Privacy and Authentication in Emerging Network Applications

He Li

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Electrical Engineering

Jung-Min (Jerry) Park, Co-chair
Yaling Yang, Co-chair
Danfeng Yao
Haibo Zeng
Michael S. Hsiao

December 2, 2020
Blacksburg, Virginia

Keywords: privacy-preserving technologies, dynamic spectrum sharing, lightweight authentication, applied cryptography.

Privacy and Authentication in Emerging Network Applications

He Li

(ABSTRACT)

In this dissertation, we studied and addressed the privacy-preserving and authentication techniques for some network applications, where existing internet security solutions cannot address them straightforwardly due to different trust and attack models and possibly constrained resources. For example, in a centralized dynamic spectrum access (DSA) system, the spectrum resource licensees called incumbent users (IUs), have strong operational privacy requirements for the DSA service provider called spectrum access system (SAS), and hence SAS is required to perform spectrum computation without knowing IUs' operational information. This means SAS can at most be considered as a semi-trusted party which is honest but curious, and common anonymization and end-to-end encryption cannot address this issue, and dedicated solutions are required. Another example is that in an intra-vehicle Controller Area Network (CAN), the transmitter can only embed 64 bits of message and its authentication tag into on message frame, which makes it difficult to achieve message authentication in real-time with sufficient cryptographic strength. The focus of this dissertation is to fill the gap of existing solutions with stronger security notion and practicability. On the topic of privacy-preserving DSA systems, we *firstly* explored existing solutions and proposed a comparative study. We additionally proposed a new metric for evaluation and showed the advantages and disadvantages of existing solutions. We *secondly* studied the IU location privacy in 3.5GHz band ESC-based DSA system and proposed a novel scheme called PriDSA. PriDSA addresses malicious colluding SAS attack model through leveraging different and relatively lightweight cryptography primitive with novel design, granting stronger

security notion and improved efficiency as well. We *thirdly* studied the operational privacy of both IU and secondary users (SUs) in a general centralized SAS based DSA system and proposed a novel framework called PeDSS. Through our novel design that integrates differential privacy with secure multi-party computation protocol, PeDSS exhibits great communication and computation overhead compared to existing solutions.

On the topic of lightweight message authentication in resource-constrained networks, we *firstly* explored message authentication schemes with high cryptographic strength and low communication-overhead and proposed a novel scheme called CuMAC. CuMAC provides a flexible trade-off between authentication delay and cryptographic strength, through the embodiment of a novel concept that we refer to as accumulation of cryptographic strength. We *secondly* explored the possibility of achieving both high cryptographic strength and low authentication delay and proposed a variant of CuMAC called CuMAC/S. By employing the novel idea of message speculation, CuMAC/S achieves enables the accumulation of cryptographic strength while incurring minimal delay when the message speculation accuracy is high.

Privacy and Authentication in Emerging Network Applications

He Li

(GENERAL AUDIENCE ABSTRACT)

The privacy-preserving and message authentication issues of some network applications are distinctive from common internet security due to different attack models and possibly constrained resources, and these security and privacy concerns cannot be addressed by applying existing internet security solutions straightforwardly. For example, in a centralized dynamic spectrum access (DSA) system, the spectrum resource licensees called incumbent users (IUs), have strong operational privacy requirements for the DSA service provider called spectrum access system (SAS), and hence SAS is required to perform spectrum computation without knowing IUs' operational information. This means SAS can at most be considered as a semi-trusted party which is honest but curious, and common anonymization and end-to-end encryption cannot address this issue, and dedicated solutions are required. Another example is that in an intra-vehicle Controller Area Network (CAN), the transmitter can only embed 64 bits of message and its authentication tag into on message frame, which makes it difficult to achieve message authentication in real-time with sufficient cryptographic strength. We addressed the privacy issue of DSA systems by proposing novel schemes incorporating efficient cryptographic primitives and various privacy-preserving techniques, achieving a greatly higher efficiency or stronger privacy-preserving level. We addressed the lightweight authentication issue of resource-constrained networks by employing the novel concept of security accumulation and message speculation, achieving high cryptographic strength, low communication overhead, and probable low latency.

Dedication

To my parents Baomian Li and Xiuhua Zhang

Acknowledgments

I would firstly like to thank my supervisors, Dr. Yaling Yang, and Dr. Jung-Min Park, for the guidance through each stage of the process. Your feedback and knowledge helped me to better understand myself and pushed my work to a higher level. Your support and openness gave me the chance to explore various research topics and get another master's degree in mathematics. Your attitude towards research and scientific thinking has shown me what makes a great scholar. I treasure all the experiences a lot.

I would like to thank my dissertation committee members, Dr. Danfeng Yao, Dr. Haibo Zeng, and Dr. Michael Hsiao, for their valuable suggestions and comments on my work.

I would also like to give thanks to my colleagues Vireshwar Kumar and Yanzhi Dou, for the deep discussions and collaborations on research. I would like to give thanks to my colleague Jinshan Liu for discussions on research and mathematics. I would like to give thanks to Curtis Zeng, Gaurang Naik, Chang Lu, Noah Luther, and all the other colleagues for the collaborations and help.

In addition, I would like to thank my parents for their love, support, and empathy. You are always there for me. Finally, I would like to thank my friends, Ren Mao and Wentao Guan, who provided distractions to rest my mind outside of my research.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter, we present an introduction to the two domains we studied: incumbent user operational privacy in centralized dynamic spectrum access systems and lightweight message authentication in resource-constrained networks.

1.1 Preserving Operational Privacy of Incumbent Users in Database-Driven Dynamic Spectrum Access Systems

Dynamic spectrum access (DSA) technique has been widely accepted as a key technique to address the spectrum scarcity issue. DSA enhances spectrum efficiency by allowing unlicensed secondary users (SUs) to opportunistically utilize the spectrum that is not used by licensed incumbent users (IUs). In the U.S., 150 megahertz in the 3550-3700 MHz band (3.5 GHz Band) has already been approved for Citizens Broadband Radio Service (CBRS) by Federal Communications Commission (FCC) [15].

It is implied from industrial and governmental acts that the database-driven dynamic spectrum sharing paradigm is one of the most promising and practical designs for DSA. In this paradigm, each SU sends a spectrum access system (SAS) their spectrum access request that includes their location, transmission power, and antenna height. If the SU will not cause

intolerable interference to IUs, it will get a valid spectrum license in return.

User privacy is one of the critical concerns for a successful DSA system design. For national security reasons, operational information of government IUs is often classified data. For example, the IUs in the 3.5 GHz DSA band in the U.S. include military and fixed satellite service licensees [15]. These IUs' operational data is highly sensitive, and thus the IUs' operational security is essential. Meanwhile, civil users of SU devices also need the protection of their location privacy. According to a 2013 Pew Research project [58], 86% of users surveyed had taken actions to hide their identities to avoid information collection online. Yet the centralized spectrum management entity, SAS, who collects information related to the operational status from both IUs and SUs, is usually operated by commercial third parties [14] that are not necessarily trusted by either IUs or SUs. Thus, privacy concern has been hindering the development of DSA systems.

1.1.1 IU Location Privacy in 3.5 the GHz Band

The Federal Communications Commission (FCC) has prescribed the creation of a Citizens Broadband Radio Service (CBRS) in the 3.5 GHz band (3500 - 3700 MHz) to enable spectrum sharing between federal and commercial systems [15]. In the scenario of 3.5 GHz CBRS along with U.S coastal areas, Environmental Sensing Capability(ESC) systems are set up to detect the presence of federal incumbent users (IUs). ESC systems inform the presence of IUs to a Spectrum Access System (SAS), which coordinates CBRS devices' (CBSDs') access to the 3.5 GHz band. Meanwhile, SAS and ESC have to guarantee that CBSDs, essentially the Secondary Users (SUs), do not generate harmful interference to IUs.

Existing works that attempt to address the IU location privacy protection problem can be divided into two categories. The first category [5][75][76] adds noise or distortion on IU

location data before the data is sent to SAS. The second category [25][26][37] encrypts IU location data using homomorphic cryptosystem so that SAS can homomorphically perform spectrum allocation computation on ciphertext domain without needing to see the underlying plaintext location data. Both categories of works, however, assume that IU must actively participate in the spectrum allocation process. The first category relies on IUs to add noise to their data according to their privacy needs and true location. The second category depends on IUs to encrypt its true location data.

None of the existing work on IU privacy-preserving can handle the non-informing IU cases in the 3.5 GHz band, where IUs do not directly interact with the DSA system. In the 3.5 GHz paradigm, SAS obtains information related to IU presence from ESC. The inputs from ESC to the SAS are IU signal detection events, not IU location information as in existing works. Thus, distortion logics used in category one of existing work cannot be applied on ESC input and the homomorphic computation over IU location inputs in category two of existing work also are not applicable.

We hereby propose PriDSA to address this issue. We leverage the partial homomorphic feature of a proxy re-encryption scheme to preserve IU privacy and achieve the goal of dynamic spectrum sharing at the same time. We additionally blind IU inputs to prevent malicious colluding SAS from extracting information from any single ESC input; spectrum allocation is achieved by leverage a commitment system, where only the overall blinding of data can be removed.

1.1.2 Preserving both IU and SU Operational Privacy in General Centralized DSA System

Existing works regarding IU and SU privacy protection can be divided into three types. The first type [75] attempts to protect IU privacy from curious SUs while assuming the SAS is trustworthy. These schemes cannot protect IU operational security from curious SAS. The second type [25][26][37] protects both IU and SU privacy by formulating IU and SU privacy protection problem as a secure multi-party computation problem and solves the problem using partially homomorphic cryptographic primitives. However, these schemes rely on the aid of an **online** trusted third party (OTTP), which brings additional overhead on maintenance. Also, all users' privacy will be breached if this OTTP service is compromised. Furthermore, these schemes' average processing time per SU spectrum request is several seconds, which is not scalable when the arrival rate of spectrum requests is high. In [47] a solution that addresses computational overhead is proposed, but the issue of SU privacy is neglected. The final type [76] applies a differential privacy mechanism to protect IU and SU privacy. While they are much faster than the second type of approach, their level of privacy protection for IU will quickly drop as more SU queries are serviced.

We hereby propose PeDSS, a novel framework that protects both IU and SU privacy from untrusted SAS in a database-driven DSA system. PeDSS leverages the homomorphic property of a proxy re-encryption scheme, called the AFGH scheme, to eliminate the need for an online trusted third party and hence mitigates the single-point-of-failure issue in existing works. Also, PeDSS successfully integrates differential privacy schemes with homomorphic encryption-based privacy protection, such that it can ensure IU privacy will not be compromised regardless of the number of SU queries while still achieving fast and scalable spectrum request service time.

1.2 Message Authentication in Resource-Constrained Networks

In emerging applications, such as home automation, industrial controllers, and sensor networks, a large number of energy-constrained computing devices are getting closely integrated with the existing computer infrastructure through bandwidth-constrained networks to form the Internet of Things (IoT) [59]. The successful adoption of those applications will partially depend on our ability to thwart security and privacy threats, including message forgery and tampering. Today, message authentication code (MAC) is the most commonly used method for providing message authenticity and integrity in wired/wireless network applications. The cryptographic strength of a MAC depends on the cryptographic strength of the underlying cryptographic primitive (which may be a hash or block cipher), the size of the MAC output, and the size and quality of the key. Hence, all standard MAC schemes produce an output of at least a few hundred bits to ensure a sufficient level of cryptographic strength.

To employ MACs in a resource-constrained (i.e., energy and/or bandwidth constrained) network, we need to consider two problems: the computational burden on the devices for generating and verifying the MACs, and the additional communication overhead incurred due to the inclusion of the MAC in each message packet/frame. The first problem can be addressed by using dedicated hardware and cryptographic accelerators [28, 66]. Hardware accelerators, particularly GPUs are becoming more and more popular on the high-performance systems, as well as smartphones and smart cars. Developers need simpler abstractions [48] and programming languages to program these accelerators [18, 21, 22]. Significant domain knowledge is also required to tune the performance of the code to achieve peak performance [19, 20]. However, the second problem is not as easy to address. In energy-constrained networks (e.g., low-power wide-area network (LPWAN)) and bandwidth-constrained networks (e.g.,

in-vehicle controller area network (CAN)), the payload size of each packet/frame is very short (less than 150 bits in some protocols), and not more than a few bits can be spared for the MAC [55, 59]. A straightforward yet naive solution to this problem is to employ a truncated MAC with a very compact size [67, 72]. However, short MACs sacrifice cryptographic strength in exchange for reduced communication overhead and energy consumption, which may be undesirable, or even unacceptable, in some applications.

Overall, we identify three challenges in employing MACs for resource-constrained networks—(1) ensuring that the cryptographic strength meets the security need of the application, (2) incurring minimal communication overhead so that the MAC can fit in a message packet while being protocol compliant, and (3) incurring minimal latency so that the MAC generation and verification processes do not cause unacceptable delays in the processing of packets.

1.2.1 Cumulative MAC

We firstly propose a novel approach for message authentication that we refer to as *Cumulative Message Authentication Code* (CuMAC) that addresses all of the three aforementioned challenges. In CuMAC, a sender utilizes a procedure called *aggregation* through which it divides each MAC output into multiple short segments and aggregates the MAC segments of multiple messages to form a short authentication tag. On the other hand, a receiver utilizes a procedure called *accumulation* through which it "accumulates" the cryptographic strength of the underlying MAC by collecting the relevant tags carried in the received packets. During the accumulation procedure, the receiver incurs a delay that is proportional to the accumulated cryptographic strength since it needs to wait for the relevant tags to be received and processed. In essence, CuMAC makes a favorable trade-off between security and latency—i.e., cryptographic strength can be increased, if needed, at the cost of increased

latency in verifying the received message's authenticity and integrity.

1.2.2 Cumulative MAC with Speculation

In latency-sensitive applications, the receiver may be required to immediately authenticate a message as it arrives. For this type of application, the trade-off made by CuMAC is not acceptable. To address this need, we also propose a variant of CuMAC called *CuMAC with Speculation* (*CuMAC/S*) that enables a receiver to accumulate the MAC's cryptographic strength while incurring a minimal delay. The core concept of CuMAC/S is motivated by the technique of *speculative execution*[1] which is widely employed in modern computer systems [10, 54]. CuMAC/S can be utilized in latency-sensitive applications where future messages can be predicted correctly with high reliability using the current and past messages along with an appropriate speculation model. In CuMAC/S, a sender speculates future messages, computes the corresponding MACs, and aggregates the MAC segments of the speculated messages into the authentication tag of the current packet. If the speculated value of a received message is correct and equal to the actual value, a receiver can accumulate the tags from the previous packets (containing MAC segments of the speculated value) and the current packet (containing a MAC segment of the actual value). In this way, CuMAC/S enables the receiver to *accumulate* cryptographic strength without having to wait for tags included in forthcoming packets; this significantly cuts down on the MAC verification delay.

Like speculative execution, if there is a misprediction in CuMAC/S, the MAC computation for the corresponding message needs to be re-executed which leads to wasted computational overhead. However, CuMAC/S does not suffer from the same drawback as that of speculative

[1]Speculative execution is an optimization technique in which a computer system performs speculative execution where some outcome is predicted and execution proceeds along a predicted path. Work is done before it is known whether it is actually needed, to prevent a delay that would have to be incurred by doing the work after it is known that it is needed.

execution—i.e., if there is a misprediction in CuMAC/S, execution does *not* need to be unrolled. Even if all of the predictions are incorrect, the receiver can verify at least one MAC segment that provides some nominal cryptographic strength which is equivalent to a conventional truncated MAC of the same size. Also, the misprediction of a message does *not* result in any wasted communication overhead, since the MAC segments associated with the message are aggregated with MAC segments of other messages.

1.3 Contributions

In this dissertation, we study two issues and propose novel solutions for two technical problems regarding authentication and privacy-enhancing in network applications: 1) how to preserve privacy in a centralized dynamic spectrum access system, and 2) how to achieve message authentication in resource-constrained networks. To address the first problem, we firstly present a comparison study across various IU privacy-preserving solutions in chapter 4. We then propose in chapter 5, which is a tailored IU operational privacy-preserving scheme. Considering a more general centralized DSA model and the privacy for both IU and SU, we propose PeDSS in chapter 6.

To address the second problem, we propose CuMAC in chapter 7 which achieves low communication overhead and high achievable cryptographic strength. To further address this issue under latency intensive scenario, we propose CuMAC/S in chapter 8, which introduce the novel concept of message speculation and achieve our goal.

The main contributions are summarized below.

1. We present a review of existing works and a comparative study on the two existing categories of proposals, and we explore different existing security metrics for evaluating

these existing works. Furthermore, we propose two new and generic metrics to evaluate IU privacy-preserving level that can be applied across different schemes.

2. We propose PriDSA, a framework for ESC-based DSA system to address the IU location privacy issue with untrusted SAS. PriDSA achieves a much higher privacy-preserving level and improved efficiency compared to existing secure multi-party computation solutions. Besides, PriDSA employs the concept of individual ESC operational security in ESC-based DSA systems, which provides a novel approach to protect IU privacy under colluding SAS and SUs.

3. We proposed PeDSS, a novel privacy-enhancing framework for database-driven spectrum sharing system, which is computationally efficient, provably secure, and free of an online trusted third party. As a result of novel design with relatively lightweight cryptographic primitive and integration with differential privacy, PeDSS achieves a great advantage on communication and computation overhead, compared to existing secure multi-party computation solutions.

4. We propose a novel message authentication scheme called CuMAC, which meets the security need of delay-tolerant, resource-constrained network applications. CuMAC is an embodiment of a novel concept that we refer to as *accumulation* of cryptographic strength in the context of MACs. This idea is fundamentally different from existing solutions.

5. We propose a variant of CuMAC called CuMAC/S that meets the security need of delay-sensitive, resource-constrained network applications. CuMAC/S enables the accumulation of cryptographic strength while incurring a minimal delay by employing the novel idea of message speculation.

Chapter 2

Related Work

In this chapter, we present related work on both domains studied in this dissertation.

2.1 Preserving Operational Privacy of Incumbent Users in Database-Driven DSA Systems

In a legacy database-driven spectrum sharing system under exclusion zone framework [38][33], SAS maintains a geo-location database indicating "exclusion zones"; spectrum request can only be approved when it is not located in any exclusion zone. Exclusion zone framework is suggested by FCC's proposal for DSA in the 3.5 GHz band [15], yet it is recommended to leverage sensing technique to define smaller exclusion zones to increase spectrum efficiency [15]. Another strategy is called as "protection zone" [16], where SAS calculates the potential interference incurred by an SU and then approve the spectrum request only when they are not harmful. In this paper, SAS manages the interference under the protection zone framework.

Currently, it is identified that addressing the location privacy of IUs is still in its infancy and more still need to be done [35]. In [5], database inference attack is identified, where SUs may try to infer IUs' operational status based on their spectrum access result. In [12] IU data obfuscation techniques are proposed and in [47] a tailored secure MPC protocol is

proposed. However, approaches proposed in [5], [12] and [47] only consider IU privacy.

In [75] a k-anonymity based approach is proposed to address both IU privacy and SU privacy. However, in its proposal, SAS is assumed to run the k-anonymity algorithm, so it cannot address IU privacy in the untrusted SAS scenario. In [76], differential privacy is applied to preserve privacy for both IUs and SUs, and a utility maximization protocol is proposed. However, the system model in [76] implies that a single IU may receive multiple queries from different SUs, and the security level will decrease when multiple queries happen [8]. This issue is not investigated in [76]. In [25][26], efficient MPC protocols dedicated for centralized DSA are proposed to address both IU privacy and SU privacy, under the protection zone model and exclusion zone mode respectively. The design of introducing an online trusted third party, the Key Distributor, is not favorable for military IUs, who do not tend to trust other parties and are not willing to participate in the spectrum allocation routine of the system. In [37], an MPC protocol called "PISA" is proposed for the DSA system at UHF TV band, yet in PISA there is also an online trusted third party called "Semi-trusted Third Party" (STP). Although STP is claimed to be "semi-trusted", it can decrypt IU private data since it owns the corresponding private key.

All of the above schemes require IUs to actively exchange messages with SAS and perform some computation based on accurate knowledge of IU operational data. Thus, they cannot handle the non-informing IU case, where IU does not interact with SAS and SU, and IU information can only be partially sensed by ESC.

2.2 Message Authentication in Resource-Constrained Networks

The legacy solution for generating a short authentication tag is to truncate the output of a *standard MAC* (e.g., from 128 bits to 16 bits) so that it fits a message packet. This type of a MAC is called a *truncated MAC*. It is utilized in Sigfox where the message packets contain 16 bit-MACs. To ensure backward-compatibility in a CAN, Szilagyi et al. proposed to replace 32 bits out of the available 64 bits of the data field with a 32-bit truncated MAC [67]. The drawback of this approach is that the cryptographic strength afforded by such schemes is limited by the size of the authentication tag.

For authenticating messages in a CAN, Nilson et al. proposed to use the 16-bit CRC field in each CAN frame to include a 16-bit MAC [55]. Their justification for this approach is that a CRC provides only error detection, whereas a MAC provides integrity as well as authentication. Hence, the CRC is redundant and can be replaced with a MAC. However, this approach would not be compatible with existing CAN bus systems, as the existing CAN bus controllers are designed to check the CRC field and reject any frames that contain an incorrect CRC value. Nevertheless, Nilson et al. proposed to use the 16-bit CRC field in four consecutive frames to include a 64-bit MAC. This type of a MAC is called a *compound MAC*. The compound MAC is calculated on a compound of multiple successive messages and distributed over successive packets. This scheme incurs significant latency in the verification of the messages because the receiver needs to receive and process all packets before being able to verify the associated compound MAC. This delay is unacceptable for CAN messages that require real-time processing.

Groza et al. proposed to utilize the full data-field of a subsequent CAN frame to send a 64-bit MAC [36]. This type of a MAC is called a *trailing MAC*. The trailing MAC significantly

increases the CAN busload. Note that the increased load on a CAN bus can negatively affect the messages' transmission latency; i.e., it may cause some of the messages to arrive at the receiving ECU after the required deadline [72, 73].

To authenticate multiple messages together, the concept of *aggregate MAC* has been proposed in the literature. Katz et al. provided the first provable aggregate MAC scheme where standard MACs of multiple messages generated by multiple senders were combined into one aggregate MAC [42]. Although their idea of bitwise XOR-ing the standard MACs for generating the aggregate MAC was simple, the authors presented proof of security, which illustrated that the security of the aggregate MAC was equivalent to the standard MAC. However, Eikemeier et al. showed that the aggregate MAC scheme introduced by Katz et al. was prone to combination attacks in which the adversary may generate an aggregate MAC of a new message using the aggregate MACs of previous messages [27]. Kolesnikov et al. provided an improved aggregate MAC scheme which performed better than the aggregate MAC by Katz et al. in the presence of packet loss and message duplicity [44].

Chapter 3

Technical Background

In this chapter, we present the technical background. we introduce the cryptosystem used to achieve a privacy-preserving DSA system, as well as its partially homomorphic feature. We also give an overview of existing privacy-preserving metrics and several application scenarios of lightweight message authentication.

3.1 Overview of AFGH Cryptosystem

PeDSS uses the AFGH scheme, which is a widely used proxy re-encryption scheme proposed by G. Ateniese et al. [4]. The homomorphic multiplication and inverse property of AFGH cryptosystem are critical for the design of PeDSS system, which enables SAS to perform spectrum computation without knowing the actual IU status. Meanwhile, the re-encryption property enables SUs to recover the potential spectrum license without the help of an extra trusted third party.

3.1.1 Overview of AFGH

AFGH cryptosystem is defined over a type 1 bilinear groups $(\mathbb{G}_1, \mathbb{G}_T)$, where a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$ exists with the following properties:

1. $\mathbb{G}_1$ and $\mathbb{G}_T$ are multiplicative cyclic groups of prime order p; g is a generator of $\mathbb{G}_1$.

2. e is an efficiently computable bilinear map with the following properties:

   - Bilinear: $e(u^a, v^b) = e(u, v)^{ab}$, $\forall u, v \in \mathbb{G}_1$, $a, b \in \mathbb{Z}_p^*$;
   - Non-degenerate: $e(g, g) \neq 1$.

The following describes the construction of the AFGH scheme, which is "the third attempt" in [4].

- **System parameters**: To setup the system, a Type 1 bilinear pairing system is required. Denote g as the generator of $\mathbb{G}_1$, p as the order of $\mathbb{G}_1$. Set $Z = e(g, g)$.
- **Key Generation($\mathsf{KG}(a_1, a_2)$)**: For two inputs $a_1, a_2 \in \mathbb{Z}_p^*$, set secret key $\mathtt{sk} = (a_1, a_2)$, compute public key $\mathtt{pk} := (Z^{a_1}, g^{a_2})$.
- **Re-Encryption Key Generation ($\mathsf{RG}(\mathtt{sk}_a, \mathtt{pk}_b)$)**: Taking private key $\mathtt{sk}_a = (a_1, a_2)$ of user A and public key $\mathtt{pk}_b = (Z^{b_1}, g^{b_2})$ of user B, the re-encryption key is computed by $\mathtt{rk}_{A \to B} := (g^{b_2})^{a_1} = g^{a_1 b_2}$.
- **First-Level Encryption ($\mathsf{E}_I(M, \mathtt{pk}_a)$)**: For a message $M \in \mathbb{G}_T$ and public key $\mathtt{pk}_a = (Z^{a_1}, g^{a_2})$, select a random nonce $r \leftarrow_\$ \mathbb{Z}_p^*$, and compute $c_1 = Z^r \cdot M$, $c_2 = Z^{r a_2}$, where $\leftarrow_\$$ means "samples from". The ciphertext is $C := (c_1, c_2)$.
- **Second-Level Encryption ($\mathsf{E}_{II}(M, \mathtt{pk}_a)$)**: For a message $M \in \mathbb{G}_T$ and public key $\mathtt{pk}_a = (Z^{a_1}, g^{a_2})$, select $r \leftarrow_\$ \mathbb{Z}_p^*$, and compute $c_1 = Z^{r a_1} \cdot M$, $c_2 = g^r$. The ciphertext is $C := (c_1, c_2)$.
- **First-Level Decryption ($\mathsf{D}_I(C_r, \mathtt{sk}_b)$)**: for a first-level ciphertext $C_r = (c_1, c_2)$ and its corresponding private key $\mathtt{sk}_b = (b_1, b_2)$, the plaintext is obtained by computing $M^* := \frac{c_1}{c_2^{1/b_2}}$.

- **Second-Level Decryption ($\mathsf{D}_{II}(C_r, \mathtt{sk}_a)$)**: for a second-level ciphertext $C_r = (c_1, c_2)$ and its corresponding private key $\mathtt{sk}_a = (a_1, a_2)$, the plaintext is obtained by computing $M^* := \frac{c_1}{e(g^{a_1}, c_2)}$.

- **Re-Encryption ($\mathsf{R}(C, \mathtt{rk}_{A\to B})$)**: for a message M encrypted by public key (Z^{a_1}, g^{a_2}), its second-level ciphertext $C = (c_1, c_2)$ can be re-encrypted to be a first-level ciphertext encrypted by public key $\mathtt{pk}_b = (Z^{b_1}, g^{b_2})$ by computing $c_2^* := e(c_2, \mathtt{rk}_{A\to B}) = Z^{(ra_1)b_2}$. The re-encrypted first level ciphertext is $C_r := (c_1, c_2^*)$.

3.1.2 Homomorphic Property of AFGH Scheme

Proposition 3.1. ***Homomorphic properties***: *Given an AFGH public and private key pair* $(\mathtt{pk}, \mathtt{sk})$, *consider two AFGH second-level encrypted ciphertexts* $C = \mathsf{E}_{II}(M, \mathtt{pk}) = (c_1, c_2)$ *and* $C' = \mathsf{E}_{II}(M', \mathtt{pk}) = (c_1', c_2')$. *The homomorphic multiplication operation* $C \otimes C' := (c_1c_1', c_2c_2')$ *produces a ciphertext of* MM'. *In another word,* $\mathsf{D}_{II}(C \otimes C') = MM'$.

The homomorphic inverse operation $\mathsf{inv}(C) := (c_1^{-1}, c_2^{-1})$ *produces a ciphertext of* M^{-1}. *In another word,* $\mathsf{D}_{II}(\mathsf{inv}(C)) = M^{-1}$.

We omit the proof since the homomorphic feature of AFGH has already been discussed in [63]. Note that this proposition implies that AFGH is homomorphic in terms of division.

It is important to note that these additional algorithms do not break the security of AFGH scheme itself. This is because the proof of Theorem 3.1 in [4] is not affected.

3.2 Cryptographic Assumptions

The security of AFGH scheme is preserved under extended decisional bilinear Diffie-Hellman (EDBDH) assumption and discrete logarithm (DL) assumption [4].

Assumption 1. (EDBDH assumption) Given instance $(g, g^a, g^b, g^c, e(g,g)^{bc^2}, Q) \in \mathbb{G}_1^4 \times \mathbb{G}_T^2$ where $a, b, c \in \mathbb{Z}_p^*$, as input, for each probabilistic polynomial time (PPT) algorithm $\mathcal{A}$, the probability with which $\mathcal{A}$ is able to differentiate whether $c = a \cdot b$, or $c \leftarrow\!\!\$\, \mathbb{Z}_p^*$ is negligibly small.

Assumption 2 ($\mathbb{G}_1$-DL Assumption)**.** Given $(P, P^a) \in \mathbb{G}_1^2$, where $a \in \mathbb{Z}_p^*$, as input for each probabilistic polynomial time (PPT) algorithm $\mathcal{A}$, the probability that $\mathcal{A}$ outputs a is negligibly small.

3.3 Existing Privacy-Preserving Metrics for IUs in DSA System

A few metrics have been used in existing DSA privacy-preserving works. In this section, we introduce these metrics and discuss whether they are appropriate for comparative studies of multiple privacy-preserving schemes.

3.3.1 Average Expected Location Estimation Error

This metric is used in [5, 13, 75]. Assume an attacker can obtain a probability density function $A(loc'|\mathcal{O})$ as the guess of an IU location given some observation $\mathcal{O}$. Assuming n_I IUs exist in a target region, the actual IU locations are used to partition the region into n_I

subregions using the Voronoi diagram approach. These subregions are denoted as $\mathcal{L}_i$. The *average expected location estimation error* metric is defined as:

$$Pri := \frac{1}{n_I} \sum_{i=1}^{n_I} \sum_{loc \in L_i} d(loc_i, loc) \frac{A(loc|\mathcal{O})}{\sum_{loc' \in \mathcal{L}_i} A(loc'|\mathcal{O})}, \tag{3.1}$$

where loc_i is the true location of the ith IU, $d(\cdot, \cdot)$ is the distance between two locations. This metric is proposed in [13] and is widely applicable. [5] and [75] use a special case of the metric that assumes $n_I = 1$.

3.3.2 Privacy Time

Privacy time used in [12] is a widely applicable metric that measures the degradation of location privacy level over time. It is the expected time that it takes for IU location estimation error to fall lower than a certain threshold.

3.3.3 Size of the Search Space

This metric used in [13] is the size of the search space of possible IU parameters. After collecting some observations, an adversarial SAS or SU can exclude some locations as possible IU locations, which means the search space of true IU locations is reduced. This metric is essentially equivalent with a special case of the "expected location estimation error" metric where $A(loc'|\mathcal{O})$ is set to be a uniform distribution in the search space area.

3.3.4 ϵ-Differential Privacy

When an attacker obtains observation $\mathcal{O}$ and attempts to distinguish the true location of an IU between l_0 and l_1 within a circle with radius r, ϵ-differential privacy[76] requires the likelihood ratio is lower than $e^{\epsilon r}$, where ϵ is the parameter of differential privacy and r can be any radius value smaller than the radius of the service area.

The formal definition is given as follows[1]:

$$\frac{P(\mathcal{O}|l_1)}{P(\mathcal{O}|l_0)} \leq e^{\epsilon r} \qquad \forall r > 0 \, \forall l_1, l_0 : d(l_1, l_0) \leq r. \tag{3.2}$$

This metric, however, cannot be applied in DSA systems where the interference management policy of SAS protects IUs from harmful interference. For example, when $\mathcal{O}$ is a positive response for an SU query at location l_0 and l_0 is extremely close to an IU at the same time, it is easy to see that the denominator $P(\mathcal{O}|l_0)$ in equation (3.2) must be 0, which makes differential privacy unachievable. Since most of the existing privacy-preserving works [5, 12, 13, 25, 26, 75] except [76] are designed for systems where harmful interference to IUs is strictly prohibited, ϵ-differential is not a suitable metric for a comparative study of DSA privacy schemes.

3.3.5 Provable Security with the Cryptographic Setting

By defining provable security with the cryptographic setting for a privacy-preserving DSA system, we attempt to abstract the attack model and formulate any attacker under this model as an adversarial algorithm $\mathcal{A}$. When the provable security is achieved, we expect that any probabilistic polynomial time (PPT) adversarial algorithm $\mathcal{A}$ cannot achieve its

[1]There are three equivalent definitions proposed in [3], and in this paper we show the third one.

goal at a non-negligible probability. Note that we call these security features "provable" because usually, we attempt to prove that it is at least harder for an adversary to achieve its goal, compared to breaking a secure cryptosystem or other underlying hard problems.

In [25, 26], provable security feature on IU operational privacy protection is proposed and proved. However, security definitions in [25, 26] are also tailored definitions towards a specific interference management policy ("protection zone" and "exclusion zone" respectively).

3.4 Message Authentication in Resource-Constrained Networks

We discuss three specific applications where the constraints of the application—either in terms of MAC size or energy/bandwidth consumption of the networked devices—prohibit the use of conventional MAC schemes.

3.4.1 Low-Power Wide-Area Network (LPWAN)

Many IoT applications (e.g., smart metering, smart city infrastructure, etc.) require a heavily-crowded network of low-cost energy-constrained battery-operated devices. The paradigm of LPWAN is aimed at fulfilling these requirements of IoT networks [59, 71]. Sigfox [64] is one example of a widely-known LPWAN technology. In Sigfox, the size of the message in each uplink packet can be from 0 to 96 bits. For instance, a Sigfox device may communicate the message corresponding to its GPS coordinates, temperature, speed, or on/off status, which can be represented using 48 bits, 16 bits, 8 bits, or 1 bit, respectively. The Sigfox devices, which are battery-powered, are expected to have a service/battery life of several years. As the energy consumption of a Sigfox device is directly proportional to the size of messages

communicated by it, it is imperative to communicate using short messages to ensure long battery life. We note that in the applications supported by Sigfox, the latency requirements may be relaxed, but the message integrity and authentication are of prime importance [62]. However, due to the small size of the messages and the long service life required by Sigfox applications, it is undesirable to employ a standard MAC scheme.

3.4.2 In-Vehicle Controller Area Network (CAN)

Today's high-end cars use a hundred or more electronic control units (ECUs) to enable advanced functionalities, such as real-time engine control [39]. ECUs in most modern vehicles communicate with each other over a broadcast channel called the Controller Area Network (CAN) bus [40]. Because the messages communicated between ECUs directly affect vital functions of a vehicle, some of which are safety-related (e.g., vehicle dynamics control system [41]), the security and reliability of the CAN bus and the integrity of the messages on it are critical [66]. Several studies have shown that a car's in-vehicle network can be compromised through either direct physical access (e.g., using the on-board diagnostics port) or a remote connection (e.g., using Bluetooth) to the CAN bus [11, 43, 45, 72]. Due to one such potential vulnerability, Jeep had to recall 1.4 million vehicles in 2015 [51]. To counter attacks on CANs, the US National Highway Traffic Safety Administration (NHTSA) recommends the inclusion of MACs to protect messages on the CAN bus [53]. To employ MACs in a CAN, we need to consider the three issues: (1) additional communication overhead due to the inclusion of a MAC in each CAN frame since the size of each frame is limited, (2) required cryptographic strength, and (3) additional latency due to the inclusion of MACs. The third issue is important because some messages sent over a CAN are delay-sensitive.

Figure 3.1 shows the multiple fields in the basic frame format in a CAN [9]. In the basic

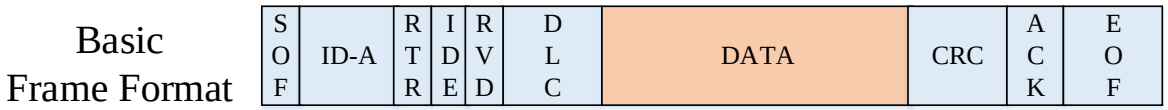


Figure 3.1: Conventional frame format in CAN.

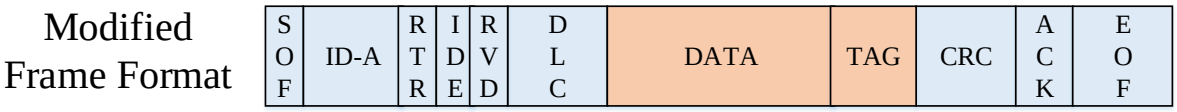


Figure 3.2: Proposed modification in frame format in CAN.

frame format, the frame comprises fields such as start-of-frame (SOF), message identifier (ID-A), data, 16-bit cyclic redundancy check (CRC), end-of-frame (EOF), etc. Except for the data field, we cannot arbitrarily change the length or the content of the fields in the CAN frame as that would make the modified frame incompatible with the existing CAN protocols. Hence, in the prior art, researchers have considered the modified frame format shown in Figure 3.2 to realize MAC-based authentication in each frame [67, 69]. In the modified frame format, the data field is used to accommodate the message payload as well as the authentication tag (TAG). This design of the modified frame format ensures that it is backward-compatible. However, the maximum allowed length of the data field in a CAN frame is only 64 bits. Hence, inserting a standard-sized MAC is not possible, and the use of even a truncated MAC severely lowers the CAN's message throughput.

3.4.3 Global Navigation Satellite System (GNSS)

GNSS utilizes multiple satellites to provide the receivers with high precision geospatial position information (i.e., longitude, latitude, and altitude) globally. The GNSS receivers are widely deployed in automobiles, ships, and aircraft. In the last decade, researchers have shown that GNSS messages can be readily spoofed to alter the course of automobiles, ships and unmanned aerial vehicles (UAVs) [74]. To thwart such attacks, some researchers have

proposed a scheme that relies on a symmetric-key based MAC and time-delayed asymmetry [29, 32]. This scheme is fully backward-compatible because it utilizes only the reserve fields in the GNSS messages to include the authentication tag. This limits the maximum size of a MAC to 10-20 bits in each 2-second long message-page. Hence, the use of a standard MAC scheme is impractical.

Chapter 4

Comparison Study across IU Privacy-Preserving Solutions

4.1 System Model

In this section, we introduce the general system model and attack model for an IU privacy-preserving DSA system.

4.1.1 Model of Database-Driven Dynamic Spectrum Access System

In this paper, we assume a general DSA service model, which consists of three entities: incumbent users (IUs, also known as "primary users" in some literature), Spectrum Access System (SAS), and secondary users (SUs). IUs update their operational status to SAS. SAS handles spectrum request from SUs by running a spectrum computation functionality $f(\cdot)$ that performs admission control for SUs. $f(\cdot)$ may also include channel assignment and/or power assignment operations. This system model summarizes the system models used in all existing privacy-protection works, including [5, 12, 13, 25, 26, 75, 76].

4.1.2 Attack Models

There exist several types of attackers focusing on breaking IU operational privacy in a database-driven DSA system:

A1: (Intruders) the attacker is an intruder, which is not an entity in the DSA system. It can overhear, intercept, and synthesize any message exchanged across the network. Specifically, it may directly extract IU operational status by looking at the messages sent from IUs to SAS. This threat model is also referred to as "Dolev-Yao model" [24]. Under this threat model, exchanging all messages under secure channels (e.g using TLS) can provide confidentiality, which ensures IU privacy. Thus, existing works are not focusing on proposing countermeasures towards this type of attack.

A2: (Honest but curious SAS, or semi-honest SAS) the adversary is a faithful SAS. While it performs all spectrum computation faithfully, it is also interested in discovering the operational parameters of IUs from the information it receives from IUs. Under this attack model, IU privacy fails if IUs directly send their plain operational status to SAS. Works [12, 13, 25, 26] consider this attack model.

A3: (Adversarial SU network) the adversary controls a group of compromised SUs so that it can obtain their spectrum request results to infer IUs' operational parameters. This type of attack is also referred to as "database inference attack" [5], which is studied in [5, 12, 13, 75].

A4: (Malicious colluding SAS) the adversary controls both a group of compromised SUs and a malicious SAS. The malicious SAS would deviate from the protocol to allure SUs to generate other observations to further infer IUs' operational status. In [26] such kind of attack is discussed.

4.2 Proposed Security Metric

In this section, we propose two additional metrics to evaluate the level of IU operational privacy. These two metrics are named *minimum adversarial estimation error* and *indistinguishable input*.

4.2.1 Minimum Adversarial Estimation Error

Suppose M IUs are operating with parameter sets $P_1^*, \cdots, P_M^* \in \mathcal{P}$, where $\mathcal{P}$ is the set of parameters with all possible values. An adversary $\mathcal{A}$ can obtain a posterior distribution of IUs' true parameters through its observations $\mathcal{O}$, which are obtained from compromised SAS or SUs. We denote $p_{\mathcal{A}}(\mathcal{P})$ as the posterior probability and

$$p_{\mathcal{A}}(\mathcal{P}) := \Pr[\text{IUs' operational parameter set}= \mathcal{P}|\mathcal{A} \text{ observes } \mathcal{O}], \tag{4.1}$$

where $\mathcal{P} := \{P_j\}_{j=1}^{M}$.

We assume that the adversary would sample a parameter set based on the posterior distribution $p_{\mathcal{A}}(\mathcal{P})$ as its guess of the IUs' true operational parameter sets. The privacy-preserving level (PPL), thus, can be defined as the expectation of the minimum estimation error, which is the minimum distance between any true IU parameters and any adversarial guess. That is,

$$PPL := E\left[\min_{i\in[M]} \min_{j\in[M]} d(P_i, P_j^*)\right]. \tag{4.2}$$

The above privacy-preserving metric definition extends the "expected location estimation error" concept to the privacy protection of any IU operational parameter. Besides, compared to the "average expected location estimation error" discussed in section 3.3, this metric

evaluates the minimum estimation error among multiple IUs.

Since different privacy-preserving techniques have a different format of observations and will result in the different posterior distribution of IU parameters from the adversary's point of view, it is not efficient or possible to derive a closed-form math expression of the PPL. Thus, we choose to use a numerical method to obtain the PPL value. Specifically, note that given the specification of a privacy-preserving system, it is straightforward to generate a large set of possible adversary observations $\mathcal{O}$ for any given IU parameter set P^*. We can therefore employ Markov Chain Monte Carlo(MCMC) [60] method to generate samples of the posterior distribution of IU parameters and use them to obtain an approximate PPL.

4.2.2 Indistinguishable Input

We introduce the new metric called *indistinguishable input* to extend the concept of provable IU operational security so that it can be applied to evaluating more privacy-preserving DSA solutions under different attack models. Indistinguishable input requires that an adversary is not able to extract much information about an IU's operation parameters (e.g. location, transmit power, etc.) from this IU's input to SAS. In other words, indistinguishable input says that when SAS receives a message from an IU, the likelihoods that the message is generated under two different IU operational parameter settings are *almost* the same.

To formally define this metric, we set up the following guessing game:

- Initialization phase: set up the DSA system faithfully.
- Challenge phase: an adversary (e.g. a compromised SAS) chooses two arbitrary different IU operational parameters P_0 and P_1 on its will. IU picks a random bit $b \leftarrow_{\$} \{0, 1\}$ and sends the corresponding IU input $\mathcal{I}_b$ to the adversary.

- Finalization phase: the adversary attempts to find out the secret bit b and return its guess b^*. We say the adversary wins the game if $b^* = b$.

Indistinguishable input means that any adversary cannot win the above game with an effectively higher probability than randomly guessing. Formally, if for any polynomial time algorithm $\mathcal{A}$ through which the adversary wins the above game at a probability $\epsilon_{\mathcal{A}}(\lambda)$ (λ is the security parameter of underlying cryptosystem), a design that has indistinguishable input property must ensure $\max_{\mathcal{A}} \left|\epsilon_{\mathcal{A}}(\lambda) - \frac{1}{2}\right|$ is negligible. Here, "negligible" means that for any integer c, there exists some λ^* such that $\forall\lambda \geq \lambda^*$,

$$\max_{\mathcal{A}} \left|\epsilon_{\mathcal{A}}(\lambda) - \frac{1}{2}\right| < \frac{1}{\lambda^c}. \tag{4.3}$$

4.3 Comparisons on Privacy-Preserving Strength

In this section, we compare the security strength for existing works [5, 12, 13, 25, 26, 75]. We analyze the indistinguishable input property for all these works and evaluate the average expected error, minimum adversarial estimation error, and privacy time for all of them under attack model **A2** and **A3**.

As we have discussed in section 3.3, "search space size" metric is essentially equivalent to a special case of "expected location estimation error" metric, and ϵ-Differential privacy is not a suitable metric since it does not apply to most of the existing works. Therefore, we are not using these two metrics for evaluation.

Note that as we have discussed in section 4.1.2, the security threat in attack model **A1** can be thwarted by using a secure channel, and the security threats in attack model **A4** has not been deeply studied in existing works. Therefore, we are not evaluating existing works under

these two attack models.

4.3.1 Comparison based on Indistinguishable Input Property

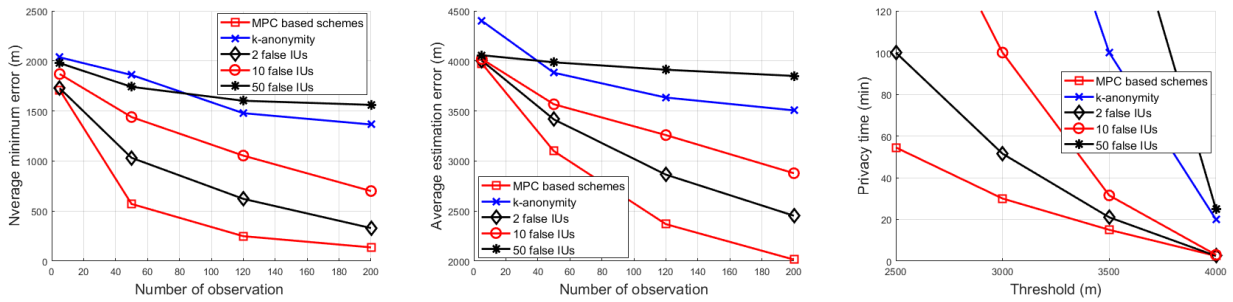
Under attack model **A2**, indistinguishable input property is only achieved for secure MPC protocol based schemes [25, 26]. This is because the obfuscated IU operational status still leaks non-negligible information. In the challenge phase of the guessing game, a semi-honest SAS can generate two IU parameter sets that lead to different obfuscated results and directly distinguish them.

Under attack mode **A3**, when adversarial SUs are taken into consideration, indistinguishable input property is not expected to be achieved for all existing schemes. Under this attack model, what an adversary can obtain includes the final spectrum allocation results. Meanwhile, under any DSA service model, the spectrum allocation result changes if the IU operational status changes. Hence, by synthesizing the spectrum allocation results, an adversary can distinguish IU operational statuses under different scenarios.

4.3.2 Comparison using Adversarial Estimation Error and Privacy Time

The comparative study in this subsection is based on simulation. In the simulation setting, IUs are deployed in a 20 km by 20 km rectangular region and there is one channel centered at frequency 3600MHz. The IUs are military radars with a 50m height and -80 dBm interference threshold; SUs are assumed to be outdoor CBSD devices and their antenna heights are 6m and transmission powers are 24 dBm. ECC-33 model [17] is used to formulate the path loss. The adversary collects two SU observations per minute.

We first compare the privacy-preserving strength under attack model **A3**, i.e. attacker is an adversarial SUs network. Figure 4.1 compares the privacy-preserving strength of MPC-based schemes [25, 26], obfuscation-based schemes designed for k-anonymity [5, 75], and obfuscation-based schemes that add false IU entries [12, 13]. MPC-based schemes behave the worst in this case since essentially in the perspective of an adversarial SU network, the secure MPC protocols in [25, 26] do not affect the spectrum computation result and hence introduce no additional confusion for the adversary to infer IU's operational parameters.



(a) Minimum error (b) Average error (c) Privacy time

Figure 4.1: Privacy-preserving strength evaluation under attack model **A3**.

We then compare the privacy-preserving strength under attack model **A2**, i.e. attacker is a semi-honest SAS. Table 4.1 shows the privacy-preserving strength against semi-honest SAS between different approaches. For MPC-based schemes in [26][25], we assume that an adversary cannot break a cryptographic system and can only obtain inferred IU locations by randomly guessing.

In the table, we see that MPC-based schemes provide strong IU privacy protection against semi-honest SAS. Obfuscation schemes designed for k-anonymity do not grant strong privacy protection under attack model **A2**, compared to the same simulation setting under attack model **A3**. This is because when a semi-honest SAS has the full knowledge of k-anonymity algorithm and the equivalent protection zone information, it can estimate the true IU locations with much smaller errors. For obfuscation-based schemes that add false IU items, it

can be observed that the privacy-preserving strength increases with more false IUs. Yet, intuitively we also expect the spectrum utility will decrease in this case, which will be analyzed in the next subsection.

Table 4.1: Privacy-preserving strength under attack model **A2**.

	Minimum estimation error(m)	Average estimation error(m)
MPC-based schemes	1743.02	4006.33
k-anonymity	509.52	2394.24
Obfuscation(2 false IUs)	0	599.83
Obfuscation(10 false IUs)	71.07	2618.74
Obfuscation(50 false IUs)	701.96	3453.90

4.3.3 Comparison based on Spectrum Utilization

In this subsection, we compare the spectrum utilization for different privacy-preserving approaches. Table 4.2 shows the privacy-preserving strength measured in minimum estimation error and spectrum utilization between different privacy-preserving solutions. We observe that MPC-based schemes provide the highest spectrum utilization and they also grant strong privacy protection against semi-honest SAS. For obfuscation-based schemes, we observe a trade-off between privacy protection and spectrum utilization under both **A2** and **A3** attack models. We also observe that schemes designed for k-anonymity sacrifice most spectrum utilization to achieve strong privacy protection under attack model **A3**.

Table 4.2: Spectrum utilization and privacy protection.

	Spectrum utilization(%)	Minimum estimation error(m)	
		Attack model **A2**	Attack model **A3**, 120 queries
MPC-based schemes	95.56	1743.02	249.14
k-anonymity	51.68	509.52	1478.35
Obfuscation(2 false IUs)	93.89	0	622.07
Obfuscation(10 false IUs)	87.92	71.07	1054.89
Obfuscation(50 false IUs)	62.67	701.96	1602.70

Chapter 5

PriDSA: Preserving the Incumbent Users' Location Privacy in the 3.5 GHz Band

In this chapter, we present PriDSA, our novel solution addressing IU location privacy in the 3.5GHz band scenario. We consider two different attack models with different assumptions about attacker capability: the non-colluding honest but curious (HBC) SAS model, and the colluding malicious SAS model. We designed two schemes, called PriDSA-v1 and PriDSA-v2, to tackle these two different cases. We leverage the partial homomorphic feature of a proxy re-encryption scheme to preserve IU privacy and achieve the goal of dynamic spectrum sharing at the same time. In PriDSA-v2, we additionally blind IU inputs to prevent malicious colluding SAS from extracting information from any single ESC input; spectrum allocation is achieved by leverage a commitment system, where only the overall blinding of data can be removed. We provide formal proofs to show the level of privacy that each PriDSA scheme provides.

5.1 System Model and Security Properties

In this section, we introduce the system model and security properties expected to be achieved.

5.1.1 System Model

We consider the 3.5 GHz spectrum sharing paradigm, consisting of a spectrum access system (SAS), IUs, environmental sensing capability (ESC) system distributed in the service area of SAS, and SUs. We assume that IUs are non-informing, such that they do not inform SAS their Exclusion zones (Ezone), where an operation CBSD can interfere with the IUs and should not obtain spectrum access permission. Instead, In the ESC-based SAS system, the ESC sensors are deployed in the vicinity of the predefined exclusion zones, which conservatively protects IUs from harmful interference. The ESC system converts part of exclusion zones to protection zones, where ESC detects the existence of IUs through spectrum sensing and SUs can get their spectrum requests approved in those areas based on the sensing result.

5.1.2 Inputs from ESC to SAS

In this paper, we assume an ESC system firstly identifies areas where an SU does not violate the spectrum access rules and hence can operate. We call such areas as safe zones. ESC determines safe zones based on its sensing results and default parameters of IUs. Protection zones are essentially areas outside of the safe zones.

Note that WINNF requires that the information relevant to federal activity passed from an ESC to a SAS shall be limited to protection area, channel, effective time, allowed retention time, and protection level [34]. The concept of safe zone indicates the protection level and

area and hence fulfills the requirement in [34]. Moreover, safe zones also indicate the area of ESC interference protection, where SUs are not allowed to transmit to prevent it causing excessive physical damage to ESC sensors. In the following subsection, we give an example on how an ESC sensor determines safe zones.

5.1.3 Example of Determining Safe Zone

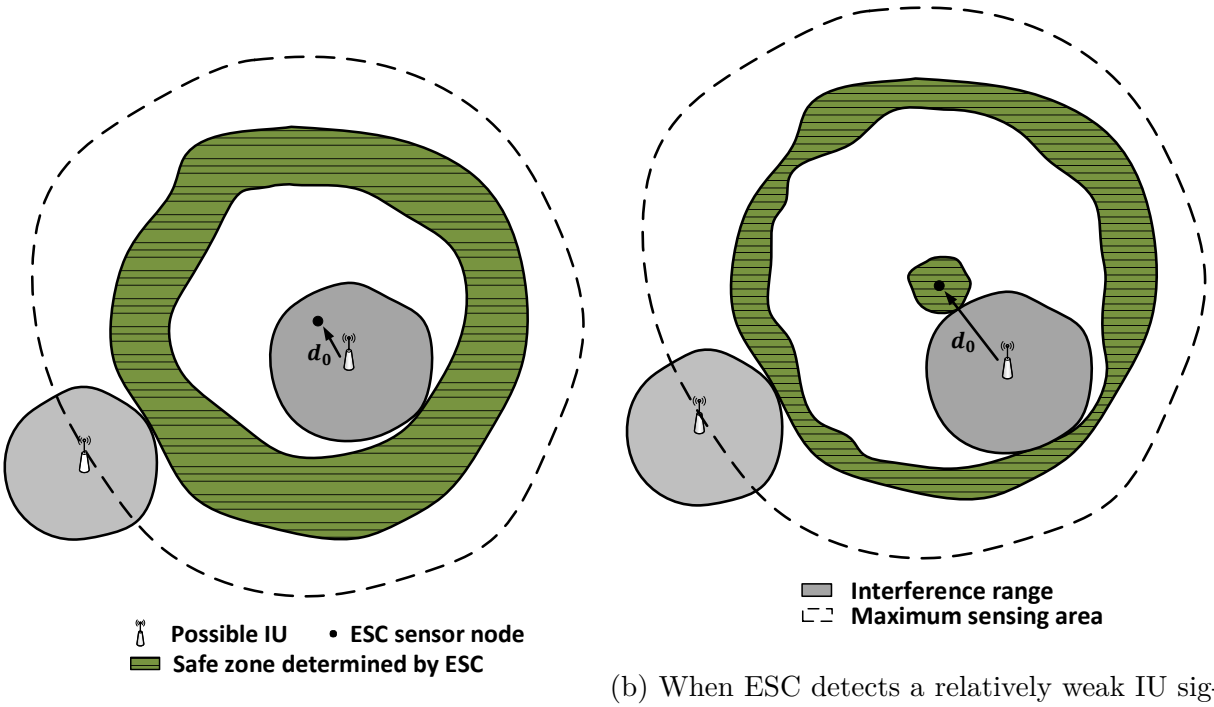
Assume that the ESC has a sensing range of r_{max}. Signals from IUs that are located outside of r_{max} cannot be detected by the ESC. Figure 5.1 illustrates our example of determining safe zone. The black dot at the center of Figure 5.1 (a) and (b) is an ESC sensor. $\mathcal{R}_{max}$, which is marked by the dashed line, is the maximum sensing area of the ESC sensor. An IU located outside of $\mathcal{R}_{max}$ cannot be sensed by the ESC. An IU inside $\mathcal{R}_{max}$ can be detected by ESC and its distance to the ESC, denoted as d_0, can be computed based on ESC sensing result as follows.

Set the ESC sensor's location to be the origin of a 2-dimensional polar coordination system. Denote the path loss between any other polar location (d, θ) and the origin as $G_f(d, \theta)$ in dBm, where θ is the polar angle, d is the radical distance, f is the center frequency and $G_f(d, \theta)$ can be obtained from any radio propagation model or software. Assume that the ESC sensor senses that the received signal strength of an IU at frequency f is ψ dBm. The ESC sensor can calculate its possible distance to the IU, denoted as d_0, by

$$P_{IU} - \psi = G_f(\theta, d_0) \Longrightarrow d_0(\theta) = G_f^{-1}(\theta, P_{IU} - \psi), \tag{5.1}$$

where $G_f^{-1}(\theta, \cdot)$ is the inverse function of $G_f(\theta, \cdot)$, and P_{IU} is the default transmission power of IU, which is assumed to be known by ESC.

Note that IUs near the ESC sensor have two types of possible locations. An IU in type one is



(a) when ESC detects a strong IU signal

(b) When ESC detects a relatively weak IU signal

Figure 5.1: Example of ESC determining safe zones.

inside r_{max} and hence its distance to the ESC can be computed as $d_0(\theta)$ based on ESC sensing result. An IU in type two is outside r_{max} and its signal cannot be detected by the ESC. For each possible IU location (d, θ), ESC can obtain an interference range, which is the area where an operating SU would impose harmful interference on this possible IU location. We denote the interference area as $O_f(d, \theta)$ and show it as a gray area in Figure 5.1. The aggregation of all such gray areas over all possible IU positions (i.e. $\mathcal{O} = \cup_{\theta \in [0,2\pi), d \in \{d_0, radius(\mathcal{R}_{max}, \theta)\}} O_f(d, \theta)$) form the non-safe zone and is colored white in figure 5.1. The safe zone that can be determined by the ESC sensor, hence, can be defined as $\mathcal{T} = \mathcal{R}_{max} - \mathcal{O}$ and is shown as the green area in figure 5.1. If an ESC sensor senses multiple IUs operating on the same frequency in its sensing range, it firstly computes the safe zone for each sensed IU signal, and then computes the intersection of all safe zones as the final sensing result passed to SAS.

It is important to note that the safe zones discovered by a single ESC are determined in a conservative way: those safe zones do not have intersection with any possible interference ranges. Hence, other ESC's sensing result will not change their safe zone status.

SAS then integrates all ESC input by taking the union of all ESC sensors' final safe zone sensing results. Figure 5.2 shows an example of aggregated safe zone information from multiple ESC sensors across a $20km \times 20km$ area. The white areas in the map are areas that no ESC is capable of defining them as the safe zone and are essentially the protection zones. SAS then performs DSA spectrum allocation according to this integration results, such that SAS will not allow SUs to access spectrum in protection zone areas.

There exists hierarchy among SUs sometimes. For example, in 3.5 GHz band FCC proposed two types of SUs in its proposal [15] priority access and general access, where the framework is referred as "three-tiered". This affects the resource allocation strategy, yet the security features and SAS service protocols of PriDSA scheme are not affected.

5.1.4 Attack Model

In this paper, we assume that SAS is not trustworthy/can be compromised and our goal is to protect IU location privacy from untrustworthy SAS. As shown in figure 5.1, SAS can violate IU location privacy by examining the safe zone information submitted by ESC sensors. This is because the inner radius of the safe zone, denoted as d_0, is the distance between an ESC sensor and an IU whose signal is sensed by the ESC. When an IU's signal is detected by three ESC sensors, the three safe zones submitted by the three ESC sensors can lead to three such d_0 estimations. A trilateration localization algorithm then can be used to pinpoint the location of the IU. This observation shows that the safe zone information from ESC sensors can be a significant threat to IU privacy when SAS operator is not trusted.

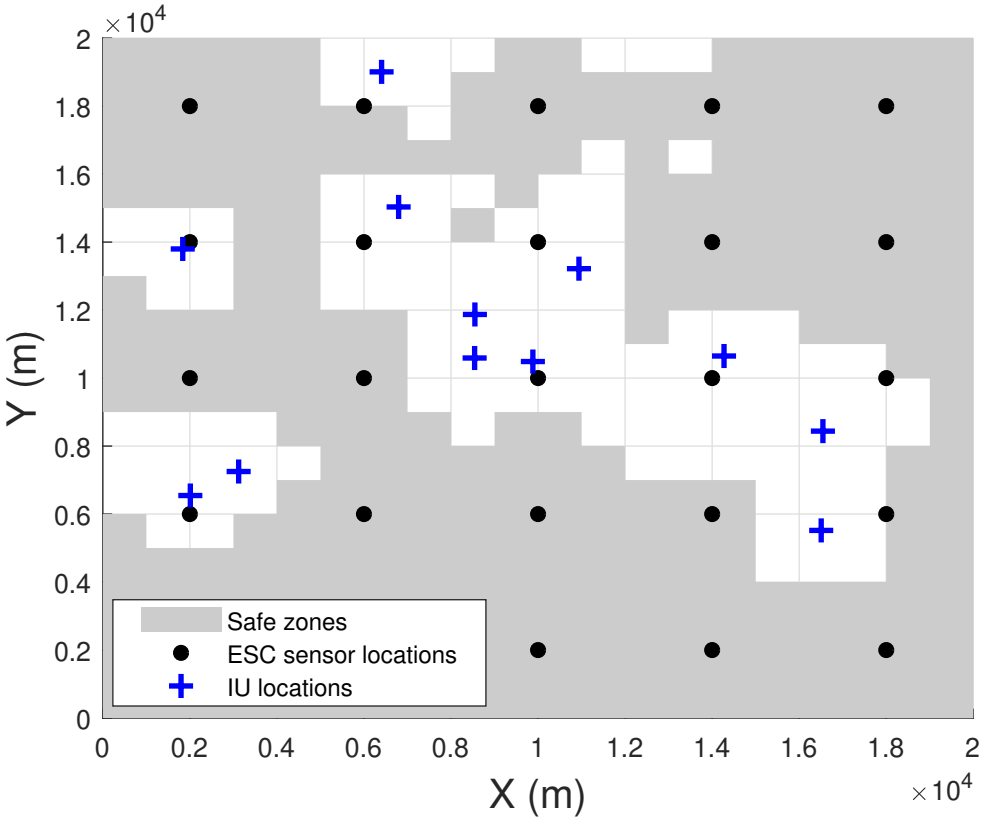


Figure 5.2: Safe zone determined by ESC over $20km \times 20km$ area

We study two attack models with ascendantly more powerful attacker capability and designed two different countermeasure schemes for these models.

- **Honest but curious (HBC) SAS Model:** This model assumes that SAS does not collude with SUs and follows protocols faithfully. Yet, it may attempt to derive sensitive IU location data from information that it receives. Our PriDSA-v1 scheme tackles this attack model.
- **Colluding malicious SAS Model:** This model assumes that SAS may collude with a small number of SUs in order to break the location privacy of IUs. SAS may also deviate from the spectrum allocation computation process to corrupt the spectrum allocation decisions. Our PriDSA-v2 scheme tackles this attack model.

In both models, we assume that ESC nodes are honest in their sensing report.

5.1.5 Security Properties

The following is an overview of the security properties of PriDSA. Formal definitions will be presented after introducing details of the system.

Correctness: This property requires that when an SU's operation location can impose interference to some IU, its spectrum request cannot be approved. i.e. SU can not receive a valid spectrum license in this case.

Location privacy for IUs: We assume SAS as the adversary, which is curious on IU locations. SAS can attempt to derive IU location by examining the safe zone information submitted by ESC nodes.

We expect that in PriDSA (both versions), when SAS is non-colluding and honest but curious, IUs have location privacy guarantee. This guarantee ensures that SAS is unable to

identify any safe zone information, which could be used by the adversary to determine the location of IUs.

Moreover, we expect PriDSA-v2 can still provide certain level of IU privacy protection under the colluding malicious SAS model. Specifically, individual ESC privacy is proposed in this case, which requires that SAS is not able to identify the safe zone information of any single ESC's data report, so as to reduce the likelihood of inferring the accurate locations of IUs.

IU individual privacy is important since by looking into the Ezone information of a single Ezone data report, SAS can infer the sensing results of some sensor node and incorporate localization techniques [70] to obtain IUs' locations.

Soundness: Soundness property requires that the denial or grant of spectrum access permissions to SUs must be strictly and correctly based on ESC sensing inputs.

5.2 PriDSA-v1: Secure DSA under HBC Model

We present the design of PriDSA-v1, which preserves IU location privacy under the assumption that SAS does not collude with SUs and follows protocols faithfully.

5.2.1 Overview

As shown in Figure 5.3, there are four parties in PriDSA-v1: (1) ESC nodes, (2) a SAS server for spectrum management, (3) SUs, and (4) a trusted Key Issuer. In a high level, SAS realizes two functions:

1. maintaining the database of encrypted safe zone maps. Periodically, from each ESC sensor, SAS receives its safe zone information that is encrypted by level-2 AFGH.

The encrypted safe zone information is denoted as $[\![\mathbf{X}]\!]_{II}$, where $[\![x]\!]_{II}$ denotes the level-2 AFGH encryption on a message x. SAS aggregates all ESCs' $[\![\mathbf{X}]\!]_{II}$ inputs by leveraging the homomorphic property of AFGH cryptosystem. The aggregated safe zone map is denoted as $[\![\mathbf{D}]\!]_{II}$ and is stored at SAS.

2. spectrum allocation based on the encrypted maps. To handle a spectrum request from an SU, SAS computes a potential spectrum license `cred` and generate a safe zone token based on $[\![\mathbf{D}]\!]_{II}$, and then sends `cred` and the safe zone token to the SU. Only when the SU is located in a safe zone, can the SU be able to successfully generate the permission proof based on the safe zone token.

In the following, we will describe the details of each step in PriDSA-v1 design.

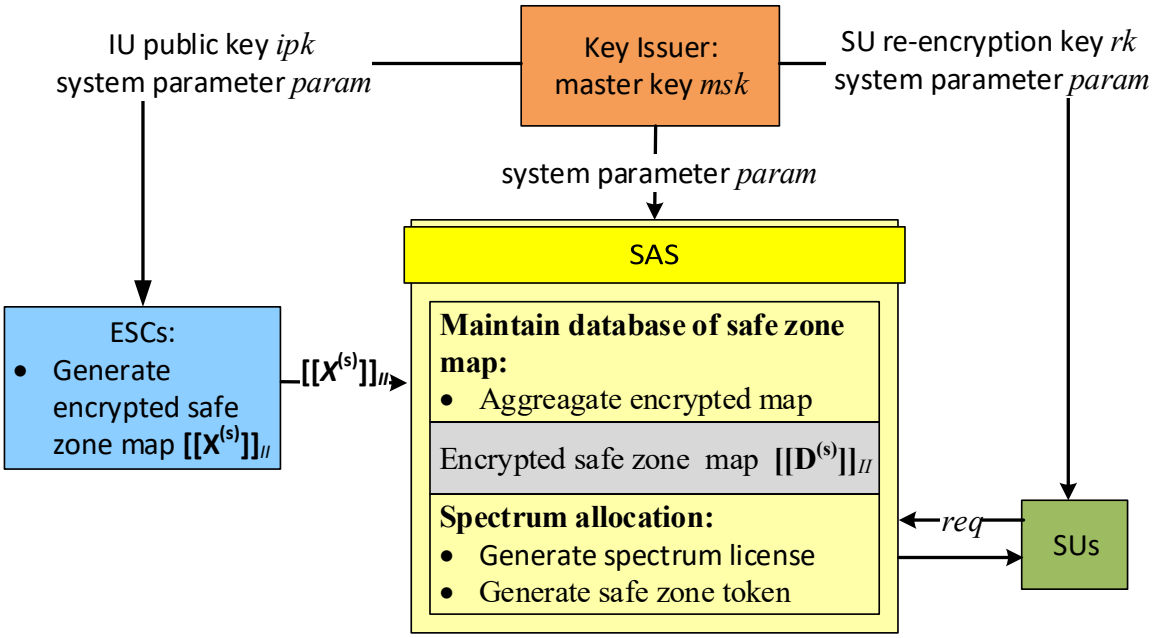


Figure 5.3: System Framework

5.2.2 System Setup

To initialize the system, the following three steps are executed by the Key Issuer, which is trusted by the IUs (e.g. it can be operated by IU operator):

1. Set up the AFGH scheme: Let the symmetric bilinear group pair be $\mathbb{G}_1, \mathbb{G}_T$ of prime order p, the corresponding bilinear mapping function be $e : \mathbb{G}_1 \times \mathbb{G}_1 \mapsto \mathbb{G}_T$, and AFGH system parameters be $g \in \mathbb{G}_1$, $Z = e(g, g)$.

2. Select $a_1, a_2 \leftarrow\!\!\$\, \mathbb{Z}_p^*$, where $\mathbb{Z}_p^*$ is the multiplicative group modulo p. Compute an AFGH key pair $(\mathtt{sk}, \mathtt{pk}) = \mathsf{KG}(a_1, a_2)$. Let the master secret key $\mathtt{msk} = \mathtt{sk}$ and IU's group public key $\mathtt{ipk} = \mathtt{pk}$. Select $f_1, h \leftarrow\!\!\$\, \mathbb{G}_1$, $H \leftarrow\!\!\$\, \mathbb{G}_T$ and compute $Y := e(f_1, h)$. Let system parameters $\mathtt{params} = (\mathbb{G}_1, \mathbb{G}_T, p, e, g, Z, f_1, h, Y)$.

3. Publish `ipk` and `params`.

An SU b must register its identity with the Key Issuer before sending any spectrum request. During the registration process, it generates a randomized AFGH key pair $(\mathtt{sk}_b, \mathtt{pk}_b)$ and sends $\mathtt{pk}_b$ to the Key Issuer. The Key Issuer then sends back the corresponding re-encryption key $\mathtt{rk}_b$, which is generated by $\mathtt{rk}_b \leftarrow \mathsf{RG}(\mathtt{msk}, \mathtt{pk}_b)$, through a secure channel.

After the initial setup, the Key Issuer can go off-line without affecting PriDSA-v1's runtime normal operation, which is outlined in the following subsections.

5.2.3 Details of Maintaining the Database of Safe Zone Map

In this subsection, we describes how ESC sends safe zone information to SAS without indirectly exposing IU location information .

ESC Input Data

The input safe zone data from an ESC to SAS is generated by the following procedure. Firstly, PriDSA system's service area is divided into a total of L same size grids and the 3.5 GHz spectrum is divided into F channels. Then, following the process outlined in Section 5.1.3, an ESC derives its own safe zone information and converts the information into a matrix $\mathbf{T} := [\mathbf{T}_{l,f}]_{L\times F}$ of dimension $L \times F$. If the entire grid l at channel f is inside the safe zone, ESC sets $\mathbf{T}_{l,f}$ to be a random non-zero element picked from $\mathbb{Z}_p$; formally,

$$\mathbf{T}_{l,f} \leftarrow\!\!\$\, \mathbb{Z}_p \backslash \{0\}. \tag{5.2}$$

Otherwise, $\mathbf{T}$'s entry $\mathbf{T}_{l,f}$ is 0; formally,

$$\mathbf{T}_{l,f} \leftarrow 0. \tag{5.3}$$

Afterwards, ESC creates matrix $\mathbf{X} := [\mathbf{X}_{l,f}]_{L\times F}$ by converting every $\mathbf{T}_{l,f}$ to group $\mathbb{G}_T$ through the exponentiation operation

$$\mathbf{X}_{l,f} \leftarrow Y^{\mathbf{T}_{l,f}}. \tag{5.4}$$

Then, ESC encrypts every entry of $\mathbf{X}$ using level 2 encryption and public key `ipk`. Finally, ESC sends the encrypted safe zone report $[\![\mathbf{X}]\!]_{II}$ to SAS. SAS cannot decrypt these encrypted reports since it does not have the decryption key.

Aggregate ESC Input

Upon receiving the encrypted safe zone maps $\{[\![\mathbf{X}(i)]\!]_{II}\}_{i=1}^{N}$ from all N ESCs, where (i) indicates the input of the ith ESC sensor, SAS integrates them together to form the aggregated

safe zone map $[\![\mathbf{D}]\!]_{II} := [[\![\mathbf{D}_{l,f}]\!]]_{L\times F}$ by computing element-wise homomorphic product of all input maps. That is:

$$[\![\mathbf{D}_{l,f}]\!]_{II} \leftarrow \otimes_{i=1}^{N} [\![\mathbf{X}_{l,f}(i)]\!]_{II}. \tag{5.5}$$

Note that an entry $\mathbf{D}_{l,f} \neq 1_{\mathbb{G}_T}$ indicates grid l is a safe zone grid in channel f.

5.2.4 Spectrum Computation

Spectrum computation requires an SU to obtain a valid spectrum license from SAS if and only if it is in a safe zone. To realize the above spectrum computation requirements, on a high level, SAS computes a safe zone token using the encrypted safe zone map, which can be further used to generate permission proof by SU. Specifically, the process works as follows.

Let l be the location of an SU b and f be its requested channel. SAS firstly generates a valid license `cred` for this request. The license contains the content of authorization for the SU b (i.e. expiration time, location, transmission power, etc), SAS certificate, an safe zone token and a digital signature over the authorization and safe zone token. The safe zone token is computed by using SU b's re-encryption key $\mathtt{rk}_b$ on safe zone map item $[\![\mathbf{D}_{l,f}]\!]_{II}$, namely,

$$[\![\mathbf{D}_{l,f}]\!]_{I,b} \leftarrow \mathsf{R}\left([\![\mathbf{D}_{l,f}]\!]_{II}, \mathtt{rk}_b\right). \tag{5.6}$$

Note that the safe zone token is a level 1 ciphertext of some random value other than $1_{\mathbb{G}_T}$ over SU b's key, if and only if SU b's location belongs to a safe zone.

Then, SAS sends spectrum license `cred` and safe zone token $[\![\mathbf{D}_{l,f}]\!]_{I,b}$ to the SU as the response to the SU's spectrum access request.

5.2.5 Operations at SU: Safe Zone Token Retrieval

Upon receiving the response to its spectrum access request, the SU b with public key $\mathtt{pk}_b = (p_1, p_2)$ decrypts the safe zone token $[\![\mathbf{D}_{l,f}]\!]_{I,b} := (D_1, D_2)$ in `cred`, namely $D^* \leftarrow \mathsf{D}_I([\![\mathbf{D}_{l,f}]\!]_{I,b}, \mathtt{sk}_b)$. If $D^* = 1_{\mathbb{G}_T}$, it means SU is not located in a safe zone. Thus, it should not access the spectrum; otherwise, SU b gains the permission to access channel k at its location l.

5.2.6 Permission Proof

The design of the `cred` ensures that an SU can prove to anyone that it indeed got a valid spectrum access permission from SAS. Firstly, the SAS certificate and signature carried in `cred` ensure that the content of `cred` is not modified and is generated by SAS. Moreover, the design of the safe zone token $[\![\mathbf{D}_{l,f}]\!]_{I,b} := (D_1, D_2)$ in `cred` ensures that SU can provide a proof of $D^* \neq 1_{\mathbb{G}_T}$ using zero-knowledge proof. Specifically, to demonstrate $D^* \neq 1_{\mathbb{G}_T}$, an SU b only needs to prove $D_2 \neq D_1^{b_2}, p_2 = g^{b_2}$, where SU b's secret key is $sk_b = (b_1, b_2)$ and public key is $pk_b = (p_1, p_2)$. To achieve this, SU b publishes a tuple $(D_1, D_3, \delta_{b_2}, c)$, which is computed by [30]:

- $D_3 \leftarrow D_1^{b_2}$, $r_{b_2} \leftarrow \mathbb{Z}_p$, $R_1 \leftarrow D_1^{r_{b_2}}$, and $R_2 \leftarrow g^{r_{b_2}}$.
- $c \leftarrow H(D_1, D_3, R_1, R_2)$ and $\delta_{b_2} \leftarrow r_{b_2} + cb_2$.

Anyone can check this zero-knowledge proof by firstly checking $D_3 \neq D_2$ and then checking $c = H(D_1, D_3, \tilde{R_1}, \tilde{R_2})$, where $\tilde{R_1} \leftarrow D_1^{\delta_{b_2}}/D_3^c$, $\tilde{R_2} \leftarrow g^{\delta_{b_2}}/p_2^c$. The proof for the correctness and soundness of this zero-knowledge proof can be found in [30].

5.3 PriDSA-v2: IU Privacy Protection against Colluding Malicious SAS

Similar to other existing MPC-based secure DSA schemes on HBC model [25][26][37], PriDSA-v1 is not secure when SAS colludes with some SUs, i.e. PriDSA-v1 and works in [25][26][37] cannot preserve IU privacy in the *colluding malicious SAS model*. To understand this, consider that SAS colludes with an SU b. SAS can use SU b's re-encryption key $\mathtt{rk}_b$ on the encrypted safe zone data $[\![\mathbf{X}_{l,f}]\!]_{II}$, namely, $[\![\mathbf{X}_{l,f}]\!]_{I,b} \leftarrow \mathsf{R}\left([\![\mathbf{X}_{l,f}]\!]_{II}, \mathtt{rk}_b\right)$, which can then be decrypted by SU b to reveal $\mathbf{X}_{l,f}$. When all $\mathbf{X}_{l,f}$ are revealed, all $\mathbf{T}_{l,f}$ are revealed. Essentially, the colluding SU b can reveal the entire Figure 5.1 of the ESC sensor. With Figure 5.1 revealed, SAS can easily derive d_0. With d_0s to at least three ESC sensors known, SAS can uniquely identify an IU's location through trilateration and compromise the location privacy of the IU.

Essentially, the failure of PriDSA-v1 in protecting IU privacy in the previous example is due to the fact that under the colluding SAS model, an individual ESC's safe zone report to SAS can be revealed by a colluding SU, which gives very accurate indications of the distance between IUs and their surrounding ESC nodes.

In this section, we introduce PriDSA-v2, which is a modification of PriDSA-v1 that mitigates the threat of SAS-SU collusion to IU location privacy. The spectrum computation function of PriDSA-v2 remains the same as PriDSA-v1, yet it prevents a colluding SAS from obtaining information on an individual ESC's safe zone report by adding a blinding factor on the ESC's protection-zone input data to the SAS. The blinding factor can only be removed after all ESC inputs have been aggregated. Thus, a colluding SAS at most can know the integrated safe zone map (i.e. the shape of the white area in Figure 5.2), which does not provide enough information to pinpoint individual IU's location.

PriDSA-v2 also includes mechanisms to ensure that a malicious SAS cannot deviate from the proper spectrum allocation computation process to produce incorrect SU spectrum permission. Essentially, PriDSA-v2 includes verifiable computation mechanism so that a SU can verify that SAS's response to its spectrum request is computed following the correct protocol. This ensures that SAS cannot launch denial-of-service attack on selected SUs or treat SUs unfairly.

In the remainder of this section, we introduce the details of PriDSA-v2 design.

5.3.1 Blinding of the Input Data

To blind the safe zone data at each location l and channel f, an ESC sensor picks a random nonce $\mathbf{A}_{l,f}$ in $\mathbb{Z}_p$, and replaces equation (5.4) by

$$\mathbf{X}_{l,f}^{(b)} \leftarrow Y^{\mathbf{T}_{l,f}+\mathbf{A}_{l,f}}. \tag{5.7}$$

The above formula adds the nonce to safe zone data $\mathbf{T}_{l,f}$. Note that the blinding factor $\mathbf{A}_{l,f}$ ensures that even when a colluding SU obtains $\mathbf{X}_{l,f}^{(b)}$, it will not be able to see the real safe zone information as it cannot remove the blinding factors.

ESC also computes the Pedersen commitment on $\mathbf{T}_{l,f} + \mathbf{A}_{l,f}$. That is:

$$\mathbf{r}_{l,f} \leftarrow_{\$} \mathbb{Z}_p; \quad \mathbf{C}_{l,f} \leftarrow Y^{\mathbf{T}_{l,f}+\mathbf{A}_{l,f}} H^{\mathbf{r}_{l,f}}. \tag{5.8}$$

The helper value $\mathbf{B} := [\mathbf{B}_{l,f}]_{L\times F}$ is computed by:

$$\mathbf{B}_{l,f} \leftarrow Y^{\mathbf{A}_{l,f}} \mathbf{C}_{l,f}. \tag{5.9}$$

Afterwards, this ESC computes $[\![\mathbf{X}_{l,f}^{(b)}]\!]_{II}$ and $[\![\mathbf{B}_{l,f}]\!]_{II}$ using level-2 encryption and sends them as input data to SAS. Meanwhile, ESC also sends all $\mathbf{C}_{l,f}$, $H^{\mathbf{r}_{l,f}}$, and $\mathbf{B}_{l,f}$ to a trusted IU tracker through a secure channel.

5.3.2 Data Aggregation and Removal of Blinding Factors

Upon receiving the input from all ESC sensors, SAS follows equation (5.5) to homomorphically aggregate the blinded encrypted safe zone data report. The aggregation results include the blinding factors and we denote it as $[\![\mathbf{D}^{(b)}]\!]_{II}$. SAS also aggregates helper value $[\![\mathbf{B}]\!]_{II}$ from all ESCs input by computing

$$[\![\mathbf{B'}_{l,f}]\!]_{II} \leftarrow \otimes_{i=1}^{N} [\![\mathbf{B}(i)_{l,f}]\!]_{II}, \tag{5.10}$$

where i is the index of ESC sensor and N is the number of ESC sensors.

The IU tracker publishes the cumulative commitment value $\mathbf{C'}$, where

$$\mathbf{C'}_{l,f} := \prod_{i=1}^{N} \mathbf{C}_{l,f}(i) \tag{5.11}$$

for all locations l and channels f.

Using $[\![\mathbf{B'}_{l,f}]\!]_{II}$ and $\mathbf{C'}$, SAS can remove the blinding factors from the aggregated safe zone map as follows:

$$[\![\mathbf{D}_{l,f}]\!]_{II} \leftarrow [\![\mathbf{D}_{l,f}^{(b)}]\!]_{II} \otimes \mathsf{inv}\left([\![\mathbf{B}'_{l,f}]\!]_{II}\right) \otimes [\![\mathbf{C}'_{l,f}]\!]_{II}. \tag{5.12}$$

$[\![\mathbf{D}_{l,f}]\!]_{II}$ is then used to perform spectrum allocation computation as described in Section 5.2.4 and 5.2.5.

5.3.3 Preventing SAS from Deviating from the Protocols

To prevent SAS from deviating from the proper spectrum computation protocol, PriDSA-v2 includes a spectrum enforcer service, which enables an SU to verify that the SAS's response to its spectrum request is computed properly. The enforcer proceeds as follows to verify the integrity of spectrum computation:

Verify the integrity of SU: the enforcer checks the content and signature of the response from SAS to ensure SU is sending the original response.

Fetch data from IU tracker: The enforcer extracts the location l and channel f from the response and gets the corresponding accumulated commitments $\mathbf{C}'_{l,f}$, helper values $\mathbf{B}'_{l,f}$, and commitment nonces $\mathbf{H}'_{l,f}$ from IU tracker.

Soundness check: A faithful response from SAS must satisfy the following condition:

$$\begin{cases} \mathbf{H}'_{l,f}\mathbf{B}'_{l,f} \neq \left(\mathbf{C}'_{l,f}\right)^2, & \text{if the response is approval} \\ \mathbf{H}'_{l,f}\mathbf{B}'_{l,f} = \left(\mathbf{C}'_{l,f}\right)^2, & \text{if the response is denial} \end{cases} \tag{5.13}$$

The soundness check condition is valid for the following reason. Note that from equation (5.8)(5.9):

$$\begin{aligned} \mathbf{H}_{l,f}\mathbf{B}_{l,f} &= \mathbf{H}_{l,f}Y^{\mathbf{A}_{l,f}}\mathbf{C}_{l,f} = \left(\mathbf{C}_{l,f}\right)^2 Y^{-\mathbf{T}_{l,f}} \\ \Rightarrow \mathbf{H}'_{l,f}\mathbf{B}'_{l,f} &= \left(\mathbf{C}'_{l,f}\right)^2 Y^{-\sum_{i=1}^{N}\mathbf{T}_{l,f}(i)}. \end{aligned} \tag{5.14}$$

Since $\mathbf{T}_{l,f}(i)$ is either 0 or some random non-zero element, we have:

- If $\mathbf{H}'_{l,f}\mathbf{B}'_{l,f} = \left(\mathbf{C}'_{l,f}\right)^2$, then $\sum_{i=1}^{N}\mathbf{T}_{l,f}(i) = 0$ and all $\mathbf{T}_{l,f}(i)$ are 0 with probability $1-\epsilon$, where ϵ is some negligibly small probability. Hence, we can conclude that all ESC nodes are **not** marking location l in channel f as safe zone. Therefore, the second condition in equation (5.13) implies the requested location and channel is not a safe

zone, so the expected response should be "decline".

- If $\mathbf{H}'_{l,f}\mathbf{B}'_{l,f} \neq \left(\mathbf{C}'_{l,f}\right)^2$, then $\mathbf{T}_{l,f}(i) \neq 0$ for some i. Hence, we can conclude that at least one ESC node is marking location l in channel f as safe zone. Thus, the expected response from SAS should be "approval".

5.4 Security Definitions and Analysis

In this section, we provide formal definitions and proofs of PriDSA's security properties.

5.4.1 Correctness

Correctness property requires that when an SU is not located in a safe zone of any ESC, its spectrum request cannot be approved. i.e. SU cannot receive a valid or useful spectrum license in this case. Denote the whole PriDSA functionality as a function f:

$$\texttt{cred}^* := f(\mathcal{T}, \mathcal{B}, \texttt{req}), \tag{5.15}$$

where $\mathcal{T}$ is the set of all received safe zone reports, $\mathcal{B}$ is its corresponding set of helper value maps.

The formal definition of correctness is given as follows:

Definition 5.1. PriDSA is correct if it satisfies the following condition: For any input $(\mathcal{T}, \mathcal{B}, \texttt{req})$ to PriDSA functionality, if the requested location l and channel f is not in any safe zone, the recovered license $\texttt{cred}^* := f(\mathcal{T}, \mathcal{B}, \texttt{req})$ is invalid.

Theorem 5.2. *The probability with which PriDSA (both version) is NOT correct is negligible.*

Proof. The correctness follows directly from the specification of the PriDSA protocols. The safe zone data can be correctly aggregated by the homomorphic feature of AFGH cryptosystem, as long as SAS aggregates them faithfully; the unforgeability of the digital signature in the spectrum license ensures SUs that are outside of safe zones cannot recover valid licenses, and the soundness feature of zero-knowledge proof [30] prevents such SUs from forging the permission proof. For PriDSA-v2, the correctness of removing the blinding factors is ensured by the homomorphic feature of AFGH cryptosystem.

We start with the proof of PriDSA-v1. Let l be the location and f be the channel in the spectrum request `req`. If the requesting SU is located in not located in any safe zone, then $\mathbf{T}_{l,f}^{(s)}(i) = 0$ for all i, and we have

$$[\![\mathbf{D}_{l,f}^{(s)}]\!]_{II} = \bigotimes_{\mathbf{X}^{(s)}(i)\in\mathcal{X}^{(s)}} [\![Y^{\mathbf{T}_{l,f}^{(s)}(i)}]\!]_{II} = [\![1_{\mathbb{G}_T}]\!]_{II}, \tag{5.16}$$

This demonstrate that the chance that PriDSA-v1 is incorrect is negligible.

For PriDSA-v2, note that:

$$\begin{aligned}
[\![\mathbf{D}_{l,f}^{(s)}]\!]_{II} &= [\![\mathbf{D}_{l,f}^{(b,s)}]\!]_{II} \otimes \mathsf{inv}\left([\![\mathbf{B}_{l,f}^{\prime(s)}]\!]_{II}\right) \otimes [\![\mathbf{C}_{l,f}^{\prime(s)}]\!]_{II} \\
&= \left(\bigotimes_{\mathbf{X}^{(s)}(i)\in\mathcal{X}^{(s)}} [\![Y^{\mathbf{T}_{l,f}^{(s)}(i)+\mathbf{A}_{l,f}^{(s)}(i)}]\!]_{II}\right) \left[\!\!\left[\frac{\mathbf{C}_{l,f}^{\prime(s)}}{\mathbf{B}_{l,f}^{\prime(s)}}\right]\!\!\right]_{II} \\
&= \left(\bigotimes_{\mathbf{X}^{(s)}(i)\in\mathcal{X}^{(s)}} [\![Y^{\mathbf{T}_{l,f}^{(s)}(i)+\mathbf{A}_{l,f}^{(s)}(i)}]\!]_{II}\right) \\
&\cdot \left(\bigotimes_{\mathbf{X}^{(s)}(i)\in\mathcal{X}^{(s)}} [\![Y^{-\mathbf{A}_{l,f}^{(s)}(i)}]\!]_{II}\right) \\
&= \bigotimes_{\mathbf{X}^{(s)}(i)\in\mathcal{X}^{(s)}} [\![Y^{\mathbf{T}_{l,f}^{(s)}(i)}]\!]_{II},
\end{aligned} \tag{5.17}$$

and similarly we can yield $[\![\mathbf{D}_{l,f}^{(s)}]\!]_{II} = \bigotimes_{\mathbf{X}^{(s)}(i)\in\mathcal{X}^{(s)}} [\![Y^{\mathbf{T}_{l,f}^{(s)}(i)}]\!]_{II}$. Therefore, we can follow the same procedure to prove the correctness of PriDSA-v2. □

5.4.2 Security Analysis of PriDSA under Non-colluding Honest but Curious (HBC) SAS Model

To formally analyze PriDSA security under the non-colluding HBC model, we setup a security experiment where an adversary tries to distinguish two groups of encrypted (and possibly blinded) safe zone reports as shown in Figure 5.4. Here, the procedure of generating the encrypted safe zone map report is denoted as algorithm $\mathsf{GenRep}(\cdot)$. The semantic security experiment depicts an adversary who has the messages exchanged between ESC and SU, and aims at distinguishing two different encrypted safe zone reports.

$\mathbf{Exp}_{\mathcal{A}}^{\text{sem}-\text{Sec}}(\lambda)$

$(\mathtt{msk}, \mathtt{ipk}, \mathtt{params}) \leftarrow \mathsf{Setup}(2^{\lambda})$.

$(\mathcal{T}^{(0)}, \mathcal{T}^{(1)}) \leftarrow \mathcal{A}(\mathtt{ipk}, \mathtt{params})$, where

$\mathcal{T}^{(0)} := \left\{\mathbf{T}^{(0)}(k)\right\}_{k=1}^{N}, \quad \mathcal{T}^{(1)} := \left\{\mathbf{T}^{(1)}(k)\right\}_{k=1}^{N};$

$b \leftarrow\$ \{0,1\}$.

$b' \leftarrow \mathcal{A}\left(\mathtt{ipk}, \mathtt{params}, \left\{\mathsf{GenRep}(\mathbf{T}^{(b)}(k))\right\}_{k=1}^{N}\right)$.

return 1 if $b = b'$; otherwise **return** 0.

Figure 5.4: Definition of semantic security experiment

The formal definition of IU privacy is shown as follows:

Definition 5.3. PriDSA is semantically secure for IUs if for all $\lambda \in \mathbb{N}$, the advantage $\mathsf{Adv}_{\mathcal{A}}^{\text{sem}-\text{Sec}}(\lambda)$ is negligible in λ for all Probabilistic Polynomial-Time (PPT) Adversaries $\mathcal{A}$,

where

$$\mathsf{Adv}_{\mathcal{A}}^{\text{sem}-\text{Sec}}(\lambda) = \left| \Pr\left[\mathbf{Exp}_{\mathcal{A}}^{\text{sem}-\text{Sec}}(\lambda) = 1\right] - \frac{1}{2} \right|$$

The definition of semantic security ensures that PriDSA preserving IU privacy under non-colluding HBC SAS model. This is because as long as an HBC SAS achieves non-negligible advantage in the semantic security experiment, it is able to extract information from ESC inputs.

Theorem 5.4. *If AFGH scheme is semantically secure, PriDSA (both versions) is semantically secure for IUs under the non-colluding HBC SAS model.*

Proof. To prove PriDSA is secure for IUs, we assume that there exists an adversary $\mathcal{A}$ which can break IU security with non-negligible probability. Then, we construct a simulator $\mathcal{S}$ which aims at breaking the semantic security of AFGH scheme by taking advantage of the adversary $\mathcal{A}$. $\mathcal{S}$ plays as the adversary in an given AFGH semantic security experiment, yet meanwhile it sets up a simulated semantic security experiment to interact with $\mathcal{A}$. Finally it can leverage the response from $\mathcal{A}$ to gain a non-negligible advantage in AFGH semantic security experiment.

In the following, we provide the full proof of Theorem 5.4.

Recall the definition A.2 in [4], i.e. definition of semantic security, which is shown in Figure. 5.5. AFGH scheme is semantically secure if:

$$\mathsf{Adv}_{adv,AFGH}^{std}(\lambda) := \left| \Pr\left[\mathbf{Exp}_{adv,AFGH}^{std}(\lambda) = 1\right] - \frac{1}{2} \right|$$

is negligible.

$\mathsf{Exp}^{std}_{\mathcal{A},AFGH}(\lambda)$

$(\mathtt{pk}_q, \mathtt{sk}_q), (\mathtt{pk}_B, \mathtt{sk}_B), (\mathtt{pk}_h, \mathtt{sk}_h) \leftarrow KG(1^\lambda)$.
$\mathtt{rk}_{q\to B} \leftarrow RG(\mathtt{sk}_q, \mathtt{pk}_B), \mathtt{rk}_{B\to h} \leftarrow RG(\mathtt{sk}_B, \mathtt{pk}_h)$.
$\mathtt{rk}_{h\to B} \leftarrow RG\mathtt{sk}_h, \mathtt{pk}_B$.
$(m_0, m_1, \alpha) \leftarrow \mathcal{A}(\mathtt{pk}_q, \mathtt{sk}_q, \mathtt{pk}_B, \mathtt{pk}_h, \mathtt{rk}_{q\to B}, \mathtt{rk}_{B\to h}, \mathtt{rk}_{h\to B})$.
$b \leftarrow\$ \{0,1\}, b' \leftarrow \mathcal{A}(\alpha, E_i(\mathtt{pk}_B, m_b))$.
return 1 if $b = b'$; otherwise **return** 0.

Figure 5.5: Definition of semantic security of AFGH scheme

Before setting up the simulated PriDSA semantic security experiment, we assume that $\mathcal{T}^{(0)}$ and $\mathcal{T}^{(1)}$ are not the same without loss of generality; otherwise the input value to the adversary $\mathcal{A}$ in the experiment will have no difference between $b = 0$ and $b = 1$, and hence $\mathcal{A}$ will not be able to distinguish them.

$\mathcal{S}$ set up the the simulated PriDSA privacy experiment as follows. Firstly $\mathcal{S}$ sets $\mathtt{msk} \leftarrow \mathtt{sk}_B$ and $\mathtt{ipk} \leftarrow \mathtt{pk}_B$. Here $\mathtt{msk}$ is actually unknown by $\mathcal{S}$. Let $\mathtt{pk}_h = (Z^{h_1}, g^{h_2})$ and $\mathtt{sk} = (B_1, B_2)$.

Afterwards, $\mathcal{S}$ submits $m_0 = 1_{\mathbb{G}_T}$ and $m_1 \leftarrow\$ \mathbb{G}_T \backslash \{1_{\mathbb{G}_T}\}$ to the challenger of AFGH semantic security experiment. It obtains $C := \mathsf{E}_{II}(\mathtt{pk}_B, m_b) \in \mathbb{G}_T \times \mathbb{G}_1$. Let $C = (c_1, c_2)$, where $c_1 = Z^{sB_1} m_b$ and $c_2 = g^s$. Upon receiving $\mathcal{T}^{(0)}$ and $\mathcal{T}^{(1)}$ submitted by $\mathcal{A}$, $\mathcal{S}$ skips the procedure of selecting b and generates simulated $\{[\![\mathbf{X}^*(k)]\!]_{II}, [\![\mathbf{B}^*(k)]\!]_{II}\}_{k=1}^N$ as follows. For any $k = 1, \cdots, M$, if $\mathbf{T}^{(0)}(k)_{l,f} = \mathbf{T}^{(1)}(k)_{l,f}$, then $\mathcal{S}$ computes the value of $[\![\mathbf{X}^*(k)_{l,f}]\!]_{II}$ and $[\![\mathbf{B}^*(k)_{l,f}]\!]_{II}$ faithfully according to algorithm $\mathsf{GenRep}(\cdot)$. Otherwise, $\mathcal{S}$ proceeds as follows:

- $\mathcal{S}$ selects $r \leftarrow\$ \mathbb{Z}_p^*$, $\mathbf{A}_{l,f} \leftarrow\$ \mathbb{Z}_p$.
- If $\mathbf{T}^{(0)}(k)_{l,f} = 0$, $\mathcal{S}$ sets $[\![\mathbf{X}^*(k)_{l,f}]\!]_{II} \leftarrow (\otimes_{i=1}^r C) \otimes [\![Y^{\mathbf{A}_{l,f}}]\!]_{II}$.

  If $\mathbf{T}^{(0)}(k)_{l,f} \neq 1$, $\mathcal{S}$ sets $[\![\mathbf{X}^*(k)_{l,f}]\!]_{II} \leftarrow ((c_1/m_1)^r, c_2^r) \otimes [\![Y^{\mathbf{A}_{l,f}}]\!]_{II}$.
- $\mathcal{S}$ samples $\mathbf{r}(k)_{l,f} \leftarrow\$ \mathbb{Z}_p$, and sets $[\![\mathbf{C}(k)_{l,f}]\!]_{II} \leftarrow [\![\mathbf{X}^*(k)_{l,f}]\!]_{II} \otimes [\![H^{\mathbf{r}(k)_{l,f}}]\!]_{II}$.

- $\mathcal{S}$ sets $[\![\mathbf{B}^*(k)_{l,f}]\!]_{II} \leftarrow [\![Y^{\mathbf{A}_{l,f}}]\!]_{II} \otimes [\![\mathbf{C}(k)_{l,f}]\!]_{II}$.

Then, $\mathcal{S}$ sends all simulated data reports, public key and system parameter to the adversary $\mathcal{A}$. Finally, $\mathcal{S}$ outputs 0 in AFGH semantic security experiment if $\mathcal{A}$ outputs 0; otherwise $\mathcal{S}$ outputs 1.

We see that for any $k = 1, \cdots, M$, if $\mathbf{T}^{(0)}(k)_{l,f} = \mathbf{T}^{(1)}(k)_{l,f}$, then $\mathcal{S}$ perfectly simulates the data report since the report is faithfully generated and it is the same whatever b is selected. In the following analysis, we argue that if $\mathbf{T}^{(0)}(k)_{l,f} \neq \mathbf{T}^{(1)}(k)_{l,f}$, then $\mathcal{S}$ perfectly simulates the data reports in a manner where b in PriDSA semantic security experiment is selected the same as the b in AFGH semantic security security experiment.

In the AFGH semantic security experiment, if b is selected as 0, then $[\![\mathbf{X}^*(k)_{l,f}]\!]_{II} = ((Z^{sB_1}m_b)^r, (g^s)^r) \otimes [\![Y^{\mathbf{A}_{l,f}}]\!]_{II} = (Z^{(sr)B_1}, g^{sr}) \otimes [\![Y^{\mathbf{A}_{l,f}}]\!]_{II}$ when $\mathbf{T}^{(0)}(k)_{l,f} = 0$; $[\![\mathbf{X}^*(k)_{l,f}]\!]_{II} = ((Z^{sB_1}m_b/m_1)^r, (g^s)^r)) \otimes [\![Y^{\mathbf{A}_{l,f}}]\!]_{II} = (Z^{(sr)B_1}m_1^{-r}, g^{sr}) \otimes [\![Y^{\mathbf{A}_{l,f}}]\!]_{II}$ when $\mathbf{T}^{(0)}(k)_{l,h,f,\gamma} = 1$. Thus in this case $\mathcal{S}$ simulates $[\![\mathbf{X}^*(k)]\!]_{II}$ perfectly in the case of setting b as 0 in the PriDSA privacy experiment.

On the other hand, if b is selected as 1, then we have

$$\begin{aligned}[\![\mathbf{X}^*(k)_{l,f}]\!]_{II} &= ((Z^{sB_1}m_b)^r, (g^s)^r) \otimes [\![Y^{\mathbf{A}_{l,f}}]\!]_{II} \\ &= (Z^{(sr)B_1}m_1^r, g^{sr}) \otimes [\![Y^{\mathbf{A}_{l,f}}]\!]_{II}\end{aligned}$$

when $\mathbf{T}^{(1)}(k)_{l,f} = 0$; meanwhile we have

$$\begin{aligned}[\![\mathbf{X}^*(k)_{l,f}]\!]_{II} &= ((Z^{sB_1}m_b/m_1)^r, (g^s)^r)) \otimes [\![Y^{\mathbf{A}_{l,f}}]\!]_{II} \\ &= (Z^{(sr)B_1}, g^{sr}) \otimes [\![Y^{\mathbf{A}_{l,f}}]\!]_{II}\end{aligned}$$

when $\mathbf{T}^{(1)}(k)_{l,f} = 1$. Thus $\mathcal{S}$ also simulates $\mathbf{X}^*(k)$ perfectly in the case of setting b as 1 in the PriDSA privacy experiment. We see $\mathcal{S}$ perfectly simulates $\mathbf{X}^*(k)$ in expected manner, so it also perfectly simulates $[\![\mathbf{B}^*(k)]\!]_{II}$ in the same manner as a result of the specification of $\mathcal{S}$.

Therefore we conclude that $\mathcal{S}$ perfectly simulate the PriDSA semantic security experiment and the advantage for breaking AFGH semantic security is the same as the advantage for breaking PriDSA semantic security.

□

Claim 1. Under EDBDH assumption and the non-colluding HBC SAS model, PriDSA is secure for IUs.

Proof. According to theorem 3.1 in [4], AFGH is semantically secure under EDBDH assumption, so PriDSA is semantically secure for IUs under EDBDH assumption. □

5.4.3 Security Analysis of PriDSA under Colluding SAS Model

We prove that PriDSA-v2 can ensure that individual ESC's safe zone input to SAS is secure from colluding SAS and SUs, which is called "individual ESC privacy". In Section 5.5.2, we will show how "individual ESC privacy" can mitigate the threat of privacy degradation as a result of colluding SAS and SUs.

To formally define individual ESC privacy, we set up a security experiment, which describes the capability of an adversary and definition of breaking individual privacy. In the privacy experiment, the adversary $\mathcal{A}$ is required to output the safe zone information at one arbitrary location that is sensed by an arbitrary ESC according to the adversary's own choice. To setup the experiment, the challenger $\mathcal{C}$ generates a randomized (yet unknown) safe zone data and send the encrypted inputs following PriDSA-v2 protocol to $\mathcal{A}$, assuming $\mathcal{C}$ is ESC and

$\mathcal{A}$ is the SAS. Since SAS may collude with SUs, we allow the adversary $\mathcal{A}$ to query a Reg oracle to simulate the process of SU registration, so that $\mathcal{A}$ can arbitrarily fetch any keys an SU may possess. The details of individual privacy experiment are shown in Figure. 5.6.

$\mathbf{Exp}_{\mathcal{A}}^{\text{ind-Priv}}(\lambda)$

$(\mathtt{msk}, \mathtt{ipk}, \mathtt{params}) \leftarrow \mathsf{Setup}(2^{\lambda})$.
for $k = 1, \cdots, M$
$\quad \mathbf{T}^{(e)}(k) \leftarrow\!\!\$\, \{0,1\}^{L \times F}, \quad \mathbf{T}(k) \leftarrow\!\!\$\, \{0,1\}^{L \times F}$.
endfor
Send $\mathtt{ipk}, \mathtt{params}, \left\{\mathsf{GenRep}(\mathbf{T}^{(e)}(k)), \mathsf{GenRep}(\mathbf{T}(k))\right\}_{k=1}^{M}$ to $\mathcal{A}$
$(k^*, l, f) \leftarrow \mathcal{A}^{\mathsf{Reg}(\cdot)}$.
return 1 if $\mathbf{T}_{l,f}^{(e)}(k^*) = 1$ or $\mathbf{T}_{l,f}(k^*) = 1$; otherwise **return** 0.

$\mathsf{Reg}(\mathtt{pk}_b)$

$\mathtt{rk}_b \leftarrow \mathsf{RG}(\mathtt{msk}, \mathtt{pk}_b)$.
return $\mathtt{rk}_b$.

Figure 5.6: Definition of individual privacy experiment

The formal definition of IU privacy is shown as follows:

Definition 5.5. PriDSA preserves individual ESC privacy if for all $\lambda \in \mathbb{N}$, the advantage $\mathsf{Adv}_{\mathcal{A}}^{\text{ind-Priv}}(\lambda)$ is negligible in λ for all Probabilistic Polynomial-Time (PPT) Adversaries $\mathcal{A}$, where

$$\mathsf{Adv}_{\mathcal{A}}^{\text{ind-Priv}}(\lambda) = \left| \Pr\left[\mathbf{Exp}_{\mathcal{A}}^{\text{ind-Priv}}(\lambda) = 1\right] - \frac{1}{4} \right|$$

Theorem 5.6. *PriDSA-v2 preserves individual ESC privacy under DLIN assumption.*

Proof. (Sketch) In PriDSA-v2, nonces are added to all Ezone and non-Ezone data, yet they remain unknown to $\mathcal{A}$, and the accumulated commitments can be determined by the sum

of all nonces. This fact allows us to perfectly simulate encrypted blinded data reports without knowing the actual Ezone or non-Ezone status, by leveraging a $DLIN$ instance. To prove PriDSA-v2 is individually private for ESC reports, we assume that there exists an adversary $\mathcal{A}$ which can break individual ESC privacy with non-negligible probability. Then we construct a simulator $\mathcal{S}$ which aims at solving DLIN problem by taking advantage of the adversary $\mathcal{A}$. $\mathcal{S}$ receives an DLIN instance, yet meanwhile it sets up a simulated PriDSA-v2 privacy experiment to interact with $\mathcal{A}$, where the DLIN instance is leveraged to simulate encrypted data reports without knowing its actual safe zone status. Finally it can leverage the response from $\mathcal{A}$ to gain a non-negligible advantage in solving DLIN problem.

To prove PriDSA is individually private for IUs, we assume that there exists an adversary $\mathcal{A}$ which can break IU individual privacy with non-negligible probability. Then we construct a simulator $\mathcal{S}$ which aims at solving DLIN problem by taking advantage of the adversary $\mathcal{A}$. $\mathcal{S}$ receives an DLIN instance, yet meanwhile it sets up a simulated PriDSA privacy experiment to interact with $\mathcal{A}$. Finally it can leverage the response from $\mathcal{A}$ to gain a non-negligible advantage in solving DLIN problem. Without loss of generality, we assume $\mathcal{A}$ is focusing on breaking individual privacy of safe zone data, i.e. finding out (k^*, l, f) such that $\mathbf{T}_{l,f}^{(s)}(k^*) = 1$ with non-negligible advantage.Suppose $\mathcal{S}$ receives a DLIN instance denoted as $(f_1, f_2, h, f_1^x, g_1^y, h') \in \mathbb{G}_t^6$, where $\mathcal{S}$ is expected to decide whether $h' = h^{x+y}$ or it is a random value drawn form $\mathbb{G}_1$.$\mathcal{S}$ set up the the simulated individual privacy experiment as follows. $\mathcal{S}$ firstly setup the whole system normally, except for setting the f_1 and h value as the one in DLIN instance, and set $H \leftarrow e(f_2, h)^\beta$. Then $\mathcal{S}$ picks two special IUs denoted as i_0 and i_1. $\mathcal{S}$ also pick a special location denoted as l. $\mathcal{S}$ generates their encrypted data reports at location l as follows:

- For i_0: assume $h' = h^z$ (z is unknown). Set $\mathbf{T}_{l,f}^{(s)}(i_0) = z - (a + b)$, select $A^* \leftarrow\!\!\$\, \mathbb{Z}_p$ and set $\mathbf{A}_{l,f}^{(s)}(i_0) = A^* + b$. In this way, $\mathbf{T}_{l,f}^{(s)}(i_0) + \mathbf{A}_{l,f}^{(s)}(i_0) = A^* + c - a$ and $\mathbf{X}_{l,f}^{(s)}(i_0) =$

$Y^{A^*+c-a} = Y^{A^*} e(f_1, h^c) e(f_1^a, h)^{-1}$. Select $\mathbf{B}_{l,f}^{(s)}(i_0) \leftarrow\$ \mathbb{G}_1$.

- For i_1: select $T', A' \leftarrow\$ \mathbb{Z}_p$,set $\mathbf{T}_{l,f}^{(s)}(i_1) = T' + b$, and set $\mathbf{A}_{l,f}^{(s)}(i_1) = A' - b$. In this way, $\mathbf{T}_{l,f}^{(s)}(i_1) + \mathbf{A}_{l,f}^{(s)}(i_1) = A' + T'$ and $\mathbf{X}_{l,f}^{(s)}(i_1) = Y^{A'+T'}$. Set $\mathbf{B}_{l,f}^{(s)}(i_1) \leftarrow \frac{Y^{A^*+A'} \mathbf{C}_{l,f}^{(s)}(i_0) \mathbf{C}_{l,f}^{(s)}(i_1)}{\mathbf{B}_{l,f}^{(s)}(i_0)}$.

The other parts of safe zone data report is generated normally. If the output from $\mathcal{A}$ is not (i_0, l), abort the game; otherwise, output $h' \leftarrow\$ \mathbb{G}_1$ for the DLIN instance.We state that $\mathcal{S}$ perfectly simulate the individual privacy experiment, since the helper value $\mathbf{B}_{l,f}^{(s)}(i_0)$ and $\mathbf{B}_{l,f}^{(s)}(i_1)$ are still randomly distributed, and the product of all helper values (i.e. $\mathbf{B}_{l,f}^{(s)}(i)$) is the same as what is generated faithfully. Therefore, as long as $\mathcal{A}$ breaks individual privacy and outputs (i_0, l) as safe zone, it implies $z-(a+b) \neq 0$ and hence h' is drawn randomly.Note that i^0, i_1 and l are drawn randomly drawn by $\mathcal{S}$. Hence the probability with which $\mathcal{S}$ doesn't abort the game is $\frac{1}{NL}$,so if $\mathcal{A}$ breaks individual privacy with non-negligible advantage ϵ, then $\mathcal{S}$ can break DLIN assumption with advantage $\frac{\epsilon}{NL}$, which is non-negligible. In other words, as long as DLIN assumption holds, the individual privacy can only be broke with non-negligible probability and hence PriDSA preserves individual privacy.

□

5.4.4 Soundness

Soundness feature requires that whenever SAS diverts from the protocol to intentionally give SU a wrong response that are not correctly computed based on safe zone information, SU can verify the response with the enforcer to dispute it. We show that PriDSA-v2 preserves the soundness property in section 5.3.3.

5.5 Evaluation

In this section, we evaluate the efficiency, accuracy and security strength of PriDSA.

5.5.1 Implementation Details

The AFGH cryptosystem is set up such that it provides approximately the same level of security as an RSA signature with a modulus size of 2048 bits. By using the pairing-based cryptography (PBC) library available at [49], we instantiate the AFGH cryptosystem and implement all SAS protocols and algorithms on a a laptop with 8x Intel(R) Core(TM) i7-4710HQ CPU @ 2.50GHz.

We deploy IUs in a 20 km by 20 km rectangular region and consider one channel centered at frequency 3600MHz. We set IUs as military radars with a 50m height and the interference threshold of IUs are -80 dBm; we assume SUs to be outdoor CBSD devices and set their antenna height as 6m and transmission power as 24 dBm. We use ECC-33 model [17] to formulate the path loss.

We compare PriDSA with one MPC-based privacy preserving system IP-SAS proposed in [26] and the data obfuscation technology proposed in [12]. IP-SAS is a MPC based protocol that assumes SAS is untrusted and achieves incumbent user privacy under exclusion zone model. The data obfuscation technology proposed in [12] can provide IU privacy in case of compromised SAS.

5.5.2 Privacy Preserving Level

When the SAS is malicious and colludes with SUs, inference attack based on safe zone information can reduce the location privacy level of IUs. Hence we can evaluate the location

privacy preserving level of PriDSA or IP-SAS using the metric mentioned in [12], which is expressed as:

$$PPL = \frac{1}{S/u^2} = \frac{u^2}{S}, \tag{5.18}$$

where S is the size of the area where an adversary believe that a specific IU may appear, and u is the unit of IU location representation. The smaller PPL is, the better the privacy protection of IU will be.

Table 5.1 compares the IU location privacy between IP-SAS, PriDSA and obfuscation technique in [12], using the PPL metric in equation (5.18). We obtain the average PPL value by repeatedly, randomly and uniformly deploying 20 IUs 1000 times. We set the side length of grids as 100m for IP-SAS and PriDSA, and assume there are 225 ESC sensor nodes with equally spaced distribution for PriDSA. The precision of u is set to be 10m. Note that for obfuscation techniques in [12] we only consider changing the parameter of "obfuscation strategy #1", where false IU entries are inserted and sent to SAS. This is because in our system model the "obfuscation strategy #2" proposed in [12] can only be processed at SAS, and thus it cannot preserve IU privacy in HBC model or malicious colluding model.

Under honest but curious (HBC) model, as a result of the provable security provided by IP-SAS and PriDSA, an adversary has to randomly guess the location of the IU across the whole SAS service area, so the PPL is very small.

Under the malicious colluding model, an adversary towards IP-SAS can directly pinpoint any given IU to a specific grid. The adversary towards PriDSA-v1 can identify all grids that have IUs, but cannot know the identify of the IU in each grid. Thus, while both PriDSA-v1 and IP-SAS's IU privacy suffers significantly when SAS can maliciously collude with SUs, PriDSA-v1 performs a little better than IP-SAS in terms of IU privacy protection. For PriDSA-v2, the adversary can only get an overall safe zone map and the location of an IU

can possibly in any grid outside of safe zone.

Table 5.1: Comparison of IU privacy level (PPL)

	HBC model	Malicious colluding model
IP-SAS	$2.5 * 10^{-7}$	0.01
PriDSA-v1	$2.5 * 10^{-7}$	0.0005
PriDSA-v2	$2.5 * 10^{-7}$	$2.049 * 10^{-6}$
Obfuscation (10 false IUs)	0.033	0.033
Obfuscation (30 false IUs)	0.02	0.02
Obfuscation (80 false IUs)	0.01	0.01

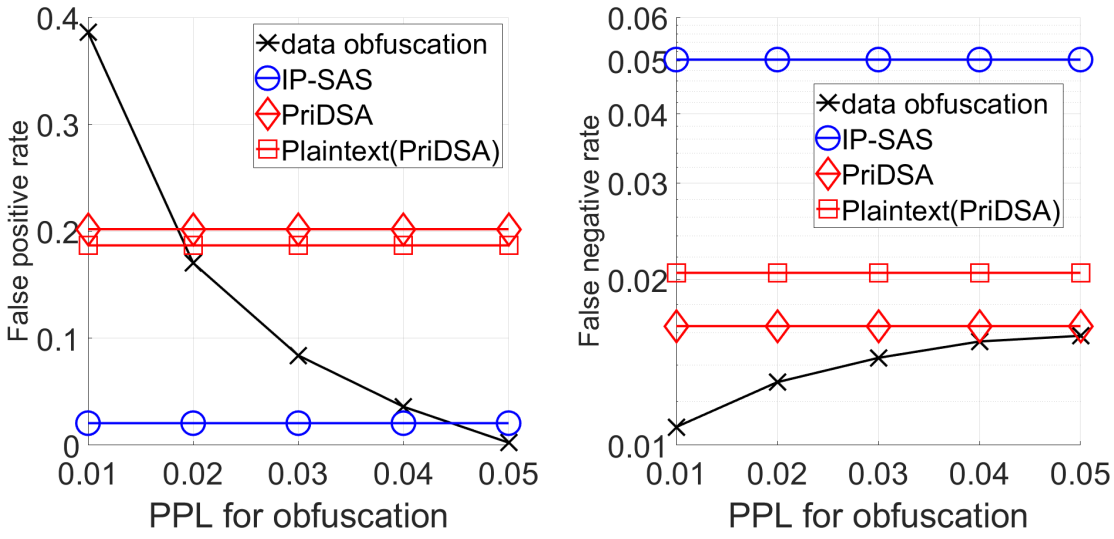
When obfuscation techniques in [12] is applied, SAS received extra IU location data entries, and the adversary (under HBC model or malicious colluding model) can find out the true location of target IU by randomly guessing. Hence we can achieve smaller PPL as long as more false IU entries are inserted. However, since in [12], SAS also protects false IUs from harmful interference, inserting more false IU entries may affect the accuracy of the system greatly. We'll evaluate this effect in the next subsection.

5.5.3 Accuracy

We use two metrics to evaluate the accuracy of spectrum allocation: false positive error rate, and false negative error rate. False positive error refers to declining an SU's request although it is safe for this SU to access the spectrum, and false negative error refers to accepting an SU's request although it may cause harmful interference to IU. To evaluate accuracy, we simulate 10000 SU requests from random locations for all three approaches under different settings, and we assume the error ECC-33 model obeys normal distributions with $\mu_e = -0.7$dB and $\sigma_e = 11.8$dB, according to the studies in [2].

Figure 5.7 shows the accuracy for different approaches. The parameters for grid side length, number of IUs and ESC nodes are the same as previous subsection. In figure 5.7a, we can observe that data obfuscation approaches in [12] sacrifices a lot of false positives even to

achieve 0.01 PPL, while PriDSA achieves a much smaller PPL using the same parameter setting. This is because to preserve IU privacy towards compromised SAS using data obfuscation technique in [12], lots of false IU entries are inserted. In figure 5.7b, we can observe the obfuscation technique in [12] achieves lower false negative rate, since some false negative error might be avoided as a result of interference to the extra false IUs. In figure 5.7, we also evaluate the performance of plaintext version of PriDSA, where ESC doesn't encrypt its messages. The plaintext version performs better in false positives and worse in false negatives, since in plaintext version the area is not discretized.



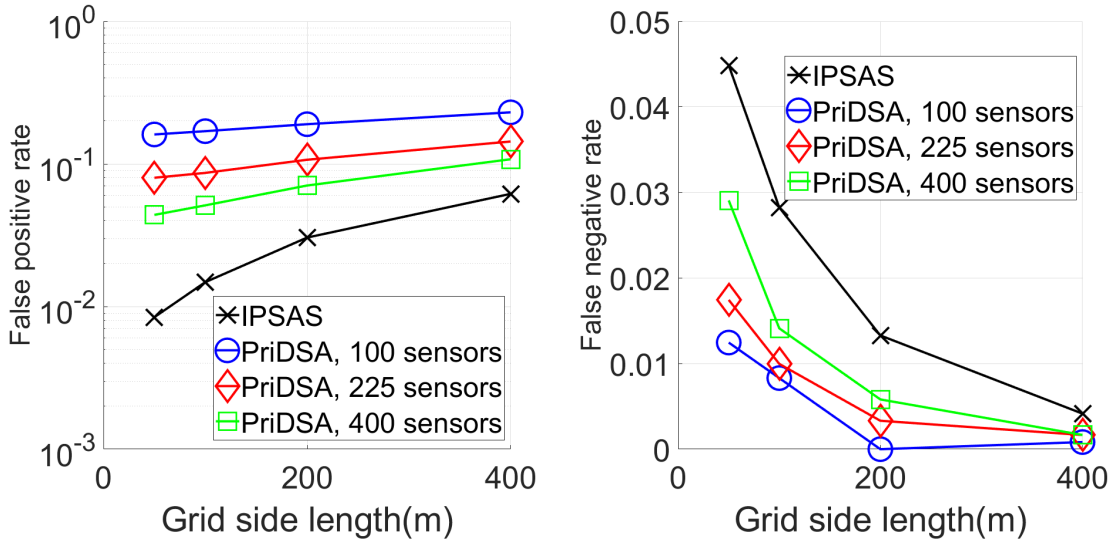
(a) False positive rate

(b) False negative rate

Figure 5.7: Accuracy for techniques in [12], IP-SAS and PriDSA(both versions)

We also evaluate accuracy for IP-SAS and PriDSA as a result of discretization error. Figure 5.8 shows the effect of grid size and number of ESC sensors on the spectrum allocation accuracy for PriDSA(in both versions) and IP-SAS. From figure 5.8a, we can see that PriDSA can achieve lower false positive rate by deploying more ESC sensors. For both PriDSA and IP-SAS, the false positive rate decreases when grid side length decreases, as a result of more accurate safe-zone representation. Meanwhile, performance on false negative errors shows

an opposite trend. In figure 5.8b we see false negative rate decreases when side length increases. This is because safe zones are defined by ESC systems only when a single grid is completely inside a safe zone, and larger grid side length makes the spectrum allocation more robust towards propagation model inaccuracy. We also observe that PriDSA(in both versions) achieves lower false negative rate compared to IP-SAS, since some false negative error might be avoided as a result of being out of the safe zone.



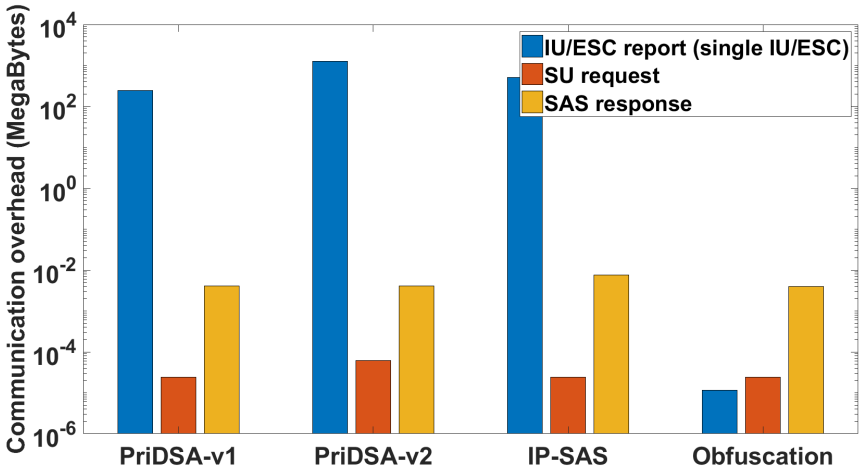
(a) False positive rate (b) False negative rate

Figure 5.8: Accuracy for techniques in [12], IP-SAS, and PriDSA(both versions) over different grid side length

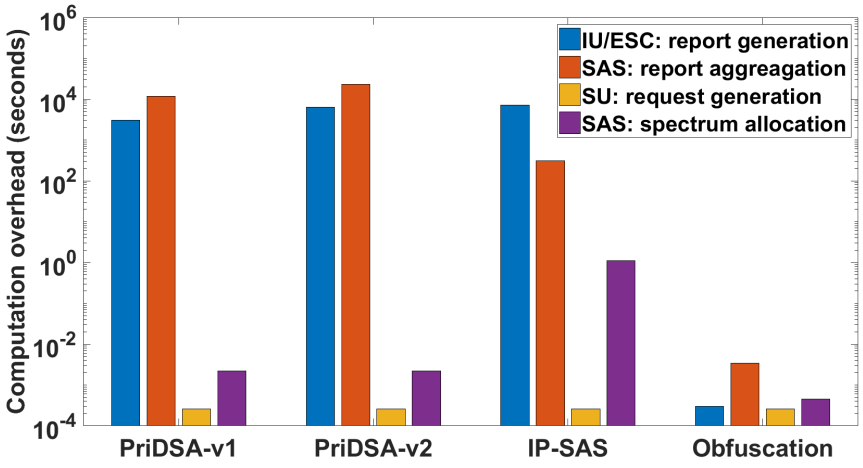
Note that for all approaches evaluated in this subsection, false negative errors happen as a result of the inaccuracy of the adopted radio propagation model, so the false negative error performance can be potentially improved by adopting more accurate model. Moreover, the design of PriDSA doesn't bring any additional false negative errors, which is guaranteed by the correctness property discussed in section 5.4.1.

5.5.4 Efficiency

Figure 5.9 shows the communication overhead and computation overhead comparison between PriDSA, IP-SAS and the approach in [12]. We evaluate the performance of the three approaches over a 400 km^2 area with 10 channels, and the grid side length is set to be 100m. We see that PriDSA achieves similar computation and communication overhead compared to IP-SAS, yet PriDSA-v2 preserves individual ESC privacy in the scenario of SAS-SU collusion. We can also observe that data obfuscation approach in [12] is more efficient in terms of IU data updating, yet as we analyzed above, its security strength is much weaker and it suffers from great accuracy loss for stronger privacy protection.



(a) Communication overhead



(b) Computation overhead

Figure 5.9: Communication overhead and computation overhead for different approaches

Chapter 6

PeDSS: Privacy Enhanced and Database-Driven Dynamic Spectrum Sharing

In this chapter we presents PeDSS, a novel framework that protects both IU and SU privacy from untrusted SAS in database-driven DSA system. PeDSS leverages the homomorphic property of a proxy re-encryption scheme, called AFGH scheme, to eliminate the need for an online trusted third party and hence mitigates the single-point-of-failure issue in existing works. In addition, PeDSS successfully integrates differential privacy schemes with homomorphic encryption-based privacy protection, such that it can ensure IU privacy will not be compromised regardless of the number of SU queries while still achieving fast and scalable spectrum request service time.

6.1 Overview of PeDSS and Security Properties

In this chapter, we present an overview of PeDSS and the security properties expected to achieve.

6.1.1 Overview of PeDSS

As shown in Figure. 6.1, there are four parties in a PeDSS system: (1) IUs, (2) a SAS server for spectrum management, (3) SUs, and (4) a Key Issuer for setting up keys for the system.

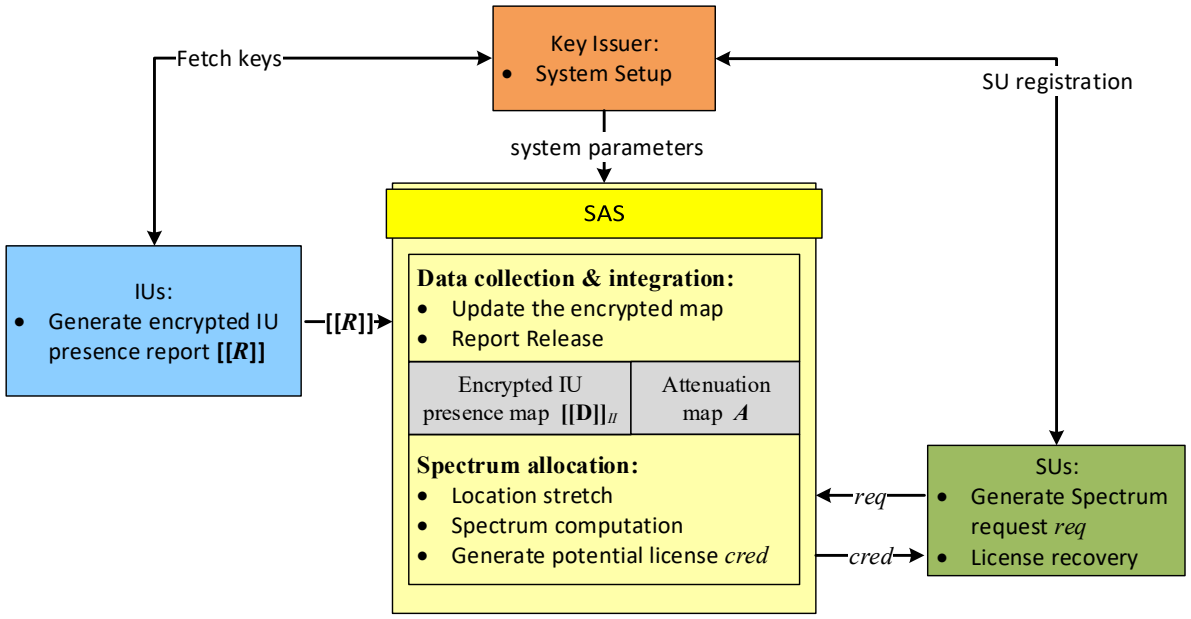


Figure 6.1: PeDSS overview

There are two routines of this system: data collection and spectrum allocation. In data collection routine, an IU obtains the IU group public key from Key Issuer and uses the key to encrypt their raw data regarding the presence of IUs. SAS leverages the homomorphic property of underlying cryptosystem to integrate encrypted data reports to form an encrypted map on IU presence, denoted as $[\![\mathbf{D}]\!]_{II}$.

In spectrum allocation routine, an SU sends the spectrum request along with its location, antenna height, and maximum transmission power to SAS. When an SU sends this spectrum request `req`, it will intentionally introduce fuzziness in its location claim so that its location privacy can be protected. Upon receiving a spectrum request along with the fuzzy location,

SAS computes the potential spectrum license `cred`, based on the encrypted IU presence map $[\![\mathbf{D}]\!]_{II}$ and the attenuation map A. The SU can manage to recover a valid license $\texttt{cred}^*$ by using its re-encryption key obtained from the Key Issuer. The recovered license is valid if and only if the potential interference does not exceed the interference sensitivity threshold of any IU.

6.1.2 Attack Model

In this paper, we assume SAS as the adversary. We assume that the SAS is *honest but curious*, which is commonly used to characterize any general service provider. In particular, SAS is trusted to faithfully conduct all algorithms and follow the protocols faithfully, but it is also interested in learning SUs' locations and IUs' operational data.

We also consider outside attackers that focus on compromising a SAS administrator in order to dump all the data stored at SAS. This type of attackers is interested in learning SUs' locations and IUs' operational data.

In database driven DSA systems, there also exists inference attacks that attempts to break IU privacy through forming an adversarial network of SUs to gather spectrum allocation results. It can be thwarted by adding fuzziness to the spectrum allocation process as shown in existing works [5][76][75]. Discussions on such types of countermeasures are beyond the scope of this paper.

6.1.3 Security Goals

In the following, we provide the security goals of PeDSS. Formal definitions will be presents in section 6.3.

Correctness: Correctness is the basic design goal of PeDSS, which ensures interference protection for IUs. This property requires that when an SU would cause interference greater than any IU's sensitivity threshold, its spectrum request cannot be approved. i.e. SU cannot receive a valid or useful spectrum license in this case.

IU privacy: IU privacy requires that an honest but curious SAS is not able to identify the IU presence status and operational data. It also requires that a compromised SAS will not leak any information related to IU presence status.

SU location privacy: SU location privacy requires that an honest but curious SAS can only extract limited information on the location of a queried SU. In this paper, we use differential privacy to formally define the concept of "limited information".

6.2 System Framework

In this section, we present the technical details of each step in PeDSS design.

6.2.1 System Setup

To initialize the PeDSS system, the Key Issuer firstly needs to run $\mathsf{Setup}(p)$ algorithm once. The algorithm has three steps:

- Step 1: Set up the AFGH scheme: Let the symmetric bilinear group pair be $\mathbb{G}_1, \mathbb{G}_T$ of prime order p, the corresponding bilinear mapping function be $e : \mathbb{G}_1 \times \mathbb{G}_1 \mapsto \mathbb{G}_T$, and AFGH system parameters be $g \in \mathbb{G}_1$, $Z = e(g, g)$.
- Step 2: Select $a_1, a_2 \leftarrow\!\!\$\, \mathbb{Z}_p^*$, and compute an AFGH key pair $(\mathtt{sk}, \mathtt{pk}) = \mathsf{KG}(a_1, a_2)$. Let the master secret key $\mathtt{msk} = \mathtt{sk}$ and IU's group public key $\mathtt{ipk} = \mathtt{pk}$. Let system

parameters $\texttt{params} = (\mathbb{G}_1, \mathbb{G}_T, p, e, g, Z)$.

- Step 3: The Key Issuer publishes $\texttt{ipk}$ and $\texttt{params}$.

An SU b sets up itself by registering itself at the Key Issuer before sending any spectrum request. During the registration process, it generates a randomized AFGH key pair $(\texttt{sk}_b, \texttt{pk}_b)$ and sends $\texttt{pk}_b$ to the Key Issuer. Upon receiving the SU's public key $\texttt{pk}_b$, the Key Issuer firstly generates the re-encryption key $\texttt{rk}_b$ through $\texttt{rk}_b \leftarrow \mathsf{RG}(\texttt{msk}, \texttt{pk}_b)$, and then sends $\texttt{rk}_b$ back to the SU through a secure channel.

After the initial setup, the Key Issuer, which is the only trusted third party in PeDSS, can go off-line and the PeDSS moves to the normal operation state. Essentially, PeDSS does not require its trusted third party to be engaged during the entire operation of SAS and the Key Issuer is only needed during the system setup stage.

6.2.2 Details of Data Collection and Integration Routine

In the normal operation state of PeDSS, one function of SAS is to collect and integrate IU input data, whose detail will be introduced in this subsection. The other function of SAS is to process spectrum allocation requests from SUs and we will discuss it in the next subsection.

6.2.3 Input Data Types

We assume that its input data to SAS includes its frequency f_{IU}, location l_{IU}, antenna height h_{IU} and sensitivity level to interference γ. These data have been identified in [25] to be related to spectrum allocation and are also private IU information.

To limit the computation overhead, all input data are discretized into a limited set of possible values. For locations, the discretization process divides PeDSS system's service area into a total of L same size grids and the location of an IU is represented by the grid it belongs to. For antenna height, frequency and interference sensitivity level, each is quantized into several value ranges and the actual value is approximated by the range it belongs to.

6.2.4 Generate Encrypted IU Presence Report

An IU runs algorithm $\mathsf{Report\_Gen}(\cdot)$ to generate an encrypted IU presence report. The IU presence information is represented as a matrix $\mathbf{R}$ of dimension $L \times H \times F \times \Gamma$, where L is the total number of grids, H, F and Γ are the total number of discretized antenna height ranges, frequency ranges and interference sensitivity level ranges, respectively.

If an IU is in grid l, has antenna height in range h, frequency in range f, interference sensitivity level in range γ, then $\mathbf{R}$'s entry $\mathbf{R}_{l,h,f,\gamma}$ is a random non-identity element picked from $\mathbb{G}_T$; formally,

$$\mathbf{R}_{l,h,f,\gamma} \leftarrow_{\$} \mathbb{G}_T \backslash \{1_{\mathbb{G}_T}\}. \tag{6.1}$$

For the other entries, we set $\mathbf{R}_{l,h,f,\gamma}$ to be the identity element of $\mathbb{G}_T$, which is denoted as $1_{\mathbb{G}_T}$; formally,

$$\mathbf{R}_{l,h,f,\gamma} \leftarrow 1_{\mathbb{G}_T} \tag{6.2}$$

Then, IU encrypts every entry of $\mathbf{R}$ using level 2 encryption and public key `ipk`. We denote the encrypted results as $[\![\mathbf{R}]\!]_{II}$.

6.2.5 Update the Encrypted Map

Upon receiving an encrypted data report $[\![\mathbf{R}]\!]_{II}$, SAS integrates the report into the encrypted map $[\![\mathbf{D}]\!]_{II}$.

The map $[\![\mathbf{D}]\!]_{II}$ is a matrix over level 2 AFGH ciphertext of the same dimension as the incoming report $[\![\mathbf{R}]\!]_{II}$. To initialize this map, SAS sets all elements to be $[\![1_{\mathbb{G}_T}]\!]_{II}$, which is the second level ciphertext of $1_{\mathbb{G}_T}$, encrypted using key `ipk`.

SAS updates the map by conducting element-wise homomorphic multiplication between $[\![\mathbf{D}]\!]_{II}$ and $[\![\mathbf{R}]\!]_{II}$. That is:

$$[\![\mathbf{D}_{l,h,f,\gamma}]\!]_{II} \leftarrow [\![\mathbf{D}_{l,h,f,\gamma}]\!]_{II} \otimes [\![\mathbf{R}_{l,h,f,\gamma}]\!]_{II} \tag{6.3}$$

for all entries of $[\![\mathbf{D}]\!]_{II}$ and $[\![\mathbf{R}]\!]_{II}$.

6.2.6 Release an Expired IU Presence Data Report

When an IU data report $[\![\mathbf{R}]\!]_{II}$ expires, SAS removes the impact of that report by element-wise homomorphically dividing $[\![\mathbf{D}]\!]_{II}$ by $[\![\mathbf{R}]\!]_{II}$. Formally, this means:

$$[\![\mathbf{D}_{l,h,f,\gamma}]\!]_{II} \leftarrow [\![\mathbf{D}_{l,h,f,\gamma}]\!]_{II} \otimes \mathsf{inv}([\![\mathbf{R}_{l,h,f,\gamma}]\!]_{II}) \tag{6.4}$$

for all entries of $[\![\mathbf{D}]\!]_{II}$ and $[\![\mathbf{R}]\!]_{II}$.

6.2.7 Details of Spectrum Allocation Routine

In order to protect its privacy, when an SU sends SAS its spectrum request, it will not tell SAS its exact location. Instead, it sends a fuzzy location that is different from its true location and guarantees differential privacy. Upon receiving the SU request with the fuzzy location, SAS needs to conduct proper spectrum allocation based on the fuzzy SU location and encrypted IU information. Finally, SAS needs to ensure that from the information that it sends to the SU, the SU can recover a proper license if and only if the spectrum computation result indicates that no harmful interference will be generated by the SU.

In the following, we describe the details of spectrum access request generation, spectrum allocation, and license recovery.

6.2.8 Generating Spectrum Access Request

Let $l_{SU} := (x, y)$ be the true location of the SU. To protect SU location privacy, the SU firstly picks a privacy parameter r_{SU}, which indicates the mean value of the distance between fuzzy location and true location. Then, the SU sets the parameter ϵ by

$$\epsilon \leftarrow \frac{2}{r_{SU}}, \tag{6.5}$$

and generates a fuzzy location l^*_{SU} by sampling from polar Laplacian distribution with parameter ϵ, which proceeds as follows[3]:

- Step 1: Select $\hat{\theta} \leftarrow\!\!\$\, [0, 2\pi)$;
- Step 2: Select $z \leftarrow\!\!\$\, [0, 1)$. Define function $C_\epsilon(r) = 1 - (1 + \epsilon r)e^{-\epsilon r}$, and find the root of equation $z = C_\epsilon(r)$ through bisection. Let the root be $\hat{r}$;

- Step 3: Set $x^* \leftarrow x + \hat{r}\cos\hat{\theta}$ and $y^* \leftarrow y + \hat{r}\sin\hat{\theta}$. Output $l^*_{SU} = (x^*, y^*)$.

Afterwards, the SU sends its spectrum request message using the fuzzy location l^*_{SU} instead of its true location, i.e., $\texttt{req} = (l^*_{SU}, r_{SU}, h_{SU}, f_{SU}, P_{\max})$, where h_{SU} is the discretized antenna height of SU and $P_{\max}$ is the maximum transmission power (in dBm).

In the following, we claim and proof that the above algorithm correctly generates fuzzy-locations following Laplacian distribution with mean value r_{SU}:

Proposition 6.1. *The above fuzzy-location generating algorithm creates locations following polar Laplacian distribution centered in the origin l_{SU} with parameter $\epsilon = \frac{2}{r_{SU}}$. That is,*

$$\Pr[l^*_{SU} = (r, \theta)] = \frac{\epsilon^2}{2\pi} r e^{-\epsilon r}, \tag{6.6}$$

where (r, θ) is the polar coordinate of fuzzy location l^*_{SU} centered at l_{SU}. Moreover, the mean value of distance between fuzzy location and true location is r_{SU}; that is,

$$E[d(l^*_{SU}, l_{SU})] = r_{SU}, \tag{6.7}$$

where $d(\cdot, \cdot)$ denotes Euclidean distance.

Proof. From the algorithm description, the fuzzy location l^*_{SU} has a polar coordinate of $(\hat{r}, \hat{\theta})$, centered at l_{SU}. Thus, $d(l^*_{SU}, l_{SU})$ has a cumulative distribution function (CDF) as $C_\epsilon(r)$, and its probability distribution function (PDF) $D_\epsilon(r)$ can be obtained as:

$$D_\epsilon(r) = \frac{dC_\epsilon(r)}{dr} = \epsilon^2 r e^{-\epsilon r}. \tag{6.8}$$

Since $\hat{\theta}$ is uniformly selected and independent of $\hat{r}$, we have:

$$\begin{aligned}\Pr[l^*_{SU} = (r,\theta)] &= \Pr[d(l^*_{SU}, l_{SU}) = r] \cdot \frac{1}{2\pi} = D_\epsilon(r)\frac{1}{2\pi} \\ &= \epsilon^2 r e^{-\epsilon r} \cdot \frac{1}{2\pi} = \frac{\epsilon^2}{2\pi} r e^{-\epsilon r}.\end{aligned} \tag{6.9}$$

Note that $D_\epsilon(r)$ is the pdf of the gamma distribution with shape 2 and scale $1/\epsilon$. So its mean value is:

$$E[d(l^*_{SU}, l_{SU})] = 2 \cdot \frac{1}{\epsilon} = r_{SU}. \tag{6.10}$$

□

6.2.9 Location Stretch

Upon receiving the SU request, the first step that SAS does is to perform a location stretch, which creates a location set $\mathcal{L}$, where the true location of the SU is expected to belong to $\mathcal{L}$.

To stretch the fuzzy location, SAS selects a stretch radius r_0, draws a circle centered at the fuzzy location with radius r_0, and puts all location grids that are located in the circle in the stretched set. That is,

$$\mathcal{L} \leftarrow \{l_{SU} : d(l_{SU}, l^*_{SU}) < r_0\}, \tag{6.11}$$

where $d(\cdot,\cdot)$ denotes Euclidean distance.

6.2.10 Spectrum Computation on Ciphertext Domain

Note the IU presence map $[\![\mathbf{D}]\!]_{II}$ is encrypted and SAS cannot directly compute the interference an SU would cause to IUs. Instead, SAS firstly enumerates all feasible operational parameters of an IU (i.e., all possible combinations of discretized IU location l, antenna height h, frequency f and sensitivity level γ). If for any IU parameter combination, the SU located in some location in $\mathcal{L}$ can cause interference larger than IU sensitivity level, the combination of IU parameters are put into a set $\mathcal{U}$.

Afterwards, the SAS homomorphically multiplies all items in the encrypted map $[\![\mathbf{D}]\!]_{II}$ with index in set $\mathcal{U}$. According to equation (6.1)(6.2)(6.3)(6.4), the result, denoted as I, will be

$$I = [\![1_{\mathbb{G}_T}]\!]_{II} \otimes [\![1_{\mathbb{G}_T}]\!]_{II} \otimes \cdots \otimes [\![1_{\mathbb{G}_T}]\!]_{II} = [\![1_{\mathbb{G}_T}]\!]_{II}, \tag{6.12}$$

if the SU will not generate harmful interference assuming it is located at *any* locations in $\mathcal{L}$; otherwise, I will be almost surely[1] a level 2 ciphertext of a random element in $\mathbb{G}_T$ other than $1_{\mathbb{G}_T}$.

Figure. 6.2 presents the pseudocode of the above spectrum computation on ciphertext domain, where $A(\cdots)$ is the attenuation map function. $A(l_{SU}, h_{SU}, f_{SU}, l, h, f)$ is assigned to be the **minimum** path loss between a transmitter located at grid l_{SU} with antenna height range h_{SU}, frequency range f_{SU} and a receiver located at grid l with antenna height range h, frequency range f. This map function can be pre-computed to reduce the online computation overhead of handling a spectrum request.

[1]The probability with which I being the ciphertext of $1_{\mathbb{G}_T}$ is $\mathcal{O}(1/p)$ where $p = 2^{80}$ if we are considering 80 bit level security.

$\mathcal{U} \leftarrow \emptyset$, $I \leftarrow \mathsf{E}_{II}(1_{\mathbb{G}_T}, \mathtt{ipk})$
for all possible (l, h, f, γ) combinations and all $l_{SU} \in \mathcal{L}$
  if $P_{\max} - A(l_{SU}, h_{SU}, f_{SU}, l, h, f) \geq \gamma$
    $\mathcal{U} \leftarrow \mathcal{U} \cup \{(l, h, f, \gamma)\}$
  fi
endfor
for all $(l, h, f, \gamma) \in \mathcal{U}$
  $I \leftarrow I \otimes [\![\mathbf{D}_{l,h,f,\gamma}]\!]_{II}$
endfor
return I

Figure 6.2: Spectrum computation on ciphertext domain

6.2.11 Generate Potential Spectrum License

A valid spectrum license consists of the PKI certificate of SAS $\mathtt{cert}_{SAS}$, license related message $\mathtt{msg}$ and the signature σ of $\mathtt{msg}$ signed by $\mathtt{sk}_{SAS}$. Moreover, $\mathtt{msg}$ is composed of SAS-stretched location set $\mathcal{L}$, SU's antenna height h, allowed maximum power $P_{\max}$ and extra information. The expiration date and SU's credential are put in the extra information field.

SAS firstly generates the valid license $\mathtt{cred}$ based on previously computed $\mathcal{L}$ and SU information, yet $\mathtt{cred}$ should only be sent to the SU if the SU causes no potential harmful interference. To achieve this, SAS firstly picks random element α in $\mathbb{G}_T$ and hashes it to random bit string $\mathtt{k}$. Then it encrypts the valid license using any symmetric key cryptosystem (say, AES) assuming $\mathtt{k}$ as the secret key, which is denoted as $\mathsf{E}_{AES}(\mathtt{cred}, \mathtt{k})$. Meanwhile, SAS encrypts α to level 2 ciphertext using key $\mathtt{ipk}$ and homomorphically multiplies it with the I returned by the algorithm in Fig. 6.2. That is, set $C \leftarrow \mathsf{E}_{II}(\alpha, \mathtt{ipk}) \otimes I$. Note that if the SU cannot create harmful interference to IU, then $I = [\![1_{\mathbb{G}_T}]\!]_{II}$ and hence C can be decrypted to α. Otherwise, I is the ciphertext of some random number and decryption of C just results in a random number.

Finally, SAS sets the potential license $\texttt{cred}_P = (C, \mathsf{E}_{AES}(\texttt{cred}, \texttt{k}))$ and sends it to the SU.

6.2.12 License Recovery

Upon receiving a potential spectrum license $\texttt{cred}_P = (C, \texttt{str})$, an SU b recovers a license by running algorithm Rec, which proceeds as follows:

- Using the re-encryption key $\texttt{rk}_b$ obtained during its registration process with the Key Distributor (See section III.B), SU b re-encrypts C to be a first level cipher text encrypted by $\texttt{pk}_b$: $C^* \leftarrow \mathsf{R}(C, \texttt{rk}_b)$.

- SU b decrypts C^* and recovers an AES key, and then obtains the recovered spectrum license using that key:

$$k^* \leftarrow H(\mathsf{D}_I(C^*, \texttt{sk}_b)),$$
$$\texttt{cred}^* \leftarrow \mathsf{D}_{AES}(\texttt{cred}, k^*).$$

Afterwards, the SU uses the signature part of $\texttt{cred}^*$ to check if $\texttt{cred}^*$ is a valid license. If the validation fails, it means that $I \neq 1_{\mathbb{G}_T}$ (i.e. SU can create harmful interference). Only when $\texttt{cred}^*$ has been successfully validated, can the SU access the spectrum from a location inside the set $\mathcal{L}$ indicated in the license.

6.3 Security Definitions and Analysis

In this section, we provide formal definitions and proof of PeDSS's security properties.

6.3.1 Correctness

We denote the whole PeDSS functionality as a function f:

$$\texttt{cred}^* := f(\mathcal{R}, \mathcal{E}, A, \texttt{req}), \tag{6.13}$$

where the $\mathcal{R}$ is the set of all received IU presence reports and $\mathcal{E}$ is the set of all expired IU presence reports. The formal definition of correctness is given as follows:

Definition 6.2. Denote SU's true location as l_{SU} and its spectrum request as $\texttt{req} = (l^*_{SU}, \epsilon, h_{SU}, f_{SU}, P_{\max})$. PeDSS is correct if for any input $(\mathcal{R}, \mathcal{E}, A, \texttt{req})$ to PeDSS functionality and an IU parameter combination (l, h, f, γ) such that

$$P_{\max} - A(l_{SU}, h_{SU}, f_{SU}, l, h, f) \geq \gamma$$
$$\exists \mathbf{R} \in \mathcal{R}, \mathbf{R} \notin \mathcal{E} \text{ s.t. } \mathbf{R}_{l,h,f,\gamma} \neq 1_{\mathbb{G}_T},$$

either the signature of $\texttt{cred}^* := f(\mathcal{R}, \mathcal{E}, A, \texttt{req})$ is invalid or $l_{SU} \notin \mathcal{L}$. ($\mathcal{L}$ is the location set in $\texttt{cred}^*$)

Theorem 6.3. *The probability with which PeDSS is NOT correct is negligible.*

Proof. If $l_{SU} \notin \mathcal{L}$, then PeDSS is correct.

Now assume $l_{SU} \in \mathcal{L}$. Define $\mathbf{R}_{l,h,f,\gamma} = Q \neq 1_{\mathbb{G}_T}$ for some $Q \leftarrow\!\!\$\, \mathbb{G}_T \backslash \{1_{\mathbb{G}_T}\}$. Note that

$$[\![\mathbf{D}_{l,h,f,\gamma}]\!]_{II} = \bigotimes_{\mathbf{R} \in \mathcal{R}, \mathbf{R} \notin \mathcal{E}} [\![\mathbf{R}_{l,h,f,\gamma}]\!]_{II}, \tag{6.14}$$

and $I = \otimes_{(l,h,f,\gamma) \in \mathcal{U}} [\![\mathbf{D}_{l,h,f,\gamma}]\!]_{II}$. Hence, I is essentially homomorphic multiplication of $Q \neq 1_{G_T}$ with several items which are either $[\![1_{\mathbb{G}_T}]\!]_{II}$ or some other random elements. Hence, the

probability with which $I = [\![1_{\mathbb{G}_T}]\!]$, is negligible. For example, for 80 bit security level, the probability is 2^{-80}. Therefore, the probability with which the recovered key is correct (i.e. $k^* = k$) and the signature of $\texttt{cred}^*$ can be verified as valid is negligible. This demonstrate that the chance that PeDSS is incorrect is negligible. □

6.3.2 Privacy for IU

To formally define IU privacy property, we setup a security experiment, which is shown in Figure. 6.3. In the privacy experiment, the adversary $\mathcal{A}$ is required to respond a challenge by showing its ability to distinguish two different IU presence data patterns. The adversary $\mathcal{A}$ submits two actually IUs presence data sequences arbitrarily on his own choice, and then the challenger will pick a random one and generate the all corresponding data reports, based on which the adversary will output a single bit indicating the guess on which sequence is picked by the challenger. In addition, we also allow the adversary $\mathcal{A}$ to query a $\mathsf{CrptReg}$ oracle, which implies the adversary can corrupt SU registration channel.

The formal definition of IU privacy is shown as follows:

Definition 6.4. PeDSS is private for IUs if for all $\lambda \in \mathbb{N}$, the advantage $\mathsf{Adv}_{\mathcal{A}}^{\text{Priv}}(\lambda)$ is negligible in λ for all Probabilistic Polynomial-Time (PPT) Adversaries $\mathcal{A}$, where

$$\mathsf{Adv}_{\mathcal{A}}^{\text{Priv}}(\lambda) = \left| \Pr\left[\mathbf{Exp}_{\mathcal{A}}^{\text{Priv}}(\lambda) = 1\right] - \frac{1}{2} \right|$$

Theorem 6.5. *If AFGH scheme is semantically secure, PeDSS is private for IUs.*

Proof. To prove PeDSS is private for IUs, we assume that there exists an adversary $\mathcal{A}$ that can break IU privacy with non-negligible probability. Then, we construct a simulator $\mathcal{S}$ that aims at breaking the semantic security of AFGH scheme by taking advantage of the

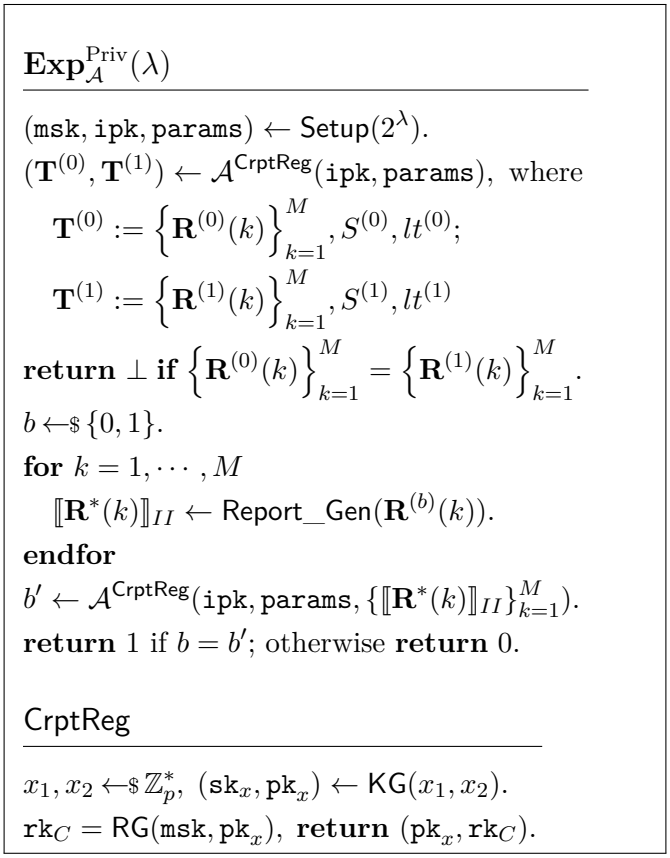


Figure 6.3: Definition of privacy experiment

adversary $\mathcal{A}$. $\mathcal{S}$ plays as the adversary in a given AFGH semantic security experiment, yet meanwhile it sets up a simulated PeDSS privacy experiment to interact with $\mathcal{A}$. Finally, it can leverage the response from $\mathcal{A}$ to gain a non-negligible advantage in AFGH semantic security experiment.

$\mathcal{S}$ set up the the simulated PeDSS privacy experiment as follows. Firstly $\mathcal{S}$ sets $\mathtt{msk} \leftarrow \mathtt{sk}_B$ and $\mathtt{ipk} \leftarrow \mathtt{pk}_B$. Here $\mathtt{msk}$ is actually unknown by $\mathcal{S}$. Let $\mathtt{pk}_h = (Z^{h_1}, g^{h_2})$ and $\mathtt{sk} = (B_1, B_2)$. Then to simulate the corruption oracle $\mathsf{CrptReg}$, $\mathcal{S}$ selects $r_1, r_2 \leftarrow\!\!\$\, \mathbb{Z}_p^*$ and compute $\mathtt{pk}_x \leftarrow ((Z^{h_1})^{r_1}, (g^{h_2})^{r_2})$ and $\mathtt{rk}_C \leftarrow (\mathtt{rk}_{B\to h})^{r_2}$. In this case $(\mathtt{sk}_x, \mathtt{pk}_x)$ is a randomly valid key pair where $\mathtt{sk}_x = (h_1 r_1, h_2 r_2)$, and $\mathtt{rk}_C = g^{B_1 h_2 r_2} = g^{B_1 (h_2 r_2)}$ is generated correctly. Hence $\mathsf{CrptReg}$ is perfectly simulated. Afterwards, $\mathcal{S}$ submits $m_0 = 1_{\mathbb{G}_T}$ and $m_1 \leftarrow\!\!\$\, \mathbb{G}_T \backslash \{1_{\mathbb{G}_T}\}$ to AFGH semantic security experiment defined in [4]. It obtains $C := \mathsf{E}_{II}(\mathtt{pk}_B, m_b) \in \mathbb{G}_T \times \mathbb{G}_1$. Let $C = (c_1, c_2)$, where $c_1 = Z^{sB_1} m_b$ and $c_2 = g^s$. Upon receiving $\mathbf{T}^{(0)}$ and $\mathbf{T}^{(1)}$ submitted by $\mathcal{A}$, $\mathcal{S}$ skips the procedure of selecting b and generates simulated $\{[\![\mathbf{R}^*(k)]\!]_{II}\}_{k=1}^{M}$ as follows. For any $k = 1, \cdots, M$, if $\mathbf{R}^{(0)}(k)_{l,h,f,\gamma} = \mathbf{R}^{(1)}(k)_{l,h,f,\gamma}$, then $\mathcal{S}$ computes the value of $[\![\mathbf{R}^*(k)_{l,h,f,\gamma}]\!]_{II}$ faithfully according to algorithm $\mathsf{Report\_Gen}(\cdot)$. Otherwise, $\mathcal{S}$ selects $r \leftarrow\!\!\$\, \mathbb{Z}_p^*$ and sets $[\![\mathbf{R}^*(k)_{l,h,f,\gamma}]\!]_{II} \leftarrow \otimes_{i=1}^{r} C$ if $\mathbf{R}^{(0)}(k)_{l,h,f,\gamma} = 0$; $\mathcal{S}$ sets $[\![\mathbf{R}^*(k)_{l,h,f,\gamma}]\!]_{II} \leftarrow ((c_1/m_1)^r, c_2^r)$ if $\mathbf{R}^{(0)}(k)_{l,h,f,\gamma} = 1$. Finally, $\mathcal{S}$ outputs 0 in AFGH semantic security experiment if $\mathcal{A}$ outputs 0; otherwise $\mathcal{S}$ outputs 1.

In the AFGH semantic security experiment, if b is selected as 0, then $[\![\mathbf{R}^*(k)_{l,h,f,\gamma}]\!]_{II} = ((Z^{sB_1} m_b)^r, (g^s)^r) = (Z^{(sr)B_1}, g^{sr})$ when $\mathbf{R}^{(0)}(k)_{l,h,f,\gamma} = 0$; $[\![\mathbf{R}^*(k)_{l,h,f,\gamma}]\!]_{II} = ((Z^{sB_1} m_b/m_1)^r, (g^s)^r)) = (Z^{(sr)B_1} m_1^{-r}, g^{sr})$ when $\mathbf{R}^{(0)}(k)_{l,h,f,\gamma} = 1$. Therefore in this case $\mathcal{S}$ simulates $\mathbf{R}^*(k)$ perfectly in the case of setting b as 0 in the PeDSS privacy experiment.

On the other hand, if b is selected as 1, then we have $[\![\mathbf{R}^*(k)_{l,h,f,\gamma}]\!]_{II} = ((Z^{sB_1} m_b)^r, (g^s)^r) = (Z^{(sr)B_1} m_1^r, g^{sr})$ when $\mathbf{R}^{(0)}(k)_{l,h,f,\gamma} = 0$; when $\mathbf{R}^{(0)}(k)_{l,h,f,\gamma} = 1$, we have $[\![\mathbf{R}^*(k)_{l,h,f,\gamma}]\!]_{II} = ((Z^{sB_1} m_b/m_1)^r, (g^s)^r)) = (Z^{(sr)B_1}, g^{sr})$. Therefore in this case $\mathcal{S}$ also simulates $\mathbf{R}^*(k)$ per-

fectly in the case of setting b as 1 in the PeDSS privacy experiment. Hence we conclude that the advantage for breaking AFGH semantic security is the same as the advantage for breaking PeDSS privacy.

□

Claim 2. Under EDBDH assumption, PeDSS is private for IUs.

Proof. According to theorem 3.1 in [4], AFGH is semantically secure under EDBDH assumption, so PeDSS is private for IUs under EDBDH assumption. □

6.3.3 Location Privacy for SU

In PeDSS, the concept of geo-indistinguishability [3] can be adopted to formally define location privacy for SUs. Intuitively, for a curious SAS, if an SU at two locations l_{SU}, l'_{SU} generates the same fuzzy location l^*_{SU} with similar probabilities, then the SAS holds little information about whether the true location is l_{SU} or l'_{SU}.

Geo-indistinguishability formalizes the above notion, since it is a generalized version of differential privacy using the Euclidean metric to measure the distance between secrets [3]. When a curious SAS receives a fuzzy location l^*_{SU} and attempts to distinguish the true location of an SU between l_{SU} and l'_{SU} within a circle with radius r, ϵ-geo-indistinguishability requires the likelihood ratio is lower than $e^{-\epsilon r}$, where ϵ is the parameter of differential privacy and r can be any radius value smaller than the radius of service area.

The formal definition of SU geo-indistinguishability is given as follows[2]:

Definition 6.6. PeDSS satisfies ϵ-geo-indistinguishability if and only if for any received

[2]There are three equivalent definitions proposed in [3], and in this paper we adopt the third one.

fuzzy location l^*_{SU}:

$$\frac{P(l^*_{SU}|l_{SU})}{P(l^*_{SU}|l'_{SU})} \leq e^{\epsilon r} \qquad \forall r > 0 \; \forall l_{SU}, l'_{SU} : d(l_{SU}, l'_{SU}) \leq r. \tag{6.15}$$

Theorem 6.7. *PeDSS preserves ϵ'-geo-indistinguishability for SUs in the service area with diameter D, with*

$$\epsilon' = \frac{2}{r_{SU}} + \frac{1}{u} \ln \frac{q - 2 + 3e^{2v\sqrt{2}/r_{SU}}}{q - 5}, \tag{6.16}$$

where u, v is the smaller and larger value of reported location precision in Cartesian coordinates; q is a parameter obtained $q = \frac{u}{D\delta_\theta}$, where δ_θ is the precision of angle.

Proof. From the analysis in [3] a mechanism drawing fuzzy locations from polar Laplacian distribution with parameter ϵ preserves ϵ-geo-indistiguishabiltiy. However, locations reported to SAS have precisions and thus are actually discrete values. From Theorem 4.1 in [3], within a given diameter $r_{\max} := \frac{u}{q\delta_\theta}$, PeDSS provides ϵ'-geo-indistinguishability for SUs in the service area with diameter D, with

$$\begin{aligned} \epsilon' &= \epsilon + \frac{1}{u} \ln \frac{q - 2 + 3e^{\epsilon v\sqrt{2}}}{q - 5} \\ &= \frac{2}{r_{SU}} + \frac{1}{u} \ln \frac{q - 2 + 3e^{2v\sqrt{2}/r_{SU}}}{q - 5}. \end{aligned} \tag{6.17}$$

Note that $r_{\max} = \frac{u}{q\delta_\theta} = D$, so ϵ'-geo-indistinguishability for SUs is preserved in the diameter D. □

6.4 Evaluation

In this section, we evaluate the efficiency and accuracy of PeDSS.

6.4.1 Implementation Details

To instantiate a secure Type 1 pairing, we utilized the pairing with "A-internal" described in [49]. The security level is symmetric 80 bits, which provides approximately the same level of security as an RSA signature with a modulus size of 1024 bits. By using the pairing-based cryptography (PBC) library available at [49], we implemented the AFGH scheme and the homomorphic operation algorithms.

To evaluate PeDSS, we set the service area to be a 405 km^2 area in Washington D.C.. We employed L-R model using SPLAT![1] to calculate the attenuation map in this area, where real world terrain data obtained from USGS and SRTM3 was used. The IU interference sensitivity level was set as -80 dBm, which is the sensitivity for a general military radar. We splitted this urban area of Washington D.C. into 36×45 grids with a side length of 500m. All the experiments were conducted on a laptop with Intel i7-4770 CPU @ 3.4GHz.

We compared the performance of PeDSS with two MPC-based privacy preserving system: P^2-SAS proposed in [25], and PISA proposed in [37]. We selected these two works as the benchmark because of the similarity that both PeDSS and them address privacy for both IUs and SUs under protection zone model with the assumption that SAS is not fully trusted.

6.4.2 Computational Overhead

Table 6.1 compares the computational overhead of PeDSS with P^2-SAS and PISA. We simulated 5000 random spectrum requests to evaluate PeDSS and utilized the benchmark of Paillier cryptosystem to evaluate P^2-SAS and PISA under *the same scale of network and the same resolution for discretization*.

From Table 6.1, we observe that in terms of spectrum allocation, PeDSS is three orders

Table 6.1: Computational overhead comparison between PeDSS, P^2-SAS and PISA

	PeDSS	P^2-SAS	PISA
IU: report generation	2.221 s	197.5 s	2.16 s
SAS: report update	34.36 ms	26.03 ms	70.20 ms
SU: request generation	0.26 ms	197.5 s	5.97 s
SAS: spectrum allocation	0.51 ms	3.48 s	5.91 s

of magnitude faster than P^2-SAS and PISA. PeDSS achieves advantage in computational overhead on spectrum allocation as a result of the novel construction that leverages homomorphic proxy re-encryption, where SAS does not have to aggregate all items in the encrypted database.

Compared to P^2-SAS, PeDSS is more than one order of magnitude faster in terms of generating IU report and performs similarly in integrating one IU report into the database. This is due to the difference between AFGH and Paillier cryptosystem. PeDSS does not achieve significant advantage over PISA in terms of generating IU report, since PISA is tailored for DSA system in UHF TV band, where some IU operational data are assumed to be known by the SAS, including the location, antenna height, and transmission power. Hence in PISA IUs are sending a smaller matrix to SAS under the same scale of network.

The computational overhead of generating an SU request in PeDSS is about two orders of magnitude faster than P^2-SAS. This is achieved by the incorporation of differential privacy in PeDSS, so that SU does not generate a whole encrypted matrix to hide its actual location.

6.4.3 Communication Overhead

Table 6.2 compares the communication overhead of PeDSS with P^2-SAS and PISA. For the spectrum license `cred` in PeDSS, each field of the message is assumed to be 32 bits long. The SAS certificate $\texttt{cert}_{SAS}$ is assumed to be a X.509 certificate. The length of $\texttt{cert}_{SAS}$ is

set to be 1888 bytes, which is the same as the site certificate of Google. The signature σ in `cred` is a 512 bit ECDSA signature.

Table 6.2: Communication overhead comparison between PeDSS, P^2-SAS and PISA

	PeDSS	P^2-SAS	PISA
IU report (single IU)	297.9 KB	3.174 MB	50.2 KB
SU request	3.65 KB	1.19 MB	783.0 KB
SAS response	3.95 KB	4.01 KB	4.01 KB

In PeDSS, the size of a single IU report is about one order of magnitude smaller than P^2-SAS. PeDSS achieves the advantage since the bilinear pairing cryptosystem of AFGH is constructed on elliptic curves, while the Paillier cryptosystem is based on number theory and consumes larger spaces under the same level of security. One can also observe that in PISA, the size of IU report is smaller than the one in PeDSS. This is because in PISA smaller matrix are sent to SAS under the same scale of network; as we mentioned above, PISA is customized for UHF TV band, where the privacy of IU locations, antenna height, and transmission power are not protected.

Note that in PeDSS, the size of a spectrum request is more than two orders of magnitude smaller than ones in P^2-SAS and PISA. This is also achieved by the incorporation of differential privacy, where SU does not need to send a whole matrix to SAS to hide its actual location.

6.4.4 Accuracy

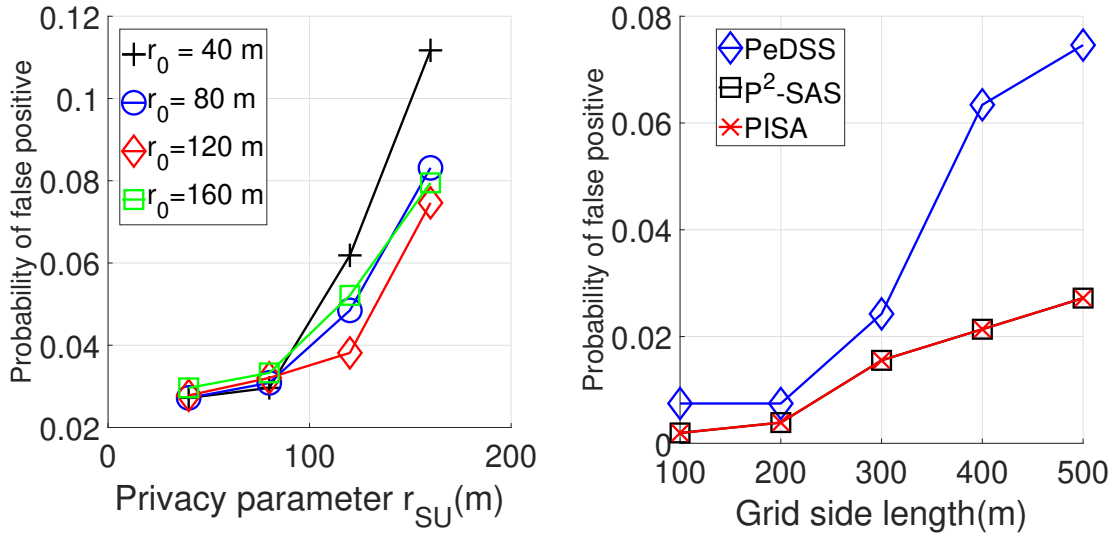
There are two types of errors in a general DSA system: false positive and false negative. A false positive error means that an SU does not get access to the spectrum although it will not generate harmful interference to IUs, and a false negative error means that an SU gets a a valid spectrum license although it will generate harmful interference to IUs. False

positives result in lost spectrum access opportunity, which is undesirable but tolerable. False negatives create interference to IUs, which violates the FCC DSA rules and should be avoided whenever possible. Because of the correctness feature of PeDSS, there is no false negative errors *assuming the measurements and attenuation calculation is accurate*. Recall that the attenuation map records the minimum attenuation values for those discretized parameters, so the discretization can only overestimate the interference and never underestimate interference (if we neglect the errors of L-R model itself). Therefore, the only possible false negatives are introduced by errors of L-R model. Yet, performing a fine-grained measurements over a 405 km^2 area to evaluate the accuracy of L-R model is beyond the scope of this paper. Compared to other privacy-preserving techniques that also introduce location fuzziness (e.g [12][76][75]), PeDSS achieves the advantage of negligible false negatives.

In PeDSS, there are two factors contributing to the false positive error: stretching error and discretization error. This is because both stretching and discretization can lead to overestimation of SU interference. Figure. 6.4a shows the false positive error of PeDSS, where r_{SU} is the mean distance between the fuzzy location submitted by an SU and the true location of the SU. From the figure we can observe the trade-off between false positive and SU privacy. When privacy parameter r_{SU} is set as 160m, parameter $\epsilon = 2/r_{SU} = 0.0125$ and from the CDF function $C_\epsilon(r)$, 97% fuzzy locations are in an radius of 428.4m. In this case PeDSS achieves false positive rate of 7.46% by setting stretch radius $r_0 = 120$m. Therefore, PeDSS incurs small false positive error by setting a proper stretch radius r_0.

By comparing different curves in Figure. 6.4a, we also observe that when stretching radius r_0 is set to be 120m, PeDSS achieves lower false positive than the case where stretching radius r_0 is set to be 80m or 160m. Intuitively stretching radius r_0 should not be too large since larger r_0 leads to more interference overestimation. Meanwhile, if the stretching radius r_0 is too small, then SAS will possibly not cover the true location of SU; as a result, the license

will not be valid and a false positive error happens.



(a) False positive vs. privacy parameter r_{SU}. (b) False positive vs. grid side length.

Figure 6.4: False positive evaluation.

Figure. 6.4b compares the false positive error between PeDSS, P^2-SAS and PISA. For PeDSS, privacy parameter r_{SU} is set as 160m, and stretch radius r_0 is set as 120m. The false positive probability of PISA and P^2-SAS shows the errors introduced by discretization, and the gap of false positive probability between PeDSS and P^2-SAS/PISA shows the errors introduced by SAS location stretching of PeDSS.

Chapter 7

CuMAC: Cumulative Message Authentication Codes for Resource-Constrained Networks

In this chapter, we propose a novel approach for message authentication that we refer to as *Cumulative Message Authentication Code* (CuMAC). In CuMAC, a sender utilizes a procedure called *aggregation* through which it divides each MAC output into multiple short segments, and aggregates the MAC segments of multiple messages to form a short authentication tag. On the other hand, a receiver utilizes a procedure called *accumulation* through which it "accumulates" the cryptographic strength of the underlying MAC by collecting the relevant tags carried in the received packets. During the accumulation procedure, the receiver incurs delay that is proportional to the accumulated cryptographic strength since it needs to wait for the relevant tags to be received and processed. In essence, CuMAC makes a favorable trade-off between security and latency—i.e., cryptographic strength can be increased, if needed, at the cost of increased latency in verifying the received message's authenticity and integrity.

7.1 Overview of CuMAC

In CuMAC, we utilize two key procedures: *aggregation* at the sender, and *accumulation* at the receiver. Aggregation refers to the procedure of generating one authentication tag from the short segments of multiple MACs, and it enables the sender to reduce the communication overhead of the authentication tag in each packet. Accumulation refers to the procedure of verifying multiple segments of the MAC together, and it enables the receiver to accumulate the cryptographic strength of the underlying MAC by accumulating more MAC segments. The formal definition of CuMAC is as follows.

Definition 7.1. CuMAC comprises of the following algorithms.

1. $\mathtt{k} \leftarrow \mathsf{KeyGen}(1^\lambda)$: This key generation algorithm is run by the sender and the receiver. It takes as input the security parameter $\lambda \in \mathbb{N}$ (in unary). It generates and outputs the secret key denoted as $\mathtt{k}$.

2. $\sigma_i \leftarrow \mathsf{MacGen}(\mathtt{k}, m_i)$: This MAC generation algorithm is run by the sender and the receiver. It takes the secret key $\mathtt{k}$, and the message represented by m_i as inputs. It generates and outputs the L bits long MAC represented by σ_i.

3. $\tau_i \leftarrow \mathsf{SegAgg}(\mathtt{segArray})$: This segment aggregation algorithm is run by the sender and the receiver. It takes as input a two-dimensional array of MAC segments, represented by $\mathtt{segArray}$, where each MAC segment is l bits long. It extracts n elements of the array $\mathtt{segArray}$, and computes and outputs the l bits long authentication tag τ_i. Here, n represents the number of segments in which each underlying MAC of L bits is divided, i.e., $L = n \cdot l$.

4. $\tau_i \leftarrow \mathsf{TagGen}(\mathtt{k}, m_i)$: This tag generation algorithm is performed by the sender. It takes as inputs the secret key $\mathtt{k}$ and the message m_i. It utilizes an array of MAC segments

of previously transmitted messages which is represented by `segTx`. The array `segTx` is stored and maintained by the sender. This algorithm runs the MacGen algorithm followed by the SegAgg algorithm to generate the authentication tag τ_i.

5. `valid/invalid` $\leftarrow$ TagVerify($\mathtt{k}, m_i, \tau_i$): This tag verification algorithm is run by the receiver. It takes as inputs the secret key `k`, the received messages m_i, and the received tag τ_i. It utilizes an array of MAC segments of previously received messages which is represented by `segRx`, and an array of number of verified segments which is represented by `accRx`. The arrays `segRx` and `accRx` are stored and maintained by the receiver. This algorithm verifies whether the tag τ_i is generated using the secret key `k`. If the verification succeeds, it outputs the value `valid`; otherwise, it outputs the value `invalid`.

In CuMAC, we assume that the sender and the receiver share a secret key `k` generated using the KeyGen algorithm. We also assume that the sender and receiver share a counter i, and they agree on the method to increment the counter i after communicating the i^{th} packet. Note that when the sender and the receiver are synchronized using the counter i, the elements in the segments arrays `segTx` at the sender and `segRx` at the receiver are the same.

Figure 7.1 illustrates the major steps performed by the sender in CuMAC. In the TagGen algorithm, the sender computes the authentication tag through two major steps. In the first step, the sender generates the MAC of the message using the MacGen algorithm. It breaks the MAC into short segments, and stores them into a segment array `segTx`. Note that the segment array at the sender also stores the segments of the MACs of the previously transmitted messages. In the second step, the sender retrieves n segments (one MAC segment of the current message and $n-1$ MAC segments of the previously transmitted messages) from the segment array `segTx`, and aggregates the segments to generate an authentication tag

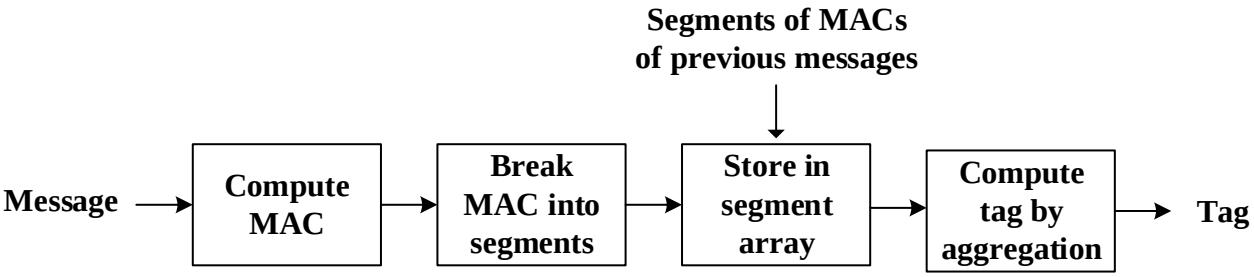


Figure 7.1: Schematic of the procedures at the sender in CuMAC.

using the SegAgg algorithm. The sender inserts a message and the associated authentication tag in a packet, and sends it to the receiver.

Having received the packet, the receiver runs the TagVerify algorithm which includes two major steps. In the first step, the receiver generates an authentication tag of the received message using the same secret key `k`. In the second step, the receiver compares the generated authentication tag with the received authentication tag. If they match, the receiver increments the number of accumulated segments of received MACs into an accumulation array `accRx`. The cryptographic strength of the MAC that authenticates message m_i is indicated by the i^{th} element of the accumulation array, which represents the number of accumulated segments of the MAC that covers message m_i.

In CuMAC, the counter i handles out-of-order delivery of packets [7]. The cases of missing tags due to lost packets are classified into two categories based on the awareness of the sender about the packet loss. In the first category of cases, before transmitting the next packet, the sender becomes aware that the current packet is lost since it has not received the acknowledgement from the receiver. In these cases, the sender decrements the value of counter i, and proceeds with the standard retransmission mechanism and the authentication scheme. In the second category of cases, the sender is not aware of the loss (e.g., in networks that only provide best effort services). In these cases, the sender utilizes the MAC segments of the lost message in $n-1$ forthcoming packets which means that one lost packet hinders

the verification of those $n-1$ forthcoming packets at the receiver. Hence, like compound MAC [55], trailing MAC [36] and aggregate MAC [42], CuMAC and CuMAC/S are not robust against a missing tag in the second category of cases. However, we highlight that in the targeted applications of these schemes, either there exists retransmission mechanism with acknowledgement (e.g., the CAN bus employs the mechanism where the receiver acknowledges the reception of the message [73]), or the probability of a missing tag is close to zero (e.g., the Sigfox network utilizes repetition of packets to ensure that 99.9 % packets are successfully communicated [64]).

7.2 Overview of Security Properties

We convey cryptographic strength in bits, where a cryptographic strength of λ bits for a scheme means that for any adversary making at most 2^{λ} queries or taking at most 2^{λ} time, the probability of successfully launching an attack on the scheme is negligibly small [7]. The cryptographic strength of a standard MAC depends on three security parameters: (1) the cryptographic strength of the underlying cryptographic primitive, (2) the size and quality of the secret key, and (3) the size of the MAC output, L. To achieve a cryptographic strength of λ bits, the minimum size of the key and the MAC output should be λ bits. In the following discussions, we present the security properties satisfied by CuMAC.

7.2.1 Authentication Levels

To compare CuMAC, and conventional MAC schemes, let us define three different levels of authentication: (1) real-time authentication, (2) full authentication, and (3) partially accumulated authentication. Figure 7.2 illustrates the three different levels of authentication,

when applied to message m_i. In this illustration, the MAC of the message m_i is distributed in tags $\tau_{i-n+1}, \cdots, \tau_{i+n-1}$. The receiver can perform real-time authentication immediately after receiving message m_i by processing the n segments received in tags $\tau_{n-i+1}, \cdots, \tau_i$. With real-time authentication, the receiver performs authentication without any delay, but it achieves the lowest cryptographic strength since there is no security accumulation using the subsequent tags. On the other hand, the receiver can perform full authentication after receiving all of the segments of the MAC associated with message m_i. With full authentication, the receiver achieves the highest cryptographic strength, but needs to incur a latency of $n-1$ packets. The receiver can perform partially accumulated authentication by accumulating and processing tags $\tau_{i-n+1}, \cdots, \tau_{i+r-1}$, where $1 < r < n$. Partially accumulated authentication enables the receiver to make a trade-off between cryptographic strength and message verification latency to meet the security and performance needs of the protocol.

In CuMAC, in the i^{th} packet, the MAC of message m_i is computed as σ_i, and divided into n segments. These segments are distributed in tags $\tau_i, \cdots, \tau_{i+n-1}$. In this case, the cryptographic strength of the full authentication depends on the same three aforementioned security parameters for the standard MAC. The cryptographic strength of the real-time authentication depends on the first two aforementioned security parameters, and the size of each MAC segment, l. The cryptographic strength of the partially accumulated authentication depends on the first two aforementioned security parameters, the size of each MAC segment, l, and the number of accumulated segments, r.

7.2.2 Security Definitions

Katz et al. provide the first concrete proof which illustrates that if multiple standard MACs with cryptographic strength of λ bits are aggregated by XOR operation to form an aggregate

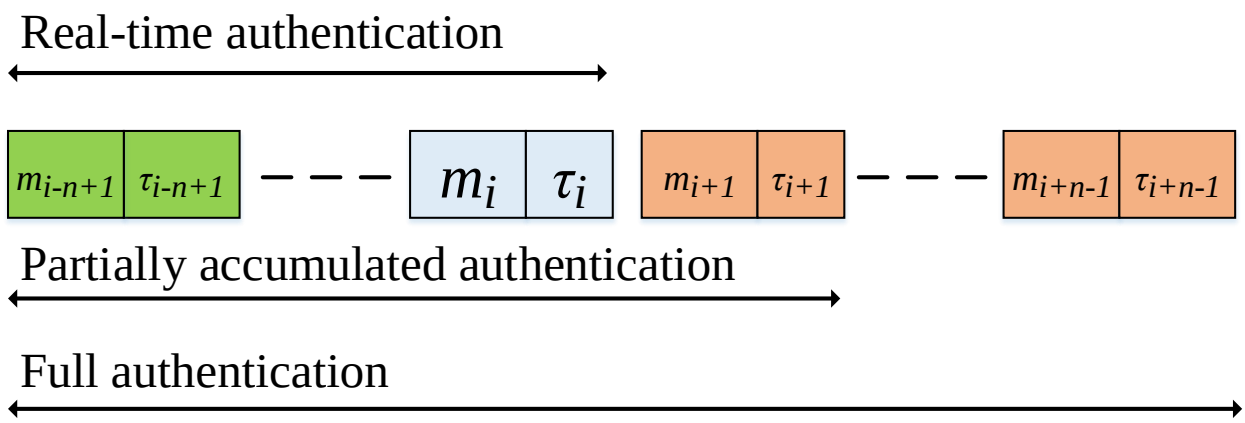


Figure 7.2: Illustration of three different levels of authentication.

MAC, then the aggregate MAC is secure with the cryptographic strength of λ bits [27, 42]. The aggregation procedure employed in CuMAC and CuMAC/S share similar attributes with the scheme proposed by Katz et al. Hence, in this paper, we present the security definitions and security proofs which closely follow those presented by Katz et al.

The security evaluation for CuMAC is centered around the notion of unforgeability under chosen message attack with parameter r (uf-cma-r), where r indicates the number of packets accumulated for tag verification. We denote by $\mathbf{Adv}^{\text{uf-cma-}r}_{\mathsf{CuMAC}}(\mathcal{A}, \lambda, q)$, the advantage of the adversary $\mathcal{A}$ in forging a message for a random key $\mathtt{k} \leftarrow \mathsf{KeyGen}(1^\lambda)$, where $\mathcal{A}$ can make q queries to the tag generating oracle of CuMAC $O_{\mathsf{CuMAC}}(\mathtt{k}, \cdot)$, and verification is performed after accumulating r segments of each MAC. CuMAC is considered to be secure if the advantage of the adversary $\mathcal{A}$ is negligibly small. Formally, the advantage can be expressed by the probability (represented by Pr[]) that the following experiment returns 1.

$\mathbf{Exp}^{\text{uf-cma-}r}_{\mathsf{CuMAC}}(\mathcal{A}, \lambda, q)$

$\mathtt{k} \leftarrow \mathsf{KeyGen}(1^\lambda)$

Invoke $\mathcal{A}^{O_{\mathsf{CuMAC}}(\mathtt{k}, \cdot)}$ who can make up to q queries to the tagging oracle of CuMAC $O_{\mathsf{CuMAC}}(\mathtt{k}, \cdot)$. $\mathcal{A}$ can query $O_{\mathsf{CuMAC}}(\mathtt{k}, \cdot)$ with n arbitrarily chosen messages and receive

their CuMAC tags in response.

$\mathcal{A}$ outputs a set of n pairs $(\{m_i\}_{i=1}^{n}, \{\tau_i\}_{i=1}^{n})$.

Return 1 if $\texttt{valid} \leftarrow \mathsf{TagVerify}(\texttt{k}, m_i, \tau_i)$ for all $1 \leq i \leq n$, and $\mathcal{A}$ did not make the query for m_{i^*} to $O_{\mathsf{CuMAC}}(\texttt{k}, \cdot)$, where $i^* = n - r + 1$.

Return 0 otherwise.

Definition 7.2. CuMAC is (t, q, ϵ, r)-uf-cma secure if for any probabilistic polynomial time (PPT) adversary $\mathcal{A}$ running in time t, $\Pr\left[\mathbf{Exp}^{\text{uf-cma-}r}_{\mathsf{CuMAC}}(\mathcal{A}, \lambda, q) = 1\right] \leq \epsilon$.

Note that if CuMAC are (t, q, ϵ, r)-uf-cma secure for all r, then they are also (t, q, ϵ)-uf-cma secure which is the standard notion of security for a MAC scheme. We utilize the aforementioned uf-cma-r security model to define the cryptographic strength for full authentication, partially accumulated authentication and real-time authentication. Note that in the uf-cma-r experiment, when the experiment returns a value of 1, it implies that $\mathcal{A}$ is able to forge a valid tag for a packet at which point the receiver has already accumulated r packets. Therefore, if CuMAC is (t, q, ϵ, n) secure, i.e., $r = n$, then CuMAC is secure in terms of full authentication. Similarly, if CuMAC is (t, q, ϵ, r) secure for all $2 \leq r \leq n - 1$, then the scheme is secure for partially accumulated authentication; and if CuMAC S is $(t, q, \epsilon, 1)$ secure, i.e., $r = 1$, then the scheme is secure in terms of real-time authentication. The security of CuMAC is based on the following assumption that defines the security of the underlying MAC algorithm [6].

Assumption 3. The underlying deterministic MAC algorithm, MacGen, is (t, q, ϵ)-uf-cma secure—i.e., the probability that an adversary will be successful in producing a forged tag after running for a polynomial time t and making q queries is negligible.

7.3 CuMAC: Cumulative Message Authentication Code

In this section, we present the technical details of the algorithms employed by CuMAC. We also provide an example that illustrates the generation of the tags and their verification.

7.3.1 Technical Details

$\mathtt{k} \leftarrow \mathsf{KeyGen}(1^{\lambda})$

This probabilistic key generation algorithm is utilized by the sender and the receiver to generate the secret key. The input to this algorithm is the security parameter $\lambda \in \mathbb{N}$. This algorithm outputs the secret key $\mathtt{k}$. The detailed discussion of this algorithm is out of scope of this paper.

$\sigma_i \leftarrow \mathsf{MacGen}(\mathtt{k}, m_i)$

This deterministic MAC generation algorithm is utilized by the sender and the receiver to compute the MAC of a message using the secret key. The inputs to this algorithm are the secret key $\mathtt{k}$, and the message m_i. A counter i is utilized in this algorithm as the tool to deter against the replay attack. This algorithm outputs the L bits long MAC represented by σ_i. This algorithm can be realized using a cipher-based (e.g., AES-CMAC) or a hash-based (e.g., SHA-3) MAC scheme. In this paper, we utilize the widely used AES-CMAC [6]. The detailed discussion of this algorithm is out of scope of this paper.

$\tau_i \leftarrow \mathsf{SegAgg}(\mathtt{segArray})$

This segment aggregation algorithm is run by the sender and the receiver to generate the authentication tag. It takes as input a two-dimensional array of MAC segments, represented by `segArray`. The `segArray` comprises of the MAC segments of the previous messages and

the current message. The i^{th} entry in `segArray` is generated as follows. The L-bit MAC σ_i is divided into n segments such that the size of each segment is l bits, i.e., $L = n \cdot l$. The j^{th} segment of σ_i is represented by s_i^j, and is extracted from σ_i as

$$s_i^j \leftarrow (\sigma_i)_{\downarrow[(j-1)\cdot l+1, j\cdot l]}. \tag{7.1}$$

This means that the bits in s_i^j correspond to the bits from $((j-1)\cdot l+1)^{\text{th}}$ bit to $(j\cdot l)^{\text{th}}$ bit in σ_i.

This algorithm extracts n elements of the array `segArray`, and computes the authentication tag τ_i as

$$\tau_i \leftarrow \bigoplus_{j=1, i-j+1>0}^{n} s_{i-j+1}^j, \tag{7.2}$$

where $\bigoplus$ represents the bit-by-bit XOR of the segments. Hence, the tag τ_i is computed by the bit-by-bit XOR of the first segment of the current MAC, s_i^1, and the segments of the previous MACs, i.e., s_{i-1}^2, s_{i-2}^3 and so on. This algorithm outputs the authentication tag τ_i.

$\tau_i \leftarrow \mathsf{TagGen}(\mathtt{k}, m_i)$

The tag generation algorithm is utilized by the sender to generate an authentication tag. It takes as inputs the secret key $\mathtt{k}$, and the message m_i. It also utilizes an array of MAC segments of transmitted messages, `segTx`, which is stored and maintained by the sender. This algorithm proceeds as follows.

1. Compute the MAC σ_i of the message m_i, i.e., $\sigma_i \leftarrow \mathsf{MacGen}(\mathtt{k}, m_i)$.

2. Divide the MAC into n segments, and append them in the array of MAC segments `segTx`.

3. Compute the current tag τ_i by aggregating the segments of the previous and current MACs, i.e., $\tau_i \leftarrow \mathsf{SegAgg}(\mathtt{segTx})$.

4. Output the tag τ_i.

$\mathtt{valid/invalid} \leftarrow \mathsf{TagVerify}(\mathtt{k}, m_i, \tau_i)$

This tag verification algorithm is utilized by the receiver to verify the authenticity of the received tag. The inputs to this algorithm are the secret key `k`, the received message m_i, and the received tag τ_i. It also utilizes an array of MACs of previously received messages `segRx`, and an array of the number of accumulated MAC segments `accRx`, which are stored and maintained by the receiver. The i^{th} entry in the array `accRx` is represented by r_i. To initialize the element r_i in the array `accRx`, the receiver sets $r_i = 0$. This algorithm verifies whether the tag τ_i is generated using the correct secret key `k` and the MAC segments in `segRx`. It proceeds as follows.

1. Compute the MAC σ_i of the message m_i, i.e., $\sigma_i \leftarrow \mathsf{MacGen}(\mathtt{k}, m_i)$.

2. Divide the MAC into n segments, and append them in the array of MAC segments `segRx`.

3. Compute the authentication tag $\widetilde{\tau}_i$ by aggregating the segments of the previous and current MACs, i.e., $\widetilde{\tau}_i \leftarrow \mathsf{SegAgg}(\mathtt{segRx})$.

4. If $\widetilde{\tau}_i = \tau_i$,

   (a) Increment the number of accumulated MAC segments in `accRx`, i.e., for each $t \in [i - n + 1, i]$, set $r_t = r_t + 1$.

   (b) Output the value `valid`.

5. Otherwise, if $\widetilde{\tau}_i \neq \tau_i$, output the value `invalid`.

Table 7.1: Example illustrating CuMAC with $L = 128$, $n = 4$, and $l = 32$.

Packet	**Previous MACs**	**Current MAC**	**Aggregation of MAC segments**	**Tag**
5	$\sigma_2, \sigma_3, \sigma_4$	σ_5	$s_2^4 \oplus s_3^3 \oplus s_4^2 \oplus s_5^1$	τ_5
6	$\sigma_3, \sigma_4, \sigma_5$	σ_6	$s_3^4 \oplus s_4^3 \oplus s_5^2 \oplus s_6^1$	τ_6
7	$\sigma_4, \sigma_5, \sigma_6$	σ_7	$s_4^4 \oplus s_5^3 \oplus s_6^2 \oplus s_7^1$	τ_7
8	$\sigma_5, \sigma_6, \sigma_7$	σ_8	$s_5^4 \oplus s_6^3 \oplus s_7^2 \oplus s_8^1$	τ_8

7.3.2 Illustration

Table 7.1 presents an example of CuMAC. In this example, the size of the tag in each packet is 32 bits (i.e., $l = 32$). The MAC is generated using the AES-CMAC algorithm. Hence, the size of the MAC is 128 bits (i.e., $L = 128$), which provides 128-bit cryptographic strength. Each MAC is divided into four segments (i.e., $n = 4$), each of size 32 bits. This means that the achievable cryptographic strength for real-time authentication and full authentication are 32 and 128 bits, respectively. To simplify the discussions, we limit the discussions to the packets which are involved in the authentication of the message transmitted in the *fifth* packet, m_5.

In this example, for each $i \in [2, 7]$, the MAC of the message m_i is computed as σ_i. The segments of σ_i are represented as $\{s_i^j : j \in [1, 4]\}$. In the fifth packet, the message m_5 is transmitted. The corresponding tag τ_5 is generated using the segments of the MACs of the three previously transmitted messages, σ_2, σ_3 and σ_4, and the MAC of the current message, σ_5. The tags τ_6, τ_7 and τ_8 are generated by the sender in a similar manner, and transmitted in the subsequent packets.

When the receiver receives the fifth packet with the message m_5 and the tag τ_5, it verifies the validity of the tag. Since tag τ_5 is generated using segment s_5^1, the successful verification of

tag τ_5 enables the real-time authentication of message m_5 with a cryptographic strength of 32 bits. Next, when the receiver receives the sixth packet with tag τ_6, it verifies its validity. Since tag τ_6 is generated using segment s_5^2, the successful verification of tag τ_6 enables the receiver to combine the segments s_5^1 and s_5^2 to "accumulate" cryptographic strength. This means that after the reception of the sixth packet, the receiver can verify the validity of m_5 with the cryptographic strength of 64 ($= 2 \times 32$) bits—this is an example of partially accumulated authentication. After the receiver has received and verified the tags $\tau_5, ..., \tau_8$ as being `valid`, it can combine the segments s_5^1, s_5^2, s_5^3 and s_5^4—which were contained in the tags τ_5, τ_6, τ_7 and τ_8, respectively—to "accumulate" cryptographic strength. This enables the receiver to carry out full authentication of m_5 with cryptographic strength of 128 ($= 4 \times 32$) bits.

7.4 Security Analysis

we present theorems and corresponding proofs of security for CuMAC. Let CuMAC be instantiated with parameters (l, n), i.e., each MAC is divided into n segments, each of length l bits. Note that in CuMAC, the receiver performs real-time authentication by setting $r = 1$, partially accumulated authentication by setting $1 < r < n$, and full authentication by setting $r = n$.

Theorem 7.3. *For any $t, q \in \mathbb{N}$ and $\epsilon > 0$, if the underlying deterministic MAC algorithm,* MacGen*, is (t, q, ϵ)-uf-cma secure, then CuMAC with parameters (l, n) is (t', q', ϵ', r)-uf-cma secure, where*

$$t' \approx t, \qquad q' = \frac{q - n + 1}{n}, \qquad \epsilon' = 2^{l(n-r)} \cdot \epsilon.$$

Proof. Let there be an adversary $\mathcal{A}$ which succeeds to create a forgery of an authentication tag for CuMAC with a non-negligible probability. We construct a simulator $\mathcal{S}$ that interacts with the adversary $\mathcal{A}$ and creates a forgery of a MAC for the MacGen algorithm with a non-negligible probability.

Let CuMAC and the MacGen algorithm utilize the same secret key $\mathtt{k}$ which is not known to the adversary $\mathcal{A}$. Also, let the MAC of a message in CuMAC be computed by a query to the tag generating oracle of underlying MAC, which is denoted as $O_{\mathsf{MacGen}}(\mathtt{k}, \cdot)$. In this way, $\mathcal{S}$ perfectly simulates $O_{\mathsf{CuMAC}}(\mathtt{k}, \cdot)$, and hence, the uf-cma-$r$ experiment. Suppose the uf-cma-r experiment for CuMAC returns 1 with the probability ϵ' in time t', where an adversary $\mathcal{A}$ outputs a valid forgery $(\{m_i\}_{i=1}^n, \{\tau_i\}_{i=1}^n)$ after q' queries to $O_{\mathsf{CuMAC}}(\mathtt{k}, \cdot)$ simulated by $\mathcal{S}$. To create a forgery of a MAC for the MacGen algorithm, the simulator $\mathcal{S}$ proceeds as follows.

For all $i \in [1, n]$ and $i \neq i^*$, the simulator $\mathcal{S}$ queries the $O_{\mathsf{MacGen}}(\mathtt{k}, \cdot)$ for the MAC of m_i, and obtains the corresponding σ_i. It divides each MAC into n segments as shown in equation (7.1). It recovers the MAC segments of the message m_{i^*} by removing the mask by the MAC segments of other messages as follows:

$$s_{i^*}^k \leftarrow \tau_{i^*+k-1} \oplus \bigoplus_{j=1, j\neq k}^{n} s_{i^*+k-j}^j. \tag{7.3}$$

Since $i^* = n - r + 1$, the simulator $\mathcal{S}$ cannot recover the segments $s_{i^*}^k$ with $k \geq r + 1$. Hence, it makes a random guess for the rest of the $n - r$ segments, such that $\widetilde{s}_{i^*}^k \leftarrow\!\!\$\, \{0,1\}^l$ for all $k \in [r + 1, n]$. Finally, to create the forgery for the underlying MAC algorithm, MacGen, it concatenates all the recovered segments and the guessed segments: $\sigma_{i^*} \leftarrow s_{i^*}^1 || s_{i^*}^2 \cdots || s_{i^*}^r || \widetilde{s}_{i^*}^{r+1} || \cdots \widetilde{s}_{i^*}^n$. This means that given a successful forgery of the authentication tag in CuMAC, the probability of creating the forgery of MacGen is $2^{-l(n-r)}$.

To achieve the forgery of MacGen as shown above, the simulator $\mathcal{S}$ conducts at most $n \cdot q'$

queries to the $O_{\mathsf{MacGen}}(\mathtt{k},\cdot)$ to reply the q' queries by $\mathcal{A}$ to $O_{\mathsf{CuMAC}}(\mathtt{k},\cdot)$. Also, the simulator $\mathcal{S}$ conducts $n-1$ queries to $O_{\mathsf{MacGen}}(\mathtt{k},\cdot)$ to obtain $\{\tau_i\}_{i=1,i\neq i^*}^{n}$. Therefore, if these exists an adversary $\mathcal{A}$ running in time t' and achieving $\Pr\left[\mathbf{Exp}^{\text{uf-cma-}r}_{\mathsf{CuMAC}}(\mathcal{A},\lambda,q')=1\right]\leq\epsilon'$, then it can be leveraged to create a forgery for the underlying MAC algorithm, MacGen, in time t' plus the time required to evaluate the equation (7.3), by making $nq'+n-1$ queries, and with probability $2^{-l(n-r)}\epsilon'$. Hence, if the underlying MAC algorithm, MacGen, is (t,q,ϵ)-uf-cma secure, then CuMAC is (t',q',ϵ',r)-uf-cma secure, where $t'\approx t$, $q'=\frac{q-n+1}{n}$, and $\epsilon'=2^{l(n-r)}\epsilon$. □

7.5 Evaluation

In this section, we evaluate the performance of CuMAC and compare them with the prior art in two different network applications. The simulation results presented here were generated using Matlab.

7.5.1 Need for a Short Tag in an Energy-Constrained Network

Simulation Setup

We consider an air quality monitoring system based on a Sigfox network which consists of a base station and multiple sensors nodes distributed over a large area [64]. The air quality data is collected by analyzing the quantity of specific particles in the samples of air. Each sensor node utilizes the Sigfox protocol to send the sensing data to the base station once in every hour [61]. The air quality data is examined at the base station and finally made available to the responsible authorities. In this application scenario, there are two important performance requirements: (1) each battery-operated sensor node needs to operate for a few

Table 7.2: Parameters utilized for computing the service life of a sensor node in a Sigfox network.

Battery capacity	8000 mAh × 3600 s/h = 28800 C
Sleep charge	1.3 μA × 86400 s/day = 0.11 C/day
Packet transmission rate	1 packet/h = 24 packets/day
Packet transmission without payload	0.20 C
Payload transmission	0.002 C/bit

years independently without any physical access which means that the network is energy-constrained; and (2) message authentication is needed to ensure secure collection of data. Note that in this scenario, the latency requirement is not stringent as the data is collected and analyzed at the base station with some inherent delay.

In this analysis, we evaluate the effect of appending an authentication tag in each packet on the service life of a sensing node. In this application scenario, the service life of the sensor node is equal to its battery life. To compute the battery/service life, we utilize the charge consumption data from a Sigfox compliant transceiver IC produced by ON Semiconductor [56]. Table 7.2 presents the parameters utilized in the computation of the service life. We employ two 1.5 V Alkaline C batteries connected in series. Each battery holds a charge of 8000 mAh. The transmission time in every hour is limited to 2 seconds, and hence the device is considered to be in sleep almost all the time. We ignore the battery self discharge in this calculation.

Results

Figure 7.3 presents the service life of a sensor node for different sizes of message and authentication tag. We observe that by appending authentication tags in transmitted packets, each sensor node consumes a significantly more energy on data transmission which short-

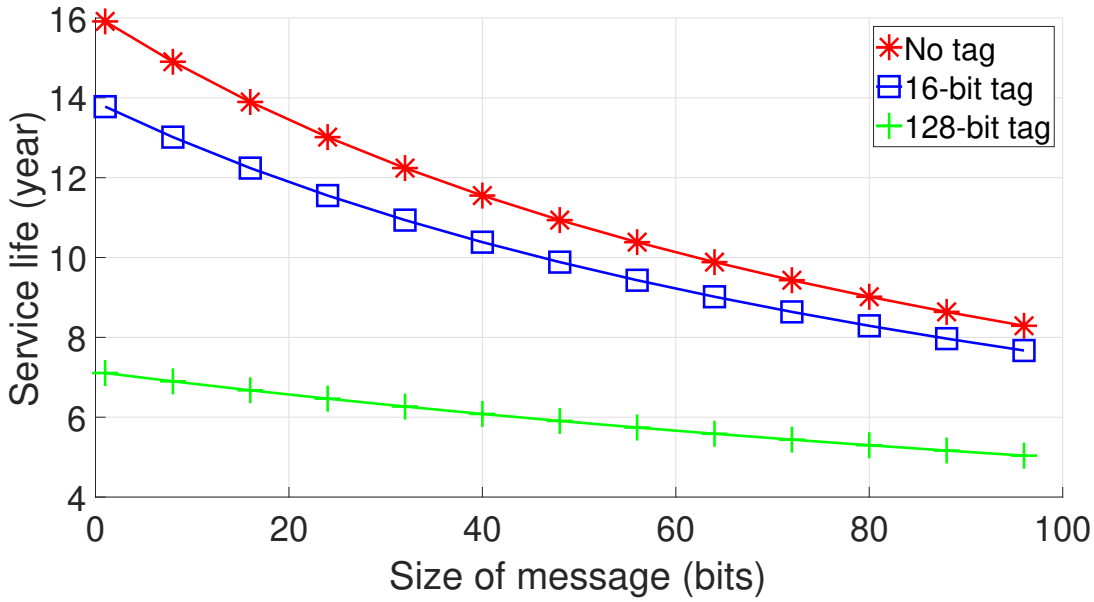


Figure 7.3: Effect of size of message and authentication tag on the service life of a sensor node in a Sigfox network.

ens the service life. Hence, the size of the tag in each packet is usually limited due to the energy constraints of the application. In this application scenario, we consider a 48-bit message without tag as the benchmark which results in the service life of around 11 years. Figure 7.3 illustrates that in comparison to this benchmark, utilizing 48-bit messages with the standard MAC with 128-bit tags results in a significant loss of around 45 % of service life. However, as compared to the benchmark, CuMAC or CuMAC/S can utilize the 16-bit tag in each packet without compromising the cryptographic strength of authentication, and with a modest (around 10 %) reduction in the battery/service life. Hence, for LPWANs like Sigfox, we assert that CuMAC or CuMAC/S is a much more viable solution for message authentication.

Table 7.3: Distribution of size and period of messages.

Size	1 byte	2 bytes	4 bytes	6 bytes
Share	35 %	49 %	13 %	3 %

Period	5 ms	10 ms	20 ms	50 ms	100 ms	200 ms	1000 ms
Share	7 %	25 %	25 %	3 %	20 %	1 %	19 %

7.5.2 Need for a Short Tag in a Bandwidth-Constrained Network

Simulation setup

We simulate a CAN bus to evaluate the bus load when the authentication tag along with the message is inserted in the CAN frames. Table 7.3 illustrates the distribution of the size and the period of messages utilized in the simulation. This distribution is based on the open-source benchmark presented by Kramer et al. [46]. The bus speed utilized in the simulation is 500 kbit/s. In the simulation, we let the maximum size of the message to be 6 bytes which means 2 bytes (16 bits) of tag can be readily inserted in the data field as shown in Figure 3.2. To utilize a 128-bit tag, each ECU needs to transmit two extra frames with the tag for each frame with the message. Recall that a maximum of 64 bits of tag can be transmitted in one CAN frame as shown in Figure 3.2.

We note that the bus load is a critical parameter for evaluating the latency performance of a CAN. Typically, the CAN bus load is between 30 % to 40 %, but with systematic approaches based on scheduling analysis, the bus load can be increased to around 80 % [23]. The bus load is directly proportional to the number of supported messages on the CAN bus. A high bus load may increase the latency of messages that may lead to problems, such as car functions being delayed and high possibility of communication fault situations [73]. Hence, it is critical to keep the bus load low.

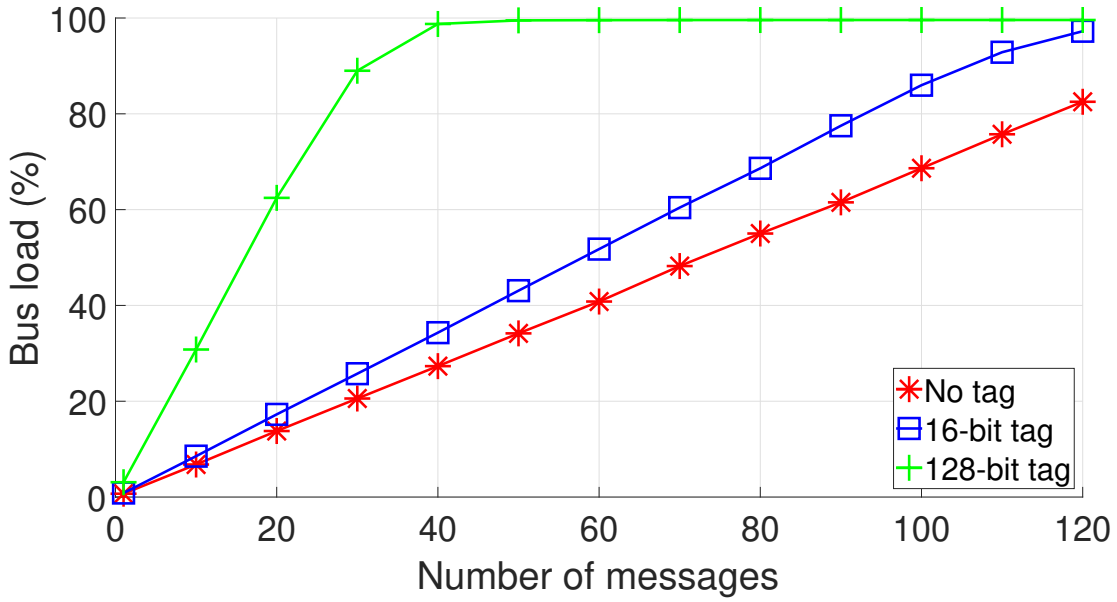


Figure 7.4: Effect of size of message and authentication tag on the CAN bus load.

Results

Figure 7.4 illustrates the effect of increasing the number of messages and inserting authentication tags in the CAN frame on the CAN bus load. We observe that at a typical bus load of 40 %, the number of supported messages without authentication is 60. While maintaining the same bus load, CuMAC or CuMAC/S with a 16-bit tag is able to support 45 messages, but a standard MAC with a 128-bit tag supports only 12 messages. Further, considering the maximum bus load of 80 %, the maximum number of messages supported by the bus with a 16-bit tag is 91, but that with a 128-bit tag is only 27. This means that to support 91 messages, a vehicle needs only one CAN bus when the messages are authenticated using CuMAC or CuMAC/S, but it needs three CAN buses when the messages are authenticated using a standard MAC scheme. Note that increasing the number of CAN buses increases the overall cost of the vehicle.

Chapter 8

CuMAC/S: Cumulative Message Authentication Codes with Speculation for Resource-Constrained Networks

In this chapter, we propose a variant of CuMAC called *CuMAC with Speculation* (*CuMAC/S*) that enables a receiver to accumulate the MAC's cryptographic strength while incurring minimal delay. The core concept of CuMAC/S is motivated by the technique of *speculative execution*[1] which is widely employed in modern computer systems [10, 54]. CuMAC/S can be utilized in latency-sensitive applications where future messages can be predicted correctly with high reliability using the current and past messages along with an appropriate speculation model. In CuMAC/S, a sender speculates future messages, computes the corresponding MACs, and aggregates the MAC segments of the speculated messages into the authentication tag of the current packet. If the speculated value of a received message is correct and equal to the actual value, a receiver is able to accumulate the tags from the previous packets (containing MAC segments of the speculated value) and the current packet (containing a

[1]Speculative execution is an optimization technique in which a computer system performs speculative execution where some outcome is predicted, and execution proceeds along a predicted path. Work is done before it is known whether it is actually needed, so as to prevent a delay that would have to be incurred by doing the work after it is known that it is needed.

MAC segment of the actual value). In this way, CuMAC/S enables the receiver to *accumulate* cryptographic strength without having to wait for tags included in forthcoming packets; this significantly cuts down on the MAC verification delay.

8.1 Overview

We propose CuMAC/S for delay-sensitive applications in which the delay incurred in the accumulation procedure of CuMAC is not acceptable. For CuMAC/S to be viable, the messages need to be time-correlated, and thus future messages can be predicted using an appropriate model. The most distinguishing attribute of CuMAC/S is *speculation*, which refers to the concept of predicting future messages based on the current and previous messages. This attribute enables a receiver to authenticate a message received in the current packet with an "accumulated" level of cryptographic strength without having to wait for subsequent packets. CuMAC/S utilizes three key procedures: speculation at the sender and the receiver, aggregation at the sender, and accumulation at the receiver. The formal definition of CuMAC/S is as follows.

Definition 8.1. CuMAC/S is composed of the following algorithms.

1. $\mathtt{k} \leftarrow \mathsf{KeyGen}(1^\lambda)$: It is defined in the same manner as the KeyGen algorithm in Definition 7.1.

2. $\sigma_i \leftarrow \mathsf{MacGen}(\mathtt{k}, m_i)$: It is defined in the same manner as the MacGen algorithm in Definition 7.1.

3. $\mathtt{msgArray}' \leftarrow \mathsf{MsgSpec}(\mathtt{msgArray})$: This message speculation algorithm is run by the sender and the receiver. It takes an array of messages `msgArray` as input. It generates the predicted value of the future message. It updates the array of messages and outputs

the updated array $\texttt{msgArray}'$. Here, we assume the speculation model is deterministic, and the corresponding parameters are pre-trained and shared between the sender and receiver.

4. $\tau_i \leftarrow \mathsf{SegAgg}(\texttt{segArray})$: This aggregation algorithm is run by the sender and the receiver. It takes as input a two-dimensional array of MAC segments `segArray`. It extracts $2n-1$ elements of the array `segArray`, and computes and outputs the authentication tag τ_i.

5. $\tau_i \leftarrow \mathsf{TagGen}(\texttt{k}, m_i)$: This tag generation algorithm is run by the sender. It takes as inputs the secret key `k`, and a messages m_i. It utilizes an array of the transmitted and speculated messages `msgTx`, and an array of MAC segments of the transmitted and speculated messages `segTx`. The arrays `msgTx` and `segTx` are stored and maintained by the sender. This algorithm runs the MsgSpec algorithm followed by MacGen algorithm and SegAgg algorithm to compute the authentication tag τ_i.

6. $\texttt{valid/invalid} \leftarrow \mathsf{TagVerify}(\texttt{k}, m_i, \tau_i)$: This tag verification algorithm is run by the receiver. It takes as inputs the secret key `k`, the received messages m_i, and the received tag τ_i. It utilizes an array of received and speculated messages `msgRx`, an array of MAC segments of received and speculated messages `segRx`, and an array of number of verified segments `accRx`. The arrays `msgRx`, `segRx`, and `accRx` are stored and maintained by the receiver. This algorithm verifies whether the tag τ_i is generated using the secret key `k`. If the verification succeeds, it outputs the value `valid`; otherwise, it outputs the value `invalid`.

In CuMAC/S, we assume that the sender and the receiver share a secret key `k` and a counter i. When the sender and the receiver are synchronized using the counter i, the elements in the segments arrays `segTx` at the sender and `segRx` at the receiver are the same. Similarly,

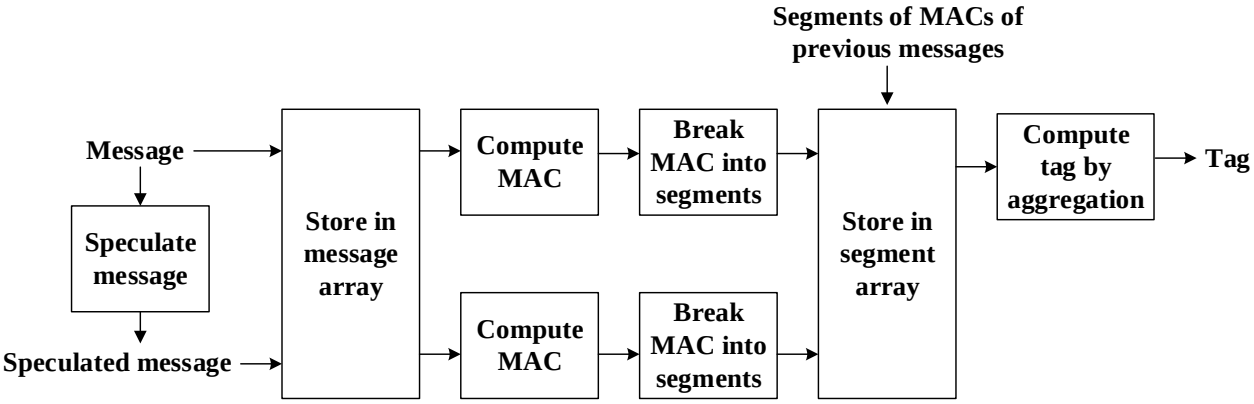


Figure 8.1: Schematic of the procedures at the sender in CuMAC/S.

the elements in the message arrays `msgTx` at the sender and `msgRx` at the receiver are the same. Note that since the packet speculation algorithm is pre-trained and deterministic, the sender and the receiver are able to conduct the same computations and obtain the same speculated messages.

Figure 8.1 illustrates the major steps performed by the sender in CuMAC/S. In the TagGen algorithm, the sender computes the authentication tag through three major steps. In the first step, if the speculated message $\widehat{m}_i$ (corresponding to message m_i) stored in the `msgTx` array matches the current message m_i, the sender proceeds to the next step; otherwise, the sender generates the MAC of the message m_i using the MacGen algorithm, breaks the MAC into short segments, and stores them into the segment array `segTx`. In the second step, the sender speculates message $\widehat{m}_{i+n-1}$ and stores it in the `msgTx` array. It computes the corresponding MAC using the MacGen algorithm, breaks the MAC into short segments, and stores them into the segment array `segTx`. In the third step, the sender retrieves $2n-1$ segments (one MAC segment of the current message, $n-1$ segments of the MACs of the previously transmitted messages and $n-1$ segments of the MACs of the speculated messages) from the array `segTx`, and aggregates the segments to generate a tag using the SegAgg algorithm. The sender inserts the message and the authentication tag in a packet, and sends the packet to the receiver.

Having received the packet, the receiver runs the TagVerify algorithm which includes two major steps. In the first step, the receiver generates an authentication tag of the received message using the same secret key `k`. In the second step, the receiver compares the generated authentication tag with the received authentication tag. If the authentication tags match, the receiver increments the number of accumulated MAC segments of received messages into an accumulation array `accRx`. If the message m_i is *successfully* speculated during the generation of the previous packets, the receiver is able to accumulate some of its MAC segments after verifying those previous packets. Hence, when the receiver receives the message m_i, it is able to verify all the accumulated segments together resulting in higher level of cryptographic strength. When the speculation of message m_i is incorrect, the performance of CuMAC/S is reduced to CuMAC, i.e., CuMAC/S still enables the receiver to authenticate the message with some minimal cryptographic strength in real-time, and with the highest level of cryptographic strength after accumulating all segments of the MAC (that covers message m_i) in the forthcoming packets. Similarly, in CuMAC/S, the counter i handles out-of-order delivery of packets [7]. The cases of missing tags due to lost packets are classified into two categories based on the awareness of the sender about the packet loss. In the first category of cases, before transmitting the next packet, the sender becomes aware that the current packet is lost since it has not received the acknowledgement from the receiver. In these cases, the sender decrements the value of counter i, and proceeds with the standard retransmission mechanism and the authentication scheme. In the second category of cases, the sender is not aware of the loss (e.g., in networks that only provide best effort services). In these cases, the sender utilizes the MAC segments of the lost message in $n-1$ forthcoming packets which means that one lost packet hinders the verification of those $n-1$ forthcoming packets at the receiver. Hence, like compound MAC [55], trailing MAC [36] and aggregate MAC [42], CuMAC and CuMAC/S are not robust against a missing tag in the second category of cases. However, we highlight that in the targeted applications of these schemes, either

there exists retransmission mechanism with acknowledgement (e.g., the CAN bus employs the mechanism where the receiver acknowledges the reception of the message [73]), or the probability of a missing tag is close to zero (e.g., the Sigfox network utilizes repetition of packets to ensure that 99.9 % packets are successfully communicated [64]).

8.2 Overview of Security Properties

In this section, we present the security properties satisfied by CuMAC/S.

8.2.1 Authentication Levels

In CuMAC/S, the message m_i is speculated as $\widehat{m}_i$ while generating τ_{i-n+1}, and the MAC of $\widehat{m}_i$ is computed as $\widehat{\sigma}_i$. The MAC $\widehat{\sigma}_i$ is divided into n segments. The last $n-1$ segments are distributed in tags $\tau_{i-n+1}, \cdots, \tau_{i-1}$. In the i^{th} packet, the MAC of message m_i is computed as σ_i, and divided into n segments. These segments are distributed in tags $\tau_i, \cdots, \tau_{i+n-1}$. In CuMAC/S, the cryptographic strength of the full authentication depends on the same three aforementioned security parameters of the standard MAC. Note that the speculation procedure in CuMAC/S enables the receiver to accumulate multiple tags received in the previous packets to increase the cryptographic strength of the MAC for real-time authentication. The cryptographic strength of the real-time authentication of m_i in CuMAC/S is indicated by l bits if the speculation of m_i is wrong, and $L(= l \cdot n)$ bits if the speculation of m_i is correct. In CuMAC/S, we compute the mean/average cryptographic strength by averaging the cryptographic strengths of the messages which are correctly speculated and those which are wrongly speculated. This means that in addition to the three aforementioned security parameters of the standard MAC, the average cryptographic strength of real-time

authentication in CuMAC/S depends on the size of the MAC segment l, and the speculation accuracy. In the similar manner, we observe that in addition to the three aforementioned security parameters of the standard MAC, the average cryptographic strength of the partially accumulated authentication of m_i in CuMAC/S depends on the size of the MAC segment l, the number of accumulated segments r, and the speculation accuracy.

8.2.2 Security Definitions

Similar to $\mathbf{Exp}^{\text{uf-cma-}r}_{\mathsf{CuMAC}}(\mathcal{A}, \lambda, q)$, the uf-cma-$r$ experiment for CuMAC/S can be readily defined as follows.

$\mathbf{Exp}^{\text{uf-cma-}r}_{\mathsf{CuMAC/S}}(\mathcal{A}, \lambda, q)$

$\mathtt{k} \leftarrow \mathsf{KeyGen}(1^\lambda)$

Invoke $\mathcal{A}^{O_{\mathsf{CuMAC/S}}(\mathtt{k},\cdot)}$ who can make up to q queries to the tagging oracle of CuMAC/S $O_{\mathsf{CuMAC/S}}(\mathtt{k}, \cdot)$. $\mathcal{A}$ can query $O_{\mathsf{CuMAC/S}}(\mathtt{k}, \cdot)$ with $2n-1$ arbitrarily chosen messages and receive their CuMAC/S tags in response.

$\mathcal{A}$ outputs a set of $2n-1$ pairs $\big(\{m_i\}_{i=1}^{2n-1}, \{\tau_i\}_{i=1}^{2n-1}\big)$.

Return 1 if $\mathtt{valid} \leftarrow \mathsf{TagVerify}\,(\mathtt{k}, m_i, \tau_i)$ for all $1 \leq i \leq 2n-1$, and $\mathcal{A}$ did not make the query for m_{i^*} to $O_{\mathsf{CuMAC/S}}(\mathtt{k}, \cdot)$, where $i^* = 2n - r$.

Return 0 otherwise.

Definition 8.2. CuMAC/S is (t, q, ϵ, r)-uf-cma secure if for any PPT adversary $\mathcal{A}$ running in time t, $\Pr\big[\mathbf{Exp}^{\text{uf-cma-}r}_{\mathsf{CuMAC/S}}(\mathcal{A}, \lambda, q) = 1\big] \leq \epsilon$.

Similarly, if CuMAC/S is (t, q, ϵ, r)-uf-cma secure for all r, then it is also (t, q, ϵ)-uf-cma secure which is the standard notion of security for a MAC scheme. We utilize the aforementioned uf-cma-r security model to define the cryptographic strength for full authentication, partially accumulated authentication and real-time authentication. Note that in the uf-cma-r experiment, when the experiment returns a value of 1, it implies that $\mathcal{A}$ is able to forge a valid tag for a packet at which point the receiver has already accumulated r packets. Therefore, if CuMAC/S is (t, q, ϵ, n) secure, i.e., $r = n$, then CuMAC/S is secure in terms of full authentication. Similarly, if CuMAC/S is (t, q, ϵ, r) secure for all $2 \leq r \leq n-1$, then the scheme is secure for partially accumulated authentication; and if CuMAC/S is $(t, q, \epsilon, 1)$ secure, i.e., $r = 1$, then the scheme is secure in terms of real-time authentication. The security of CuMAC/S is also based on the assumption that defines the security of the underlying MAC algorithm [6].

8.3 CuMAC/S: CuMAC with Speculation

In this section, we present the technical details of the algorithms employed by CuMAC/S. We also provide an illustrative example of CuMAC/S.

8.3.1 Technical Details

The KeyGen and MacGen algorithms in CuMAC (discussed in Section 7.3) and those in CuMAC/S are the same, and hence we do not provide their details in this section. We present the details of other algorithms in CuMAC/S as follows.

$\texttt{msgArray}' \leftarrow \mathsf{MsgSpec}(\texttt{msgArray})$

This deterministic message speculation algorithm is utilized by the sender and receiver for

the speculation of future message values. It takes an array of the transmitted and speculated messages `msgArray` as input. At the i^{th} instance, the array `msgArray` can be represented as $\{m_1, m_2, \cdots, m_{i-1}, m_i, \widehat{m}_{i+1}, \widehat{m}_{i+2}, \cdots \widehat{m}_{i+n-2}\}$. This algorithm generates the predicted value of the message m_{i+n-1}, which is represented by $\widehat{m}_{i+n-1}$. This speculated message is appended to the array `msgArray`. It outputs the updated array `msgArray'`. Since the speculation model used in this algorithm is deterministic, the sender and the receiver run the same set of steps, and obtain the same speculated messages given the same input messages.

$\tau_i \leftarrow \mathsf{SegAgg}(\texttt{segArray})$:

This segment aggregation algorithm is run by the sender and the receiver. It takes as input a two-dimensional array of MAC segments `segArray`. The `segArray` comprises of the segments of the MACs of the transmitted and speculated messages. The i^{th} entry in `segArray` is generated by the segment $s_i^j \ \forall j \in [1, n]$ using the equation (7.1). This algorithm extracts $2n-1$ elements from `segArray` ($n-1$ previous MAC segments, current MAC segment, and $n-1$ speculated MAC segments), and computes the authentication tag τ_i as follows.

$$\tau_i \leftarrow \left(\bigoplus_{j=1, i-j+1>0}^{n} s_{i-j+1}^{j} \right) \oplus \left(\bigoplus_{j=2}^{n} \widehat{s}_{i+j-1}^{j} \right). \tag{8.1}$$

This algorithm outputs the authentication tag τ_i.

$\tau_i \leftarrow \mathsf{TagGen}(\mathtt{k}, m_i)$

This tag generation algorithm is utilized by the sender to generate an authentication tag. It takes as inputs the secret key `k` and a messages m_i. It utilizes an array of the transmitted and speculated messages `msgTx`, and an array of MAC segments of the transmitted and speculated messages `segTx`. The arrays `msgTx` and `segTx` are stored and maintained by the sender. This algorithm proceeds as follows.

1. Verify that $m_i = \widehat{m}_i$.

   (a) If $m_i = \widehat{m}_i$, set $\sigma_i = \widehat{\sigma}_i$.

   (b) Otherwise, if $m_i \neq \widehat{m}_i$, compute the MAC of the message m_i and set it as σ_i, i.e., $\sigma_i \leftarrow \mathsf{MacGen}(\mathtt{k}, m_i)$. Divide the MAC into n segments and replace the MAC segments of $\widehat{\sigma}_i$ in the array `segTx`.

2. Predict the value of the message m_{i+n-1} and append the speculated message $\widehat{m}_{i+n-1}$ to the array `msgTx`, i.e., $\mathtt{msgTx}' \leftarrow \mathsf{MsgSpec}(\mathtt{msgTx})$.

3. Compute the MAC of the message $\widehat{m}_{i+n-1}$ and set it as $\widehat{\sigma}_{i+n-1}$, i.e., $\widehat{\sigma}_{i+n-1} \leftarrow \mathsf{MacGen}(\mathtt{k}, \widehat{m}_{i+n-1})$. Divide the MAC into n segments and append to the array `segTx`.

4. Compute the current tag τ_i by aggregating the segments of MACs of the previous, current and future messages, i.e., $\tau_i \leftarrow \mathsf{SegAgg}(\mathtt{segTx})$.

5. Output the tag τ_i.

$\mathtt{valid/invalid} \leftarrow \mathsf{TagVerify}(\mathtt{k}, m_i, \tau_i)$

This verification algorithm is utilized by the receiver for verifying the authenticity of the received message and tag. It takes as inputs the secret key `k`, the received messages m_i, and the received tag τ_i. It utilizes an array of received and speculated messages `msgRx`, an array of MAC segments of received and speculated messages `segRx`, and an array of number of verified segments `accRx`. These arrays are stored and maintained by the receiver. The i^{th} entry in the array `accRx` is represented by r_i. To initialize the value of r_i in the array `accRx`, the receiver sets $r_i = 0$. This algorithm proceeds as follows.

1. Verify that $m_i = \widehat{m}_i$.

   (a) If $m_i = \widehat{m}_i$, set $\sigma_i = \widehat{\sigma}_i$.

(b) Otherwise, if $m_i \neq \widehat{m}_i$, compute the MAC of the message m_i and set it as σ_i, i.e., $\sigma_i \leftarrow \mathsf{MacGen}(\mathtt{k}, m_i)$. Divide the MAC into n segments and replace the MAC segments of $\widehat{\sigma}_i$ in the array `segRx`. Also, set the number of accumulated MAC segments corresponding to m_i in `accRx` to zero, i.e., $r_i = 0$.

2. Predict the value of the message m_{i+n-1}, and append the speculated message $\widehat{m}_{i+n-1}$ to the array `msgRx`, i.e., $\mathtt{msgRx}' \leftarrow \mathsf{MsgSpec}(\mathtt{msgRx})$.

3. Compute the MAC of the message $\widehat{m}_{i+n-1}$ and set it as $\widehat{\sigma}_{i+n-1}$, i.e., $\widehat{\sigma}_{i+n-1} \leftarrow \mathsf{MacGen}(\mathtt{k}, \widehat{m}_{i+n-1})$. Divide the MAC into n segments and append to the array `segRx`.

4. Compute the current tag $\widetilde{\tau}_i$ by aggregating the segments of MACs of the previous, current and future messages, i.e., $\widetilde{\tau}_i \leftarrow \mathsf{SegAgg}(\mathtt{segRx})$.

5. If $\widetilde{\tau}_i = \tau_i$,

   (a) Increment the number of accumulated MAC segments in `accRx`, i.e., for each $t \in [i-n+1, i+n-1]$, if $r_t \neq n$, set $r_t = r_t + 1$.

   (b) Output the value `valid`.

6. Otherwise, if $\widetilde{\tau}_i \neq \tau_i$, output the value `invalid`.

8.3.2 Illustration

Table 8.1 presents an example of CuMAC/S. Similar to the example shown in Table 7.1, the size of the tag in each packet is 32 bits (i.e., $l = 32$). The MAC is generated using the AES-CMAC algorithm. Hence, the size of the MAC output is 128 bits (i.e., $L = 128$), which provides cryptographic strength of 128 bits. Each MAC is divided into four segments (i.e., $n = 4$). This means that the achievable cryptographic strengths for real-time authentication

Table 8.1: Example illustrating CuMAC/S with $L = 128$, $n = 4$, and $l = 32$.

Packet	Previous MACs	Current MAC	Previous speculated MACs	Current speculated MAC	Aggregation of MAC segments	Tag
2	σ_1	σ_2	$\widehat{\sigma}_3, \widehat{\sigma}_4$	$\widehat{\sigma}_5$	$s_1^2 \oplus s_2^1 \oplus \widehat{s}_3^2 \oplus \widehat{s}_4^3 \oplus \widehat{s}_5^4$	τ_2
3	σ_1, σ_2	σ_3	$\widehat{\sigma}_4, \widehat{\sigma}_5$	$\widehat{\sigma}_6$	$s_1^3 \oplus s_2^2 \oplus s_3^1 \oplus \widehat{s}_4^2 \oplus \widehat{s}_5^3 \oplus \widehat{s}_6^4$	τ_3
4	$\sigma_1, \sigma_2, \sigma_3$	σ_4	$\widehat{\sigma}_5, \widehat{\sigma}_6$	$\widehat{\sigma}_7$	$s_1^4 \oplus s_2^3 \oplus s_3^2 \oplus s_4^1 \oplus \widehat{s}_5^2 \oplus \widehat{s}_6^3 \oplus \widehat{s}_7^4$	τ_4
5	$\sigma_2, \sigma_3, \sigma_4$	σ_5	$\widehat{\sigma}_6, \widehat{\sigma}_7$	$\widehat{\sigma}_8$	$s_2^4 \oplus s_3^3 \oplus s_4^2 \oplus s_5^1 \oplus \widehat{s}_6^2 \oplus \widehat{s}_7^3 \oplus \widehat{s}_8^4$	τ_5
6	$\sigma_3, \sigma_4, \sigma_5$	σ_6	$\widehat{\sigma}_7, \widehat{\sigma}_8$	$\widehat{\sigma}_9$	$s_3^4 \oplus s_4^3 \oplus s_5^2 \oplus s_6^1 \oplus \widehat{s}_7^2 \oplus \widehat{s}_8^3 \oplus \widehat{s}_9^4$	τ_6
7	$\sigma_4, \sigma_5, \sigma_6$	σ_7	$\widehat{\sigma}_8, \widehat{\sigma}_9$	$\widehat{\sigma}_{10}$	$s_4^4 \oplus s_5^3 \oplus s_6^2 \oplus s_7^1 \oplus \widehat{s}_8^2 \oplus \widehat{s}_9^3 \oplus \widehat{s}_{10}^4$	τ_7
8	$\sigma_5, \sigma_6, \sigma_7$	σ_8	$\widehat{\sigma}_9, \widehat{\sigma}_{10}$	$\widehat{\sigma}_{11}$	$s_5^4 \oplus s_6^3 \oplus s_7^2 \oplus s_8^1 \oplus \widehat{s}_9^2 \oplus \widehat{s}_{10}^3 \oplus \widehat{s}_{11}^4$	τ_8

and full authentication are 128 bits. To simplify the discussion, we limit the discussions to the packets which are involved in the authentication of the message transmitted in the fifth packet, m_5.

In the second packet, message m_2 is transmitted. To compute the corresponding tag τ_2, the sender speculates the message to be transmitted in the fifth packet $\widehat{m}_5$, and aggregates the segments of the MACs of the generated messages, σ_1 and σ_2, and the MACs of the speculated messages, $\widehat{\sigma}_3$, $\widehat{\sigma}_4$ and $\widehat{\sigma}_5$. Specifically, tag τ_2 is computed using the segment $\widehat{s}_5^4$. Further, tag τ_3 is computed using the segment $\widehat{s}_5^3$, and the tag τ_4 is computed using the segment $\widehat{s}_5^2$. In the fifth packet, if the current message m_5 matches with the speculated message $\widehat{m}_5$, the corresponding MAC σ_5 is mapped to $\widehat{\sigma}_5$; otherwise, the MAC σ_5 is computed. The tags τ_5, τ_6, τ_7 and τ_8 are computed using the segments s_5^1, s_5^2, s_5^3 and s_5^4, respectively.

When the receiver receives the fifth packet with the message m_5, it verifies whether it matches the speculated message $\widehat{m}_5$. If they do not match, the successful verification of the tag τ_5 enables the real-time authentication of message m_5 with a cryptographic strength of 32 bits, which is the same as in CuMAC. Next, the receiver receives and verifies the validity of tags $\tau_5, \cdots, \tau_8$. If all four tags are verified as `valid`, the receiver combines the segments s_5^1, s_5^2, s_5^3 and s_5^4—which are contained in tags τ_5, τ_6, τ_7 and τ_8, respectively—to accumulate the cryptographic strength. This enables the receiver to perform full authentication of message

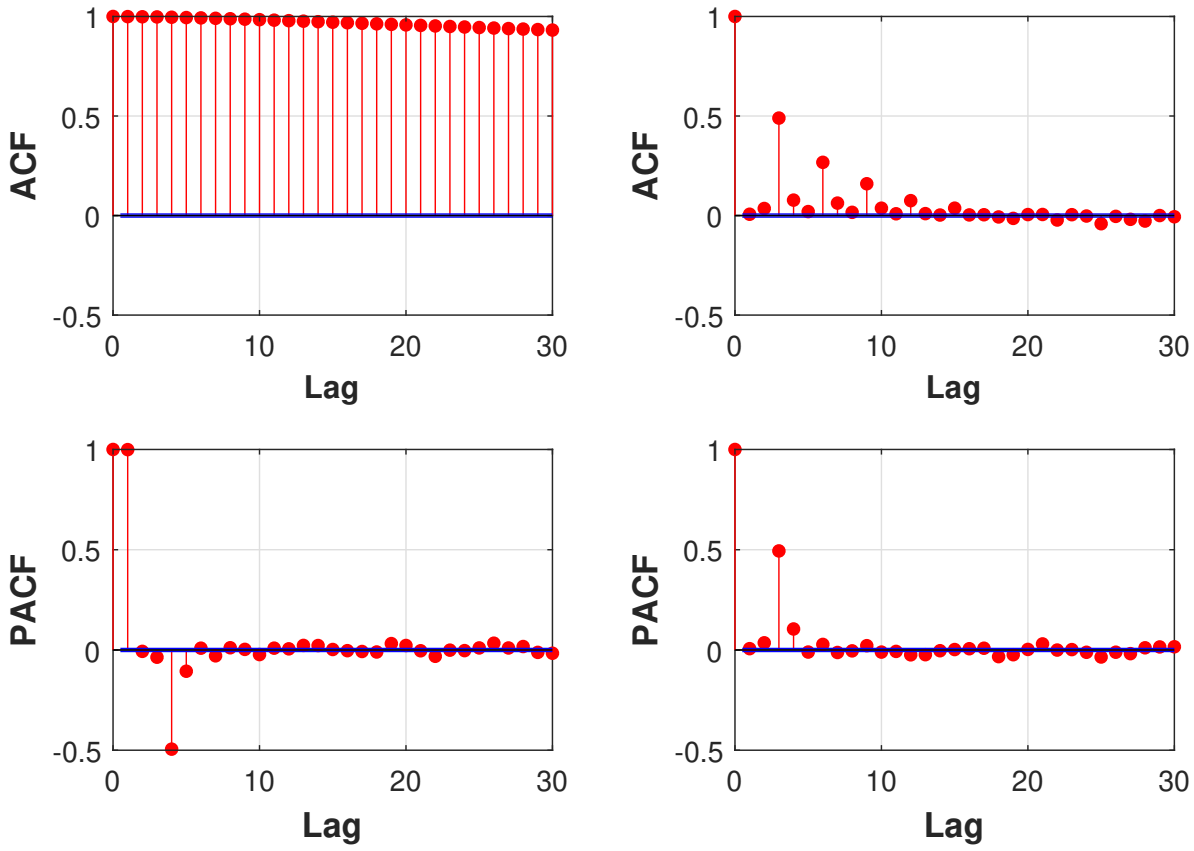
m_5 with a cryptographic strength of 128 ($= 4 \times 32$) bits.

However, if the message m_5 and $\widehat{m}_5$ match, and tags τ_2, τ_3, τ_4 and τ_5 are verified as `valid`, the receiver combines the segments s_5^1, $\widehat{s}_5^2$, $\widehat{s}_5^3$ and $\widehat{s}_5^4$—which are contained in tags τ_5, τ_4, τ_3 and τ_2, respectively. This enables the receiver to perform real-time authentication of message m_5 with a cryptographic strength of 128 bits. Note that the receiver may also perform full authentication of message m_5 with a cryptographic strength of 128 bits after verifying tags τ_5, τ_6, τ_7 and τ_8.

8.3.3 Feasibility of Speculation

We discuss the feasibility of speculation of future messages by analyzing the messages communicated on a CAN bus in a typical vehicle. To evaluate the speculation accuracy for different CAN messages, we utilize trace files of a real vehicle, which have been recorded using the OpenXC platform [31]. These files present different types of CAN messages which can be identified and interpreted by OpenXC libraries. To speculate future message values, we utilize autoregressive integrated moving average (ARIMA) model, which is a widely used model for time series analysis.

The ARIMA model with hyper parameters (p, d, q) implies that for the d^{th}-order difference of the time series values, a speculated future message value is the linear combination of p previous values, and q previous error values in the speculation. We utilize the Box-Jenkins [50] method to compute the hyper parameters of the ARIMA model for each type of CAN messages. In this method, we determine the values for p, d and q by observing the autocorrelation and partial autocorrelation of the message values. We illustrate this procedure in Figure 8.2 which presents the autocorrelation and partial autocorrelation of the values of the message corresponding to the torque at transmission in a vehicle. From the



(a) Original data.

(b) First-order differenced data.

Figure 8.2: Example illustrating the feasibility of speculation of a vehicle's transmission torque values through an ARIMA model whose parameters are determined using the auto-correlation function (ACF) and partial autocorrelation function (PACF).

results shown in Figure 8.2a, we observe that there is high autocorrelation between message values. Further, from the results shown in Figure 8.2b, we observe that the autocorrelation decays gradually, and the partial autocorrelation is close to zero after a lag of 3 message values. Hence, according to the rules of Box-Jenkins approach, we set ARIMA(3,1,0) model to speculate the message values corresponding to the torque at transmission.

In our analysis, we train the ARIMA model using the first 90% of the message values, and then we employ the model on the last 10% of message values for the test. Here, the accuracy of correct speculation/prediction of future message values is measured using a metric called speculation error rate (SER), which is defined as the ratio between the number of incorrect speculations and the total number of speculations. Note that the speculation is correct only if the message is correctly predicted up to the least significant bit (LSB). Table 8.2 shows the speculation error rate of ten message-types. We observe that certain types of CAN messages (e.g. the first five message types listed in Table 8.2) can be predicted with high reliability using the ARIMA model.

We can improve the speculation accuracy by using more sophisticated and further tuned models. Moreover, we can mitigate the impact of speculation errors by increasing robustness of the MAC scheme against such errors. For example, if the message contains some values for which some of the least significant bits can be safely ignored (without impacting performance or security), then these bits do not need to be protected by a MAC, and hence the MAC calculation can be limited to only the part of a message that can be predicted with high reliability. In the rightmost column of Table 8.2, we show that the SER can be significantly improved for some types of messages by ignoring the last three least LSBs.

Now we present the technical details of the algorithms in CuMAC/S, and an instantiation that illustrates the generation and verification of the tags in CuMAC/S.

Table 8.2: Speculation accuracy for typical CAN messages.

Signal	**SER**	**SER after ignoring 3 LSBs**
Longitude	<0.0001	<0.0001
Latitude	<0.0001	<0.0001
Odometer	<0.0001	<0.0001
Fuel level	<0.0001	<0.0001
Fuel consumed since restart	<0.0001	<0.0001
Accelerator pedal position	0.0030	0.0002
Torque at transmission	0.0100	0.0020
Engine speed	0.2329	0.0880
Vehicle speed	0.2478	0.0975
Steering wheel angle	0.4763	0.3595

8.4 Security Analysis

In this section, we present theorems and corresponding proofs of security for CuMAC/S. Let CuMAC/S be instantiated with parameters (β, l, n), i.e., the speculation error rate for the messages is β, and each MAC is divided into n segments, each of length l bits. Note that in CuMAC/S, the receiver performs real-time authentication by setting $r = 1$, partially accumulated authentication by setting $1 < r < n$, and full authentication by setting $r = n$.

Theorem 8.3. *For any $t, q \in \mathbb{N}$ and $\epsilon > 0$, if the underlying deterministic MAC algorithm,* MacGen*, is (t, q, ϵ)-uf-cma secure, then CuMAC/S with parameters (β, l, n) is (t', q', ϵ', r)-uf-cma secure, where*

$$t' \approx t, \quad q' = \frac{q - 3n + 3}{2n - 1}, \quad \epsilon' = \frac{\epsilon}{(1 - \beta) + \beta 2^{-l(n-r)}}.$$

Proof. Let there be an adversary $\mathcal{A}$ which succeeds to create a forgery of an authentication

tag for CuMAC/S with a non-negligible probability. We construct another simulator $\mathcal{S}$ that interacts with the adversary $\mathcal{A}$ and creates a forgery of a MAC for the MacGen algorithm with a non-negligible probability.

Let CuMAC/S and the MacGen algorithm utilize the same secret key $\mathtt{k}$ which is unknown to the adversary $\mathcal{A}$. Also, let the MAC of a message in CuMAC/S be computed by a query to $O_{\mathsf{MacGen}}(\mathtt{k},\cdot)$. In this way, $\mathcal{S}$ perfectly simulates $O_{\mathsf{CuMAC/S}}(\mathtt{k},\cdot)$, and hence the uf-cma-$r$ experiment for CuMAC/S. Suppose the uf-cma-r experiment for CuMAC/S returns 1 with the probability ϵ' in time t', where an adversary $\mathcal{A}$ outputs a successful forgery $\left(\{m_i\}_{i=1}^{2n-1}, \{\tau_i\}_{i=1}^{2n-1}\right)$ after q' queries to $O_{\mathsf{CuMAC/S}}(\mathtt{k},\cdot)$ simulated by $\mathcal{S}$. To create a forgery of a MAC for the MacGen algorithm, the simulator $\mathcal{S}$ proceeds as follows.

For all $i \in [1, 2n-1]$ and $i \neq i^*$, the simulator $\mathcal{S}$ queries $O_{\mathsf{MacGen}}(\mathtt{k},\cdot)$ for the MAC of m_i, and obtains the corresponding σ_i. Additionally, it queries $O_{\mathsf{MacGen}}(\mathtt{k},\cdot)$ for the MAC of the speculated messages $\widehat{m}_i$ and obtains $\widehat{\sigma}_i$ for all $i \in [i^*+1, i^*+n-1]$. It divides each MAC into n segments as shown in equation (7.1). It recovers the MAC segments of the message m_{i^*} by removing the mask by the MAC segments of other messages as follows:

$$s_{i^*}^k \leftarrow \tau_{i^*+k-1} \oplus \bigoplus_{j=1, j\neq k}^{n} s_{i^*-j+k}^j \oplus \bigoplus_{j=2}^{n} \widehat{s}_{i^*+j+k-2}^j. \tag{8.2}$$

By following the above procedure, the simulator $\mathcal{S}$ recovers r MAC segments. For all $k \geq r+1$, the simulator $\mathcal{S}$ attempts to recover $s_{i^*}^k$ from the tags received before tag τ_{i^*} as follows:

$$s_{i^*}^k \leftarrow \tau_{i^*-k+1} \oplus \bigoplus_{j=1}^{n} s_{i^*-k-j+2}^j \oplus \bigoplus_{j=2, j\neq k}^{n} \widehat{s}_{i^*-k+j}^j. \tag{8.3}$$

These segments can be recovered with a probability $1-\beta$. If an speculation error occurs, then the corresponding MAC segment is not recovered. In this case, $\mathcal{S}$ sets the value of the MAC

segment by randomly guessing the bits. Finally, the simulator $\mathcal{S}$ creates a fresh forgery for the underlying deterministic MAC algorithm, MacGen, by concatenating all recovered and guessed segments. The probability that such forgery is correct is $(1-\beta)+\beta\cdot 2^{-l(n-r)}$.

To achieve the forgery of MacGen as shown above, the simulator $\mathcal{S}$ conducts at most $(2n-1)q'$ queries to $O_{\mathsf{MacGen}}(\mathtt{k},\cdot)$ to answer q' queries by $\mathcal{A}$ to $O_{\mathsf{CuMAC/S}}(\mathtt{k},\cdot)$. In order to compute operations in equations (8.2) and (8.3), the simulator $\mathcal{S}$ conducts at most $2n-2$ queries to $O_{\mathsf{MacGen}}(\mathtt{k},\cdot)$ to obtain $\{\tau_i\}_{i=1,i\neq i^*}^{2n-1}$, and at most $n-1$ queries to obtain $\{\widehat{\sigma}_i\}_{i=i^*+1}^{i^*+n-1}$. Therefore, if for an adversary $\mathcal{A}$ running in time t', we have $\Pr\left[\mathbf{Exp}^{\text{uf-cma-}r}_{\mathsf{CuMAC/S}}(\mathcal{A},\lambda,q')=1\right]\leq\epsilon'$, then we can leverage it to break the underlying MAC algorithm, MacGen, in time t' plus the time required to evaluate equations (8.2) and (8.3), by making $(2n-1)q'+3n-3$ queries, and with probability δ. Hence, if the underlying MAC algorithm, MacGen, is (t,q,ϵ)-uf-cma secure, then CuMAC/S is (t',q',ϵ',r)-uf-cma secure, where $t'\approx t$, $q'=\frac{q-3n+3}{2n-1}$, and $\epsilon'=\frac{\epsilon}{(1-\beta)+\beta 2^{-l(n-r)}}$. □

8.5 Evaluation

In this section, we evaluate CuMAC/S and compare it with CuMAC through simulation.

8.5.1 Comparison of CuMAC and CuMAC/S with the Prior Art

We evaluate the performance of CuMAC and CuMAC/S by comparing them with two other schemes from the prior art: the truncated MAC [67] and the compound MAC [55]. For all four schemes, AES-CMAC with a MAC output of 128 bits is utilized as the underlying MAC algorithm. We set the same communication overhead in all the four schemes, i.e., the size of the tag in each scheme is set to 16 bits. Recall that in the truncated MAC, each

MAC is truncated to 16 bits, and transmitted as the tag. In the compound MAC scheme, one compound MAC of 128 bits is computed over eight messages. The compound MAC is divided into eight segments each of size 16 bits, and transmitted in each of the eight frames as the tag. In CuMAC, each MAC is divided into eight segments each of size 16 bits, and each tag is generated by aggregating MAC segments of seven previously transmitted messages and the current message. In CuMAC/S, each MAC is divided into eight segments each of size 16 bits, and each tag is generated by aggregating MAC segments of seven previously transmitted messages, the current message, and the seven speculated messages.

Figure 8.3 presents the curves for the cryptographic strengths of six schemes versus their authentication delay. For CuMAC/S, three curves, with SER $= 0, 0.5, 1$, are shown[2]; the average cryptographic strength was used in plotting the curves for CuMAC/S. We observe that the truncated MAC is authenticated without any delay, but the cryptographic strengths of the real-time authentication and the full authentication are limited to only 16 bits. The compound MAC delivers full authentication with cryptographic strength of 128 bits, but it can only be authenticated with a delay of seven frames. CuMAC provides real-time authentication with cryptographic strength of 16 bits and full authentication with cryptographic strength of 128 bits with a delay of seven frames. Note that although both the compound MAC and CuMAC split a MAC into multiple segments, and provide the same cryptographic strength for full authentication, there is an important difference between the two approaches. The former does not provide any real-time authentication whereas the latter does. The compound MAC cannot provide real-time authentication because the verification of a MAC requires the receiver to wait and receive *all* of the packets that contain the segments of that particular MAC. Findings shown in Figure 8.3 highlight the unique attribute of CuMAC and CuMAC/S—i.e., they enable a receiver to make a trade-off between (accumulated) crypto-

[2]CuMAC/S with SER $= 1$ is equivalent to CuMAC in terms of cryptographic strength versus delay.

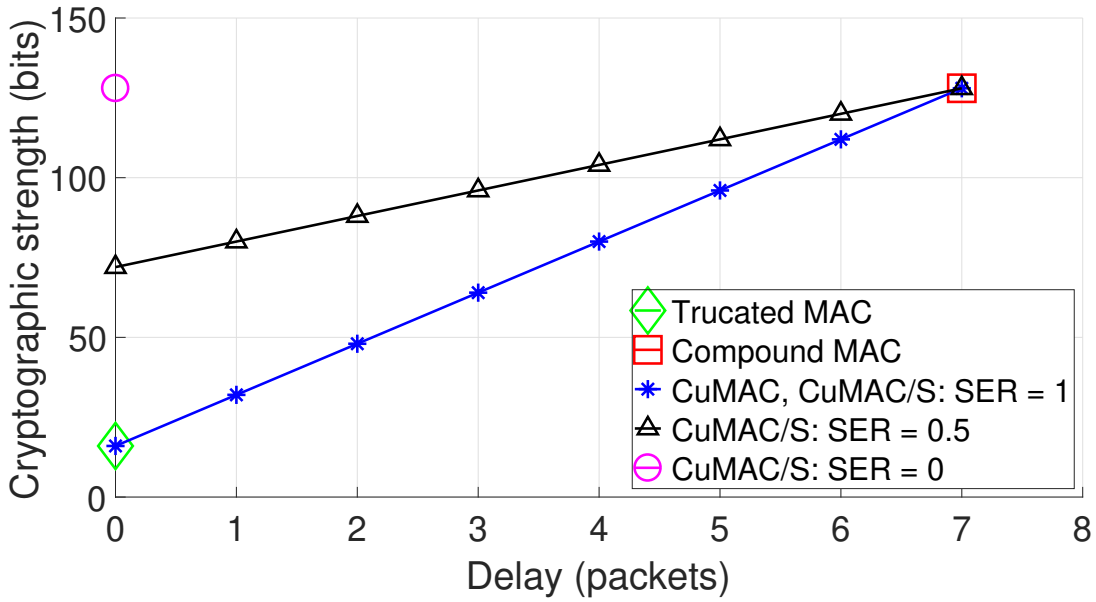


Figure 8.3: Cryptographic strength vs. delay for CuMAC, CuMAC/S, and prior art.

graphic strength and authentication delay. In some applications (e.g., air quality monitoring system discussed in Section 7.5.1), especially in latency-tolerant applications, this attribute provides the receiver with operational flexibility to vary the security level and/or packet processing delay based on particular needs of a protocol or rules prescribed by network traffic processing policies. From the curves of CuMAC/S, we can observe that CuMAC/S is able to accumulate cryptographic strength faster if SER is reduced. In the case of SER = 0, CuMAC/S enables the receiver to achieve 128 bits of full cryptographic strength without any delay (i.e., immediately after the message and tag are received and verified).

8.6 Implementation Results

In this section, we present results from experiments performed with a prototype implementation of CuMAC and CuMAC/S on an ECU development board. The prototype implementation played a critical role in evaluating performance using various metrics, including

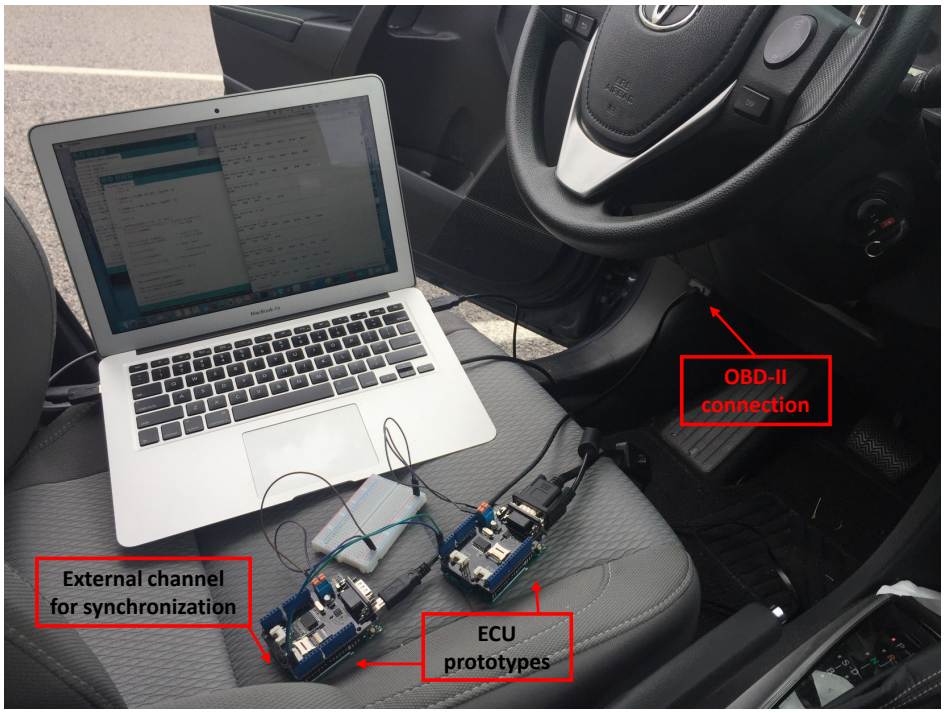


Figure 8.4: Prototype connected to a car's CAN bus.

computational cost of generating a tag and end-to-end latency of tag generation/verification. The prototype implementation was also used to compare CuMAC and CuMAC/S with conventional MAC schemes.

8.6.1 Details of the Prototype Implementation

Figure 8.4 illustrates the prototype implementation and the setup that were used for running our experiments. The prototype implementation comprises of two ECU prototypes connected to the on-board diagnostics (OBD) port of the CAN bus of a 2016 Toyota Corolla LE. The ECU prototype consists of an Arduino UNO board and a Seeedstudio CAN shield. The Arduino UNO board is used to emulate the controller unit of an ECU, and the Seeedstudio CAN shield works as the interface between the Arduino UNO board and the CAN. The Arduino UNO board utilizes an Atmel ATmega328P chip, which includes a low-power 8-

bit micro-controller running at 16 MHz clock speed along with a 32 KB flash memory and a 2 KB RAM. These specifications of the ECU prototype are representative of a typical state-of-the-art automotive-grade controller [52].

We utilized an open-source cryptography library to implement the following four schemes: standard MAC (i.e., a conventional MAC), truncated MAC, CuMAC and CuMAC/S. For all four schemes, AES-CMAC with a MAC output of 128 bits was utilized as the underlying MAC algorithm. For CuMAC, CuMAC/S, and truncated MAC, the size of the tag was set to 16 bits, and the message and the tag were inserted into the data field of the same CAN frame, as shown in Figure 3.2. For the standard MAC, the size of the tag was set to 128 bits, and the tag was split into two segments, and inserted into the data field of two consecutive CAN frames. These frames were transmitted immediately after the CAN frame containing only the message.

8.6.2 Results

Table 8.3 summarizes the results from the experiments. The computation time of tag generation was calculated by averaging the computation time over 1000 executions to compute a tag. The end-to-end delay shown in the table is the worst case delay for the highest priority message type. Table 8.3 also presents the cryptographic strength for real-time authentication in each scheme, where for CuMAC/S, we consider the average cryptographic strength. From Table 8.3, we can observe that the code space required by the implementation of the four schemes are similar. We can also observe that the computational cost of generating a tag for CuMAC/S increases as the speculation error rate (SER) increases. This is because in the TagGen algorithm of CuMAC/S, when a message is wrongly speculated, the sender executes one additional instance of the MacGen algorithm. Further, we note that CuMAC/S

Table 8.3: Comparison of the MAC schemes using the prototype implementation.

Scheme	**Code space**	**Tag generation**	**End-to-end delay**	**Crypto. strength**
Standard MAC	7410 bytes	786.13 μs	5616.02 μs	128 bits
Truncated MAC	7410 bytes	785.71 μs	3440.10 μs	16 bits
CuMAC	7474 bytes	796.08 μs	3798.08 μs	16 bits
CuMAC/S$_{(SER=0)}$	7522 bytes	801.62 μs	3809.17 μs	128 bits
CuMAC/S$_{(SER=0.05)}$	7522 bytes	840.65 μs	3887.23 μs	122 bits
CuMAC/S$_{(SER=0.5)}$	7522 bytes	1191.88 μs	4589.69 μs	72 bits
CuMAC/S$_{(SER=1)}$	7522 bytes	1582.14 μs	5370.21 μs	16 bits

with low SER enables the receiver to perform real-time authentication with a significantly higher cryptographic strength than that in the truncated MAC and CuMAC.

To evaluate the end-to-end delay performance, we utilized one ECU prototype (called Tx-ECU) to transmit 6-byte messages with the tags on the CAN bus, and another ECU prototype (called Rx-ECU) as the receiver to measure the end-to-end delay. In the experiment, the Rx-ECU requests the Tx-ECU (through an external synchronization channel) to send a message, and starts the timer. The Rx-ECU stops the timer after verifying the tag and authenticating the message. The end-to-end delay is measured as the time between starting the timer and stopping the timer. The message identifier (ID-A) field (see Figure 3.2) in the transmitted CAN frames were set to the highest priority—this enables these messages to win arbitration on the CAN bus against all the other messages [73]. The most stringent latency requirement for any type of message on an in-vehicle CAN bus with the bus speed of 500 kbit/s is 5 ms [65, 68]. Note that the latency requirement of a message is equal to the period of the message—i.e., a message is expected to be received before the lapse of the message's period. The forth column of Table 8.3 compares the worst case end-to-end delay

incurred by the four schemes. Our experimental results indicate that the standard MAC (as implemented on the ECU prototype) cannot satisfy the latency requirement of the most latency-sensitive CAN messages (since the delay is greater than 5 ms). The same is true for CuMAC/S with SER = 1 according to our results. According to Table 8.3, CuMAC incurs significantly less end-to-end delay compared to the standard MAC; the same is true for CuMAC/S with low SER, but to a lesser degree.

Chapter 9

Conclusions and Future Work

In this chapter, we conclude the dissertation and discuss possible future research directions.

9.1 Conclusions

In this dissertation, we discuss the privacy issue of centralized DSA systems and the message authentication problem of resource-constrained networks. To address these issues, we have designed tailored secure multiparty computation protocols, and novel message authentication schemes based on various cryptographic primitives and introducing novel yet solid concepts. We validate our proposed approaches through comprehensive evaluation and implementation and show that they achieve our design goal.

In chapter 4, we present a comparative study on existing solutions that preserves incumbent user's operational privacy. We additionally propose a new metric to evaluate privacy-preserving strength, and we propose the indistinguishable input property to generalize the concept of provable security. Our study shows the effectiveness of MPC-based solutions against attacks from semi-honest SAS, and the trade-off between spectrum utilization and privacy-preserving strength for obfuscation-based solutions. We also discover that obfuscation-based schemes provide stronger privacy protection against malicious SUs compared to MPC-based schemes.

In chapter 5, we propose a novel ESC-based DSA scheme called PriDSA, which preserves IU privacy when SAS is not entirely trusted. We also study the situation when SAS may intentionally deviate from the protocol and collude with some SUs, and propose PriDSA-v2 to tackle this problem. In PriDSA-v2, ESC input data is further blinded such the security goal can be achieved without making system accuracy as a trade-off.

In chapter 6, we proposed a novel DSA framework called PeDSS, which preserves privacy for both IUs and SUs under the scenario of untrusted SAS. It is well known that a general MPC solution is heavy-duty, and a differentially private mechanism harms the usefulness of data. By leveraging the homomorphic property of the AFGH scheme and incorporation of proxy re-encryption, PeDSS manages to achieve the privacy protection goal while eliminating the online trusted third party. The location privacy of SUs is preserved by differentially private mechanisms and the potential false negative error from this mechanism is mitigated through the location stretch algorithm. PeDSS also achieves significantly lower computational overhead compared to the prior art, as a result of faster cryptographic primitive and the usage of differential privacy technique.

In chapter 7, we proposed a novel concept for message authentication that we refer to as *cumulative MAC* (CuMAC). CuMAC provides a receiver with the flexibility to make a trade-off between cryptographic strength (of the underlying MAC) and authentication latency.

In chapter 8, we have proposed a variant of CuMAC called *CuMAC with speculation* (CuMAC/S) that is more suitable for latency-sensitive applications. Our simulation and experimental results are promising and show that the two schemes provide significant advantages over conventional MAC schemes when deployed in several emerging network applications, including those that run on energy-constrained or bandwidth-constrained networks.

9.2 Future Work

Since our comparative study shows that both MPC-based and obfuscation-based schemes have their merely-overlapping advantages and disadvantages, combining both MPC-based and obfuscation-based schemes so that both adversarial SAS and SUs can be handled can be an interesting and promising future direction for IU operational privacy protection.

Future work may also focus on adapting PriDSA to more sophisticated ESC and SAS service models.

Proposed privacy-preserving protocols achieve proved security, yet we expect to illustrate their privacy-preserving strength through more comprehensive analysis, simulation, and comparison.

Future work may also focus on further eliminating the offline trusted third party by leveraging secure hardware such as Trusted Platform Module (TPM) and deeper study on the impact of location stretch algorithm on false positives.

We expect to explore more message speculation models to further prove the effectiveness of CuMAC/S.

Future work on authentication in resource-constrained networks may also focus on designing practical asymmetric message authentication. TESLA[57] is a broadcast authentication protocol that achieves public key message authentication, where the key distribution and storage cost is greatly reduced. While public-key cryptosystem usually brings increased computation cost, it would be interesting to study the trade-off or leverage dedicated hardware to address the issue.

Bibliography

[1] Splat! a terrestrial rf path analysis application for linux/unix. URL http://www.qsl.net/kd2bd/splat.html.

[2] VS Abhayawardhana, IJ Wassell, D Crosby, MP Sellars, and MG Brown. Comparison of empirical propagation path loss models for fixed wireless access systems. In *Vehicular Technology Conference, 2005. VTC 2005-Spring. 2005 IEEE 61st*, volume 1, pages 73–77. IEEE.

[3] Miguel E. Andrés, Nicolás E. Bordenabe, Konstantinos Chatzikokolakis, and Catuscia Palamidessi. Geo-indistinguishability: Differential privacy for location-based systems. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, CCS '13, pages 901–914, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2477-9. doi: 10.1145/2508859.2516735. URL http://doi.acm.org/10.1145/2508859.2516735.

[4] Giuseppe Ateniese, Kevin Fu, Matthew Green, and Susan Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. *ACM Trans. Inf. Syst. Secur.*, 9(1):1–30, February 2006. ISSN 1094-9224.

[5] Behnam Bahrak, Sudeep Bhattarai, Abid Ullah, Jung-Min Park, Jeffery Reed, and David Gurney. Protecting the primary users' operational privacy in spectrum sharing. In *Dynamic Spectrum Access Networks (DYSPAN), 2014 IEEE International Symposium on*, pages 236–247. IEEE, 2014.

[6] M. Bellare, J. Kilian, and P. Rogaway. The security of the cipher block chaining message authentication code. *Journal of Computer and System Sciences*, 61(3):362–399, 2000.

[7] Daniel J Bernstein and Tanja Lange. Non-uniform cracks in the concrete: the power of free precomputation. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 321–340, 2013.

[8] Avrim Blum, Katrina Ligett, and Aaron Roth. A learning theory approach to noninteractive database privacy. *Journal of the ACM (JACM)*, 60(2):12, 2013.

[9] R. Bosch. CAN specification - Version 2.0, 1991.

[10] Fay Chang and Garth A. Gibson. Automatic I/O hint generation through speculative execution. In *Proceedings of the Third Symposium on Operating Systems Design and Implementation (OSDI)*, pages 1–14, 1999.

[11] S. Checkoway, D. Mccoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, and T. Kohno. Comprehensive experimental analyses of automotive attack surfaces. In *Proceedings of the 20th USENIX Security Symposium*, 2011.

[12] Matthew Clark and Konstantinos Psounis. Can the privacy of primary networks in shared spectrum be protected? In *Computer Communications, IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on*, pages 1–9. IEEE, 2016.

[13] Matthew A Clark and Konstantinos Psounis. Trading utility for privacy in shared spectrum access systems. *IEEE/ACM Transactions on Networking (TON)*, 26(1):259–273, 2018.

[14] Federal Communications Commission. Wireless telecommunications bureau and office of engineering and technology conditionally approve seven spectrum access system administrators for the 3.5 ghz band. 2016.

[15] Federal Communications Commission et al. Report and order and second further notice of proposed rulemaking. *Amendment of the Commission's Rules with Regard to Commercial Operations in the*, pages 3550–3650, 2015.

[16] Commerce Spectrum Management Advisory Committee et al. Final report of working group 1–1695-1710 mhz meteorological-satellite. *National Telecommunications and Information Agency, Washington DC*, 22, 2013.

[17] Electronic Communication Committee et al. within the european conference of postal and telecommunications administration (cept)," the analysis of the coexistence of fwa cells in the 3.4-3.8 ghz band,". Technical report, tech. rep., ECC Report 33, 2003.

[18] Xuewen Cui. *Directive-Based Data Partitioning and Pipelining and Auto-Tuning for High-Performance GPU Computing*. PhD thesis, Virginia Tech, 2020.

[19] Xuewen Cui and Wu-chun Feng. Iterative machine learning (iterml) for effective parameter pruning and tuning in accelerators. In *Proceedings of the 16th ACM International Conference on Computing Frontiers*, pages 16–23. ACM, 2019.

[20] Xuewen Cui and Wu-chun Feng. Iterml: Iterative machine learning for intelligent parameter pruning and tuning in graphics processing units. *Journal of Signal Processing Systems*, pages 1–13, 2020.

[21] Xuewen Cui, Thomas RW Scogland, Bronis R de Supinski, and Wu-Chun Feng. Directive-based pipelining extension for openmp. In *2016 IEEE International Conference on Cluster Computing (CLUSTER)*, pages 481–484. IEEE, 2016.

[22] Xuewen Cui, Thomas RW Scogland, Bronis R de Supinski, and Wu-chun Feng. Directive-based partitioning and pipelining for graphics processing units. In *2017 IEEE*

International Parallel and Distributed Processing Symposium (IPDPS), pages 575–584. IEEE, 2017.

[23] Robert I Davis, Alan Burns, Reinder J Bril, and Johan J Lukkien. Controller area network (CAN) schedulability analysis: Refuted, revisited and revised. *Real-Time Systems*, 35(3):239–272, 2007.

[24] Danny Dolev and Andrew Yao. On the security of public key protocols. *IEEE Transactions on information theory*, 29(2):198–208, 1983.

[25] Yanzhi Dou, Kexiong Curtis Zeng, He Li, Yaling Yang, Bo Gao, Chaowen Guan, Kui Ren, and Shaoqian Li. P 2-sas: preserving users' privacy in centralized dynamic spectrum access systems. In *Proceedings of the 17th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pages 321–330. ACM, 2016.

[26] Yanzhi Dou, He Li, Kexiong Curtis Zeng, Jinshan Liu, Yaling Yang, Bo Gao, and Kui Ren. Preserving incumbent users' privacy in exclusion-zone-based spectrum access systems. In *Distributed Computing Systems (ICDCS), 2017 IEEE 37th International Conference on*, pages 2486–2493. IEEE, 2017.

[27] O. Eikemeier, M. Fischlin, J.-F. Götzmann, A. Lehmann, D. Schröder, P. Schröder, and D. Wagner. History-free aggregate message authentication codes. In *International Conference on Security and Cryptography for Networks*, pages 309–328, 2010.

[28] R. Escherich, I. Ledendecker, C. Schmal, B. Kuhls, C. Grothe, and F. Scharberth. SHE: Secure hardware extension - Functional specification, Version 1.1. *Hersteller Initiative Software (HIS) AK Security*, 2009.

[29] I. Fernández-Hernández, V. Rijmen, G. Seco-Granados, J. Simon, I. Rodríguez, and

J. D. Calle. A navigation message authentication proposal for the Galileo open service. *Navigation*, 63(1):85–102, 2016.

[30] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Conference on the Theory and Application of Cryptographic Techniques*, pages 186–194. Springer, 1986.

[31] Ford Motor Company, CrossChasm, and Bug Labs. OpenXC platform. URL http://openxcplatform.com/resources/traces.html. Accessed: August 1, 2018.

[32] Galileo. European GNSS (Galileo) open service: Signal-in-space interface control document. Technical report, 2016.

[33] Bo Gao, Jung-Min Park, and Yaling Yang. Supporting mobile users in database-driven opportunistic spectrum access. In *Proceedings of the 15th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, MobiHoc '14, pages 215–224, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2620-9. doi: 10.1145/2632951.2632984. URL http://doi.acm.org/10.1145/2632951.2632984.

[34] Oded Goldreich. Requirements for commercial operation in the u.s. 3550-3700 mhz citizens broadband radio service band. *Wireless Innovation Forum*, (WINNF-15-S-0112), 2016.

[35] Mohamed Grissa, Bechir Hamdaoui, and Attila A Yavuz. Location privacy in cognitive radio networks: A survey. *IEEE Communications Surveys & Tutorials*, 2017.

[36] B. Groza, S. Murvay, A. V. Herrewege, and I. Verbauwhede. LiBrA-CAN: A lightweight broadcast authentication protocol for controller area networks. In *Cryptology and Network Security*, pages 185–200, 2012.

[37] Chaowen Guan, Aziz Mohaisen, Zhi Sun, Lu Su, Kui Ren, and Yaling Yang. When smart tv meets crn: Privacy-preserving fine-grained spectrum access. In *Distributed Computing Systems (ICDCS), 2017 IEEE 37th International Conference on*, pages 1105–1115. IEEE, 2017.

[38] David Gurney, Greg Buchwald, Larry Ecklund, Steve L Kuffner, and John Grosspietsch. Geo-location database techniques for incumbent protection in the tv white space. In *New Frontiers in Dynamic Spectrum Access Networks, 2008. DySPAN 2008. 3rd IEEE Symposium on*, pages 1–9. IEEE, 2008.

[39] K. Han, S. D. Potluri, and K. G. Shin. On authentication in a connected vehicle: Secure integration of mobile devices with vehicular networks. In *ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS)*, pages 160–169, 2013.

[40] International Organization for Standardization. ISO/IEC 11898-1:2015: Road vehicles - Controller area network (CAN) - Part 1: Data link layer and physical signalling. Standard.

[41] K. H. Johansson, M. Törngren, and L. Nielsen. Vehicle applications of controller area network. In *Handbook of Networked and Embedded Control Systems*, pages 741–765. 2005.

[42] J. Katz and A. Lindell. Aggregate message authentication codes. *Topics in Cryptology–CT-RSA*, pages 155–169, 2008.

[43] P. Kleberger, T. Olovsson, and E. Jonsson. Security aspects of the in-vehicle network in the connected car. In *2011 IEEE Intelligent Vehicles Symposium (IV)*, pages 528–533, 2011.

[44] V. Kolesnikov and W. Lee. MAC aggregation protocols resilient to DoS attacks. *International Journal of Security and Networks*, 7(2):122–132, 2012.

[45] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, and S. Savage. Experimental security analysis of a modern automobile. In *IEEE Symposium on Security and Privacy*, pages 447–462, 2010.

[46] Simon Kramer, Dirk Ziegenbein, and Arne Hamann. Real world automotive benchmarks for free. In *6th International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS)*, 2015.

[47] He Li, Yanzhi Dou, Chang Lu, Doug Zabransky, Yaling Yang, and Jung-Min Park. Preserving incumbent users' location privacy in the 3.5 ghz band. In *Dynamic Spectrum Access Networks (DYSPAN), 2018 IEEE International Symposium on*. IEEE, 2018.

[48] Wenqiang Li, Guanghao Jin, Xuewen Cui, and Simon See. An evaluation of unified memory technology on nvidia gpus. In *2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pages 1092–1098. IEEE, 2015.

[49] Ben Lynn. PBC: Pairing-based cryptography. https://crypto.stanford.edu/pbc/.

[50] Spyros Makridakis and Michele Hibon. Arma models and the box–jenkins methodology. *Journal of Forecasting*, 16(3):147–163, 1997.

[51] C. Miller and C. Valasek. Remote exploitation of an unaltered passenger vehicle. *Black Hat USA*, 2015.

[52] P.-S. Murvay, A. Matei, C. Solomon, and B. Groza. Development of an AUTOSAR compliant cryptographic library on state-of-the-art automotive grade controllers. In

11th IEEE International Conference on Availability, Reliability and Security (ARES), pages 117–126, 2016.

[53] National Highway Traffic Safety Administration. Cybersecurity best practices for modern vehicles. *No. DOT HS 812*, 333, 2016.

[54] Edmund B. Nightingale, Peter M. Chen, and Jason Flinn. Speculative execution in a distributed file system. In *Proceedings of the Twentieth ACM Symposium on Operating Systems Principles (SOSP)*, pages 191–205, 2005.

[55] D. K. Nilsson, U. E. Larson, and E. Jonsson. Efficient in-vehicle delayed data authentication based on compound message authentication codes. In *IEEE 68th Vehicular Technology Conference*, pages 1–5, 2008.

[56] ON Semiconductor. Ultra-low power, AT command controlled, sigfox compliant transceiver IC for up-link and down-link. URL https://www.onsemi.com/pub/Collateral/AX-SIGFOX-D.PDF. Accessed: August 1, 2018.

[57] A. Perrig, R. Canetti, J. D. Tygar, and D. Song. The TESLA broadcast authentication protocol. *RSA Cryptobytes*, 5, 2005.

[58] Lee Rainie, Sara Kiesler, Ruogu Kang, Mary Madden, Maeve Duggan, Stephanie Brown, and Laura Dabbish. Anonymity, privacy, and security online. *Pew Research Center*, 5, 2013.

[59] U. Raza, P. Kulkarni, and M. Sooriyabandara. Low power wide area networks: An overview. *IEEE Communications Surveys Tutorials*, 19(2):855–873, 2017.

[60] Christian P Robert. *Monte carlo methods*. Wiley Online Library, 2004.

[61] Robert A. Rohde and Richard A. Muller. Air pollution in china: Mapping of concentrations and sources. *PLOS ONE*, 10(8):1–14, 2015.

[62] E. Ronen, A. Shamir, A. O. Weingarten, and C. O'Flynn. IoT goes nuclear: Creating a ZigBee chain reaction. In *IEEE Symposium on Security and Privacy (S&P)*, pages 195–212, 2017.

[63] Hossein Shafagh, Anwar Hithnawi, Lukas Burkhalter, Pascal Fischli, and Simon Duquennoy. Secure sharing of partially homomorphic encrypted iot data. In *Proceedings of the 15th ACM Conference on Embedded Network Sensor System*, 2017.

[64] Sigfox. Technical overview. URL https://www.disk91.com/wp-content/uploads/2017/05/4967675830228422064.pdf. Accessed: August 1, 2018.

[65] Society of Automotive Engineers. J2056/1: Class C application requirement considerations. *SAE Handbook*, pages 23–366, 1993.

[66] R. Soja. Automotive security: From standards to implementation. URL https://www.nxp.com/docs/en/white-paper/AUTOSECURITYWP.pdf. Accessed: August 1, 2018.

[67] C. Szilagyi and P. Koopman. A flexible approach to embedded network multicast authentication. In *Proceedings of the 2nd Workshop on Embedded Systems Security (WESS)*, 2008.

[68] K. Tindell and A. Burns. Guaranteeing message latencies on control area network (CAN). In *Proceedings of the 1st International CAN Conference*. Citeseer, 1994.

[69] H. Ueda, R. Kurachi, H. Takada, T. Mizutani, M. Inoue, and S. Horihata. Security authentication system for in-vehicle network. *SEI Technical Review*, (81), 2015.

[70] Gang Wang, H Chen, Youming Li, and Ming Jin. On received-signal-strength based localization with unknown transmit power and path loss exponent. *IEEE Wireless Communications Letters*, 1(5):536–539, 2012.

[71] H. Wang and A. O. Fapojuwo. A survey of enabling technologies of low power and long range machine-to-machine communications. *IEEE Communications Surveys Tutorials*, 19(4):2621–2639, 2017.

[72] S. Woo, H. J. Jo, and D. H. Lee. A practical wireless attack on the connected car and security protocol for in-vehicle CAN. *IEEE Transactions on Intelligent Transportation Systems*, 16(2):993–1006, April 2015.

[73] G. M. Zago and E. P. de Freitas. A quantitative performance study on CAN and CAN FD vehicular networks. *IEEE Transactions on Industrial Electronics*, 65(5):4413–4422, 2018.

[74] K. C. Zeng, Y. Shu, S. Liu, Y. Dou, and Y. Yang. A practical GPS location spoofing attack in road navigation scenario. In *Proceedings of the 18th International Workshop on Mobile Computing Systems and Applications*, pages 85–90, 2017.

[75] Long Zhang, Chenliaohui Fang, Yi Li, Haojin Zhu, and Mianxiong Dong. Optimal strategies for defending location inference attack in database-driven crns. In *Communications (ICC), 2015 IEEE International Conference on*, pages 7640–7645.

[76] Zhikun Zhang, Heng Zhang, Shibo He, and Peng Cheng. Achieving bilateral utility maximization and location privacy preservation in database-driven cognitive radio networks. In *Mobile Ad Hoc and Sensor Systems (MASS), 2015 IEEE 12th International Conference on*, pages 181–189. IEEE, 2015.