section, we describe our new signature scheme, MSS (Mergeable Stateful Signatures). MSS allows a client to authenticate herself to a single server efficiently multiple times typically spread over different transactions/times. As mentioned above, the assumption that the receiver is always the same party allows maintaining state across multiple signatures which allows improving efficiency.

4.1.1 Signature Preliminaries

Digital signatures [74] are fundamental objects from cryptography. Formally, a signature scheme consists of three algorithms $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ satisfying the syntax:

- KeyGen takes a security parameter as a unitary input and generates a key pair $(\mathsf{vk}, \mathsf{sk})$;
- Sign takes sk and a message as input and outputs a signature σ;
- Verify takes a message/signature pair and vk as input and outputs a bit indicating whether the signature is valid.

Additionally, the two properties *correctness* and *security* must hold. Correctness says that for all messages msg, if $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^n)$ and $\sigma \leftarrow \mathsf{Sign}(\mathsf{msg}, \mathsf{sk})$, then $\mathsf{Verify}(\mathsf{vk}, \mathsf{msg}, \sigma) = 1$. Intuitively, security demands that without possession of the secret key, nobody can produce a valid signature for a new message. We formally prove security of MSS in Section 5.4.

Our main construction, MSS, builds on top of signature schemes which support a special malleability property which we call *mergeability.*[1] This property is the

[1] We present MSS via mergeable signatures for modularity − By abstracting the main part of the construction so it builds on top of a general intermediate primitive, we are able to concretize MSS based on several different cryptographic assumptions (e.g., RSA, BLS, or even Lattice assumptions).

same as that of homomorphism. Roughly speaking, a signature scheme is *mergeable* if two message/signature pairs $(\mathsf{msg}_1, \sigma_1)$ and $(\mathsf{msg}_2, \sigma_2)$ can be merged to obtain a new message/signature pair $(\mathsf{msg}^*, \sigma^*)$. This operation must also be invertible in the sense that given $(\mathsf{msg}^*, \sigma^*)$ and $(\mathsf{msg}_1, \sigma_1)$ one can recover $(\mathsf{msg}_2, \sigma_2)$. It is important that this must be a *public* operation, so it does not require knowledge of the secret key. Security demands that without the secret key, nobody can produce a valid signature on a new message even one that can be created by merging together two (or more) message/signature pairs which they have already seen signed.

Formally, we say that a signature scheme $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ is *mergeable* if there exist two additional algorithms Merge and Reconstruct $(\mathsf{Merge}, \mathsf{Rec})$ which both take two message/signature pairs $(\mathsf{msg}_1, \sigma_1), (\mathsf{msg}_2, \sigma_2)$ as input and output a message/signature pair $(\mathsf{msg}^*, \sigma^*)$. Moreover, two additional properties must hold: 1) Merge and Rec are inverses of each other; and 2) if the input signatures are valid then so is the merged signature. Formally, (1) requires that $(\mathsf{msg}_2, \sigma_2)$ equals

$$\mathsf{Rec}\Big((\mathsf{msg}_1, \sigma_1), \mathsf{Merge}\big((\mathsf{msg}_1, \sigma_1), (\mathsf{msg}_2, \sigma_2)\big)\Big).$$

(2) requires that if $\mathsf{Verify}(\mathsf{vk}, \mathsf{msg}_i, \sigma_i) = 1$ for $i = 1, 2$, then $\mathsf{Verify}(\mathsf{vk}, \mathsf{msg}^*, \sigma^*) = 1$ where $(\mathsf{msg}^*, \sigma^*)$ is one of

$$\big\{\mathsf{Merge}, \mathsf{Rec}\big\}\big((\mathsf{msg}_1, \sigma_1), (\mathsf{msg}_2, \sigma_2)\big).$$

The reader may wish to keep in mind the example

$$\mathsf{Merge}\big((\mathsf{msg}_1, \sigma_1), (\mathsf{msg}_2, \sigma_2)\big) = (\mathsf{msg}_1 \cdot \mathsf{msg}_2, \sigma_1 \cdot \sigma_2).$$

This will essentially be the case in both of our constructions (MSS on RSA and on elliptic curve Boneh-Lynn-Shacham, BLS) with the precise meaning of multiplication $(\cdot)$ customized

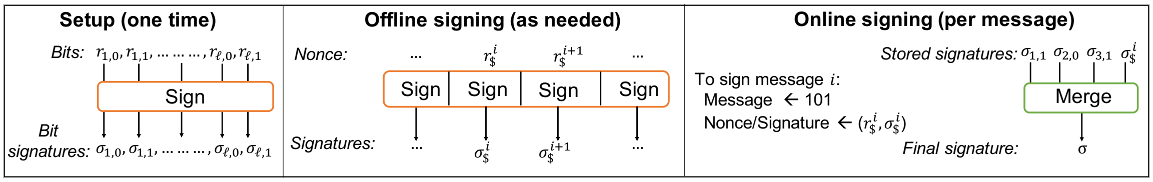
Setup (one time)
Bits: $r_{1,0}, r_{1,1}, \ldots\ldots\ldots, r_{\ell,0}, r_{\ell,1}$
Sign
Bit signatures: $\sigma_{1,0}, \sigma_{1,1}, \ldots\ldots\ldots, \sigma_{\ell,0}, \sigma_{\ell,1}$
Offline signing (as needed)
Nonce: … $r_\$^i$ $r_\$^{i+1}$ …
Sign
Sign
Sign
Sign
Signatures: … $\sigma_\$^i$ $\sigma_\$^{i+1}$ …
Online signing (per message)
Stored signatures: $\sigma_{1,1}$ $\sigma_{2,0}$ $\sigma_{3,1}$ $\sigma_\$^i$
To sign message i:
Message ← 101
Nonce/Signature ← $(r_\$^i, \sigma_\$^i)$
Merge
Final signature: σ

Figure 4.1: MSS overview: 1) Setup: user signs ℓ pairs of random strings (*i.e.,* signs 2ℓ random strings) where each pair is mapped to a bit location and each pair string is a random value representing either 0 or 1; 2) Offline signing: user apriori signs a fresh nonce and keeps it in storage, which will exclusively be used for next message; 3) Online signing: user signs a message by merging/multiplying already stored signatures (in setup and offline signing) corresponding to the message bits and nonce.

for each cryptographic assumption. For example, it would be modular multiplication for the RSA assumption and point addition for elliptic-curve based assumptions.

4.1.2 MSS Overview

Fig. 4.1 overviews how MSS works. The main idea is, instead of signing a message string all at once, use pre-computed signatures of "representative random strings" for each bit of (a hash of) the message. So as a starting (flawed) example, suppose the message hash h is $\ell-$bits long, and the scheme had chosen ℓ pairs of random strings, $\{r_{i,0}, r_{i,1}\}_{i=1,\ldots,\ell}$, (which are made public) and individually signed each string obtaining signatures $\{\sigma_{i,0}, \sigma_{i,1}\}_{i=1,\ldots,\ell}$ (which are kept private). Now each pair of signatures is mapped to a bit location to represent signatures on its bit values (*i.e.,* 0 or 1). Then one could sign a message by sending the signatures which correspond to the bits of h. So, for example, if $h = 101$, the signature would be $(\sigma_{1,1}, \sigma_{2,0}, \sigma_{3,1})$ such that $\sigma_{3,1}$ represents the signature on the value of the third digit in h, which is 1. Note that in this naive implementation, signing a single message requires signing multiple random strings and sending multiple signatures, thus sign time, signature size and verification time are drastically increased. On the other hand, the signer can precompute the $\sigma_{i,b}$'s , $b \in \{0, 1\}$, once during a setup phase and reuse them every time she signs a message, thus obtaining a scheme with very fast online sign-time. The main problem is that this naive scheme will not be secure since if an adversary sees many messages signed, she will eventually learn all of the $\sigma_{i,b}$'s and be able to sign new messages by herself.

To fix this problem with security, we use mergeable signatures and rather than sending all of the representative signatures in the clear (which is inefficient and insecure), we merge them into a single signature. For security reasons, we also merge in a signature on a fresh random nonce (which can be generated offline). So to summarize, the random representative strings and their signatures (two for each bit) are computed one time during setup, then several nonces and nonce signatures are computed offline and stored. Then once this data is in place, all the client has to do to compute the signature is merge together several of the signatures she has already computed. In our instantiation of mergeable signatures, the merge algorithm is much faster than the signing algorithm. Thus, the online cost of signing a message in MSS is greatly reduced.

MSS makes online time/offline time/space tradeoffs available to the signer; while the signer needs to have *one nonce signature* ahead of time to achieve fast online signing for the next message, she typically would store a number of nonce signatures at a time (*e.g.,* computed when idle/charging [19], or replenished periodically from a trusted proxy that is assigned the offline computation) as shown in Fig. 4.1. Thus, we do not view the number of available nonces as a limit on the number of signatures available between the signer and verifier.

4.1.3 MSS via Mergeable Signatures - The Main Contribution

In this section we build our main contribution "MSS" assuming a general mergeable signature scheme. In order to make authentication efficient for devices with small computational power as promised in the introduction, we analyze MSS in the online/offline model of Even, Goldreich and Micali [63]. In this model, the Sign algorithm is split into two algorithms $(\mathsf{Sign}_{\mathsf{off}}, \mathsf{Sign}_{\mathsf{on}})$ representing the offline and online procedures. The syntax is that $\mathsf{Sign}_{\mathsf{off}}$ takes sk (but not the message to sign) as input and produces output τ; $\mathsf{Sign}_{\mathsf{on}}$ takes $(\mathsf{sk}, \mathsf{msg}, \tau)$ and outputs the signature σ. Ideally, $\mathsf{Sign}_{\mathsf{on}}$ should be significantly more efficient than $\mathsf{Sign}_{\mathsf{off}}$. The intended use case is that $\mathsf{Sign}_{\mathsf{off}}$ is run offline before the message to sign is known to allow considerable speed and energy advantages for the online signing procedure.

Assume $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Merge}, \mathsf{Rec})$ to be a mergeable signature scheme, let H be a hash function modeled as a random oracle and let $\ell \in \mathbb{N}$ be a length parameter. MSS consists of four algorithms $(\mathsf{KeyGen}', \mathsf{Sign}'_{\mathsf{off}}, \mathsf{Sign}'_{\mathsf{on}}, \mathsf{Verify}')$ and supports signatures on $\ell-$bit messages. In the following, we assume that verification keys are included as part of the signing keys (this saves some syntax since it prevents us from having to explicitly pass the verification keys to the signing algorithms). We allow Merge and Rec to take many inputs rather than just two, without loss of generality: we can repeatedly apply the two-input version.

Algorithm 1 $\mathsf{KeyGen}'(1^n)$

```
$(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^n)$;
Initialize two $2 \times \ell$ arrays $B$, initialized with random strings, and $\Sigma$, empty; also initialize $U = [\,]$ an empty list;
Set $\Sigma = B.\mathsf{map}\big(r \rightarrow \mathsf{Sign}(\mathsf{sk}, r)\big)$;
**Output** $(\mathsf{vk}', \mathsf{sk}') = \big((\mathsf{vk}, B, U), (\mathsf{sk}, \Sigma)\big)$;
```

Algorithm 2 $\mathsf{Sign}'_{\mathsf{off}}(\mathsf{sk}')$

```
Choose a random nonce $r_\$$ and set $\sigma_\$ \leftarrow \mathsf{Sign}(\mathsf{sk}, r_\$)$;▷ where $\mathsf{vk}, \mathsf{sk}$ are part of $\mathsf{sk}'$
**Output** $\tau' = (r_\$, \sigma_\$)$;
```

Algorithm 3 $\mathsf{Sign}'_{\mathsf{on}}(\mathsf{sk}', \mathsf{msg}, \tau')$

```
Parse $\mathsf{msg} = b_1 \cdots b_\ell$ as bits, and parse $\tau' = (r_\$, \sigma_\$)$;
Set $(r, \sigma) = \mathsf{Merge}\big(\{(r_{j,b_j}, \sigma_{j,b_j})\}_j, (r_\$, \sigma_\$)\big)$;     ▷ $r_{j,b_j} \in B$ and $\sigma_{j,b_j} \in \Sigma$
**Output** $\sigma' = (r, \sigma)$;
```

Algorithm 4 $\mathsf{Verify}'(\mathsf{vk}', \mathsf{msg}, \sigma')$

```
Parse $\mathsf{msg} = b_1 \cdots b_\ell$ and $\sigma' = (r, \sigma)$;
Compute $(r_\$, \sigma_\$) = \mathsf{Rec}\big(\{(r_{j,b_j}, \sigma_{j,b_j})\}_j, (r, \sigma)\big)$; if $r_\$ \in U$, reject and exit;     ▷ This means the nonce $r_\$$ was used previously
$U.\mathsf{push}(r_\$)$
**Output** $\mathsf{Verify}(\mathsf{vk}, r, \sigma) = 1$
```

Remarks. Some remarks on Algorithm 1–4 are in order.

1. Notice that the offline signing algorithm, $\mathsf{Sign}'_{\mathsf{off}}$, signs a random string, while the online algorithm, $\mathsf{Sign}'_{\mathsf{on}}$, calls Merge. For all of the mergeable schemes we build

in subsequent sections, Merge will be significantly more efficient than Sign. This is obvious because the number of mathematical operations are much smaller.

2. The list U represents some persistent state maintained by the verifier over time. The purpose is to keep track of all nonces (the $r_{\$}$ values) used so far to force the signing algorithm to choose a fresh nonce for each new signature. This is important for security.

3. Note that the runtime of $\mathsf{Sign}_{\mathsf{on}}$ grows with ℓ since it requires merging $\ell + 1$ signatures together. Thus our scheme is most efficient when ℓ is small.

We have not yet discussed where one finds a mergeable signature scheme to use for the construction. Mergeable signatures are, in fact, not hard to find. We show that the two very common signature schemes, RSA signatures and BLS (or, a discrete log/elliptic curve based scheme with a formal security proof [37]) signatures, both support Merge and Rec operations (see sections 4.1.4 and 4.1.5). We have also analyzed the security of our construction in Section 5.4.

4.1.4 Mergeable Signatures via RSA

The arithmetic for the RSA signature scheme [157] takes place modulo a composite integer $N = pq$ which is the product of two primes. The scheme works as follows.

- $\mathsf{KeyGen}(1^n)$**:** draws $N = pq$ and e according to the RSA distribution, computes $d = e^{-1} \pmod{\phi(N)}$ (using its knowledge of p and q)[2] and outputs $(\mathsf{vk}, \mathsf{sk})$ where $\mathsf{vk} = (N, e)$ and $\mathsf{sk} = (N, e, d)$;

[2] Here $\phi(N)$ is Euler's totient function: $\Phi(N) = (p - 1)(q - 1)$

- $\mathsf{Sign}\big(\mathsf{msg}, (N, e, d)\big)$**:** let $r = \mathsf{msg} \pmod N$ and output $\sigma = r^d \pmod N$;
- $\mathsf{Verify}\big((N, e), \mathsf{msg}, \sigma\big)$**:** compute $r = \mathsf{msg} \pmod N$, if $\sigma^e \equiv r \pmod N$ output 1, otherwise output 0.

The Merge and Rec algorithms for RSA are simply modular multiplication:

- $\mathsf{Merge}\big((r_1, \sigma_1), (r_2, \sigma_2)\big) = (r_1 r_2, \sigma_1 \sigma_2)$
- $\mathsf{Rec}\big((r_1, \sigma_1), (r_2, \sigma_2)\big) = (r_1 r_2^{-1}, \sigma_1 \sigma_2^{-1})$

Note that if $\mathsf{vk} = (N, e)$ and (σ_1, σ_2) are valid signatures on messages (r_1, r_2), then $\sigma_i^e = r_i \pmod N$ holds for $i = 1, 2$. Therefore, if

$$(r, \sigma) = \mathsf{Merge}\big((r_1, \sigma_1), (r_2, \sigma_2)\big) = (r_1 \cdot r_2, \sigma_1 \cdot \sigma_2),$$

then

$$\sigma^e = (\sigma_1 \cdot \sigma_2)^e = \sigma_1^e \cdot \sigma_2^e \equiv r_1 \cdot r_2 = r,$$

and so σ is a valid signature of r.

4.1.5 Mergeable Signatures via BLS

The arithmetic in BLS scheme [37] takes place in a cyclic group G with generator g that is equipped with a pairing map $e : G \times G \to G_T$ for another group G_T (called the target group) such that $e(g, g) \neq 1$ and $e(g^a, g^b) = e(g, g)^{ab}$ for all integer exponents a, b. The syntax of the scheme is as follows:

- $\mathsf{KeyGen}(1^n)$**:** draws G and g, and draws a random exponent x, and outputs $(\mathsf{vk}, \mathsf{sk})$ where $\mathsf{vk} = (G, g, g^x)$ and $\mathsf{sk} = (G, g, x)$;

- $\mathsf{Sign}\big(\mathsf{msg}, (G, g, x)\big)$**:** put $r = \mathsf{msg}$ and output (r, r^x);
- $\mathsf{Verify}\big((G, g, g^x), \mathsf{msg}, \sigma\big)$**:** parse $\sigma = (r, r')$; check that (g, g^x, r, r') is a DDH tuple [38], if so output 1, otherwise output 0.

The Merge and Rec operations here are also based on multiplication:

- $\mathsf{Merge}\big((r_1, \sigma_1), (r_2, \sigma_2)\big) = (r_1 r_2, \sigma_1 \sigma_2)$
- $\mathsf{Rec}\big((r_1, \sigma_1), (r_2, \sigma_2)\big) = (r_1 r_2^{-1}, \sigma_1 \sigma_2^{-1})$

Similar to RSA, the mergeability can be verified as follows. If $\mathsf{vk} = (G, g, g^x)$ and (σ_1, σ_2) are valid signatures on messages (r_1, r_2), then $\sigma_i = (r_i, r_i^x)$ holds for $i = 1, 2$. Therefore, if

$$(r, \sigma) = \mathsf{Merge}\big((r_1, \sigma_1), (r_2, \sigma_2)) = (r_1 \cdot r_2, \sigma_1 \cdot \sigma_2),$$

then

$$\sigma = \sigma_1 \cdot \sigma_2 = (r_1 \cdot r_2, r_1^x \cdot r_2^x) = (r_1 \cdot r_2, (r_1 \cdot r_2)^x),$$

therefore, when the verification algorithm parses σ as (r, r') and checks whether $e(g^x, r) = e(g, r')$, it passes. This is because, $e(g^x, r) = e(g^x, r_1 \cdot r_2) = e(g, (r_1 \cdot r_2)^x)$.

4.2 MSS Application: Time based One Time Password (TOTP) Systems

Two factor authentication and similar techniques have been introduced to combat user password weaknesses. Several hardware tokens (*e.g.,* YubiKey [198]) are used today as second factor authenticators, which rely on standard bidirectional communication based challenge-response authentication. As it is becoming more popular to use devices such as IoT devices as second factor authenticators, it might be the case that such devices are offline or have only one-way communication (*e.g.,* from second factor device to authenticating device); in such case, systems like Duo [164] fall back to one-time password implementations (*e.g.,* [116]). In these implementations, however, the next password stays valid until the next authentication (which could be a long time); this makes it subject to various attacks such as phishing. Time-based OTP systems mitigate this issue by assigning each authentication time period tp a different password, and thus limiting the password window-of-use and protecting users from such attacks.

Hash chains have been proposed to implement TOTP systems that do not share secrets with the server while providing time/space tradeoffs for user efficiency (refer to Section 4.4 for hash chain authentication). Assuming N represents the hash chain-TOTP system lifespan, a standard choice of parameters indicates $N \approx 2^{21}$ (lifespan of roughly 2 years), with each hash in the chain representing half a minute; nonetheless, their time$\times$space complexity is $\mathcal{O}(N)$, which is expensive for constrained users and servers. MSS can provide attractive properties (in terms of time *and* space) compared to hash chains (see Section 4.4 for details).

In this section, we implement an MSS based TOTP protocol, which goes through setup and authentication phases (Fig. 4.2). It allows the device's online authentication algorithm to rely on only lightweight (*i.e.*, non-cryptographic) operations in order to produce/verify a TOTP for a *tp.* Our RSA-based implementation of the protocol significantly speeds up signing at the online time, compared with traditional RSA while our ElGamal-

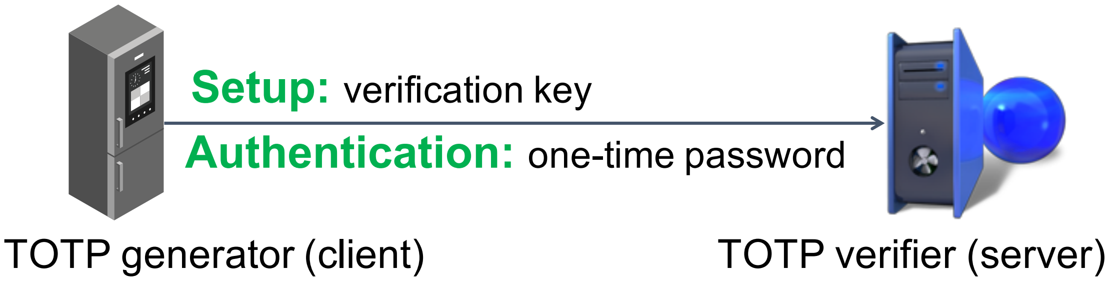

Figure 4.2: TOTP system overview.

based implementation improves the efficiency of verification by taking advantage of the merging construction for BLS. Optimizing verification is important for situations where the constrained devices are verifiers (*e.g.*, smart locks).

Remark on ElGamal Signatures. Although ElGamal-based signatures are similar to BLS-based ones in that they are both implemented in discrete-log groups, they lack security proof [5]. Nearly all of the arithmetic used in BLS carries over to the ElGamal setting, so we think of the implementation here *morally* as an implementation of our scheme MSS. We stress to the reader that the discrepancy here is in some sense unavoidable: had we worked with ElGamal signatures in Section 4.1 and 5.4, we would not have been able to prove rigorous security since even the basic ElGamal scheme has no security proof. Likewise, if we were to implement the exact BLS-based protocol our results would have given us an unfair advantage because base BLS signatures are so much slower than base ElGamal.

4.2.1 RSA-based TOTP System

Setup Whenever a client wishes to use a TOTP system as a means for authentication, it has to first go through a one-time setup so that both nodes are configured with proper keys

Algorithm 5 Authentication

```
$\sigma \leftarrow \mathsf{Sign}_{\mathsf{off}}(arg_1, arg_2)$;
$\hat{\sigma} \leftarrow \mathsf{Sign}_{\mathsf{on}}(\mathsf{sk}, tp, \sigma_{\$}, arg_3)$;        ▷ Server runs Verify()
if $\mathsf{Verify}(\mathsf{vk}, tp, \hat{\sigma}, arg_3) = 1$ then
    ACCEPT;
else
    REJECT;
end if
```

using the steps of KeyGen' described in Algorithm 1. The client defines $B := \{r_{i,b} \in \mathbb{Z}_N^* | i \in [1, d], b \in \{0, 1\}\}$ that represents all the bit values, where every $r_{i,b}$ is randomly chosen. The client computes the set of bit signatures Σ, which contains $\mathsf{Sign}(\mathsf{sk}, r)$ for each $r \in B$, where Sign works as Sign described in Appendix 4.1.4. Next, it shares the tuple (B, vk, st) with the server, where 'st' denotes the initial time to use the system.

Authentication Whenever the client wants to authenticate itself, it uses st to infer the current authentication time period tp represented in binary (*i.e.,* $tp \in \{0, 1\}^d$ with some pre-agreed on encoding of the time). The authentication procedure is described in Algorithm 5, for which details are as follows:

- Defines $\hat{B} := \{r_{i,b} \in B | i \in [1, d], b \in \{tp_i\}\}$ and $\sigma^* := \{\sigma_{i,b} \in \Sigma | i \in [1, d], b \in \{tp_i\}\}$;
- Draws a nonce $r_{\$}$ and computes $H(\mathsf{vk}, r_{\$})$, $GCD(H(\mathsf{vk}, r_{\$}), r) = 1$ for all $r \in \hat{B}$ to avoid signature overlap;

- Computes nonce signature as $\sigma_{\$} \leftarrow \mathsf{Sign}_{\mathsf{off}}(\mathsf{sk}, r_{\$}) = H(\mathsf{vk}, r_{\$})^{\mathsf{sk}.d} \mod N$ and sets $\sigma^* := \sigma^* \cup \{\sigma_{\$}\}$;
- Runs $\hat{\sigma} \leftarrow \mathsf{Sign}_{\mathsf{on}}(\mathsf{sk}, tp, \sigma_{\$}, r_{\$}) = \prod_{\sigma_i^* \in \sigma^*} \sigma_i^* \mod N$;

It then sends the TOTP tuple $(r_{\$}, \hat{\sigma})$ to authenticate itself.

Upon receipt of the TOTP, the server infers the same tp from its own clock and the shared st to avoid accepting void TOTPs that might have been stolen/replayed. In order to verify whether the received TOTP is legitimate, it does the following:

- Creates a verification subset $\hat{V} := \hat{B} \cup \{H(\mathsf{vk}, r_{\$})\}$;
- Runs $\mathsf{Verify}(\mathsf{vk}, tp, \hat{\sigma}, r_{\$})$, which checks if the following equation holds $\prod_{v \in \hat{V}}(v) \overset{?}{=} \hat{\sigma}^{\mathsf{vk}.e} \mod N$;

If successful, the client is authenticated.

Efficiency and Overhead In order to avoid communicating nonces $r_{\$}$ during authentication, both nodes can be configured to derive unique and coprime nonces using a pseudo-random function (*e.g.,* $r_{\$}^i = H(counter)$). Since, in TOTP systems, tp increases overtime and that the client uses its TOTP generation algorithm only once in each time-interval, tp can be used by both nodes to also derive the nonces $\{r_{\$}\}_i$ (*e.g.,* $r_{\$}^i = H(tp)$) without loss of security; this releases the server from keeping state in memory as well as allows both nodes to avoid communicating such nonces over the network during authentication. The enhancement in [39] can also be used in our case to reduce the size of signature set Σ and bit value set B to half – essential for edge devices with limited storage; instead of explicitly dealing with values representing 0 for each different digit, we assume presence of

1 implies absence of 0 and vice versa. This allows clients to store signatures corresponding to values representing 1 for all digits only, namely $\Sigma' = \{\sigma_{i,1} \in \Sigma\}_{i\in[1,d]}$. Similarly, the server only keeps values representing 1 for all digits, namely $B' = \{r_{i,1} \in B\}_{i\in[1,d]}$. It can also use one set of coprime values representing different bits (*i.e.,* , the set B') for all clients in the system. Clients can also store only one key of the key pair, preferably vk since the recommended size of the public verification exponent $\mathsf{vk}.e \in \{0,1\}^{\geq 17}$ is smaller than the private signing exponent $\mathsf{sk}.d \in \{0,1\}^{\geq 0.292\cdot|N|}$ [36], and derive one key from the other as needed.

4.2.2 ECElGamal-based TOTP System

In several DDH-based signatures (*e.g.,* ECDSA), the most expensive part of the signing operation can by design be done offline; however, signature verification still has to incur at least 2 exponentiations in the critical online time, which can be expensive for a limited-resource verifier (*e.g.,* a smart lock). Our MSS-based ECElGamal construction allows us to replace one of the exponentiations with only a few multiplications (*e.g.,* 21 multiplications for a 2-year TOTP system), thus reducing online verification time significantly.

Setup When the instantiation of Algorithm 1 is done using ECElGamal signatures, the output key pair is $(\mathsf{sk} = (G, g, x), \mathsf{vk} = (G, g, xg))$; where g is the public base point for the elliptic curve, $\mathsf{sk}.x \in [2, N-1]$ is a secret random scalar and $\mathsf{vk}.xg$ is a point on the elliptic curve G. In the second step, B is chosen in such a way which guarantees that the overlap amongst signatures of different authentication times is avoided. Hence, the following constraint: $\sum_{r^* \in B^*}(r^*) \neq \sum_{r' \in B'}(r'), \forall B^* \neq B' \subset B$ must be satisfied. Therefore, we choose $B = \{r_{i,b} = \ell^e \mod N | i \in I = [1,d], b \in J = \{0,1\}, \ell \in \mathbb{Z}_N, e \in [1, |I| \cdot |J|]\}$, where N is a large prime determining the order of an elliptic curve group. KeyGen' in this case is the same as the one in Algorithm 1 except that the client sends (B, vk, st) to the server and the server computes the last step of KeyGen' as $\Sigma = B.\mathsf{map}\big(r \rightarrow g \times r\big)$.

Authentication This procedure follows the same protocol described in Algorithm 5 and the three procedures ($\mathsf{Sign}_{\mathsf{off}}$, $\mathsf{Sign}_{\mathsf{on}}$ and Verify) are computed as follows. The signature of tp (current time to authenticate) consists of the tuple $(\sigma_\$, \hat{\sigma})$. Similar to normal ECElGamal signatures, $\sigma_\$ = (x_1, y_1)$ is a point on the curve calculated by $\sigma_\$ = \mathsf{Sign}_{\mathsf{off}}(r_\$, g) = r_\$ \times g$, where $r_\$ \in [2, N-1]$ is an integer nonce that is coprime with the modulus N (*i.e.*, $GCD(r_\$, N) = 1$) so that it has a multiplicative inverse $r_\$^{-1}$ that will be needed to calculate the second part of the signature $\hat{\sigma}$; $r_\$$ must be unique and random for each signature to prevent $\mathcal{A}$ from recovering sk [59]. In order to compute any $\sigma_\$$, we need to only have the set $P = \{\alpha \times g : \alpha \in \{2^0, \ldots, 2^{\log(N)-1}\}\}$ pre-computed and stored in memory.

In order to calculate $\hat{\sigma}$, the client creates the subset $\hat{B} = \{r_{i,b} \in B\}_{i \in [1,d], b \in \{tp_i\}}$ that contains the digit values corresponding to its current tp. It then runs $\hat{\sigma} \leftarrow \mathsf{Sign}_{\mathsf{on}}(\mathsf{sk}, tp, \sigma_\$, r_\$)$ which outputs the second part of the signature, $\hat{\sigma} = r_\$^{-1}(\sum_{r \in \hat{B}}(r) - \mathsf{sk}.x \cdot \sigma_\$.x_1) \mod N$. It communicates the TOTP tuple $(\sigma_\$, \hat{\sigma})$ to authenticate itself. The server now creates a signature verification subset $\hat{\Sigma} = \hat{B}.\mathsf{map}\big(r \rightarrow g \times r\big) \subset \Sigma$, which contains already computed values (at setup) corresponding to tp; it then runs the procedure $\mathsf{Verify}(\mathsf{vk}, tp, \sigma_\$, \hat{\sigma})$ to check the validity of $\sigma_\$.x_1 \times \mathsf{vk}.xg + \hat{\sigma} \times \sigma_\$ \overset{?}{=} \sum_{v \in \hat{\Sigma}} v$.

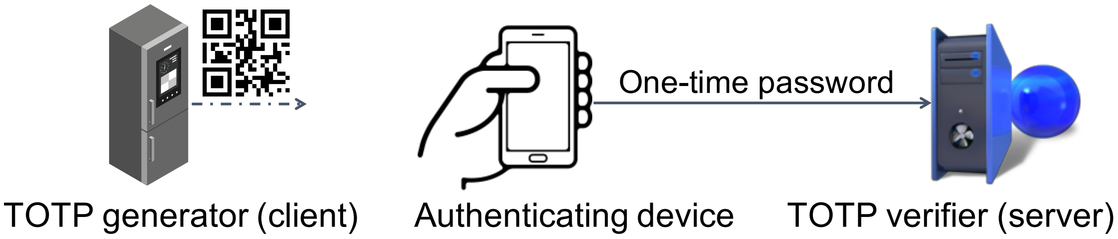

Figure 4.3: Offline second factor authentication.

Efficiency and Overhead The technique mentioned earlier that suggests keeping values corresponding to 1 only for all digits is also applicable here (*i.e.,* Σ' and B'). Since $\sigma_\$$ is independent of *tp*, it can be calculated and sent ahead of time, allowing the server to calculate part of the left side of the verification equation apriori, namely $\sigma_\$.x_1 \times \mathsf{vk}.xg$; thus, the only expensive operation left for signature verification becomes $\hat{\sigma} \times \sigma_\$$. This approach enables our TOTP system to be faster in the online authentication time. It also allows clients to send only half the signature at the critical time of authentication, namely $\hat{\sigma} \in \{0,1\}^{|N|}$. NIST suggests that elliptic curves with $|N| = 256$ can provide 128-bit security [24]; if we follow this recommendation, our system can have clients send only 256-bit TOTP tokens at the online authentication time. The server can also use one set of values representing the different digits (*i.e.,* Σ') for all clients to avoid per user storage.

4.2.3 Other Considerations

Offline Second Factor Authentication Our systems require one-way communication, which makes them a good fit for offline second factor authenticators. Mechanisms facilitating communication of TOTPs generated by offline devices (*e.g.,* a fridge with a screen) mentioned in [100] can be used in our systems (*e.g.,* QR encoding). Fig. 4.3 illustrates an anticipated system model for edge devices using the new TOTP systems for offline second factor authentication.

Clock Synchronization Similar to standard TOTP systems, our systems require synchronized clocks at the client and server. Natural delay between the time of generating a TOTP at the client and the TOTP verification at the server can cause authentication failure. The server can allow a small window of time skew (*e.g.,* 30 seconds before the current time) and thus can verify the received TOTP accordingly, as used in standard TOTP systems.

4.3 Implementation and Evaluation

We implemented a prototype of the RSA and ECElGamal-based TOTP systems (in Java, and using the mainline Java security and Bouncy Castle Crypto [110] libraries). For comparisons with counterparts, we implemented TOTP systems that rely on SHA-256 [57]-based 1-dimensional hash chains (referred to as 1D/1DHC) [100], RSA-based multi-dimensional hash chain (MDHC) [133], traditional ECElGamal, ECDSA, EC-Schnorr and traditional RSA with full domain hash (RSA-FDH). RSA is used with 3072-bit modulus, and all elliptic curve cryptography (ECC)-based signatures use the NIST P-256 curve.

4.3.1 Experimental Setups

The experiments were taken for TOTP systems that had a lifespan of ∼2 years (*i.e.,* 2^{21} TOTPs). They were done on a constrained Raspberry Pi Zero W (RPi), with a single-core 1.0 GHz CPU and 512MB RAM, and a laptop, with a dual-core 3.0 GHz

CPU and 16GB RAM. The client and server play the role of the TOTP generator and the TOTP verifier, respectively. The setups include: a laptop and a RPi (*e.g.,* a smart lock scenario); a RPi and a laptop (*e.g.,* a smart watch scenario); and two RPis (*e.g.,* a car gate scenario). We refer to first and last hash in a chain as *head* and *tail*, respectively. For readers not familiar with hash chain based authentication, it is described in Section 4.4. We set RSA public exponents (used for verification or hashing) to the recommended size, $\geq$17 bits [36]. In case of 1DHC, the server constantly replaces its tail with the hash used at last successful authentication. We also consider a case where the client already has in storage hashes across the chain that correspond to the beginning of each month/week, which helps expedite calculating any target hash in the chain. We also note that in this implementation, we use a simpler SHA hash as in the scheme of Boneh et al. [100]. We also propose and evaluate an MDHC-TOTP system with 21 dimensions (referred to as 21D), with each dimension being 2 hashes long so as to offer 2^{21} possible TOTPs and decrease the chain diameter (*i.e.,* make the number of hashes smaller, although we have to use a commutative hash like RSA). All ECC-based systems calculate the first part of the signature before knowledge of next *tp*.

4.3.2 Performance and Overhead Analysis

We discuss the performance and overhead of our TOTP systems and the other alternatives, with focus on the online authentication phase. We break down the authentication phase into two sub-tasks: (1) Generation of a TOTP at the client side, which is generating a target hash in the chain for hash chains, and a signature for the rest of the schemes; and (2) Verify of the TOTP at the server side, which is checking if the received

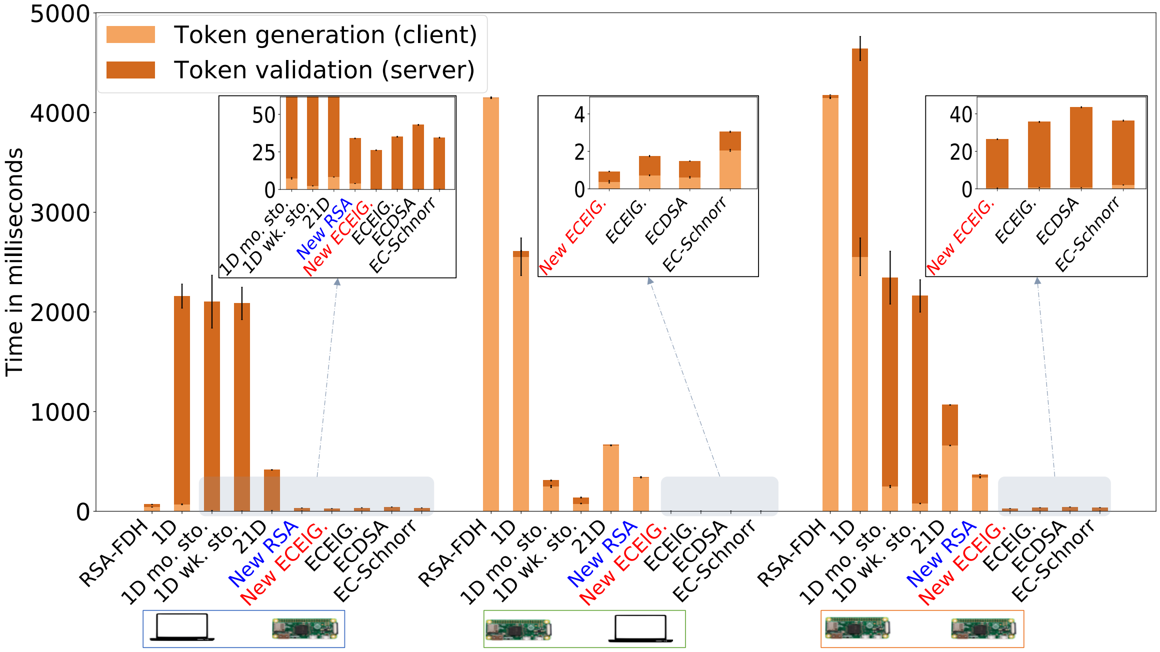

Figure 4.4: Authentication latency of TOTP systems.

TOTP can hash forward to the tail for hash chain-based schemes, or verifying a signature for the other schemes.

Fig. 4.4 demonstrates that our MSS-based ECElGamal-TOTP system reduces latency to complete authentication compared to the following alternatives as follows: RSA-FDH (158x reduction), 1DHC (best case with client storing hashes of every week beginnings, 82x), MDHC (41x) and traditional ECElGamal/ECDSA/EC-Schnorr systems (2x), respectively for the IoT to IoT scenario. The other proposed RSA-TOTP system also cuts down authentication latency by ∼12, 6 and 3 times compared to alternatives based on RSA-FDH, 1DHC (with client hash storage as in the previous case), and MDHC, respectively. Even though computations required at the client can be reduced in TOTP systems based on 1DHC by keeping some hashes along the chain as shown in the results, the server cannot

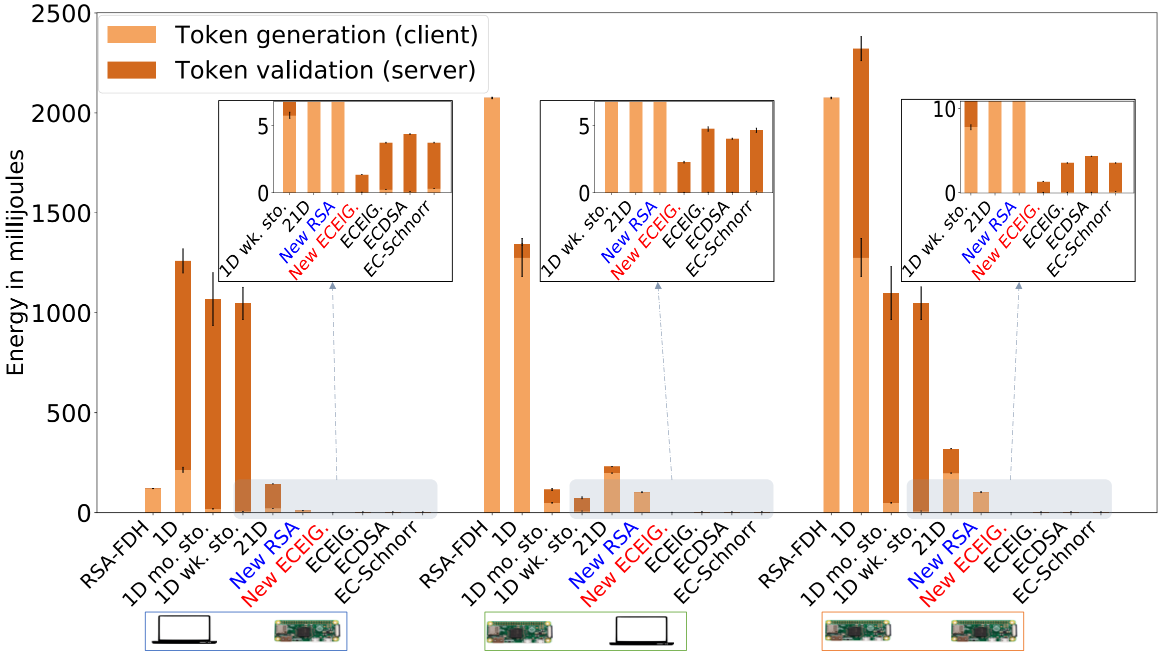

Figure 4.5: Authentication energy consumption of TOTP systems.

use this technique and keep such hashes since this reveals future passwords and makes them susceptible to theft in case of server compromise attacks, necessitating the full chain traversal.

Fig. 4.5 shows the energy consumption for each scheme (measured using a power meter). Our MSS-based ECElGamal-TOTP system provides energy savings of a factor of 1572, 792, 243 and 3 compared to its alternatives: RSA-FDH, 1DHC (with client hash storage as in the previous case), MDHC and traditional ECElGamal/ECDSA/EC-Schnorr systems, respectively. Our MSS-based RSA-TOTP system also reduces energy consumption by 20, 10 and 3 times compared to RSA-FDH, 1DHC (with client hash storage as in the previous case) and MDHC TOTP systems, respectively.

Table 4.3 shows the parameters kept at each node and amount of storage needed for them across all systems. Since online-offline signature schemes keep parameters that are pre-processed offline, they require more storage than traditional signature schemes. As an optimization to traditional ECElGamal, ECDSA, EC-Schnorr and our ECElGamal, we during setup pre-compute and store different bases P that allow such schemes to generate 2^N nonce signatures using only $\mathcal{O}(\log N)$ storage where N is the total number of signatures in the life of system. For all elliptic curve schemes (including our new ECElGamal), the client side requires more storage than other systems based on hash chains and RSA; however, this storage is still less than 16KBytes which is reasonable for modern IoT devices given the saving in power and computational delay [185]. The server storage requirement is reduced by 4x and 2x compared to MDHC and RSA-FDH, respectively. Our ECElGamal outperforms traditional ECElGamal without requiring extra storage at the server side while requiring only a few extra bytes at the client. The new RSA system improves on traditional RSA, but underperforms the simpler hash and EC based systems. However, since it offers formal security proof (not available in ECDSA, EC-Schnorr or ECElGamal), it may be of interest to high assurance applications.

4.4 Asymptotic Efficiency of MSS

In this section, we analyze MSS's time and space complexity in comparison to hash chains. MSS provides similar functional properties to hash chains and can be used in any context where hash chains are applied. We first provide an analysis of hash chains to

Table 4.3: Storage for TOTP systems.

TOTP System	Storage (bytes)			
	Client		Server	
	Size	Param.	Size	Param.
RSA-FDH	391	$st, \mathsf{vk}.N, \mathsf{vk}.e$	388	$st, \mathsf{vk}.N$
1D, 1D mo. sto., 1D wk. sto.	31, 409, 1705	$st, salt,$ $head$, hashes	31	$st, salt, tail$
MDHC (21D)	817	$st, \mathsf{vk}.N, \{\mathsf{vk}.e_i\}_i, head$	772	$st, \mathsf{vk}.N, head$
New RSA	8839	$st, \mathsf{vk}.N, \mathsf{vk}.e$ $\Sigma', \sigma_\$$	388	$st, \mathsf{vk}.N,$
New ECElGamal	16513	$st, \mathsf{sk}.x, r_\$^{-1},$ $\sigma_\$.x_1, B, P$	196	$st, \mathsf{vk}.xg,$ $\mathsf{vk}.xg \cdot \sigma_\$.x_1, \sigma_\$$
ECElGamal	16484	$st, \mathsf{sk}.x, r_\$^{-1},$ $\sigma_\$.x_1, P$	196	$st, \mathsf{vk}.xg,$ $\mathsf{vk}.xg \cdot \sigma_\$.x_1, \sigma_\$$
ECDSA	16484	$st, \mathsf{sk}.x, r_\$^{-1},$ $\sigma_\$.x_1, P$	100	$st, \mathsf{vk}.xg, \sigma_\$$
EC-Schnorr	16516	$st, \mathsf{sk}.x, r_\$, \sigma_\$, P$	68	$st, \mathsf{vk}.xg$

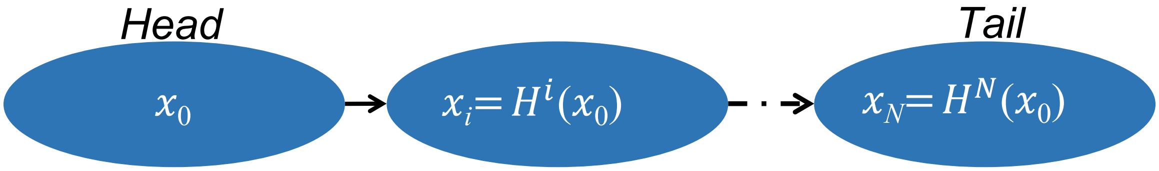

Figure 4.6: A one-dimensional hash chain.

be able to provide a baseline. We show that MSS reduces the time×space complexity from $\mathcal{O}(N)$ to $\mathcal{O}(\mathsf{polylog}(N))$.

Given a hash function H, a *hash chain* (Fig. 4.6) is a list of vertices $\{v_0, \ldots, v_N\}$ where each v_i is obtained by hashing the value before it ($i.e.$, v_i is labeled by a string x_i such that $x_{i+1} = H(x_i)$ holds for all i). The first vertex v_0 is called the *head* of the chain, v_N is the *tail*. The labels in a hash chain can be computed in the "forwards" direction ($i.e.$, given x_i one can compute x_{i+1} efficiently by applying H), while the hardness of inverting H ensures that it is computationally infeasible to compute labels in the "backwards" direction ($i.e.$, given x_{i+1}, it is hard to find x_i).

Given a hash chain, a client holding the head x_0 authenticates herself to the server holding x_N by sending the tail's preimage: $x_{N-1} = H^{N-1}(x_0) = (H \circ \cdots \circ H)(x_0)$. The server, on receiving x_{N-1}, validates by hashing and comparing with the tail since it is possible to carry out the forward hash. The server then overwrites the tail with x_{N-1}, and the next time the client wishes to authenticate, she will send x_{N-2}.

Hash chains offer a low bandwidth alternative to the signature-based solution since the labels need only have 256 bits.[3] The drawbacks of the hash-chain solution are 1) hash-chain-based authentication systems have a lifespan – a client can use a hash chain with N

[3] In fact, 128 bits suffice since we assume the hardness of inverting H, and not that of finding a collision which halves the required size of the key.

vertices to authenticate herself only N times; and 2) the client must compute $\mathcal{O}(N)$ hashes in order to authenticate herself. Boneh et al. [100] observe that there is a time/space tradeoff available to the client, since hash-chain labels can be precomputed and stored. For example, if the client stores the labels $x_{\sqrt{N}}, x_{2\sqrt{N}}, \ldots$, then authentication only requires computing $\mathcal{O}(\sqrt{N})$ hashes. Thus, a more accurate statement is 2) the time×space required by the client during authentication is $\mathcal{O}(N)$. In any case, there is a tradeoff since one wants to set N small for client efficiency, but this results in a scheme with a short lifespan.

MSS can be viewed as a hash chain with special hash functions which support faster traversal from the head label to the (preimage of the) tail label; we call these hash functions *mergeable hash functions*. In essence, they allow the generation of any point in a hash chain of length N using a cost of $\mathcal{O}(\log N)$, an exponential improvement. The offline cost of the scheme is the cost of generating a set of basis component hash functions, leaving the online cost to be that of merging these (which mathematically consists only of multiplication operations). The scheme requires additional storage space to store the pre-computed basis hashes, but the total time×space required by the client is $\mathsf{polylog}(N)$.

MSS online signing cost is based only on the number of times the Merge procedure is applied to create a signature for $\mathsf{msg} = b_1 \cdots b_\ell$, which is $\mathcal{O}(\log N)$. Regardless of message bit size $|N|$, online signing in standard signature schemes requires $\mathcal{O}(\log \mathsf{sk})$ multiplications, and usually $|\mathsf{sk}| \gg |N|$ (*e.g.,* $|\mathsf{sk}| \simeq 3072$ in RSA and $\simeq 256$ in BLS). Therefore, MSS provides two advantages over standard signatures: 1) it gives the ability to fine-tune the signing cost in different applications, which makes it valuable for applications with short messages (*i.e.,* small N, which is common in the IoT setting) and perhaps crucial for resource-constrained signers. 2) unlike the sk size which usually increases overtime[4], application requirements usually do not (*e.g.,* a smart thermometer), and therefore MSS cost is not exacerbated overtime. The amount of storage required by MSS is $\mathcal{O}(\log N)$ although storage enhancements are available as discussed in Section 4.2. As an added advantage, ECElGamal-based MSS offline signing cost is significantly reduced to $\mathcal{O}(\mathsf{polylog}\ N)$ because we mainly need to pre-compute the set P to generate any nonce signature.

4.5 MSS Correctness and Security Proof

We first define the conditions under which standard signature scheme are correct and secure. We then provide formal proofs of correctness and security for MSS.

In *standard signature schemes*, correctness implies that for all $(\mathsf{vk}, \mathsf{sk})$ in the support of KeyGen, and messages msg, $\mathsf{Verify}(\mathsf{vk}, \mathsf{msg}, \sigma) = 1$ holds with probability 1 where $\sigma = \mathsf{Sign}(\mathsf{msg}, \mathsf{sk})$. On the other hand, the security is evaluated using a game-based paradigm [73] as follows. For any efficient adversary $\mathcal{A}$, the chance that $\mathcal{A}$ wins the signature forgery game described below is negligible for a secure scheme.

Signature Security Game. The game is played between a challenger $\mathcal{C}$ and adversary $\mathcal{A}$ as follows:

1. $\mathcal{C}$ draws $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^n)$, sends vk to $\mathcal{A}$.

2. The following steps are repeated until $\mathcal{A}$ decides to move to step 3 (repetitions indexed by i):

[4] 3072-bit RSA, 256-bit ECC and SHA-256 provide security equivalent to 128 bits in symmetric key cryptography [24]; keys are usually increased due to advancements in hardware and algorithms.

- $\cdot$ $\mathcal{A}$ sends $\mathsf{msg}^{(i)}$ to $\mathcal{C}$;
- $\cdot$ $\mathcal{C}$ computes $\sigma^{(i)} \leftarrow \mathsf{Sign}(\mathsf{msg}^{(i)}, \mathsf{sk})$ and sends it to $\mathcal{A}$.

3. Finally, $\mathcal{A}$ sends $(\mathsf{msg}^*, \sigma^*)$ to $\mathcal{C}$ and wins if $(\mathsf{msg}^*, \sigma^*) \neq (\mathsf{msg}^{(i)}, \sigma^{(i)})$ for all i and $\mathsf{Verify}(\mathsf{vk}, \mathsf{msg}^*, \sigma^*) = 1$.

In other words, the attacker wins if she is able to correctly sign a new message after observing any number of messages and their signature. We will also work with a version of the above game where the messages to sign are all chosen at random by $\mathcal{C}$. Formally, this game is identical to the one above except for two changes:

(a) The two repeated items in Step 2 are replaced by the single item: $\mathcal{C}$ chooses $r^{(i)}$ at random, computes $\sigma^{(i)} \leftarrow \mathsf{Sign}(r^{(i)}, \mathsf{sk})$ and sends $(r^{(i)}, \sigma^{(i)})$ to $\mathcal{A}$.

(b) Step 3 is replaced by two steps: 1) $\mathcal{C}$ sends a random r^* to $\mathcal{A}$; 2) $\mathcal{A}$ sends σ^* to $\mathcal{C}$ and wins if $\mathsf{Verify}(\mathsf{vk}, r^*, \sigma^*) = 1$.

In this modified game, the attacker wins if she forges the signature for a random message sent by the challenger. We say a signature scheme is *secure under random (resp. adaptive chosen) message attack* if it is hard for $\mathcal{A}$ to win the modified (resp. original) game. It is clear that security under adaptive chosen message attack is the stronger notion of security. However, in the random oracle model [27], security under random message attacks are sufficient to trivially construct schemes with security under adaptive chosen message attacks. Moreover, the scheme with stronger security will have almost exactly the same performance as the scheme with weaker security. For this reason, we focused on constructing signature schemes with security under random message attacks given that scheme based on RSA and BLS are secure under random message attacks [157, 37]. This clearly explains the distinction and why we take this direction in proving security.

Correctness for MSS says that for all $(\mathsf{vk}, \mathsf{sk})$, and any msg, the following holds: $\mathsf{Verify}(\mathsf{vk}, \mathsf{msg}, \sigma) = 1$, where $\sigma = \mathsf{Sign}_{\mathsf{on}}(\mathsf{msg}, \mathsf{sk}, \tau)$, and $\tau = \mathsf{Sign}_{\mathsf{off}}(\mathsf{sk})$. On the other hand, in the MSS security game, any efficient adversary $\mathcal{A}$ wins the incremental forgery game below is negligible.

Incremental Forgery Game. The game is played between a challenger $\mathcal{C}$ and adversary $\mathcal{A}$ and works similar to the adaptive chosen message security game above, except for one change:

(a) in the second point of Step 2, $\mathcal{C}$ computes $\sigma^{(i)}$ according to $\mathsf{Sign}_{\mathsf{on}}(\mathsf{msg}^{(i)}, \mathsf{sk}, \tau^{(i)})$ where $\tau^{(i)} = \mathsf{Sign}_{\mathsf{off}}(\mathsf{sk})$;

After formally defining the correctness and security of the underlying scheme, we will now prove in Theorem 1 that the proposed scheme is an MSS scheme.

Theorem 1 Assume $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Merge}, \mathsf{Rec})$ is a mergeable signature scheme and that H is a random oracle. Then $(\mathsf{KeyGen}', \mathsf{Sign}'_{\mathsf{off}}, \mathsf{Sign}'_{\mathsf{on}}, \mathsf{Verify}')$ is MSS.

Proof. The syntax and correctness follow from correctness and mergeability of the underlying scheme (e.g., RSA). We now prove the security by contradiction. Let's assume that $\mathcal{A}$ is an efficient adversary who wins the incremental forgery game with non-negligible probability. We design another adversary $\mathcal{A}'$ who can win the random message attack game on the underlying mergeable scheme with non-negligible probability. For proof purpose we consider slightly modified games, where we replace all outputs of H with truly random values drawn by the challengers. These games are indistinguishable assuming H is a random oracle. $\mathcal{A}'$ works as follows:

- $\mathcal{A}'$ receives from the challenger the verification key vk, random $r_{j,b}$ and $\sigma_{j,b}$ for $j = 1, \ldots, l$ and $b \in \{0,1\}$.

- $\mathcal{A}'$ then sends $(\mathsf{vk}, \{r_{j,b}\})$ to $\mathcal{A}$.

- Whenever $\mathcal{A}'$ receives msg from $\mathcal{A}$, $\mathcal{A}'$ parses msg into bits $b_1 \cdots b_\ell$, and then receives $(r_\$, \sigma_\$)$ from the challenger. Then it computes $(r, \sigma) = \mathsf{Merge}\big(\{(r_{j,b_j}, \sigma_{j,b_j})\}, (r_\$, \sigma_\$)\big)$ and returns $\sigma' = (r, \sigma)$.

- The challenger sends $r^*_\$$ to $\mathcal{A}'$ and it forwards it to $\mathcal{A}$ and gets back msg^*, σ^*.

- $\mathcal{A}'$ parses the message msg^* as $b^*_1, \ldots, b^*_l$ and computes the following $\mathsf{Rec}(\mathsf{Merge}(\{r_{j,b^*_j}, \sigma_{j,b^*_j}\}, (r^*_\$, \sigma^*_\$)), \{r_{j,b^*_j}, \sigma_{j,b^*_j}\})$ to get $(r^*_\$, \sigma^*_\$)$ and sends it to the challenger.

In the above reduction, $\mathcal{A}'$ answers $\mathcal{A}$'s queries correctly by inspection and thus emulates the security game of the constructed scheme for $\mathcal{A}$. Thus, whenever $\mathcal{A}$ wins, $\mathcal{A}'$ also wins with the same probability. This contradicts the security of the mergeable signature scheme $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Merge}, \mathsf{Rec})$, proving that the proposed scheme is secure.

Chapter 5

Token-based Vehicular Security System (TVSS): Scalable, Secure, Low-latency Public Key Infrastructure for Connected Vehicles

Technology which allows road vehicles to communicate with one another and with pieces of road infrastructure has the potential to drastically improve the safety and efficiency of the transportation system. Indeed, though this technology is in its infancy, numerous products have already been proposed. Safety applications such as *Basic Safety Messages* are projected to reduce road fatalities by 80%, a decrease of roughly 30,000 per year [131, 64].

More than 50 intersections in Pittsburgh, PA were fitted with intelligent traffic signal control systems between 2012 to 2016, which reduced travel times through these intersections by 26% [44]. Coordinated driving applications such as *Cooperative Adaptive Cruise Control* allow connected vehicles to travel closely in a convoy/platoon, reducing aerodynamic drag, improving fuel efficiency, and lowering the carbon footprint of the entire transportation system by a projected 15% [25].

Needless to say, security in these types of transportation applications is extremely important as a breach could cause accidents or otherwise disrupt the flow of traffic [6, 34, 35, 149, 114, 13]. In this work, we consider the authentication layer lying underneath these potential applications. Designing such a system requires addressing some non-standard security issues having to do with the fact that a vehicle's identity and position over time are sensitive information.

Vehicular Public Key Infrastructure Significant prior work on this topic has culminated in the Security Certificate Management System (SCMS) [134] which has been adopted by the US Department of Transportation (a similar European standard is outlined in [60]). These standards provide a public key infrastructure (PKI) for vehicles to use to authenticate themselves which promises a standard unforgeability security guarantee as well as two additional security features called *anonymity* and *unlinkability.* Roughly speaking, anonymity says that a vehicle should be able to authenticate itself without revealing its long-term vehicle identity; unlinkability demands that it should not be possible to identify the same vehicle authenticating itself in two different time periods. So if Alice's car authenticates itself today on highway 1, anonymity guarantees that no one can deduce "that car belongs to Alice", while unlinkability ensures that no one can say "the same car authenticated itself yesterday on highway 2". In this work, we refer to a PKI which guarantees anonymity and unlinkability as *vehicular PKI* (VPKI). Such a system is also desirable in other related domains such as unmanned aerial vehicles (UAVs) [14].

Revocation Given the damage potential of a malicious user in VPKI, a revocation mechanism is needed to deactivate a malicious user's credentials, rendering the user unable to participate in the system. Revocation solutions involving a global centralized certificate revocation list (CRL) are not ideal for VPKI. On the one hand, the large scale and distributed nature of the system make it unreasonable to expect that all vehicles would constantly maintain an up-to-date local copy of the CRL. On the other hand, network-based solutions where a vehicle queries a CRL on the cloud before interacting with another vehicle are not ideal because of latency and network intermittency. Because of these issues, revocation is a major pain point of all prior VPKI systems.

Pseudonym Certificate Generation Time Typically in VPKI systems, users authenticate themselves on the road using temporary credentials called *pseudonym certificates* (PCs). These PCs are periodically refreshed when the vehicle executes the PC generation protocol with a stationary road-side unit (RSU). An extremely important (and often overlooked) feature in the design of VPKIs is the execution time of PC generation as this governs the possible use cases of the system. Essentially, the issue is that a fast vehicle passing by an RSU would have only a very brief period of network connectivity (less than one second if traveling at highway speeds). Therefore, if the PC generation protocol is too slow, the vehicle and RSU will fail to complete their protocol with high probability, and thus the vehicle would not reliably be able to update its PC. This issue is aggravated by the fact that channel uncertainty in wireless networks intensifies in highly mobile environments [79]. Unreliability of PC generation at high speeds means, for example, that cars might wind up using the same PC for hours at a time, which harms system security. Our results show that, because of its slow PC generation protocol, attempting to use SCMS in highway scenarios requires PCs to persist for 6.7 hours, while TVSS reduces this window to 18 minutes (a $\sim$22.5x privacy improvement).

5.0.1 Our Contributions

Edge-based VPKI: We propose Token-based Vehicular Security System (TVSS), a new system architecture for VPKI with properties which are essential for a large scale mobile PKI system. The core novel feature of TVSS is that it takes advantage of the compute power of the network of roadside units (rather than using RSUs simply as a network of proxies connecting vehicles to the backend servers). We find that computationally able RSUs fit seamlessly into VPKI, yielding improvements across the board. Specifically:

1. **Low latency PC generation:** TVSS has a lightweight PC generation protocol consisting essentially of just a handshake between the vehicle and an RSU, as shown in Figure 5.1. In particular, PC generation requires *no online involvement from the back-end.* This enables new use cases which were unsupported by previous systems (*e.g.*, high speed PC generation).

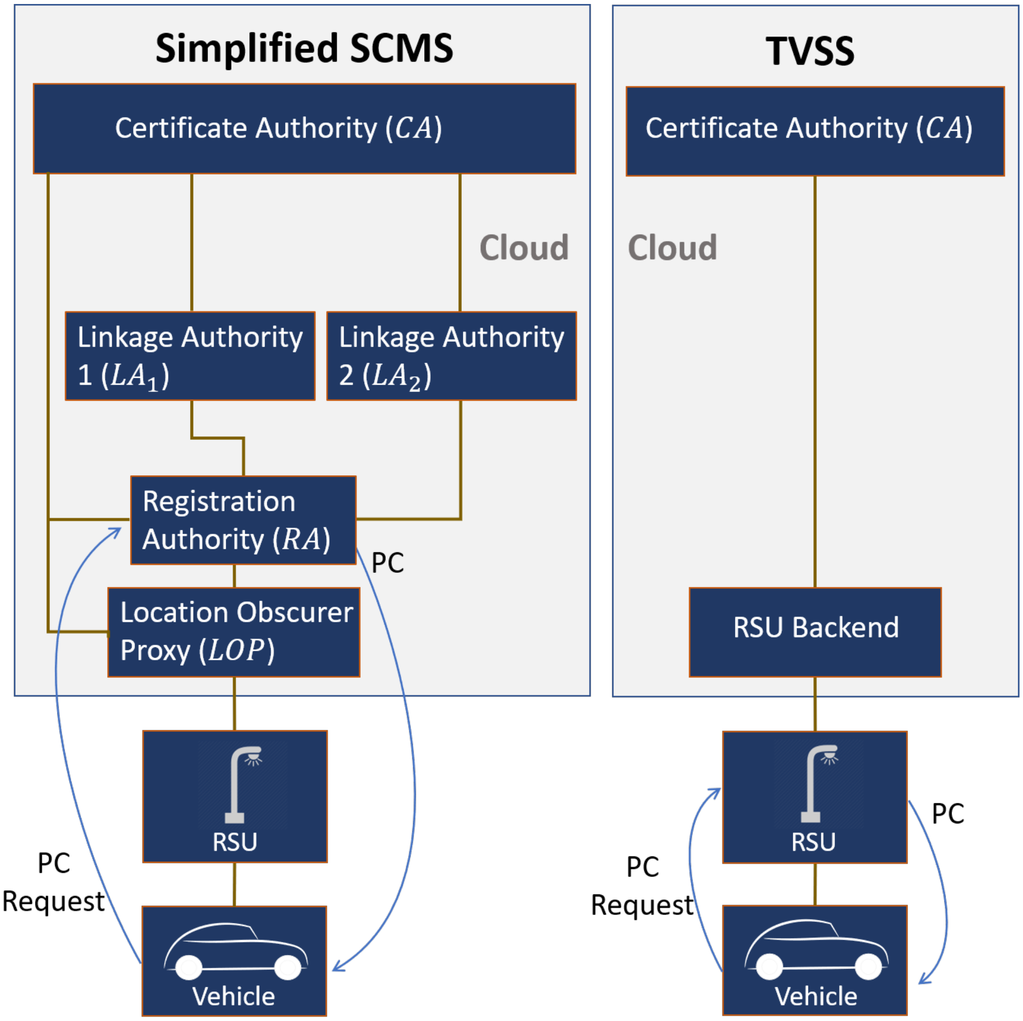
Simplified SCMS
Certificate Authority (CA)
Cloud
Linkage Authority 1 (LA_1)
Linkage Authority 2 (LA_2)
Registration Authority (RA)
PC
Location Obscurer Proxy (LOP)
RSU
PC Request
Vehicle
TVSS
Certificate Authority (CA)
Cloud
RSU Backend
RSU
PC
PC Request
Vehicle

Figure 5.1: Simplified SCMS and TVSS architectures.

2. **Low Communication Revocation:** Utilizing compute power of the RSUs enables reliable revocation where global CRLs are maintained by the (stationary) RSUs, while only short (location-specific) CRLs need to be shared with the vehicles. The result is that revocation in TVSS requires drastically less total communication than in prior systems.

3. **Simple architecture:** Passing computation to the RSUs greatly simplifies the system as a whole. As shown in Figure 5.1, the overall footprint of TVSS is much smaller than that of SCMS. Thus, maintenance costs of TVSS would be much lower with no loss to security.

Formalizing VPKI Security: In order to foster future work on VPKI, we give a formal game-based security definition for VPKI incorporating unforgeability, anonymity and unlinkability. Additionally, we consider other attacks on our system and show how to neutralize them. This discussion includes a new type of attack called a *clone attack* which was previously unconsidered in the VPKI literature, and which affects all prior systems. Clone attacks are similar in spirit to *sybil attacks*[1] [55], and occur when an authorized vehicle shares its credentials with an unauthorized vehicle in an attempt for both cars to participate in the system in different locations. Similar attacks have been considered in cryptocurrencies [45, 202], transportation toll collection [159] and network authorization schemes [140]. We show how to handle clone attacks against TVSS; no discussion or defense to clone attacks is given in other VPKI systems.

Open Source Testbed: We build and assemble a real testbed of on-board units (OBUs) and RSUs that have technical specifications similar to commercial OBUs and RSUs. Specifically, we set up the networking standard specifically designed for connected vehicles, IEEE 802.11p/dedicated short range communication. Our OBUs and RSUs are open source and re-programmable and so hopefully will be useful to other research and application development.

[1] Sybil attacks occur when a single vehicle obtains several different copies of valid credentials in order to pretend to be several different vehicles.

VPKI Implementation and Field Experiments: We implement TVSS and other VPKI systems on our OBUs and RSUs and conduct a series of highway and in-city street experiments at different velocities ranging from 25mph to 85mph. Our tests observed that TVSS achieves a 28.5x reduction in its PC generation latency and a 13x reduction in total communication during revocation compared to other systems. At extreme speeds, TVSS is 3.85x more likely to successfully complete PC generation and 6.5x more likely to successfully update its local CRL compared to other systems.

5.1 Cryptographic Preliminaries

In this section, we describe the basic cryptographic systems that are utilized in SCMS and TVSS, namely encryption, digital signatures, decisional Diffie-Hellman (DDH), and hash chains.

Encryption Schemes. Encryption schemes are used in cryptography to ensure confidentiality of the information being sent over a network. It can be formally defined as a set of three algorithms $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$. The algorithms $\mathsf{KeyGen}(1^n)$ outputs the key pair $(\mathsf{ek}, \mathsf{dk})$, $\mathsf{Enc}(\mathsf{msg}, \mathsf{ek})$ outputs a cipher text ct and $\mathsf{Dec}(\mathsf{dk}, \mathsf{ct})$ outputs the message, msg, by using the decryption key dk. It satisfies the properties of correctness and the security. Correctness says that for all messages, if $(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{KeyGen}(1^n)$, then $\mathsf{Dec}(\mathsf{dk}, \mathsf{Enc}(\mathsf{msg}, \mathsf{ek})) = \mathsf{msg}$. Informally, the security property is that no efficient adversary can decrypt the ciphertext without having access to the decryption key.

Digital Signatures. Digital signatures are basic schemes used in cryptography and used for authentication purposes. A signature scheme can be formally defined as a set of three

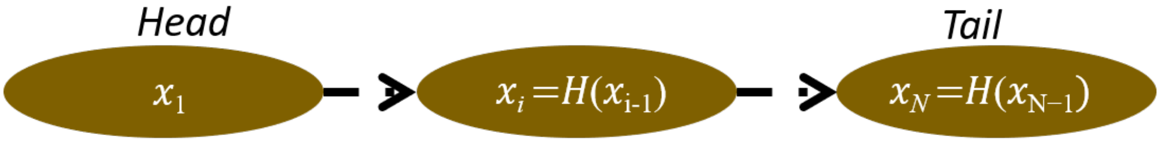

Figure 5.2: A hash chain.

algorithms $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$. The algorithms $\mathsf{KeyGen}(1^n)$ outputs the key pair $(\mathsf{vk}, \mathsf{sk})$, $\mathsf{Sign}(\mathsf{msg}, \mathsf{sk})$ outputs a signature σ and $\mathsf{Verify}(\mathsf{vk}, \mathsf{msg}, \sigma)$ checks whether the signature is valid or not.

Additionally, the correctness and security of the digital signatures must also hold. Correctness says that for all messages msg, if $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^n)$ and $\sigma \leftarrow \mathsf{Sign}(\mathsf{msg}, \mathsf{sk})$, then $\mathsf{Verify}(\mathsf{vk}, \mathsf{msg}, \sigma) = 1$. Intuitively, security demands that without possession of the secret key, no adversary can produce a valid signature for a new message.

Decisional Diffie-Hellman. We define the decisional Diffie-Hellman (DDH) problem in a cyclic group G with a generator g. It is defined as follows. Given a tuple (g, g^x, h, h') where x is a random exponent, $h \in G$ is random and h' either equals h^x or else is a random group element, decide which is the case. In DDH based encryption schemes, the private/public key pair (x, g^x) corresponds to the decryption and encryption keys $(\mathsf{dk}, \mathsf{ek})$, respectively. Similarly, the key pair (x, g^x) corresponds to the signature and verification keys $(\mathsf{sk}, \mathsf{vk})$, respectively, in digital signature schemes.

Hash Chains. Hash chains are important primitives which appear in electronic currencies and authentication schemes. Formally, we can define a hash chain as follows. Given a hash function H, a *hash chain* is a list of vertices $\{v_1, \ldots, v_N\}$ where each v_i is labeled by a string x_i such that $x_{i+1} = H(x_i)$ holds for all i (Figure 5.2). Since it is computationally infeasible to invert H, hash chains acquire a notion of direction. The first vertex v_1 is called the *head* of the chain, v_N is the *tail.* The property that is required is as follows. Given x_i, one can compute $H(x_i) = x_{i+1}$ efficiently, but not the other way because of the hardness assumption.

5.2 System Overview

The goal of this section is to explain 1) how our system TVSS works at a high level; and 2) how the three features mentioned in Section 5.0.1 are achieved. We stress this is an oversimplified discussion, the full scheme is presented in the next section.

System Players and the Parameter T. The main players in TVSS are the vehicles, the distributed network of RSUs and the certificate authority (CA). Additionally, we assume that the RSUs are all connected to a backend server called the "RSU backend" which communicates with, but is distinct from the CA (this is important for revocation). We model the connection between the vehicles and the RSUs as a secure private channel (in reality this will be implemented using encryption). Time in TVSS is broken into distinct periods of T minutes each, for a system parameter T. Roughly speaking, smaller T means stronger security and greater overhead on the system.

The Infrastructure/Backend Separation Assumption. It is critical to security in TVSS to assume that the CA and the RSUs are controlled by separate entities which do not collude; we call this the *infrastructure/backend separation assumption.* Some version of this assumption is implicit in all prior work on VPKI. Though not ideal, we believe the infrastructure/backend separation assumption necessary for security in TVSS is plausible since the CA would likely be controlled by a specialized security company while the RSUs would be part of the public infrastructure. Designing a VPKI system which does not require any separation assumptions about infrastructure and the backend is an excellent open problem.

System Setup. Life begins for a vehicle when it obtains an enrollment certificate from the CA. We think of this as occurring at the manufacturing plant, once during the vehicle's lifetime.

Token Generation. Once a vehicle possesses its enrollment certificate, it can request a batch of tokens from the CA which it will later trade in to the RSUs in order to get PCs. In order to request tokens, the vehicle simply authenticates itself using the EC in order to receive a large number of tokens (say 2,880 for a one month's supply when $T = 15$ minutes). Each token is signed using CA's private signing key and can be validated by verifying the signature using CA's public signing key. Each token is valid for one specified period of T minutes; when the final token expires, v will need to request new tokens from CA. Token generation can be performed offline.

PC Provisioning. Once a vehicle has tokens, it can request PCs from any RSU. To do this, the vehicle simply presents the token for the current time period to the RSU who verifies authenticity against CA's public credentials and, if valid, returns a PC. The PC is signed by the RSU and marked with a geographical tag corresponding to the RSU's location, and is valid only when nearby this location and during the same time window that the token is valid for. This means that new PCs must be requested every T minutes. The geographic radius of validity should be an upper bound on the distance one can drive in T minutes.

The RSU sends the token it received to the RSU backend to check for duplicates. If the same token is used twice to generate two different PCs in two different geographical areas (*i.e.*, if a clone attack is being launched), then the RSU backend will notice and will begin the revocation procedure for this vehicle (described next).

Notice that from the vehicle's point of view, PC generation consists of a simple "handshake-type" interaction with the RSU. This is in stark contrast to SCMS where PC generation requires the vehicle's messages to be forwarded all the way to the RSU backend which is four "network hops" away from the vehicle. The ability to perform PC generation over a short-term connection unlocks use cases which are not supported by SCMS. For example, our experiments in Section 5.6 demonstrate that PC generation in TVSS can be performed at high speeds on the freeway, while this would not be possible in SCMS. Indeed, our experiments indicate that in these driving conditions, our PC generation protocol is at least $10x$ less likely to fail than PC generation in SCMS. This is an important improvement since resolving the issues resulting from failed PC generation attempts consumes extra system resources. Additionally, because PC generation can be performed with a much weaker connection between the vehicle and RSU, the frequency with which a connection occurs which can support PC generation increases considerably. This allows us to set our system up so that PCs are refreshed more frequently (every fifteen minutes, rather than once per week) which translates to better security and easier revocation.

Revocation. When the PC of a malicious vehicle is identified, the RSU backend and the CA are alerted, and they cooperate to recover the current and future tokens of the offending vehicle. These tokens are shared with the RSUs who update their token blacklists (TBLs), thus preventing the offending vehicle from obtaining any new PCs in the future. In order to deactivate the current PC, the CA shares the offending PC *only with the RSUs which are in the geographic radius of the misbehaving vehicle*, these RSUs in turn share the PC with all vehicles in their region, who update their local CRL (or what we also refer to as pseudonym certificate revocation lists, PCRLs) and make sure to avoid the offending vehicle.

Note that the honest vehicles receive only the PCs of the misbehaving cars in their geographical area, not the list of all misbehaving vehicles in the whole system. Moreover, this list must only be maintained for the remainder of the time period, after which point it can be discarded. Thus, the amount of revocation information which needs to be downloaded by each vehicle is small. The large TBLs which must be maintained by the RSUs (consisting of all current and future tokens of all offending vehicles in the system) represent less of a problem as the RSUs are stationary and so should have a stable connection. Our experiments in Section 5.6 demonstrate that this change to requiring the vehicles only to maintain a geographically-based CRLs leads to a system-wide $13x$ savings in total communication size.

We remark that while the idea to use geographic PCs to improve revocation seems, at first glance, to be a generic solution which can be applied in any VPKI, this is not the case. The key feature of TVSS which makes it possible is the short lifespan of the PCs (*i.e.*, small T). In SCMS, the PCs are live for an entire week and so it is not possible to constrain a PC to a small geographic region (one week is enough time to drive from Lisbon, Portugal to Vladivostok, Russia). The short lifespan of PCs is only possible in TVSS because of the system-wide efficiency improvements gained by passing computation to the RSUs.

Security. Anonymity and unlinkability both hold in TVSS intuitively because the tokens which are used to get the PCs have no information about the vehicle. For example, even if the RSU is corrupt and can connect the PC to the token, there is no way to connect the token to the vehicle's enrollment certificate as long as the RSU does not collude with the CA. Thus, anonymity should hold under the infrastructure/backend separation assumption; unlinkability should hold for similar reasons. Formal security definitions and proofs are given in Section 5.4.

5.3 TVSS System

In this section, we start by describing the system model and assumptions. We then illustrate the detailed protocols forming our system, namely (Setup, TokenGen, PseudoGen, Revoke). We finally discuss some implementation considerations.

5.3.1 System Model and Assumptions

As mentioned before, the TVSS system consists of a network of RSUs and backend servers residing in the cloud. The backend servers are the RSU backend and the certificate authority (CA). The CA is the root of trust in the VPKI and it is responsible for providing vehicles with ECs and tokens while providing RSUs with signing certificates. Vehicles use ECs to obtain tokens from the CA while RSUs use their signing certificates to generate and sign PCs for vehicles in exchange for tokens (see Figure 5.3). Vehicles use dedicated short range communication (DSRC/IEEE 802.11p) to communicate with one another, and they can reach the internet and the backend servers through the RSUs only. The communication

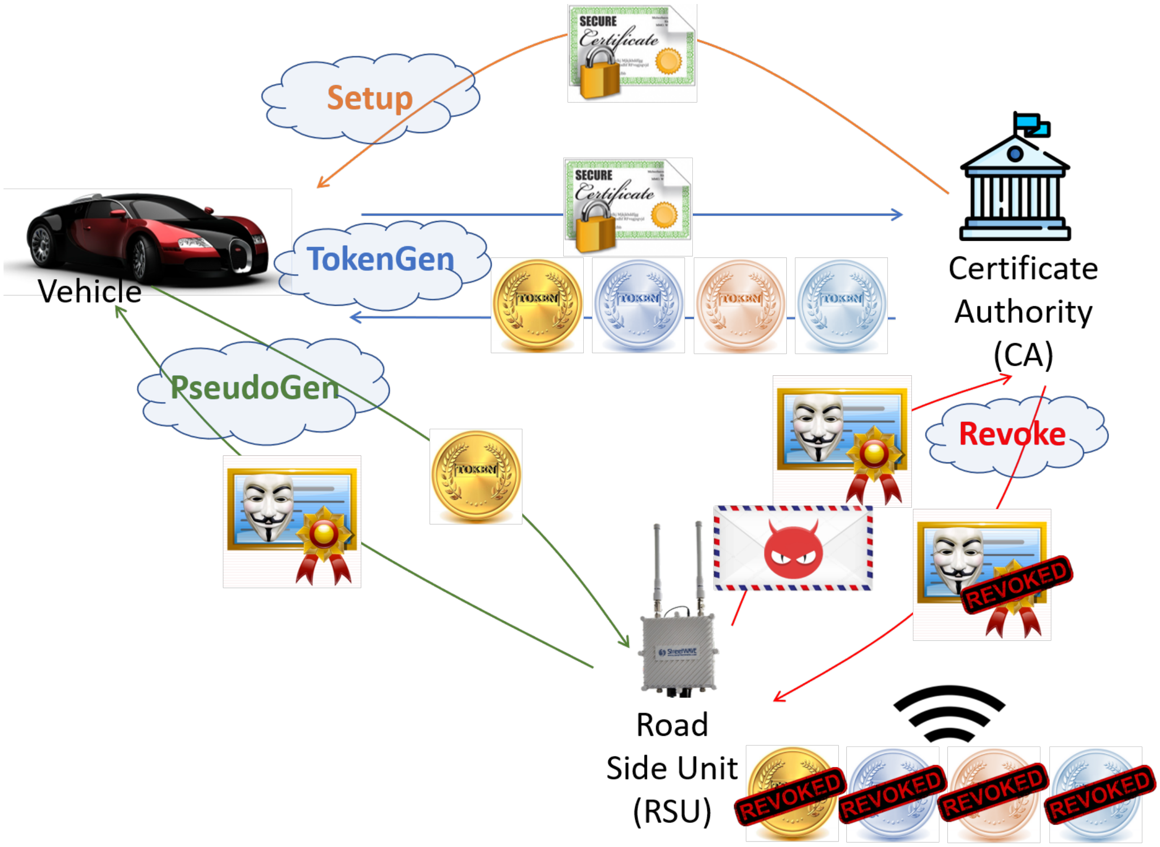

Figure 5.3: Overview of all protocols of TVSS.

channels between system's entities (i.e. vehicles, RSUs, backend servers, etc) are assumed to be always secure (e.g., over TLS). The system protocols described next make use of a standard digital signature scheme.

5.3.2 System Protocols

The four protocols (Setup, TokenGen, PseudoGen, Revoke) make up the TVSS system.

- **Setup:** This allows a vehicle $v \in \mathsf{V}$ to acquire a long-term enrollment certificate, EC_v, which acts as the vehicle identity. Explicitly EC_v is a signing key pair consisting of

Algorithm 6 $\mathsf{Setup}(\mathsf{sk}_{\mathsf{CA}})$

```
Compute $(\mathsf{vk}_v, \mathsf{sk}_v) \leftarrow \mathsf{KeyGen}()$;
Compute $\sigma_v \leftarrow \mathsf{Sign}_{\mathsf{sk}_{\mathsf{CA}}}(\mathsf{vk}_v)$
$\mathrm{EC}_v \leftarrow (\mathsf{vk}_v, \sigma_v)$
**Output** $\mathrm{EC}_v$
```

a private signing key and a public verification key, and additionally a signature on the public verification key using the CA's private signing key. The vehicle v will use EC_v to request services from the CA when needed using a TLS-style handshake. The enrollment certificate EC_v has a long expiration time and v probably gets it once in its lifetime or upon ownership transfer. Figure 5.3 shows an overview of this protocol, the full protocol is given in Algorithm 6.

- **TokenGen:** This provides a vehicle with a list of authorization tokens $\mathsf{T} = \{\tau_1, \cdots, \tau_N\}$ that can be used to anonymously request pseudonym certificates from the RSUs. Each token $\tau \in \mathsf{T}$ is active during a specified time window only, and so once the final token in T expires, the vehicle will have to rerun this procedure to get more. Explicitly, each token is a random nonce, a time window, and a signature on the nonce/time window pair. The vehicle obtains the tokens after completing a TLS-style handshake with the CA using EC_v. Figure 5.3 shows an overview of TokenGen, and Algorithm 7 shows the full protocol.

- **PseudoGen:** This is used by vehicles to request PCs from the RSUs in exchange for tokens. Explicitly, a PC is a signing key pair consisting of a private signing key (generated by the vehicle and kept secret), and a public verification key, and a signature on the

Algorithm 7 $\mathsf{TokenGen}(\mathrm{EC}_v, Req, \mathsf{sk}_{\mathsf{CA}})$

```
Parse $\mathrm{EC}_v = (\mathsf{vk}_v, \sigma_v)$;
Parse $Req = (\mathsf{st}', \mathsf{et}')$;
if $\mathsf{revoked}(\mathrm{EC}_v) = \mathrm{False}$ & $\mathsf{hasTokens}(\mathrm{EC}_v, \mathsf{st}', \mathsf{et}') = \mathrm{False}$ then
    Compute $\mathsf{T} = \{\tau_1, \cdots, \tau_N\},$ where $\tau_j \leftarrow (\mathsf{id}_j, \mathsf{tw}_j, \sigma_j)$
    $tc \leftarrow (\mathsf{id}_j, \mathsf{tw}_j)$
    $\sigma_j \leftarrow \mathsf{Sign}_{\mathsf{sk}_{\mathsf{CA}}}(tc)$
    Output $\mathsf{T}$
else
    Output $\perp$
end if
```

public verification key using the RSU's secret signing key. The vehicle simply presents its token and its verification key to the RSU who validates the token and then signs the and returns the key. The RSU also shares this token and the certificate with the RSU backend to detect if token double use occurs; this allows our system to detect and throttle clone attacks. Figure 5.3 shows an overview of PseudoGen, the full protocol is shown in Algorithm 8. This protocol is the frequently used to achieve the unlinkability property.

- **Revoke:** When a vehicle is identified as adversarial, the RSU backend and the CA cooperate to run this protocol to deactivate the offending vehicle's credentials. This works, as described in the previous section, by updating the RSU TBLs to include all future tokens of the offending vehicle, by updating the vehicle CRLs of all vehicles which are

Algorithm 8 $\mathsf{PseudoGen}(\tau, \mathsf{sk}_{RSU}, \mathsf{vk}_{\mathsf{CA}})$

```
Parse $\tau = (\mathsf{id}, \mathsf{tw}, \sigma)$
Compute $(\mathsf{vk}, \mathsf{sk}) \xleftarrow{R} \mathsf{KeyGen}()$
if $\mathsf{timeValid}(\tau) = \text{True} \;\&\; \mathsf{revoked}(\tau) = \text{False} \;\&\; \mathsf{Verify}_{\mathsf{vk}_{\mathsf{CA}}}((\mathsf{id}, \mathsf{tw}), \sigma) = \text{True}$ then
    Compute $\sigma' \leftarrow \mathsf{Sign}_{\mathsf{sk}_{RSU}}(\mathsf{vk})$
    Compute PC$\leftarrow (\mathsf{vk}, \sigma')$
    Output PC
else
    Output $\perp$
end if
```

nearby the offending vehicle to include the malicious PC, and by updating the CA blacklist BL to include the enrollment certificate (EC_v). Figure 5.3 shows an overview of Revoke; the full protocol is in Algorithm 9.

5.3.3 Implementation Considerations

RSU Blackout Areas. Suppose a vehicle needs to travel to an area with limited RSU deployment for an extended period, but wants to precompute PCs for use during its trip. Normally the RSUs in our system will only give a PC for the current time window. However, in such a situation, our system could allow an RSU to grant a batch of PCs in exchange for a batch of tokens and the vehicle would be able to authenticate itself as usual. Note however that the unlinkability guarantee would fail to hold as the RSU would likely infer that the batch of PCs all belong to the same vehicle.

Algorithm 9 Revoke(PC)

```
if decideRevoke(PC) = True then
    $\tau_j \leftarrow$ getToken(PC)
    $\mathrm{EC}_v \leftarrow$ findLTC($\tau_j$)
    Update(BL, $\mathrm{EC}_v$)
    Update(TBL, $\{\tau_j, \cdots, \tau_N\}$)
    Update(PCRL, PC)
    Output (BL, TBL, PCRL)
end if
```

Linked Tokens via Hash Chains. We propose a novel approach that allows for generating token ids for a single vehicle v that look random and independent but can still be selectively linked using a secret that is kept at CA only. Particularly, it provides two advantages: 1) It allows CA to publish a single τ id (and a secret) per revoked v, which can significantly reduce token blacklist (TBL) bandwidth requirements since the TBL size becomes based on the number of revoked vehicles rather than revoked tokens. 2) The CA can keep a single τ id (and a secret) at any time window and derive the ids of tokens of future time intervals; the same approach can also be utilized by RSU $\in$ RSU when tokens are revoked. This can significantly reduce the storage requirements of keeping such token ids at CA and RSU.

Before explaining our hash chain approach (Figure 5.4), we first discuss the intuition behind it. The approach utilizes hash chains because each hash value of theirs look random and thus can be used as a τ id. However, since the hash function H is public, RSU can determine if two hashes x_i, x_j belong to the same chain by using one to derive the other using H, which violates unlinkability. To fix this issue, CA uses a per vehicle secret r as

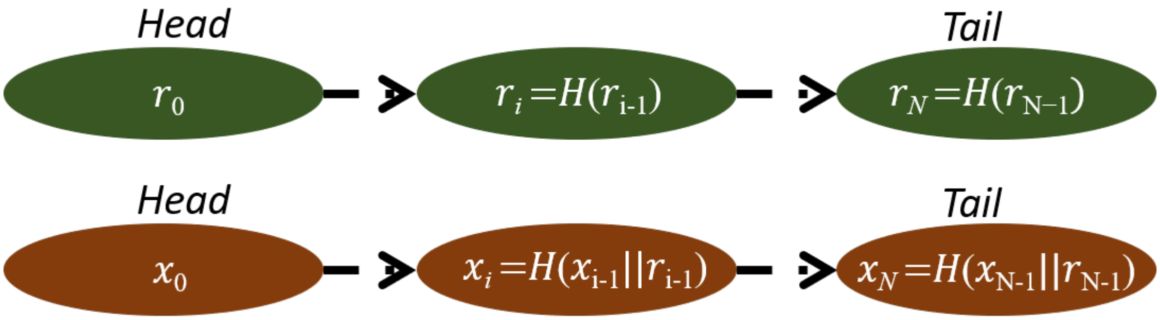

Figure 5.4: Our new hash chain mechanism.

part of the H input to traverse the τ id hash chain of v; thus, RSU is unable to link τ ids without knowledge of r due to the inversion hardness assumption of H. Additionally, upon revocation and revealing the secret r, our mechanism needs to preserve unlinkability of v before it was revoked (*e.g.,* protect privacy of a stolen v). To achieve this property, the per vehicle secret r is also constructed as another hash chain as shown in Figure 5.4; thus, when v is revoked at a time interval, only corresponding hash values from both hash chains are revealed, which only allow RSU to derive current and future revoked tokens of v.

Our hash chain mechanism works as follows; At vehicle Setup, CA picks two random values, x_0,r_0, called heads. Upon executing PseudoGen, CA uses H and the heads x_0,r_0 to create two hash chains of length $N + 1$, where N is the number of tokens generated as a result of this request; the next PseudoGen request heads become x_N,r_N. In the hash chain of secrets (*i.e.,* $r_0, \cdots, r_N$), at the i^{th} position, the value is calculated by $H(r_{i-1})$. The second hash chain, which produces the τ ids, makes use of the values from the first chain. Specifically, the value at the i^{th} position is calculated by applying $H(x_{i-1}||r_{i-1})$. Whenever a malicious vehicle is detected, then the remaining tokens are revoked by publishing the corresponding input values to the current token x_i being revoked, that is, (x_{i-1}, r_{i-1}). With this, the RSU will be able to invalidate the other remaining tokens acquired by the revoked v, which allows RSU to detect and refuse providing new PCs to revoked vehicles.

5.4 Security

In this section we discuss the security of our new system TVSS. We begin in Section 5.4.1 by considering Sybil and clone attacks. Then in Section 5.4.2 we give a formal game-based security definition for VPKI which captures anonymity and unlinkability. Then, in Section 5.4.3, we prove that TVSS satisfies the security definition.

5.4.1 Sybil and Clone Attacks

Sybil Attacks. A Sybil attack occurs when a single malicious vehicle impersonates many independent vehicles. Sybil attacks are a major concern in SCMS as the vehicles are given many valid PCs simultaneously and instructed, but not forced, to use them one at a time. However, in our system vehicles get only one valid PC at a time and so a Sybil attack is only possible if the malicious vehicle obtains (either consensually or illicitly) active PCs from other actual users. Thus, launching a Sybil attack on TVSS is much more difficult than on SCMS since it requires either convincing several other users to misbehave, or it requires obtaining their credentials via some other means. In this case, if the Sybil attack were noticed, the offending vehicle's credentials would be revoked, along with the credentials of all vehicles whose PCs were used.

Clone Attacks. A clone attack occurs when a malicious vehicle shares its token with another vehicle in another geographic area and both of them use the token to obtain a valid

PC and participate in the system. Clone attacks are more problematic for our system since there is nothing tying the token to the vehicle, and so there is no way for the second RSU to notice that the token does not "belong" to the clone vehicle. However, when the RSUs send the tokens to the RSU backend for analysis, the RSU backend will discover the clone attack since it will notice that the same token was used in two different places. Immediate revocation will follow, and so the next time the offending vehicle tries to get a PC from a RSU, it will find that its tokens no longer work.

5.4.2 VPKI Security Formalization

In this section, we formalize the security notions of anonymity and unlinkability for a VPKI system using a game-based security definition.

Security Game Overview. Let $(\mathsf{Setup}, \mathsf{TokenGen}, \mathsf{PseudoGen})$ be the algorithms from a TVSS system (the Revoke procedure plays no part in this game), and furthermore let $(\mathsf{Sign}, \mathsf{Verify})$ be the sign and verify algorithms for the signature scheme used in the VPKI (recall that the PCs consist of a signature key pair and a signature on the public key using the RSU's secret key). In the security game, we give the adversary $\mathcal{A}$ full control of all parties in the system: $\mathcal{A}$ can create vehicles using Setup, $\mathcal{A}$ can request tokens/PCs as v using $\mathsf{TokenGen}/\mathsf{PseudoGen}$ which $\mathcal{A}$ can choose to either expose or not. Eventually $\mathcal{A}$ decides how it will attempt to win the game; it has three options, it can win via forgery, it can break anonymity, or it can break unlinkability. The challenger $\mathcal{C}$ then issues a corresponding challenge to $\mathcal{A}$, and $\mathcal{A}$ attempts to win the game. For example, if $\mathcal{A}$ decides it wants to win via anonymity, then $\mathcal{A}$ will specify two different vehicles and $\mathcal{C}$ will a PC of one of them and

$\mathcal{A}$ must guess to which vehicle the PC belongs. The security game, and the full set of legal queries and winning conditions are shown in below. Definition 1 simply says that the VPKI scheme is secure if no efficient adversary can win with any non-negligible advantage. Note that $\mathcal{A}$ can trivially win the anonymity or unlinkability version of the game by guessing randomly, therefore advantage in this case means $\Pr\left[\mathcal{A} \text{ wins}\right] - 1/2$, while in the forgery version of the game $\mathcal{A}$'s advantage is simply $\Pr\left[\mathcal{A} \text{ wins}\right]$.

Definition 1 *We say that a VPKI scheme is* secure*, if for all efficient adversaries $\mathcal{A}$, $\mathcal{A}$'s advantage in winning the VPKI security game is negligible.*

VPKI Security Game

1. **Initialization:** The challenger $\mathcal{C}$ initializes several data structures, specified in the next section, to keep track of what transpires in the system during the course of the game. *For example, $\mathcal{C}$ creates sets $V_{\mathsf{corrupt}} \subset V$, both initialized to empty, which keep track of the corrupt vehicles in the system. Or $\mathcal{C}$ can keep track of the corrupt tokens of a vehicle v in the set $T^v_{\mathsf{corrupt}} \subset T^v$.*

2. **Online Phase:** The adversary $\mathcal{A}$ interacts with $\mathcal{C}$ using the specified queries, defined later. *For example, if $\mathcal{A}$ wants to add a new vehicle to the system, $\mathcal{A}$ would use the $CreateVehicle(\cdot)$ query, at which point $\mathcal{C}$ would run Alg 1 and add (v, EC_v) to V. Or, if $\mathcal{A}$ decides it wishes to attack the anonymity of our scheme, it would issue the Anonymity query.*

3. **The Final Message:** $\mathcal{A}$ sends its final message and the game ends. $\mathcal{A}$ wins if it meets one of the winning conditions specified in the next section.

Initialization and Queries

Intialization Data: The Long-term keys to the vehicle, pseudonym certificates, and tokens are the critical components of the security model. For these data to be used and presented, we will need algorithms 6, 7, and 8.

(1) A set V which will contain public/private key pairs (v, EC_v); V represents the set of all cars which ever exist in the system; V is initialized to the emptyset;

(2) A set $V_{\mathsf{corrupt}} \subset V$ of corrupt cars; V_{corrupt} is initialized to the emptyset;

(3) For every $v \in V$, $\mathcal{C}$ keeps track of a set T^v which represents the set of tokens for the vehicle v; $\mathcal{C}$ also maintains $T^v_{\mathsf{corrupt}} \subset T^v$, the set of corrupted tokens for v; every time a new v is added to V, T^v and T^v_{corrupt} are initializes to empty sets;

(4) For each $v \in V$, $\mathcal{C}$ keeps track of a set of valid pseudonym certificate P^v at a given time interval; If the vehicle's PC is exposed, then the same is added to the set P^v_{corrupt};

Interaction Queries: These queries emulate how the adversary might interact with the system in the real world.

1. CreateVehicle$(\cdot)$: $\mathcal{C}$ runs Alg 6, generates (v, EC_v) which it adds to V, $\mathcal{C}$ also creates sets T^v and T^v_{corrupt} initialized to the emptyset, finally $\mathcal{C}$ sends v to $\mathcal{A}$.

2. CorruptVehicle(v) : $\mathcal{C}$ finds (v, EC_v) in V (if there is no pair in V with first coordinate v then $\mathcal{C}$ does nothing) and returns EC_v to $\mathcal{A}$ and adds the pair to V_{corrupt};

3. GetToken(v) : $\mathcal{C}$ runs Alg 7 with input v and receives a token τ, and adds τ to the set T^v.

4. CorruptToken(v) : $\mathcal{C}$ finds the token $\tau \in T^v$, adds it to the set T^v_{corrupt} and sends it to the adversary.

5. GetPC(v) : $\mathcal{C}$ searches a valid token $\tau \in T^v$ and runs the Algorithm 8 with input (v, τ). When it receives a PC, it adds it to the set P^v.

6. ExposePC(v) The challenger selects a valid pseudonym certificate from the set P^v, adds it to P^v_{corrupt}, and sends it to $\mathcal{A}$.

Challenge Queries: These queries indicate that the adversary is ready to break some facet of the scheme's security.

1. Anonymity(v_0, v_1): $\mathcal{C}$ selects valid pseudonym certificates P^{v_0}, P^{v_1} for the vehicles v_0, and v_1 respectively. It then chooses a bit b at random from the set $\{0, 1\}$ and sends P^{v_b} to $\mathcal{A}$.

2. Unlinkability(v_0, v_1): $\mathcal{C}$ selects valid pseudonym certificates from P^{v_0}, P^{v_1} for the vehicles v_0, and v_1 respectively. $\mathcal{C}$ sends them to $\mathcal{A}$. It then chooses a bit b at random from the set $\{0, 1\}$ and sends a different PC from the set P^{v_b} to $\mathcal{A}$.

3. Forgery(v): $\mathcal{C}$ selects pseudonym certificates from P^v and sends them to the adversary $\mathcal{A}$. These PCs are added to the set P^v_{corrupt}.

Winning Conditions

In our security formalization, we give the adversary $\mathcal{A}$ the power to query before and after the challenge message is received from $\mathcal{C}$. To win, the adversary must satisfy one of the conditions below.

- When the adversary sends its final message(the guess bit b') in case of an Anonymity challenge, the probability of the correct message must be $1/2 + non - negl$.
- When the adversary sends its final message(the guess bit b') in case of an Unlinkability challenge, the probability of the correct message must be $1/2 + non - negl$.
- When the adversary sends its final message, m signed by a PC of the vehicle v, in case of a Forgery challenge, the probability of the correct message must be $non - negl$.

5.4.3 Security Proof

Assume for contradiction that an efficient adversary $\mathcal{A}$ wins the VPKI game with non-negligible advantage. We construct another efficient adversary $\mathcal{B}$ which breaks the security of the signature scheme $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ with related advantage. The first thing $\mathcal{B}$ does is it guesses how $\mathcal{A}$ will try to win the VPKI game; it will proceed differently in the three cases.

Suppose first, that $\mathcal{B}$ decides $\mathcal{A}$ will win the forgery version of the VPKI game. In this case, $\mathcal{B}$ begins playing as the adversary in the security game for $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$. Specifically, the signature game challenger $\mathcal{C}$ draws $(\mathsf{vk}, \mathsf{sk}) \sim \mathsf{G}$ and sends vk to $\mathcal{B}$. Then $\mathcal{B}$

invokes $\mathcal{A}$ and plays as the challenger in the VPKI game. $\mathcal{B}$ picks a random vehicle v and a random PC for v and uses vk as the public verification key portion of the PC; all other queries of $\mathcal{A}$ are answered honestly by $\mathcal{B}$, just as the honest challenger would answer them. If $\mathcal{A}$ does indeed decide to win the forgery version of the game, and moreover if $\mathcal{A}$ intends to use the selected vehicle v and the selected PC to produce the forgery then $\mathcal{B}$ proceeds, otherwise $\mathcal{B}$ aborts. In case $\mathcal{B}$ proceeds, $\mathcal{B}$ simply forwards $\mathcal{A}$'s forged message to $\mathcal{C}$. It is clear that $\mathcal{B}$ wins the forgery game whenever $\mathcal{A}$ wins the forgery version of the VPKI game using the vehicle v and the selected PC. Thus, if $\mathcal{A}$ has a non-negligible advantage of winning the forgery version of the VPKI game, then $\mathcal{B}$ has non-negligible probability of breaking the signature scheme.

Now, let us suppose that $\mathcal{A}$ has a non-negligible advantage of breaking either the unlinkability or the anonymity versions of the VPKI game. We derive an information theoretic contradiction in this case by showing that $\mathcal{A}$ has the ability to predict a random bit with positive advantage. The key point here is that the PCs are completely independent of the vehicle and of each other and so there is no way $\mathcal{A}$ can win the anonymity or unlinkability branch of the game with probability better than $1/2$. We prove this for anonymity, the case of unlinkability is similar. Consider an adversary $\mathcal{B}$ who emulates the VPKI challenger in all ways except that it chooses two random vehicles v_0 and v_1 and it generates two PCs: P_0 and P_1. Neither PC is associated yet with either vehicle. Now $\mathcal{B}$ flips a coin; if heads, $\mathcal{B}$ associates P_b with v_b for $b = 0, 1$; if tails $\mathcal{B}$ associates P_b with v_{1-b} for $b = 0, 1$. Now, if $\mathcal{A}$ decides to win the anonymity branch of the VPKI game using vehicles v_0 and v_1, then $\mathcal{B}$ sends P_0 to $\mathcal{A}$ (if $\mathcal{A}$ decides to win a different branch, or decides to use different vehicles, then $\mathcal{B}$ outputs a random bit. Now, note that if $\mathcal{B}$'s coin landed on b, then $\mathcal{A}$ needs to return b to win. However, as $\mathcal{B}$'s challenge is independent of b, the probability that $\mathcal{A}$ returns b is exactly $1/2$. Therefore, it is not possible for $\mathcal{A}$ to have an advantage in the anonymity branch of the VPKI game. The same argument shows that $\mathcal{A}$ cannot have any advantage in the unlinkability branch.

So in summary, it must be that if $\mathcal{A}$ has a non-negligible advantage in winning the VPKI game, then this advantage must lie in the forgery branch of the game. However, as shown above, in this case we can design an efficient adversary $\mathcal{B}$ which breaks the security of the underlying signature scheme.

5.5 Experimental Testbed

In this section, we first present the hardware used for our experiments. We also provide technical details on our connected vehicles networking technology IEEE 802.11p/DSRC setup. We then discuss implementation details and issues when conducting experiments on connected vehicles. We finally explain the design of our field experiments on connected vehicles; Figure 5.5 and 5.6 shows a snapshot of our highway testbed.

5.5.1 Hardware

We obtain 5 PC Engine APU1D4 embedded devices that can act as either on-board units (OBUs) or road-side units (RSUs). APU1D4 devices are carefully chosen as they have technical specifications that are similar to commercial OBUs and RSUs shipped by

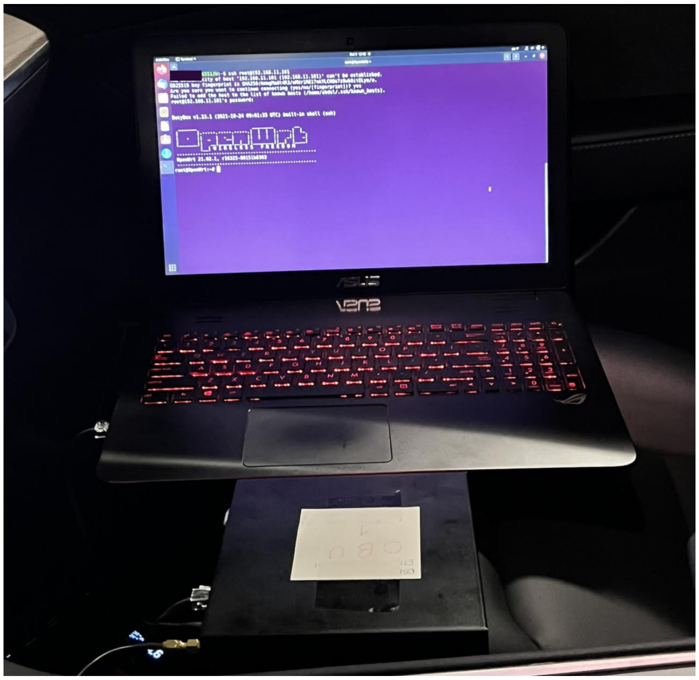

Figure 5.5: An OBU controlled by a laptop.

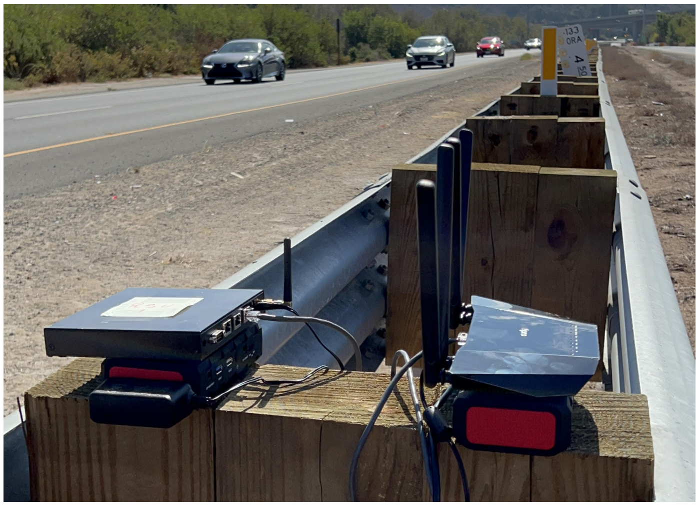

Figure 5.6: An RSU connected to a cellular router on the highway.

well-known vendors like commsignia (ITS-OB4 OBU and ITS-RS4 RSU) and Savari (MW-1000 OBU and SW-1000 RSU). Each of our PC Engine APU1D4 devices has AMD G series T40E - 1 GHz dual Bobcat core, 4 GB DDR3-1066 DRAM and a 16GB SSD hard drive. We build APU1D4 based OBUs and RSUs from scratch so that we can have an open source environment where we can implement various applications/VPKI protocols to get real field measurements of their performance in a vehicular environment. Our OBUs and RSUs run on a patched version of OpenWrt 21.02.1, a well-known firmware mainly used with embedded and networking devices and is utilized by Savari commercial OBUs and RSUs.

Moreover, we set up two cloud servers at a well-known cloud provider, Digital Ocean, acting as the certificate authority (CA) and registration authority (RA), which are needed by alternative cloud-based VPKI systems. Each of the CA and RA runs on an independent server with 4 CPUs, 8GB of DDR3 RAM and 160GB of SSD hard drive. We also obtain Cudy N300 routers, which are equipped with LTE SIM cards from a cellular service provider; RSUs are connected to cellular routers to enable internet connectivity. Whenever OBUs need to reach the internet (*e.g.,* for CA, RA ...etc), they make use of RSUs to relay their requests to the cellular router.

5.5.2 IEEE 802.11p/DSRC Setup

In order to enable our OBUs and RSUs to use the networking standard IEEE 802.11p (DSRC) - which is specifically designed for automotive communication, we install UNEX DHXA-222 wireless network interface cards (NICs), which are compatible with IEEE 802.11p/DSRC. These NICs have been validated and proven to deliver reliable IEEE 802.11p/DSRC communication [150, 151]. To have IEEE 802.11p/DSRC properly set up, we configure our NICs to use the Outside Context of a BSS (OCB) mode; this allows direct communication between OBUs and RSUs (*i.e.,* V2I/V2V) in an instant fashion with no association/authentication handshakes needed at the link layer. We also use the vehicular standard 5.9GHz frequency band with a channel width of 10MHz (namely, channel 178 at 5.890GHz) as specified by the standard. While the NICs allow us to set bit rates of 3-27Mbps, we configure our testbed to have the baseline standard DSRC bit rate of 6Mbps [7]. We also use the IEEE 802.11p compatible antennas from MobileMark (for OBUs) and PulseLarsen (for RSUs) and set the transmission power to 15dBm.

5.5.3 Implementation

We develop Bash scripts that utilize the ICMP protocol in order to determine the contact time between OBUs and RSUs when CVs pass by RSUs. We also use Python to implement 3 VPKI systems to measure their performance, namely 1) TVSS, 2) SECMACE [96], a recent VPKI system, and 3) SCMS [40], the US VPKI standard.

In SCMS, an OBU can request a fresh PC only from the cloud. In SECMACE, the OBU requests tickets from the cloud. Then the OBU uses these tickets to obtain fresh PCs from a Pseudonym Certificate Authority (PCA). Note that in SECMACE, the authors assumed the PCA to be in the cloud as well, but in order to perform a fair comparison between SECMACE and DVSS, we decided to place the implementation of PCA for SECMACE inside the RSUs. In DVSS, the OBU requests tokens from the cloud and uses these tokens to obtain fresh PCs from the RSUs. The PC in both SECMACE and DVSS is valid only within a specific region but the main difference is that when a vehicle leaves the validity region in SECMACE, the OBU must request new tickets from the cloud, while in DVSS the OBU simply uses a token to request a new PC from the RSU for the new region.

In all of the cloud based VPKIs, any interaction with the cloud must go through an RSU.

Each RSU is equipped with a switching and a routing fabric and configured with a routing table to quickly and efficiently forward any OBU request to a cloud-based IP address via the connected 4G gateway (see Section 5.5.1 and Figure 5.5 and 5.6). Note that from an OBU point-of-view, the RSU is the gateway to the cloud servers. In addition, all communications between OBUs, RSUs and cloud servers use reliable and secure channels TCP/TLS1.3, and for that the mainstream Python libraries socket and ssl are utilized. We use hashlib library for hashing using SHA256.

For public key cryptography, we utilize PyNaCl, a Python binding to libsodium, which is a fork of the Networking and Cryptography library (NaCl). We use the library to implement public key encryption using the Elliptic Curve Integrated Encryption Scheme (ECIES) algorithm with the curve25519 elliptic curve and EdDSA digital signature algorithm with the ed25519 elliptic curve. Both algorithms use 256-bit long private keys and are adopted by SCMS. All VPKI components (OBU, RSU, CA and RA) use certificates for authentication.

The nature of experimenting on fast moving vehicles makes the contact time between OBUs and RSUs (*i.e.,* gateways) limited as will be shown in Section 5.6. We initially experienced difficulties measuring cloud-based VPKI systems accurately since they tend to have longer round trip time (RTTs) and OBUs could try to communicate with cloud by sending a packet just right before it is in contact with the RSU and wait till after it is out of coverage. We address this issue by letting OBUs periodically create a new independent thread and send a new request to the cloud servers. We set the timeout to 30ms for OBU-RSU communication and 2s for OBU-cloud communication; a new request thread is created every 500ms to make sure VPKIs get a chance to make a request while in contact with the RSU and to not saturate the receivers.

5.5.4 Experimental Scenarios

We evaluate all 3 VPKI systems under the following scenarios: 1) highway and 2) in-city scenarios. For the highway scenario, the highway has 4 lanes (2 are inbound and the other 2 are outbound) and a median strip in the middle. 1 OBU is installed in a vehicle and 4 RSUs are placed on the median strip (*i.e.,* middle of highway). The vehicle passes by the 4 geographically distributed RSUs at various highway speeds ranging from 55mph to 85mph. For the in-city scenario, the street has 2 opposing lanes with no median strip; 1 OBU is installed in the vehicle and 4 RSUs are placed on the side of the street. The vehicle passes by the 4 geographically distributed RSUs at relatively slow speeds, namely 25mph and 35mph. In both scenarios, RSUs do not have overlapping coverage and are connected to cellular routers, and the OBU antenna is placed outside the vehicle for better signal transmission and reception. In each scenario, we conduct two sets of experiments, PC Procurement (obtaining a PC from the VPKI) and PCRL Download (retrieve a complete and up-to-date PCRL from the RSU). For each experiment and each speed, we run and repeat the experiment 25 times which generate 100 results per run (25 trials x 4 RSUs).

5.6 Performance Evaluation

In this section, we present a real field-based study of TVSS and its two alternatives (SECMACE and SCMS) using the measurements collected by our testbed of OBUs, RSUs, and cloud servers on highway and in-city streets at a range of driving velocities as mentioned in Section 5.5.4. As described in Section 5.3, TVSS is edge-based, pushing primary services to the edge of the network (the RSU) to enable operation compatible with potentially short connection times. As a result, we avoid communicating with centralized back-end services during critical operations, shortening response time and increasing the reliability of the offered services. A primary advantage of TVSS in a V2I communication setting, is that it enables anonymized reauthentication of connected vehicles (CVs) as they drive by RSUs. A CV may have access to the infrastructure services for a short period of time (depending on its speed, range, and location) before it exists the coverage of the serving RSU. The purpose of our experiments is to measure the effect of the network coverage time on the service performance and achieved security for different traveling speeds. We additionally discuss the scalability of actual RSUs if TVSS were to be deployed. For all VPKIs, we collected 100 field samples for each of the experiments in order to get statistically reasonable results. Next we present our evaluation campaigns; each campaign evaluates a specific metric in the design of the VPKIs.

Evaluation 1: Measuring Basic Coverage Times. In the first evaluation, we aim to capture the time a vehicle stays in the coverage of an RSU under different speeds and directions to study the implications of vehicle network connectivity time on VPKI. The results are presented in Table 5.1. As we can see, in a worst-case scenario, for a vehicle traveling at a high speed (85mph) that passes by an RSU from the outer region of its coverage area (i.e. far away from the RSU), the coverage time can be lower than 100ms. Our results also show that vehicles driving on the highway (55mph-85mph) have coverage

Speed	Coverage Time (seconds)			
	Minimum	Median	Average	Maximum
85mph	<0.1	1	1.39	5
75mph	0.32	1	1.91	5
65mph	0.61	2	2.07	6
55mph	1	3	3.30	8
35mph	3	6	5.75	10
25mph	3	6	6.30	12

Table 5.1: Highway and in-city OBU-RSU coverage time.

time with the RSU ranging from 1.39s to 3.30s, which is very limited. This suggests that all communications with the RSU need to be completed in a very short period before vehicles are out of coverage. Furthermore, the operations that require the backend/cloud involvement require the link between the RSU and backend servers to always be reliable, which might be infeasible in vehicular environments (*e.g.,* RSU might have to rely on an unstable wireless service, with only intermittent connectivity.) In addition, Table 5.1 measures the coverage times for vehicles driving inside the city at relatively slow speeds (25mph-35mph). While vehicles have more contact time with RSUs (5.75s to 6.30s) in the city, the results show that there are cases where contact time can be limited (as low as 3s). This is an expected outcome in realistic testbeds as the conditions of wireless channels fluctuate in dynamic environments such as vehicular networks.

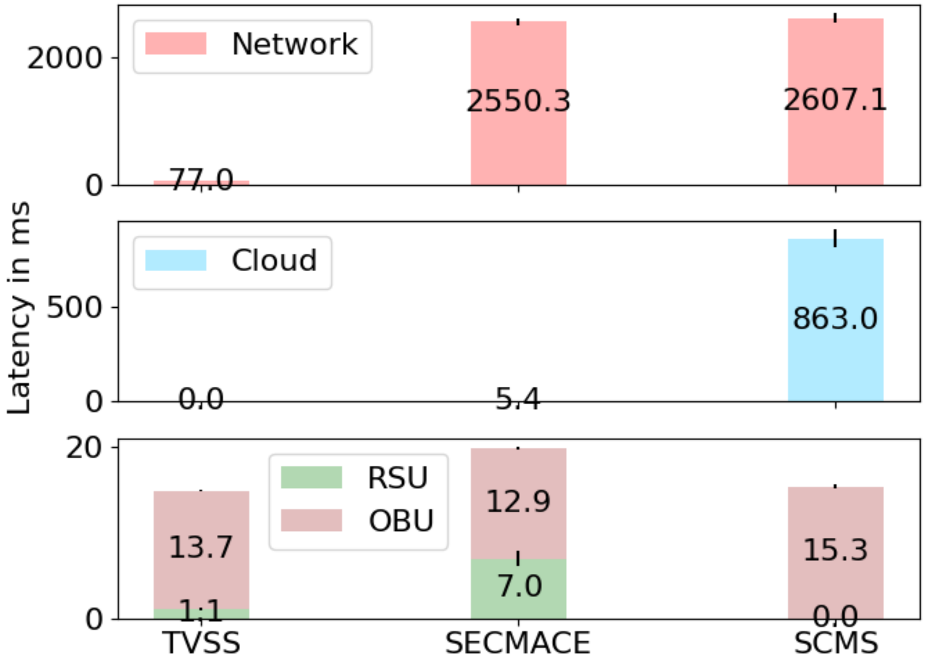

Figure 5.7: PC generation latency of VPKI systems.

Evaluation 2: PC Generation Latency. In order to get a better understanding of the design implications of VPKI systems, we measure the end-to-end latency of a PC request operation using different VPKI designs; TVSS, our proposed VPKI system, SECMACE, a recent VPKI system, and SCMS, the US VPKI standard. Recall that for PC generation, TVSS completely relies on the RSU while SECMACE relies on both the RSU and a cloud-based CA. For SCMS, it relies completely on the cloud servers. Figure 5.7 presents the elapsed time at the OBU, RSU, network and cloud servers for completing a single PC generation request. For all VPKI protocols, the most dominating factor is the network time. Our results show that relying completely on the edge for refreshing PCs, as in TVSS, helps cut down network latency by 33.5x compared to relying on the cloud. Thus, these results show that TVSS reduces total PC generation latency by 28.5x and 38.5x compared to SECMACE and SCMS, respectively. The figure additionally indicates that

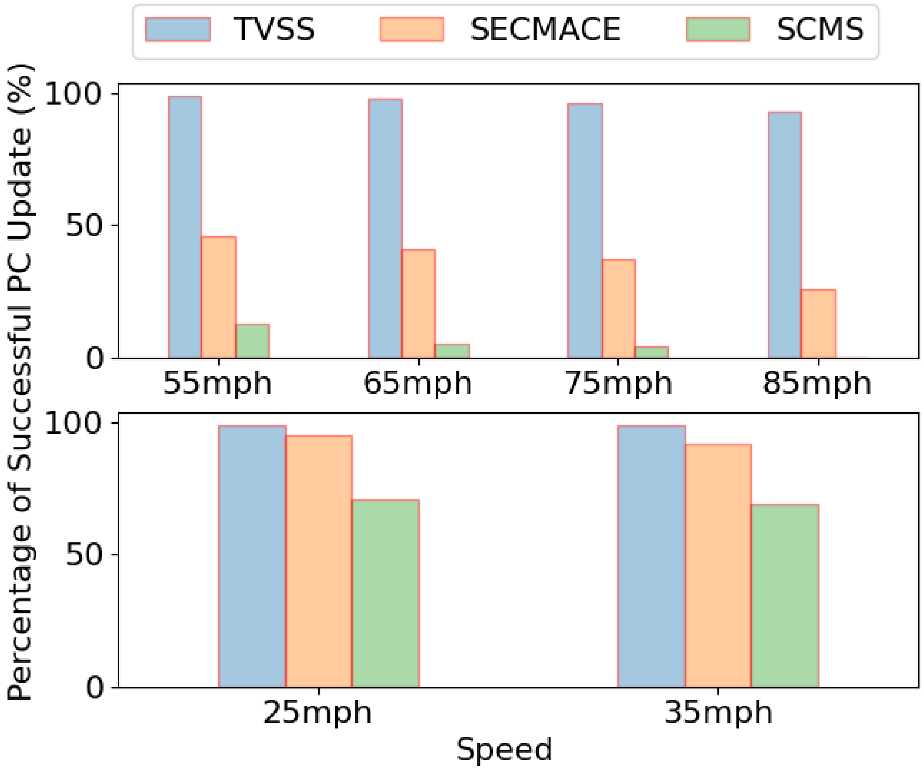

Figure 5.8: Success ratio of vehicles refreshing a PC via all VPKIs under highway speeds (top) and in-city speeds (bottom).

RSU overhead is reduced by 6.4x in TVSS compared to SECMACE; this has bad implications on the scalability of RSUs for SECMACE as will be discussed later in this section. Even though SECMACE PC generation cannot rely completely on the edge for security reasons, TVSS cuts down the combined edge overhead (OBU and RSU) by 1.36x compared to SECMACE. This implies that TVSS is a more efficient solution compared to its alternatives.

Evaluation 3: PC Renewal Success Ratio. In this evaluation, we want to measure the ability of various VPKI systems to issue a PC under the constraint of a vehicle coverage time when it drives by an RSU. We test the three VPKIs under two scenarios: 1) highways with speeds of 55mph-85mph and 2) in-city with speeds of 25mph-35mph. The results are reported in Figure 5.8 (top). From the figure, we can see that for the case of a regular highway speed of 55mph, PC refreshes can be completed successfully with a ratio of 99% for TVSS, 46% for SECMACE and 13% for SCMS; this means that TVSS achieves 2.15x and 7.62x better reliability compared to SECMACE and the standard VPKI SCMS, respectively. For the the worst case of a CV driving on the highway at 85mph, PC refresh success ratio are 93% for our proposed system TVSS and 26% for SECMACE. For SCMS, however, it was unable to refresh PCs at such small RSU-OBU contact time. This shows that TVSS improves performance over SECMACE by 3.58x and is a reliable under such harsh conditions.

Furthermore we test all VPKI protocols on a city street under low speeds (*i.e.,* in-city speeds), namely 25mph and 35mph, and show their PC issuance success ratio in Figure 5.8 (bottom). The results show that when vehicles are driving at 25mph, the PC generation success ratios are 99% for TVSS, 95% for SECMACE and 71% for SCMS. While TVSS outperforms SECMACE and SCMS, it is clear that cloud-based VPKI protocols are acceptable for slow moving vehicles but not for fast velocities at highways, which usually constitute most of the vehicle's trip. One can see that SECMACE and SCMS have a difference in their performance although both rely on the cloud; this is because SECMACE has a more efficient design and divides the work needed to issue a new PC between the cloud and an RSU while SCMS completely relies on the cloud. Our protocol, on the other hand, utilizes minimal cryptographic operations for the PC

Region	PCRL Size (MB)	
	Regular Revocation	Mass Revocation
	(1.29%)	(5%)
Entire US	176.4	683.6
California	19.9	77.3
Texas	14.7	56.9
Florida	11.4	44.1
New York	7.3	28.2
Delta PCRL (weekly)	3.4	13.1
Delta PCRL (daily)	0.49	1.87
Local PCRL	0.04	0.16

Table 5.2: PCRL types and sizes.

generation issuance process and moves it completely to the edge (*i.e.,* RSU), which allows for this significant improvement.

Evaluation 4: PCRL Size. In this evaluation, we discuss the feasibility of pseudonym certificate revocation lists (PCRLs) used in TVSS and other VPKI systems. Recall that PCRLs hold pseudonym identities of revoked vehicles and are used to notify and protect CVs from malicious vehicles. In all calculations of other VPKI protocol PCRLs, we consider the hash chain optimizations used in SCMS to reduce the PCRL size while in transit. TVSS on the other hand, does not need to use these optimizations as each OBU can have a single PC at any time interval. In order to provide a reasonable projection of PCRL sizes in VPKI, we consider the revocation ratios in another similar domain, namely the internet; [173] shows that the regular revocation ratio in the internet is ∼1.29% of the whole PKI system. Because there might also be cases of mass revocation such as the cyber attack in 2015 which allowed hackers to disable vehicles and triggered Chrysler to recall 1.4 million vehicles [2], we also consider mass revocation with a ratio of 5%. Considering the US scenario with a total of 350 million vehicles, this would result in a PCRL of ∼177 MB in case of regular revocation and ∼684 MB in case of mass revocation as shown in Table 5.2, which both require high bandwidth. In order to reduce the bandwidth requirements of downloading a PCRL, VPKIs additionally consider utilizing delta PCRLs, where the PCRL is incrementally updated so that the vehicle only downloads the newly revoked vehicles (*i.e.,* delta PCRL) when it gets network connectivity with the RSU/backend (*e.g.,* weekly/daily); nonetheless, the PCRL storage requirement at the vehicle stays the same as mentioned earlier. We also divide the size of the entire US PCRL size by the number of weeks and days to compute sizes of the weekly/daily delta PCRLs, respectively, as shown in Table 5.2.

In order to reduce the PCRL size, VPKI systems also suggest dividing the revoked certificates on different PCRLs based on some common factor, such as region of revocation (*e.g.,* a state) [40], so that a vehicle only downloads and keeps a relevant PCRL to it. In Table 5.2, we show the anticipated size of PCRLs in the top four states in the US based on number of vehicles in each state. All states PCRLs require high bandwidth and storage as shown in Table 5.2. Notice that this approach is not secure because it can still allow a revoked vehicle to maliciously participate in the system outside the region of revocation without being detected as long as the PCs are valid (*e.g.,* SCMS PCs are valid for a week); this can degrade the safety and efficiency of the vehicular environment.

TVSS, on the other hand, ties each PC to a very small region (*i.e.,* an area between RSUs), which allows for significantly reduced PCRLs, without having the aforementioned vulnerability. To predict a realistic size of a local PCRL, we consider the total number of vehicles that can fill up an area between RSUs in a highway. Particularly, we consider the real Wyoming deployment of RSUs in which RSUs are at most 20 miles apart [183]. Our results show that our technique of local PCRLs allows them to be extremely small even in case of mass revocation (∼160KB), which are 13x smaller than delta PCRLs (*i.e.,* the best SCMS choice) which eventually improves the security of the system. Please note that SCMS can not be adapted to tie PCs to small regions as this would violate anonymity and allow for tracking by the backend.

Evaluation 5: Ratio of Successful Download of PCRL. In this evaluation, we conduct a second set of field experiments to show the ratio of success to download PCRLs by CVs while driving on highways and in the city when passing by RSUs for both cases of regular revocation (Figure 5.9) and mass revocation (Figure 5.10). Note that these results were collected using the baseline DSRC data rate of 6Mbps as specified in [7]. The PCRL sizes considered in this experiment are the PCRLs that cover the entire US, delta PCRLs (that are weekly/daily updated), specific states PCRLs, and small regions (*i.e.,* our novel local PCRLs). The size of each PCRL type is shown in Table 5.2. From the experiment results, we observe that in both regular and mass revocations, OBUs are not able to

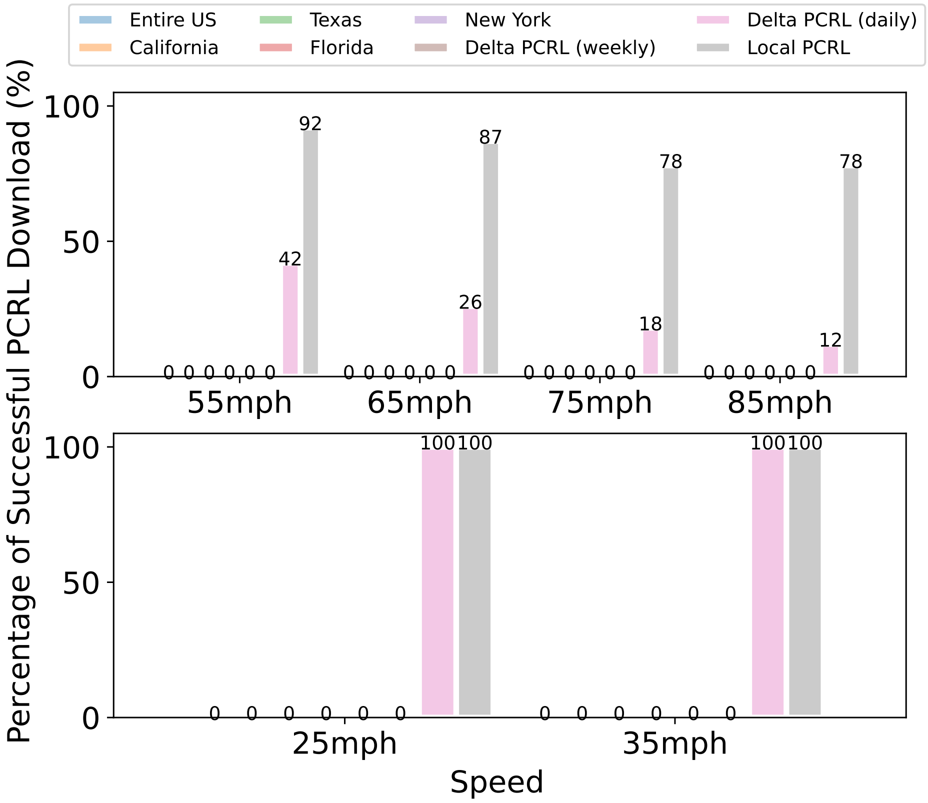

Figure 5.9: The success ratio of downloading a complete PCRL under *regular* revocation for various PCRL sizes in highway speeds (top) and in-city speeds (bottom).

download the whole nation-wide, states, and weekly updated PCRLs due to their massive size, which leaves OBUs unable to detect malicious OBUs; thus, these PCRL types are not feasible solutions as they make OBUs become oblivious of revoked and malicious OBUs. Note that for all PCRL types, we assume RSUs already have them stored locally and thus OBUs download them directly from the RSUs without the need to contact cloud servers so as to reduce network latency. Also note that alternatives to PCRLs such as certificate status inquiry protocols requiring communication with the CA or some central servers (*e.g.,* OCSP) are not feasible as OBUs use DSRC only and do not always have contact

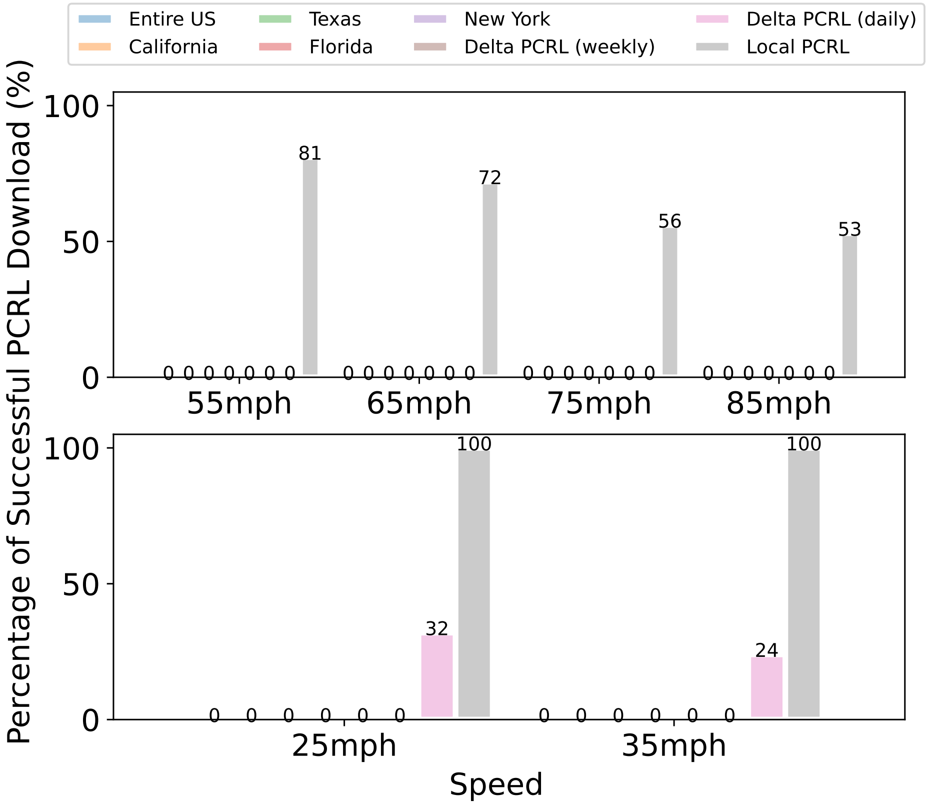

Figure 5.10: The success ratio of downloading a complete PCRL under *mass* revocation for various PCRL sizes in highway speeds (top) and in-city speeds (bottom).

with RSUs.

Under regular revocation (Figure 5.9); for the regular highway speed of 55mph, only 42% can download daily delta PCRLs successfully, leaving 58% of legitimate OBUs vulnerable to bogus messages sent by malicious OBUs. On the other hand, our local PCRLs show promising results as 92% of the OBUs are able to download them. For the highest highway speed of 85mph, only 12% can download daily delta PCRLs, leaving 88% of legitimate vehicles unaware of malicious vehicles. Our local PCRLs show promising results as 78% of OBUs can download them at this high speed. This improves the safety and efficiency of the vehicular environment as CVs can quickly detect malicious messages broadcast by revoked OBUs. For in-city speeds (*i.e.,* 25mph and 35mph), the results show that 100% success ratio of downloading local PCRLs and daily delta PCRLs under regular revocation, as expected.

Due to the large scale of VPKI, we additionally consider the effect of mass revocation on the system (Figure 5.10). Our experimental results show that in case of mass revocation (*i.e.,* 5% revocation rates), OBUs now cannot download daily delta PCRLs at highway speeds (55mph-85mph), which indicates unscalability of this technique under mass revocation. In contrast, TVSS shows substantial tolerance to such mass revocation cases since 81% and 53% of vehicles are able to download local PCRLs at 55mph and 85mph, respectively. For in-city speeds under mass revocation, local PCRLs achieves 100% download success ratio while only 32% of OBUs can download the alternative daily delta PCRLs. This makes local PCRLs a scalable solution, that improves the security of CVs.

Evaluation 6: RSU Scalability. Since TVSS proposes to move the whole workload to generate PCs to the edge network, we, in this experiment, examine the scalability of current road-side equipment (*i.e.,* RSUs) to handle such workloads. This is important because we want to make sure that we do not have to incur extra costs by upgrading current RSUs or otherwise end up with a fragile deployment. Specifically, we think about extreme situations where an RSU has to service many OBUs to refresh their PCs in a short time period. We also include SECMACE in our discussion of this experiment to show its RSU scalability (even though SECMACE does not rely on the RSU completely). As shown earlier in Figure 5.7, latency to generate a PC using TVSS and SECMACE on a real RSU are 1.08ms and 6.95ms, respectively. Cryptographic operations dominate latency.

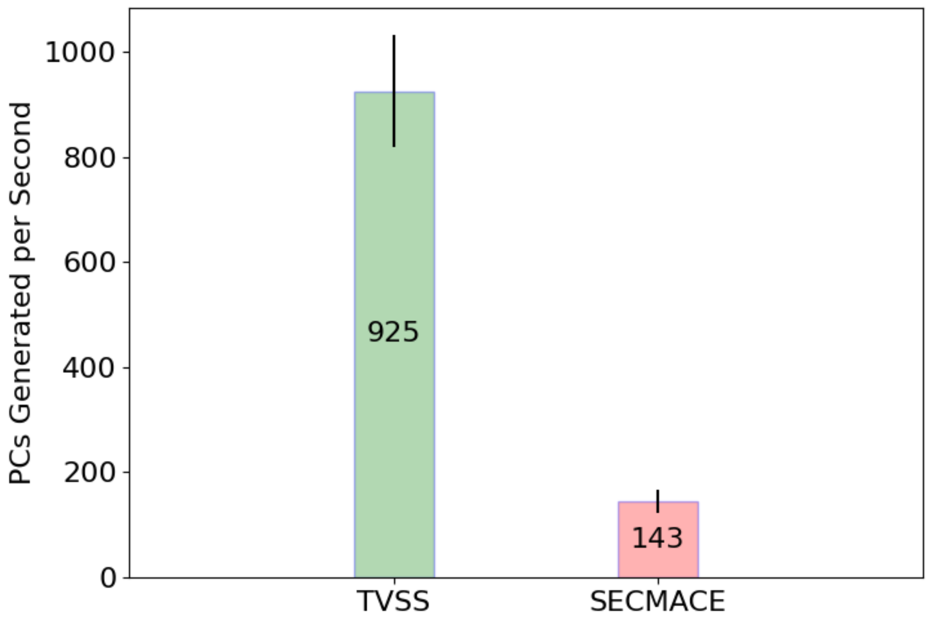

Figure 5.11: Number of PCs generated per second by an RSU.

Consequently, Figure 5.11 shows that one RSU could generate 925 PCs per second for TVSS and 143 PCs per second for SECMACE. To put this into context, we consider a busy interstate and estimate the number of vehicles in a single RSU coverage area. Considering the radius of 150 meters as specified by US DoT [143], an interstate with 8 lanes could contain at most 500 vehicles. Our results show that the RSU is able to service 1.85x the vehicles at a busy interstate when using TVSS. This indicates that RSUs have the enough computational capability to generate PCs without having to upgrade them and incur extra costs. It also shows that TVSS presents a scalable solution for VPKI. SECMACE, however, does not scale as it can service only 28.6% of PC generation requests.

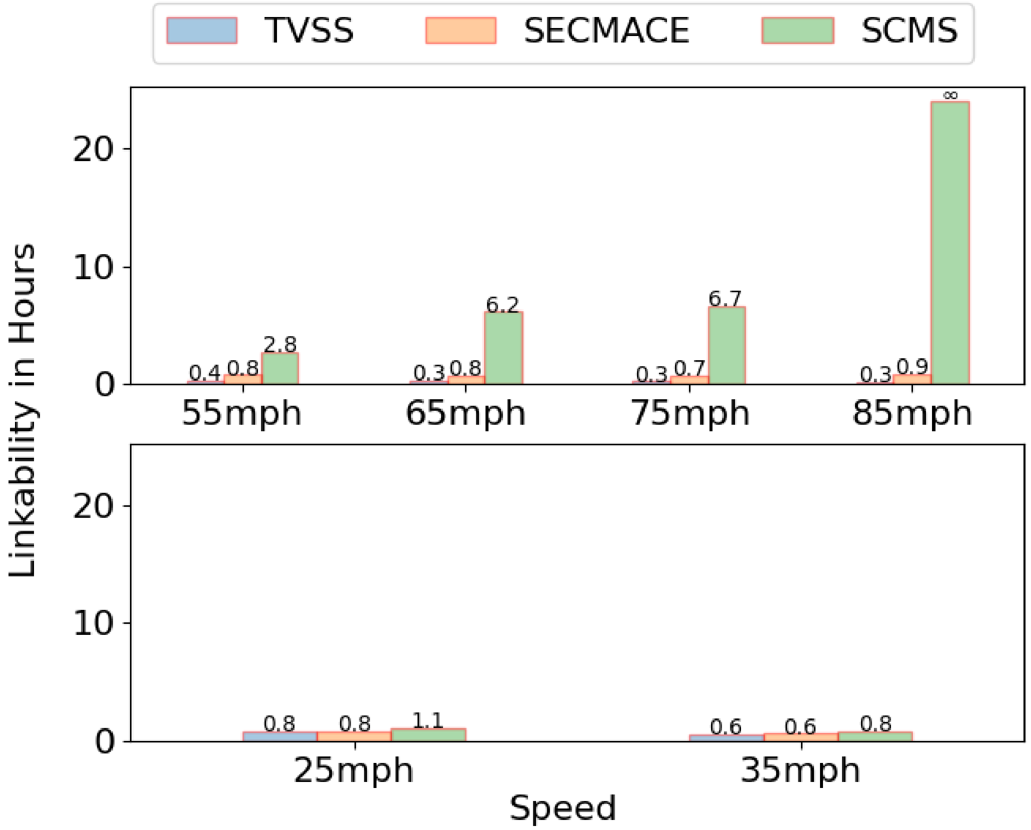

Figure 5.12: Vulnerability window to track vehicles in TVSS, SECMACE and SCMS.

Evaluation 7: Linkability Window in VPKIs. Recall that the unreliability of the CV wireless networks causes a CV to reuse the same PC for longer periods if the CV is unable to renew the PC when driving by an RSU. We, in Figure 5.12, show the linkability windows that curious vehicles/authorities have to link messages of a single vehicle when using different VPKIs for two different scenarios, namely in-city and highways; this eventually leads to other vehicles/authorities tracking vehicle driving activities. In order to provide realistic linkability windows, we consider the aforementioned Wyoming deployment in which RSUs are placed 20 miles apart. Intuitively, we expect that the faster the CV drives, the faster it reaches the next RSU (*i.e.,* smaller duration between two RSUs) and updates its PC; this allows PC validity T to become smaller and thus the vulnerability window. As expected in our system, the results show that the faster the vehicle drives, the smaller the linkability window becomes; this is because CVs successfully refresh PCs whenever in vicinity of an RSU in our system. In contrast, SCMS and SECMACE show a similar pattern but a sharp increase in the vulnerability window when CVs drive on the highway due to the increased failure ratio of refreshing PCs at high speeds. In particular, SCMS and SECMACE show that a single CV driving on the highway at 65mph can be tracked for 6.2 hours and 0.8 hours, respectively, whereas TVSS significantly reduces that vulnerability window to 18 minutes (*i.e.,* $\sim$22.5x and 2.7x reduction in linkability window compared to SCMS and SECMACE, respectively). This makes it harder for adversaries to track CV driving activities due to the limited linkability window. Therefore, TVSS can considerably decrease T, which improves the privacy of the overall system.

Chapter 6

Concluding Remarks and Future Work

It is likely that the number of IoT-related attacks will continue to increase as the number of IoT devices in use grows. This is due, in part, to the fact that many IoT devices have weak security measures and are easy to hack. In addition, the proliferation of IoT devices has created a large attack surface for hackers to exploit. There have been several high-profile IoT attacks in recent years, including the Mirai botnet attack in 2016, which used a network of compromised IoT devices to launch a massive distributed denial of service (DDoS) attack, and the 2017 WannaCry ransomware attack, which affected numerous healthcare organizations and disrupted critical infrastructure around the world. To mitigate the risk of IoT attacks, it is important to prioritize security in the design and development of the IoT products. In this dissertation, we take a deeper a look into the security and privacy mechanisms of IoT devices, taking into consideration their limited compute, energy, memory and network connectivity. These constraints govern the design of systems and protocols used to achieve secure communication in IoT environments. First, we study the problem of periodically exchanging secret encryption and authentication keys in order to provide confidential and authenticated communication of sensitive IoT data. In particular, we identify the attack surface of IoT devices and emphasize an important security property called Perfect Forward Secrecy (PFS). In case of device/long term key compromise, PFS guarantees that previous confidential communication hold their security. PFS is typically offered by the use of public key cryptography (Diffie-Hellman Key exchange). We first demonstrate that the use of public key cryptography on IoT devices is computationally expensive. We propose a computationally lightweight cryptographic protocol that allows IoT devices to securely refresh their secret keys using the alternative private key cryptography, which is computationally lightweight. We then evaluate our novel protocol on IoT devices, and it shows significant reductions in computational latency, memory, energy consumption and network latency. Finally, we provide formal proofs of security of the propose protocol. Second, we investigate the problem of frequently authenticating messages sent by computationally weak IoT devices in order to protect IoT messages in transit from illegitimate manipulations. We first test the performance of different mechanisms used for authenticating messages in IoT devices and show that they are inefficient due to high computational latency and energy consumption. Specifically, we show that the standard solution from cryptography, digital signatures, is infeasible to be used on IoT devices. Thus, we introduce an online/offline digital signature scheme that allows IoT devices to efficiently authenticate messages using lightweight cryptographic operations. We build two constructions from two cryptographic families, namely RSA and Diffie-Hellman assumptions. We also use our signature scheme to demonstrate a time based one time password (TOTP) application and construct two TOTP systems. We also provide formal proofs of security of our proposed signature scheme. We finally evaluate our TOTP systems on IoT devices and they significantly reduce computational latency and energy consumption.

Finally, we study the problem of high scale and privacy preserving key management in connected vehicles (CVs). CVs is a significant branch of the emerging IoT technology and is expected to bring major benefits to the transportation system. We first study the privacy requirements of vehicular public key infrastructure (VPKIs) systems to protect CVs from tracking by other vehicles or the authorities. We formally define the two privacy properties required in VPKIs, namely anonymity and unlinkability. Unlinkability requires CVs to frequently renew their digital certificates from authorities, used to authenticate their messages. We then identify a significant problem that prevents other VPKIs from providing strong privacy guarantees to vehicles. This problem stems from the nature of CV networks as they are highly mobile and intermittent; this prevents vehicles from having long time contact with the infrastructure, which is the gateway for vehicles to renew their certificates. Thus, this results in high failures when renewing certificates which has a considerable impact on the privacy of the vehicles. We introduce a low latency, scalable and privacy preserving VPKI that allows highly mobile vehicles to quickly renew certificates by relying on the edge infrastructure. This also allows us to offer low bandwidth revocation lists for improved security of the vehicles against revoked malicious vehicles. We build an open-source testbed of CV on-board units (OBUs) and infrastructure road-side units (RSUs) that can be used for general vehicle to vehicle and vehicle to infrastructure applications. We also evaluate our VPKI on this testbed on in-city roadways and on highways. Our results show that our VPKI offers high success ratio of renewing certificates, which protects vehicles against long-term tracking. Our results also shows significant reductions in revocation list bandwidth requirements, making it a suitable choice for high scale systems.

For future work, I plan to investigate the possibility of offering a key exchange protocol that offers the strong security guarantee PFS while fully relying on symmetric key cryptography which requires low computation. I additionally plan to generalize our VPKI system to the whole IoT ecosystem in order to offer privacy preservation and efficient revocation for the whole IoT ecosystem. Finally, I plan to utilize our connected vehicle testbed to perform empirical experiments to identify security issues specific to the connected vehicle environment and propose corresponding solutions.

Bibliography

[1] RFC 8446.

[2] Cyber hack triggers mass fiat chrysler car recall — financial times. https://www.ft.com/content/2bafe3e0-321f-11e5-8873-775ba7c2ea3d.

[3] Intel® software guard extensions — intel® software. https://software.intel.com/en-us/sgx. (Accessed on 11/27/2019).

[4] Ieee standard for information technology-telecommunications and information exchange between systems-local and metropolitan area networks-specific requirements-part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications: Amendment 6: Medium access control (mac) security enhancements. *IEEE Std 802.11i-2004*, pages 1–190, 2004.

[5] M. Abdalla et al. Dhaes: An encryption scheme based on the diffie-hellman problem. Cryptology ePrint Archive, Report 1999/007, 1999.

[6] Ahmed Abdo, Sakib Md Bin Malek, Zhiyun Qian, Qi Zhu, Matthew Barth, and Nael Abu-Ghazaleh. Application level attacks on connected vehicle protocols. In *22nd International Symposium on Research in Attacks, Intrusions and Defenses ({RAID} 2019)*, pages 459–471, 2019.

[7] Ed Adams, Scott Andrews, Mona Asudegi, Seyithan Ayhan, Justin Fisher, Ben Fuja, Dennis Fleming, Patrick Chuang, John Collins, Larry Frank, et al. Development of dsrc device and communication system performance measures recommendations for dsrc obe performance and security requirements. Technical report, United States Department of Transportation Intelligent Transportation Systems Joint Program Office, 2016.

[8] D. Airehrour, J. Gutierrez, and S.K. Ray. Secure routing for internet of things: A survey. In *Journal of Network and Computer Applications*. Elsevier, 2016.

[9] A. Alabrah and M. Bassiouni. Efficient user and broadcast authentication scheme for wsns. In *Wireless Communications and Networking Conference (WCNC)*. IEEE, 2014.

[10] Nikolaos Alexiou, Marcello Laganà, Stylianos Gisdakis, Mohammad Khodaei, and Panagiotis Papadimitratos. Vespa: Vehicular security and privacy-preserving architecture. In *Proceedings of the 2nd ACM workshop on Hot topics on wireless network security and privacy*, pages 19–24, 2013.

[11] Fatemah Alharbi, Arwa Alrawais, Abdulrahman Bin Rabiah, Silas Richelson, and Nael B Abu-Ghazaleh. Csprop: Ciphertext and signature propagation low-overhead public-key cryptosystem for iot environments. In *USENIX Security Symposium*, pages 609–626, 2021.

[12] Wi-Fi Alliance. Wpa3 specification version 1.0, 2019.

[13] Anas Alsoliman, Marco Levorato, and A Chen. Vision-based two-factor authentication & localization scheme for autonomous vehicles. In *Third International Workshop on Automotive and Autonomous Vehicle Security (AutoSec) 2021 (part of NDSS)*, 2021.

[14] Anas Alsoliman, Abdulrahman Bin Rabiah, and Marco Levorato. Privacy-preserving authentication framework for uas traffic management systems. In *2020 4th Cyber Security in Networking Conference (CSNet)*, pages 1–8. IEEE, 2020.

[15] Muhammad Naveed Aman, Kee Chaing Chua, and Biplab Sikdar. Secure data provenance for the internet of things. In *Proceedings of the 3rd International Workshop on IoT Privacy, Trust, and Security*, IoTPTS '17, 2017.

[16] Amazon. Display people visitor counter, wireless, non directional.

[17] Ross Anderson, Charalampos Manifavas, and Chris Sutherland. Netcard—a practical electronic-cash system. In *International Workshop on Security Protocols*, pages 49–57. Springer, 1996.

[18] Apple. Your heart rate. what it means, and where on apple watch you'll find it., Mar 2020.

[19] M. Arslan, I. Singh, S. Singh, H.V. Madhyastha, K. Sundaresan, and S.V. Krishnamurthy. Computing while charging: Building a distributed computing infrastructure using smartphones. In *CoNEXT*. ACM, 2012.

[20] Nadarajah Asokan, Phil Janson, Michael Steiner, and Michael Waidner. State of the art in electronic payment systems. In *Advances in Computers*, volume 53, pages 425–449. Elsevier, 2000.

[21] Gildas Avoine, Sébastien Canard, and Loïc Ferreira. Iot-friendly ake: forward secrecy and session resumption meet symmetric-key cryptography. In *European Symposium on Research in Computer Security*, pages 463–483. Springer, 2019.

[22] Gildas Avoine, Sébastien Canard, and Loïc Ferreira. Symmetric-key authenticated key exchange (sake) with perfect forward secrecy. In Stanislaw Jarecki, editor, *Topics in Cryptology – CT-RSA 2020*, pages 199–224, Cham, 2020. Springer International Publishing.

[23] Andrew Banks, Ed Briggs, Ken Borgendale, and Rahul Gupta. Mqtt version 5.0. *OASIS Standard*, 7, 2019.

[24] E. B Barker, W.C. Barker, W.E. Burr, W.T. Polk, and M.E. Smid. Sp 800-57. recommendation for key management, part 1: General (revision 4). *Tech. Rep.*, 2016.

[25] Matthew J Barth, Guoyuan Wu, and Kanok Boriboonsomsin. Intelligent transportation systems and greenhouse gas reductions. *Current Sustainable/Renewable Energy Reports*, 2(3):90–97, 2015.

[26] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Keying hash functions for message authentication. In *Crypto*, volume 96, pages 1–15. Springer, 1996.

[27] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *CCS '93, Proceedings of the 1st ACM Conference on Computer and Communications Security, Fairfax, Virginia, USA, November 3-5, 1993*, pages 62–73, 1993.

[28] Mihir Bellare, Phillip Rogaway, and David Wagner. The eax mode of operation. In Bimal Roy and Willi Meier, editors, *Fast Software Encryption*, pages 389–407, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[29] B. Bezawada, Xiaojiang Liang, A. Liu, and Rui Li. A template approach to group key establishment in dynamic ad-hoc groups. In *ICNP*. IEEE, 2016.

[30] Abdulrahman Bin Rabiah, KK Ramakrishnan, Elizabeth Liri, and Koushik Kar. A lightweight authentication and key exchange protocol for iot. In *Workshop on Decentralized IoT Security and Standards (DISS)*, February 2018.

[31] Abdulrahman Bin Rabiah, KK Ramakrishnan, Silas Richelson, Ahmad Bin Rabiah, Elizabeth Liri, and Koushik Kar. Haiku: Efficient authenticated key agreement with strong security guarantees for iot. In *Proceedings of the 22nd International Conference on Distributed Computing and Networking*, pages 196–205, 2021.

[32] Norbert Bißmeyer, Sebastian Mauthofer, Jonathan Petit, Mirko Lange, Martin Moser, Daniel Estor, Michel Sall, Michael Feiri, Rim Moalla, Marcello Lagana, et al. Preparing secure vehicle-to-x communication systems. 2014.

[33] Norbert Bißmeyer, Jonathan Petit, and Kpatcha M Bayarou. Copra: Conditional pseudonym resolution algorithm in vanets. In *2013 10th annual conference on wireless on-demand network systems and services (WONS)*, pages 9–16. IEEE, 2013.

[34] Felipe Boeira, Marinho P Barcellos, Edison P de Freitas, Alexey Vinel, and Mikael Asplund. Effects of colluding sybil nodes in message falsification attacks for vehicular platooning. In *2017 IEEE Vehicular Networking Conference (VNC)*, pages 53–60. IEEE, 2017.

[35] Felipe Boeira, Marinho P Barcellos, Edison P de Freitas, Alexey Vinel, and Mikael Asplund. On the impact of sybil attacks in cooperative driving scenarios. In *2017 IFIP networking conference (IFIP networking) and workshops*, pages 1–2. IEEE, 2017.

[36] D. Boneh. Twenty years of attacks on the rsa cryptosystem. *Notices of the AMS*, 1999.

[37] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the weil pairing. *J. Cryptology*, 2004.

[38] Dan Boneh. The decision diffie-hellman problem. In Joe P. Buhler, editor, *Algorithmic Number Theory*, pages 48–63, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[39] Jurjen N. Bos and David Chaum. Provably unforgeable signatures. In *CRYPTO*. Springer, 1992.

[40] B. Brecht, D. Therriault, A. Weimerskirch, W. Whyte, V. Kumar, T. Hehn, and R. Goudy. A security credential management system for v2x communications. *IEEE Transactions on Intelligent Transportation Systems*, 19(12):3850–3871, 2018.

[41] J. Buchmann, E. Dahmen, S. Ereth, Andreas Hülsing, and Markus Rückert. On the security of the winternitz one-time signature scheme. In *International Conference on Cryptology in Africa*. Springer, 2011.

[42] Lander Casado and Philippas Tsigas. Contikisec: A secure network layer for wireless sensor networks under the contiki operating system. *Identity and Privacy in the Internet Age*, pages 133–147, 2009.

[43] Chien-Ming Chen, Bing Xiang, Tsu-Yang Wu, and King-Hang Wang. An anonymous mutual authenticated key agreement scheme for wearable sensors in wireless body area networks. *Applied Sciences*, 8(7):1074, 2018.

[44] Alexandra Chiang. V2x promises a new era of smart transportation. `https://www.advantech.com/en-us/resources/faq/v2x-promises-a-new-era-of-smart-transportation`, 2022.

[45] Alessandro Chiesa, Matthew Green, Jingcheng Liu, Peihan Miao, Ian Miers, and Pratyush Mishra. Decentralized anonymous micropayments. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 609–642. Springer, 2017.

[46] European Commission. Certificate Policy for Deployment and Operation of European Cooperative Intelligent Transport Systems (C-ITS) Release 1.1. `https://transport.ec.europa.eu/system/files/2018-05/c-its_certificate_policy-v1.1.pdf`, June 2018.

[47] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk. Internet x.509 public key infrastructure certificate and certificate revocation list (crl) profile. RFC 5280, May 2008.

[48] Adesto Technologies Corporation. Adesto, ibm and nxp demonstrate robust end-to-end security for industrial iot, Apr 2019.

[49] B. Cox, J.D. Tygar, and M. Sirbu. Netbill security and transaction protocol. In *WOEC*. USENIX, 1995.

[50] A. Czeskis, M. Dietz, Tadayoshi Kohno, Dan Wallach, and Dirk Balfanz. Strengthening user authentication through opportunistic cryptographic identity assertions. In *CCS*. ACM, 2012.

[51] E. Dauterman, H. Corrigan-Gibbs, D. Mazières, D. Boneh, and D. Rizzo. True2f: Backdoor-resistant authentication tokens. In *S&P*. IEEE, 2019.

[52] National Control Devices. Wireless soil moisture sensor long range iot transmitter.

[53] Whitfield Diffie, Paul C. Van Oorschot, and Michael J. Wiener. Authentication and authenticated key exchanges. *Designs, Codes and Cryptography*, 2(2):107–125, Jun 1992.

[54] L. Dignan. Dyn confirms mirai botnet involved in distributed denial of service attack. *ZDNet*.

[55] John R Douceur. The sybil attack. In *International workshop on peer-to-peer systems*, pages 251–260. Springer, 2002.

[56] Mohammad Sadeq Dousti and Rasool Jalili. Forsakes: a forward-secure authenticated key exchange protocol based on symmetric key-evolving schemes. *Advances in Mathematics of Communications*, 9(4):471–514, 2015.

[57] D. Eastlake and T. Hansen. Us secure hash algorithms (sha and hmac-sha). RFC 4634, RFC Editor, July 2006.

[58] Mohammad Ehdaie, Nikos Alexiou, Mahmoud Ahmadian, Mohammad Reza Aref, and Panos Papadimitratos. 2d hash chain robust random key distribution scheme. *Information Processing Letters*, 116(5):367–372, 2016.

[59] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *TIT*. IEEE, 1985.

[60] TCITS ETSI. Intelligent transport systems (its); vehicular communications; basic set of applications; definitions. *Tech. Rep. ETSI TR 102 6382009*, 2009.

[61] TS ETSI. 102 940 v2.1.1—intelligent transport systems (its); security; its communications security architecture and security management. Technical report, Technical Report, 2021.

[62] TS ETSI. 102 941 v1.4.1—intelligent transport systems (its); security; trust and privacy management. *Standard, TC C-ITS*, 2021.

[63] S. Even, O. Goldreich, and S. Micali. On-line/off-line digital signatures. *Journal of Cryptology*, 1996.

[64] Traffic Safety Facts. Motor vehicle crashes: Overview. *Traffic safety facts: research note*, 2018.

[65] Amos Fiat and Moni Naor. Broadcast encryption. In *Annual International Cryptology Conference*, pages 480–491. Springer, 1993.

[66] Fitbit. Fitbit ionic.

[67] Y. Gao, P. Zeng, K. Choo, and F. Song. An improved online/offline identity-based signature scheme for wsns. In *IJ Network Security*. Springer, 2016.

[68] Oscar Garcia-Morchon, Sye Loong Keoh, Sandeep Kumar, Pedro Moreno-Sanchez, Francisco Vidal-Meca, and Jan Henrik Ziegeldorf. Securing the ip-based internet of things with hip and dtls. In *Proceedings of the Sixth ACM Conference on Security and Privacy in Wireless and Mobile Networks*, WiSec '13, 2013.

[69] S. Garriss, R. Cáceres, S. Berger, R. Sailer, L. van Doorn, and X. Zhang. Trustworthy and personalized computing on public kiosks. In *MobiSys*. ACM, 2008.

[70] Matthias Gerlach, Andreas Festag, Tim Leinmuller, Gabriele Goldacker, and Charles Harsch. Security architecture for vehicular communication. In *Workshop on intelligent transportation*, 2007.

[71] M. Girault, G. Poupard, and J. Stern. On the fly authentication and signature schemes based on groups of unknown order. In *Journal of Cryptology*. Springer, 2006.

[72] Stylianos Gisdakis, Marcello Laganà, Thanassis Giannetsos, and Panos Papadimitratos. Serosa: Service oriented security architecture for vehicular communications. In *2013 IEEE Vehicular Networking Conference*, pages 111–118. IEEE, 2013.

[73] S. Goldwasser and S. Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In *STOC*. ACM, 1982.

[74] S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 1988.

[75] Vinodh Gopal, James D Guilford, Gilbert M Wolrich, and Daniel F Cutter. Look-ahead hash chain matching for data compression, February 28 2017. US Patent 9,584,155.

[76] Christoph G Günther. An identity-based key-exchange protocol. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 29–37. Springer, 1989.

[77] Ankur Gupta, Meenakshi Tripathi, Tabish Jamil Shaikh, and Aakar Sharma. A lightweight anonymous user authentication and key establishment scheme for wearable devices. *Computer Networks*, 149:29–42, 2019.

[78] Jason J Haas, Yih-Chun Hu, and Kenneth P Laberteaux. Efficient certificate revocation list organization and distribution. *IEEE Journal on Selected Areas in Communications*, 29(3):595–604, 2011.

[79] Martin Haenggi. Outage, local throughput, and capacity of random wireless networks. *IEEE Transactions on Wireless Communications*, 8(8):4350–4359, 2009.

[80] N. Haller. The s/key one-time password system. RFC 1760, RFC Editor, February 1995.

[81] Dan Harkins and Dave Carrel. The internet key exchange (ike). RFC 2409, November 1998.

[82] Ralf Hauser, Michael Steiner, and Michael Waidner. *Micro-payments based on iKP*. IBM TJ Watson Research Center, 1996.

[83] HedeDanmark. Digital water level sensor.

[84] HedeDanmark. People counting sensor.

[85] Tobias Heer, Oscar Garcia-Morchon, René Hummen, Sye Loong Keoh, Sandeep S Kumar, and Klaus Wehrle. Security challenges in the ip-based internet of things. *Wireless Personal Communications*, 61(3):527–542, 2011.

[86] Stephen Hemminger et al. Network emulation with netem. In *Linux conf au*, pages 18–23, 2005.

[87] C. Herley and P. Van Oorschot. A research agenda acknowledging the persistence of passwords. In *S&P*. IEEE, 2012.

[88] Jose L Hernandez-Ramos, Marcin Piotr Pawlowski, Antonio J Jara, Antonio F Skarmeta, and Latif Ladid. Toward a lightweight authentication and authorization framework for smart objects. *IEEE Journal on Selected Areas in Communications*, 33(4):690–702, 2015.

[89] H. R. Hussen, G. A. Tizazu, Miao Ting, Taekkyeun Lee, Youngjun Choi, and Ki-Hyung Kim. Sakes: Secure authentication and key establishment scheme for m2m communication in the ip-based wireless sensor network (6l0wpan). In *Proceedings of Fifth International Conference on Ubiquitous and Future Networks (ICUFN)*, pages 246–251, 2013.

[90] Twilio Inc. Twilio partners with microsoft azure iot, adds identity and authentication capabilities to its iot connectivity offering, May 2019.

[91] Markus Jakobsson, Jean-Pierre Hubaux, and Levente Buttyán. A micro-payment scheme encouraging collaboration in multi-hop cellular networks. In *International Conference on Financial Cryptography*, pages 15–33. Springer, 2003.

[92] S. Jana et al. On the effectiveness of secret key extraction from wireless signal strength in real environments. In *MobiCom*. ACM, 2009.

[93] Jakob Jonsson. On the security of ctr+ cbc-mac. In *International Workshop on Selected Areas in Cryptography*, pages 76–93. Springer, 2002.

[94] M. Joye. An efficient on-line/off-line signature scheme without random oracles. In *Conference on Cryptology And Network Security*. Springer, 2008.

[95] A. Kalamandeen et al. Ensemble: cooperative proximity-based authentication. In *MobiSys*. ACM, 2010.

[96] Mohammad Khodaei, Hongyu Jin, and Panagiotis Papadimitratos. Secmace: Scalable and robust identity and credential management infrastructure in vehicular communication systems. *IEEE Transactions on Intelligent Transportation Systems*, 19(5):1430–1444, 2018.

[97] Mohammad Khodaei, Hamid Noroozi, and Panos Papadimitratos. Scaling pseudonymous authentication for large mobile systems. In *Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks*, pages 174–184, 2019.

[98] Mohammad Khodaei and Panos Papadimitratos. Evaluating on-demand pseudonym acquisition policies in vehicular communication systems. In *Proceedings of the First International Workshop on Internet of Vehicles and Vehicles of Internet*, pages 7–12, 2016.

[99] T. Kivinen. Minimal internet key exchange version 2 (ikev2) initiator implementation. RFC 7815, March 2016.

[100] D. Kogan, N. Manohar, and D. Boneh. T/key: Second-factor authentication from secure hash chains. In *CCS*. ACM, 2017.

[101] John Kohl and Clifford Neuman. The kerberos network authentication service (v5). Technical report, 1993.

[102] T. Krovetz and P. Rogaway. The ocb authenticated-encryption algorithm. RFC 7253, RFC Editor, May 2014.

[103] Pardeep Kumar, An Braeken, Andrei Gurtov, Jari Iinatti, and Phuong Hoai Ha. Anonymous secure framework in connected smart home environments. *IEEE Transactions on Information Forensics and Security*, 12(4):968–979, 2017.

[104] K. Kurosawa and K. Schmidt-Samoa. New online/offline signature schemes without random oracles. In *PKC*. Springer, 2006.

[105] Kenneth P Laberteaux, Jason J Haas, and Yih-Chun Hu. Security certificate revocation list distribution for vanet. In *Proceedings of the fifth ACM international workshop on VehiculAr Inter-NETworking*, pages 88–89, 2008.

[106] J. Lai, Y. Mu, and F. Guo. Efficient identity-based online/offline encryption and signcryption with short ciphertext. In *International Journal of Information Security*. Springer, 2017.

[107] L. Lamport. Password authentication with insecure communication. *Communications of the ACM*, 1981.

[108] A. Langley, M. Hamburg, and S. Turner. Elliptic curves for security. RFC 7748, January 2016.

[109] K. Lee et al. Voltkey: Continuous secret key generation based on power line noise for zero-involvement pairing and authentication. *Interact. Mob. Wearable Ubiquitous Technol.*, 2019.

[110] Legion of the Bouncy Castle. Bouncy castle crypto apis.

[111] LevelSense. Level sense sump pump failure alarm with rechargeable battery back-up.

[112] Elizabeth Liri, Prateek Kumar Singh, Abdulrahman BIN Rabiah, Koushik Kar, Kiran Makhijani, and KK Ramakrishnan. Robustness of iot application protocols to network impairments. In *2018 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)*, pages 97–103. IEEE, 2018.

[113] RSA Security LLC. Identity and access management: Rsa securid suite.

[114] Nai-Wei Lo and Hsiao-Chien Tsai. Illusion attack on vanet applications-a message plausibility problem. In *2007 IEEE Globecom Workshops*, pages 1–8. IEEE, 2007.

[115] SpyMonde Ltd. Sm mini gps tracker.

[116] David M., M. Bellare, F. Hoornaert, D. Naccache, and O. Ranen. Hotp: An hmac-based one-time password algorithm. Technical report, 2005.

[117] Zhendong Ma, Frank Kargl, and Michael Weber. Pseudonym-on-demand: a new pseudonym refill strategy for vehicular communications. In *2008 IEEE 68th Vehicular Technology Conference*, pages 1–5. IEEE, 2008.

[118] Prerna Mahajan and Abhishek Sachdeva. A study of encryption algorithms aes, des and rsa for security. *Global Journal of Computer Science and Technology*, 2013.

[119] M.S. Manasse. The millicent protocols for electronic commerce. In *USENIX Workshop on Electronic Commerce*, 1995.

[120] M. Mannan and P.C. Van Oorschot. Using a personal device to strengthen password authentication from an untrusted computer. In *International Conference on Financial Cryptography and Data Security*. Springer, 2007.

[121] Mohamed M. Mansour, Fatty M. Salem, and Elsayed M. Saad. A secure mutual authentication scheme with perfect forward-secrecy for wireless sensor networks. In Aboul Ella Hassanien, Mohamed F. Tolba, Khaled Shaalan, and Ahmad Taher Azar, editors, *Proceedings of the International Conference on Advanced Intelligent Systems and Informatics 2018*, pages 446–456, Cham, 2019. Springer International Publishing.

[122] MarCELL. Marcell® 4g multisensor.

[123] Digital Matter. Dart2 — vehicle tracking device.

[124] U. Maurer and J. Sjödin. A fast and key-efficient reduction of chosen-ciphertext to known-plaintext security. In *Eurocrypt*. Springer, 2007.

[125] David McGrew and John Viega. The galois/counter mode of operation (gcm). *Submission to NIST Modes of Operation Process*, 20, 2004.

[126] Medtronic. Guardian™ connect continuous glucose monitoring (cgm) system.

[127] R. Merkle. Secrecy, authentication, and public key systems. *Ph. D. Thesis, Stanford University*, 1979.

[128] S. Micali and R. Rivest. Micropayments revisited. In *Cryptographers' Track at the RSA Conference*. Springer, 2002.

[129] M. Miettinen et al. Context-based zero-interaction pairing and key evolution for advanced personal devices. In *CCS*. ACM, 2014.

[130] David M'Raihi, Salah Machani, Mingliang Pei, and Johan Rydell. Totp: Time-based one-time password algorithm. Technical report, 2011.

[131] Wassim G Najm, Jonathan Koopmann, John D Smith, John Brewer, et al. Frequency of target crashes for intellidrive safety systems. Technical report, United States. National Highway Traffic Safety Administration, 2010.

[132] M. Naor and O. Reingold. Synthesizers and their application to the parallel construction of pseudo-random functions. In *JCSS*. Elsevier, 1999.

[133] Q.S. Nguyen. Multi-dimensional hash chains and application to micropayment schemes. In *WCC*. Springer, 2005.

[134] U.S. Department of Transportation (US DOT). Security credential management system (scms). `https://its.dot.gov/resources/scms.htm`, 2022.

[135] Onset. Hobonet soil moisture 10hs sensor.

[136] Panagiotis Papadimitratos, Levente Buttyan, Tamás Holczer, Elmar Schoch, Julien Freudiger, Maxim Raya, Zhendong Ma, Frank Kargl, Antonio Kung, and Jean-Pierre Hubaux. Secure vehicular communication systems: design and architecture. *IEEE Communications Magazine*, 46(11):100–109, 2008.

[137] Panagiotis Papadimitratos, Levente Buttyan, Jean-Pierre Hubaux, Frank Kargl, Antonio Kung, and Maxim Raya. Architecture for secure and private vehicular communications. In *2007 7th International Conference on ITS Telecommunications*, pages 1–6. IEEE, 2007.

[138] Panagiotis Papadimitratos, Ghita Mezzour, and Jean-Pierre Hubaux. Certificate revocation list distribution in vehicular communication systems. In *Proceedings of the fifth ACM international workshop on VehiculAr Inter-NETworking*, pages 86–87, 2008.

[139] Panos Papadimitratos. “on the road”-reflections on the security of vehicular communication systems. In *2008 IEEE International Conference on Vehicular Electronics and Safety*, pages 359–363. IEEE, 2008.

[140] Bhrat Patel and Jon Crowcroft. Ticket based service access for the mobile user. In *Proceedings of the 3rd annual ACM/IEEE international conference on Mobile computing and networking*, pages 223–233, 1997.

[141] Torben P Pedersen. Electronic payments of small amounts. In *International Workshop on Security Protocols*, pages 59–68. Springer, 1996.

[142] A. Perrig, R. Canetti, J.D. Tygar, and D. Song. The tesla broadcast authentication protocol. In *Rsa Cryptobytes*. Citeseer, 2002.

[143] Frank Perry, Kelli Raboy, Ed Leslie, Zhitong Huang, Drew Van Duren, et al. Dedicated short-range communications roadside unit specifications. Technical report, United States. Dept. of Transportation, 2017.

[144] Robert Pettersen, Håvard D Johansen, and Dag Johansen. Secure edge computing with arm trustzone. In *IoTBDS*, pages 102–109, 2017.

[145] Pawani Porambage, Corinna Schmitt, Pardeep Kumar, Andrei Gurtov, and Mika Ylianttila. Pauthkey: A pervasive authentication protocol and key establishment scheme for wireless sensor networks in distributed iot applications. *International Journal of Distributed Sensor Networks*, 10(7):357430, 2014.

[146] Nachiketh R. Potlapally, Srivaths Ravi, Anand Raghunathan, and Niraj K. Jha. Analyzing the energy consumption of security protocols. In *Proceedings of International Symposium on Low Power Electronics and Design*, ISLPED ’03, 2003.

[147] Y. Qiu and M. Ma. A mutual authentication and key establishment scheme for m2m communication in 6lowpan networks. *IEEE Transactions on Industrial Informatics*, 12(6):2074–2085, 2016.

[148] Abdulrahman Bin Rabiah, Yugarshi Shashwat, Fatemah Alharbi, Silas Richelson, and Nael Abu-Ghazaleh. Mss: Lightweight network authentication for resource constrained devices via mergeable stateful signatures. In *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*, pages 282–292. IEEE, 2021.

[149] Khaled Rabieh, Mohamed MEA Mahmoud, Terry N Guo, and Mohamed Younis. Cross-layer scheme for detecting large-scale colluding sybil attack in vanets. In *2015 IEEE International Conference on Communications (ICC)*, pages 7298–7303. IEEE, 2015.

[150] Francesco Raviglione, Marco Malinverno, and Claudio Casetti. Characterization and performance evaluation of ieee 802.11 p nics. In *Proceedings of the 1st ACM MobiHoc Workshop on Technologies, mOdels, and Protocols for Cooperative Connected Cars*, pages 13–18, 2019.

[151] Francesco Raviglione, Marco Malinverno, Stefano Feraco, Giuseppe Avino, Claudio Casetti, Carla Fabiana Chiasserini, Nicola Amati, and Joerg Widmer. Experimental assessment of ieee 802.11-based v2i communications. In *Proceedings of the 18th ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, & Ubiquitous Networks*, pages 33–40, 2021.

[152] Maxim Raya, Panagiotis Papadimitratos, Imad Aad, Daniel Jungels, and Jean-Pierre Hubaux. Eviction of misbehaving and faulty nodes in vehicular networks. *IEEE Journal on Selected Areas in Communications*, 25(8):1557–1568, 2007.

[153] Shahid Raza, Ludwig Seitz, Denis Sitenkov, and Göran Selander. S3k: Scalable security with symmetric keys — dtls key establishment for the internet of things. *IEEE Transactions on Automation Science and Engineering*, 13(3):1270–1280, 2016.

[154] E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3. Internet-Draft draft-ietf-tls-tls13-28.txt, IETF Secretariat, March 2018.

[155] E Rescorla, H Tschofenig, and N Modadugu. Rfc 9147: The datagram transport layer security (dtls) protocol version 1.3, 2022.

[156] R. Rivest and A. Shamir. Payword and micromint: Two simple micropayment schemes. In *International workshop on security protocols*. Springer, 1996.

[157] R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 1978.

[158] M. Rostami et al. Heart-to-heart (h2h) authentication for implanted medical devices. In *CCS*. ACM, 2013.

[159] Andy Rupp, Gesine Hinterwälder, Foteini Baldimtsi, and Christof Paar. P4r: Privacy-preserving pre-payments with refunds for transportation systems. In *International Conference on Financial Cryptography and Data Security*, pages 205–212. Springer, 2013.

[160] Samsung. Monitor your heart rate on your samsung smart watch.

[161] S. Santesson, M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams. X.509 internet public key infrastructure online certificate status protocol - ocsp. RFC 6960, June 2013.

[162] Florian Schaub, Frank Kargl, Zhendong Ma, and Michael Weber. V-tokens for conditional pseudonymity in vanets. In *2010 IEEE Wireless Communication and Networking Conference*, pages 1–6. IEEE, 2010.

[163] C.P. Schnorr. Efficient identification and signatures for smart cards. In *CRYPTO*. Springer, 1989.

[164] Duo Security. Guide to two-factor authentication. `https://guide.duo.com/prompt`.

[165] SensorPush. The easiest way to monitor the environmental conditions affecting the things you care about.

[166] A. Shamir and Y. Tauman. Improved online/offline signature schemes. In *CRYPTO*. Springer, 2001.

[167] G. Sharma, S. Bala, and A.K. Verma. Pf-ibs: Pairing-free identity based digital signature algorithm for wireless sensor networks. In *Wireless Personal Communications*. Springer, 2017.

[168] Alan T. Sherman and David A. McGrew. Key establishment in large dynamic groups using one-way function trees. *IEEE Trans. Softw. Eng.*, 29(5):444–458, May 2003.

[169] M. Shirvanian, S. Jarecki, N. Saxena, and N. Nathan. Two-factor authentication resilient to server compromise using mix-bandwidth devices. In *NDSS*, 2014.

[170] Mengxia Shuai, Bin Liu, Nenghai Yu, and Ling Xiong. Lightweight and secure three-factor authentication scheme for remote patient monitoring using on-body wireless networks. *Security and Communication Networks*, 2019, 2019.

[171] D. Simon, B. Aboba, and R. Hurst. The eap-tls authentication protocol. RFC 5216, March 2008.

[172] S. Singh, P.K. Sharma, S.Y. Moon, and J.H. Park. Advanced lightweight encryption algorithms for iot devices: survey, challenges and solutions. In *JAIHC*. Springer, 2017.

[173] Trevor Smith, Luke Dickinson, and Kent Seamons. Let's revoke: Scalable global certificate revocation. In *27th Annual Network and Distributed System Security Symposium, NDSS*, 2020.

[174] S. Srinivas, D. Balfanz, E. Tiffany, A. Czeskis, and F. Alliance. Universal 2nd factor (u2f) overview. *FIDO Alliance Proposed Standard*, pages 1–5, 2015.

[175] Koutarou Suzuki, Kunio Kobayashi, and Hikaru Morita. Efficient sealed-bid auction using hash chain. In *International Conference on Information Security and Cryptology*, pages 183–191. Springer, 2000.

[176] Petr Švenda. Basic comparison of modes for authenticated-encryption (iapm, xcbc, ocb, ccm, eax, cwc, gcm, pcfb, cs), 2016.

[177] Irfan Syamsuddin, Tharam Dillon, Elizabeth Chang, and Song Han. A survey of rfid authentication protocols based on hash-chain method. In *2008 Third International Conference on Convergence and Hybrid Information Technology*, volume 2, pages 559–564. IEEE, 2008.

[178] Wireless Sensor Tags. External power sensor for humidity monitoring and logging.

[179] Andrea Tesei, Luca Di Mauro, Mariano Falcitelli, Sandro Noto, and Paolo Pagano. Iota-vpki: a dlt-based and resource efficient vehicular public key infrastructure. In *2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*, pages 1–6. IEEE, 2018.

[180] Thales. Global pki trends study. `https://www.thalesesecurity.com/2018/pki-trends-study`, 2018.

[181] H. Tschofenig and T. Fossati. Transport layer security (tls) / datagram transport layer security (dtls) profiles for the internet of things. RFC 7925, July 2016.

[182] UbiBot. Ubibot wifi humidity sensor — wireless humidity monitoring system.

[183] USDOT. Wydot update at the application design stage webinar q&a. `https://www.its.dot.gov/pilots/wydot_webinar_qa.htm`, 2022.

[184] B. Vaidya, J.H. Park, S.S. Yeo, and J.J. Rodrigues. Robust one-time password authentication scheme using smart card for home network environment. In *Computer Communications*. Elsevier, 2011.

[185] A. Wang, A. Mohaisen, and S. Chen. Xlf: A cross-layer framework to secure the internet of things (iot). In *ICDCS*. IEEE, 2019.

[186] WAVIoT. Em 1 electricity meter.

[187] D. Whiting, R. Housley, and N. Ferguson. Counter with cbc-mac (ccm). RFC 3610, September 2003.

[188] Zack Whittaker. Homeland security warns of brickerbot malware that destroys unsecured internet-connected devices. *ZDNet*, Apr 2017.

[189] Björn Wiedersheim, Michel Sall, and Guillaume Reinhard. Sevecom—security and privacy in car2car ad hoc networks. In *2009 9th International Conference on Intelligent Transport Systems Telecommunications,(ITST)*, pages 658–661. IEEE, 2009.

[190] P. Wouters, H. Tschofenig, J. Gilmore, S. Weiler, and T. Kivinen. Using raw public keys in transport layer security (tls) and datagram transport layer security (dtls). RFC 7250, June 2014.

[191] S. Xiao, W. Gong, and D. Towsley. Secure wireless communication with dynamic secrets. In *Proceedings of IEEE INFOCOM 2010*, pages 1–9, 2010.

[192] J. Xu, J. Qi, and X. Ye. Otp bidirectional authentication scheme based on mac address. In *International Conference on Computer and Communications (ICCC)*. IEEE, 2016.

[193] S. Xu, Y. Mu, and W. Susilo. Online/offline signatures and multisignatures for aodv and dsr routing security. In *ACISP*. Springer, 2006.

[194] Z. Xu, G. Dai, and D. Yang. An efficient online/offline signcryption scheme for manet. In *Conference on Advanced Information Networking and Applications Workshops*. IEEE, 2007.

[195] L. Yang et al. Secret from muscle: Enabling secure pairing with electromyography. In *SenSys*. ACM, 2016.

[196] Ning Ye, Yan Zhu, Ru-chuan Wang, Reza Malekian, and Lin Qiao-Min. An efficient authentication and access control scheme for perception layer of internet of things. *Applied Mathematics & Information Sciences*, 8(4):1617, 2014.

[197] P. Yu and S. R. Tate. An online/offline signature scheme based on the strong rsa assumption. In *Conference on Advanced Information Networking and Applications Workshops*. IEEE, 2007.

[198] Yubico. Yubikey strong two factor authentication. `https://www.yubico.com`.

[199] Sorin Zamfir, Titus Balan, I Iliescu, and F Sandu. A security analysis on standard iot protocols. In *2016 International Conference on Applied and Theoretical Electricity (ICATE)*, pages 1–6. IEEE, 2016.

[200] K. Zetter. Hacker spies hit security firm rsa. *Wired*.

[201] Jiansong Zhang, Zeyu Wang, Zhice Yang, and Qian Zhang. Proximity based iot device authentication. In *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*, pages 1–9. IEEE, May 2017.

[202] Shijie Zhang and Jong-Hyouk Lee. Double-spending with a sybil attack in the bitcoin decentralized network. *IEEE transactions on Industrial Informatics*, 15(10):5715–5722, 2019.

[203] Yang Zongkai, Lang Weimin, and Tan Yunmeng. A new fair micropayment system based on hash chain. In *IEEE International Conference on e-Technology, e-Commerce and e-Service, 2004. EEE'04. 2004*, pages 139–145. IEEE, 2004.