Toward Privacy-Preserving and Secure Dynamic Spectrum Access

Yanzhi “Aaron” Dou

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

Computer Engineering

November 9, 2017

Blacksburg, Virginia

Keywords: Dynamic spectrum access, Cognitive radio, Privacy-preserving system, Malware detection, Machine learning, Homomorphic encryption, Proxy re-encryption

Toward Privacy-Preserving and Secure Dynamic Spectrum Access

Yanzhi "Aaron" Dou

(ABSTRACT)

Dynamic spectrum access (DSA) technique has been widely accepted as a crucial solution to mitigate the spectrum scarcity problem. Spectrum sharing between the government incumbents and commercial wireless broadband operators/users is one of the key forms of DSA. Two categories of spectrum management methods for shared use between incumbent users (IUs) and secondary users (SUs) have been proposed, i.e., the server-driven method and the sensing-based method. The server-driven method employs a central server to allocate spectrum resources while considering incumbent protection. The central server has access to the detailed IU operating information, and based on some accurate radio propagation model, it is able to allocate spectrum following a particular access enforcement method. Two types of access enforcement methods – exclusion zone and protection zone – have been adopted for server-driven DSA systems in the current literature. The sensing-based method is based on recent advances in cognitive radio (CR) technology. A CR can dynamically identify white spaces through various incumbent detection techniques and reconfigure its radio parameters in response to changes of spectrum availability. The focus of this dissertation is to address critical privacy and security issues in the existing DSA systems that may severely hinder the progress of DSA's deployment in the real world.

Firstly, we identify serious threats to users' privacy in existing server-driven DSA designs and propose a privacy-preserving design named $\boldsymbol{P}^2$-SAS to address the issue. $\boldsymbol{P}^2$-SAS realizes the complex spectrum allocation process of protection-zone-based DSA in a privacy-preserving way through Homomorphic Encryption (HE), so that none of the IU or SU operation data would be exposed to any snooping party, including the central server itself.

Secondly, we develop a privacy-preserving design named IP-SAS for the exclusion-zone-based server-driven DSA system. We extend the basic design that only considers semi-honest adversaries to include malicious adversaries in order to defend the more practical and complex attack scenarios that can happen in the real world.

Thirdly, we redesign our privacy-preserving SAS systems entirely to remove the somewhat-trusted third party (TTP) named Key Distributor, which in essence provides a weak proxy re-encryption online service in $\boldsymbol{P}^2$-SAS and IP-SAS. Instead, in this new system, RE-SAS, we leverage a new crypto system that supports both a strong proxy re-encryption notion and MPC to realize privacy-preserving spectrum allocation. The advantages of RE-SAS are that it can prevent single point of vulnerability due to TTP and also increase SAS's service performance dramatically.

Finally, we identify the potentially crucial threat of compromised CR devices to the ambient wireless infrastructures and propose a scalable and accurate zero-day malware detection system called GuardCR to enhance CR network security at the device level. GuardCR leverages a host-based anomaly detection technique driven by machine learning, which makes it autonomous in malicious behavior recognition. We boost the performance of GuardCR in terms of accuracy and efficiency by integrating proper domain knowledge of CR software.

Toward Privacy-Preserving and Secure Dynamic Spectrum Access

Yanzhi “Aaron” Dou

(GENERAL AUDIENCE ABSTRACT)

With the rapid development of wireless technologies in recent years, wireless spectrum which all the wireless communication signals travel over is becoming the bottleneck of the fast growing wireless market. The spectrum scarcity problem is largely due to the current spectrum allocation scheme. Some spectrum bands, like the cellular bands, are overly crowded, while some government-held spectrum bands are used inadequately. By allowing users from the crowded spectrum bands to dynamically access to those less frequently used spectrum bands, the spectrum scarcity problem can be significantly alleviated. However, there are two critical issues that hinder the application of dynamic spectrum access in the real world: privacy and security. For privacy, in order to determine when, where, and how the spectrum can be reused, users need to bear the risk of sharing their sensitive operation data. This is especially frustrating for governmental and military parties whose operation data is highly classified. We solve the privacy problem by designing a privacy-preserving dynamic spectrum access system. The system is based on secure multi-party computation, which keeps users' input operation data private when performing spectrum allocation computation over those inputs. The system achieves 128-bit industry-level security strength, and it is also computation and memory efficient for real-world deployment. For security, dynamic spectrum access requires radio devices to contain many software components so that the radio devices can be dynamically programmed to access different spectrum bands. However, the software also exposes the radio devices to the risk of malware infection. We develop a malware detection system to capture the anomalous behaviors in radio software executions. By adopting advanced machine learning techniques, our system is even able to detect first-seen malware.

To the most significant people in my life.

My parents Runxi Dou, Xiaoxia Zhang, my sister Yanxin Dou,

who offered unconditional love and support for me.

Acknowledgments

My doctoral journey at Virginia Tech has been an extremely rewarding experience. I have had the opportunity to work with the best minds in the most beautiful college town in America. There are many people that I would like to acknowledge.

First, I must thank my advisor, Prof. Yaling Yang, a constant source of knowledge and inspiration for me. She was instrumental in arranging my admission, enrollment, and funding in our program, supervising my research projects with great wisdom and insights. In particular, the advice and encouragement she gave me after my final defense has been in my mind ever since. I will always be grateful to her for believing in me.

My dissertation committee members, Prof. Y. Thomas Hou, Prof. Wenjing Lou, Prof. Patrick R. Schaumont, and Prof. Kui Ren, all spent precious time in guiding my research progress and providing thoughtful suggestions, without which I would not have finished.

I must express my unending gratitude to my undergraduate advisors, Prof. Wei Chen and Dr. Bo Bai, who led me onto the research path, including publishing my first paper. I would also like to thank my collaborators, Prof. Danfeng Yao, He Li, Chang Lv, Kapil Kale, Douglas Zabransky, Kexiong Zeng, Bo Gao, Chang Liu, and Jinshan Liu, for all the rich brainstorming and discussions.

I especially thank Kexiong, since we began our Ph.D. journey together, in the same lab, at the same time, taking classes together, staying up countless nights, beating deadlines, and surviving ... all together.

All of my friends who made my life colorful and memorable: Jianxun Wang, Daren Chen, Sijing Guo, Yiming Zhang, Rui Sun, Duotong Yang, Hao Wang, Minjun Chen, and Long Xia. Life wouldn't be nearly as exciting without you all in my life.

Finally, and most importantly, I would like to thank my parents for their unconditional and boundless love and support. This dissertation is dedicated to them with my utmost love and respect.

Grant Information

This dissertation was supported by the National Science Foundation (NSF) under grants CNS-1228903, CNS-1054697, CNS-1547366. Any opinions, results and conclusions or recommendations expressed in this material and related work are those of the author(s) and do not necessarily reflect the views of the National Science Foundation (NSF).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The vigorous development of wireless communication technologies brings about enormous novel services, which have profoundly changed modern society, including communication, healthcare, education, entertainment, and transportation. For example, implants and wearables can alert a doctor to a patient's changing health by accurately and speedily delivering physiological measurements on a continuous basis. Another example is car-to-car communication, which can help to significantly reduce car crashes, especially in blind spot cases. All these novel wireless services consume radio spectrum, which is, unfortunately, a limited and scarce resource. With more and more novel services coming out, there is a common belief that the radio spectrum would be depleted soon. Until now, the U.S. spectrum allocation chart has already been overly crowded 1.1. However, thanks to the proposal of the Dynamic Spectrum Access (DSA) technique, the spectrum scarcity problem can be mitigated to a great extent. In this chapter, we first introduce the concept of DSA, then expose the privacy and security challenges in the existing DSA systems. We then present our research contributions toward addressing these challenges and finally outline the roadmap of this dissertation.

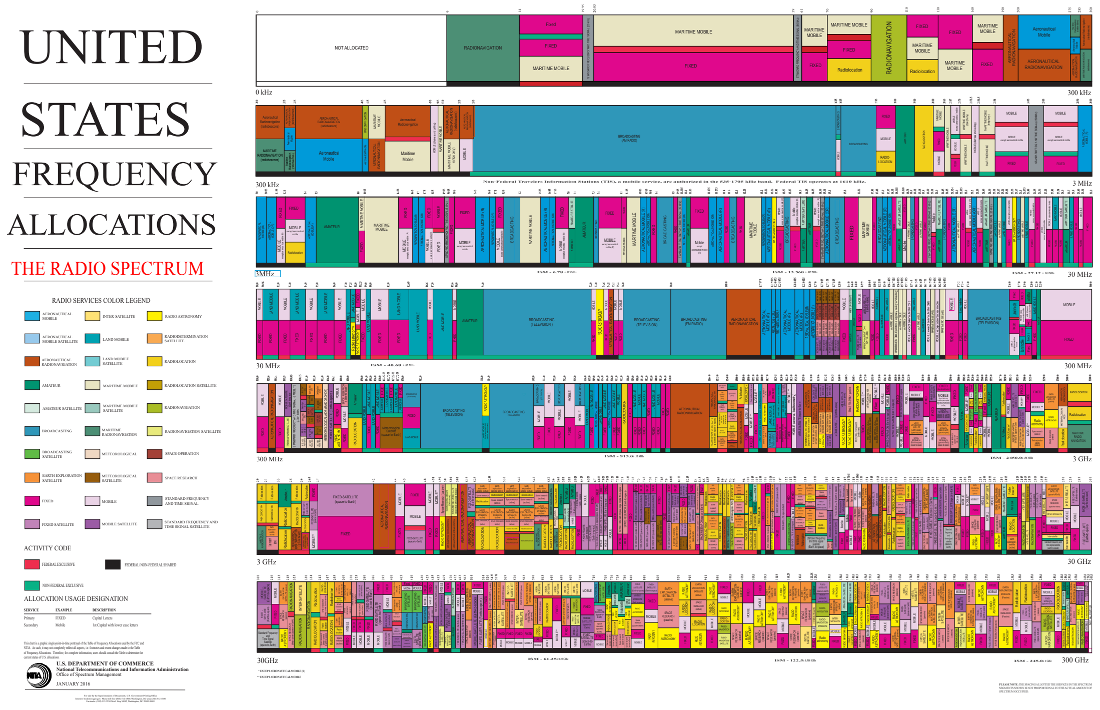


Figure 1.1: The overly crowded U.S. spectrum allocation chart. Source: NTIA

1.1 Dynamic Spectrum Access

Current wireless networks are regulated by a static spectrum allocation policy. According to the policy, radio spectrum is regulated by governmental agencies, such as the Federal Communications Commission (FCC) in the U.S. and the Office of Communications (Ofcom) in the U.K. These regulation agencies allocate spectrum to licensees for exclusive use in large geographical area on a long term basis. This static spectrum allocation policy served well in the past. However, with the rapid development of wireless communication in recent years, especially the prevalence of mobile devices and the burgeoning growth of Internet of Things (IoTs), there is a dramatically increasing demand of access to the limited spectrum for mobile services, which stimulates people to question the effectiveness of the current spectrum allocation policy. According to FCC's Spectrum Policy Task Force report [1], the current static spectrum allocation policy is inefficient since a significant portion of the spectrum is underutilized. Actual spectrum usage measurements in the report show that the utilization of the allocated spectrum varies from 15% to 85% in temporal and geographical dimension.

To solve the spectrum inefficiency problem and explore the existing spectrum opportunities, the DSA technique is proposed. Standing for the opposite of the current static spectrum allocation scheme, in the DSA scheme, unlicensed secondary users (SUs) are allowed to reuse the licensed spectrum under the constraint that SUs do not impose harmful interference to incumbent users (IUs). This can greatly improve spectrum utilization and extend regular services [2, 3]. Spectrum sharing between government incumbents (i.e. federal or non-federal agencies) and commercial wireless broadband operators/users is one of the key forms of DSA that is recommended by the FCC [4] and the National Telecommunications and Information Administration (NTIA) [5, 6]. Recommendations in the President's Council of Advisors on Science and Technology (PCAST) report have identified 1,000 MHz of federal spectrum to create "the first shared-use spectrum superhighways" [7].

Two categories of spectrum management methods for shared use between IUs and SUs have been proposed: the server-driven method and the sensing-based method. The server-driven method employs a central server to allocate spectrum resources while considering incumbent protection. The central server has access to the detailed operating information of IUs, such as geo-location, operating frequency, and protection requirement. Based on the up-to-date incumbent characteristics and some accurate radio propagation model, the central server is able to allocate spectrum following a particular access enforcement method. Two types of access enforcement methods – exclusion zone and protection zone – have been explored for server-driven DSA systems in the current literature [8, 9, 10]. The exclusion zone method introduces the notion of exclusion zone, which is defined as a spatial region around an IU where SUs are forbidden to operate to avoid interfering with the IU. For the protection zone method, interference from multiple SUs can be accumulated. An SU's operation is permitted as long as the accumulated interference will not exceed any IU's interference sensitivity threshold. The server-driven method has been recommended by the PCAST report and is also reflected in the FCC's recent proposal for DSA in 3.5 GHz [8]. In these official documents, the central sever for spectrum management is named as a "Spectrum Access System" (SAS). In Europe, a similar server-driven DSA system named "Licensed Shared Access" (LSA) is also being developed, and 2.3-2.4 GHz band has been identified for an initial deployment of LSA. Without loss of generality, we refer to the server-driven DSA system as SAS in the remainder of this dissertation.

In comparison to the server-driven method, the sensing-based method is based on recent advances in cognitive radio (CR) technology [11]. A CR refers to an intelligent radio that can be programmed and configured dynamically. It can sense the surrounding spectrum environment and adjust its radio parameters accordingly. In the DSA scenario, a CR can dynamically identify white spaces (spectrum holes) through various incumbent detection techniques and reconfigure its radio parameters (e.g., operating frequency, transmit power etc.) in response to the changes in spectrum availability. Proven to provide drastically improved spectrum utilization, CR has been called out in the U.S. Federal government's National Broadband Plan as one of the major technologies to provide broadband services to millions of US residences [12].

1.2 Privacy and Security Issues in DSA

While the benefits of DSA are obvious and its future is promising, it also brings along unique security and privacy issues. This dissertation focuses on two crucial and challenging issues, which would severely hinder the progress of DSA if left unaddressed. The first issue is users' privacy issue in SAS. In current SAS designs, the operation data of both government IUs and commercial SUs needs to be shared with SAS. However, the operation data of government IUs is often classified information and the SU operation data may also be commercial secrets. The current system design dissatisfies the privacy requirement of both IUs and SUs since SAS is not necessarily trust-worthy for holding such sensitive operation data.

The second issue is CR security. Due to CR's software-oriented design, a CR can be exploited by adversaries through levering the inherent vulnerabilities of software to launch large scale attacks on a wide range of critical wireless infrastructures at low cost. This potential threat of a compromised CR makes it much more damaging than a traditional radio and its security issues must be addressed proactively before this technology becomes widespread. In the following subsections, we present the details of these two issues.

1.2.1 Users' Privacy in SAS

One of the critical concerns in light of the increasing prospects of the SAS-driven spectrum sharing is the privacy issue [8]. For national security reasons, operation 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 [8]. In Europe, the IUs of the 2.3-2.4 GHz LSA band include military aircraft services and police wireless communications [13]. These IUs' operation data is highly sensitive. Similarly, SUs' operation parameters may also be sensitive commercial secrets for their operators. It is highly likely that SU network operators will be reluctant to share their base stations' deployment and configuration strategies.

Yet, to realize efficient spectrum access, current SAS-driven designs require IUs and SUs to send their operation data to SAS for spectrum allocation. It exposes IUs and SUs to potentially severe privacy violation since SAS is not necessarily trust-worthy for holding such sensitive operation data. For example, according to FCC and PCAST reports [7, 14], SAS may be operated by some commercial third parties to enhance its efficiency and scalability. In fact, Google has already developed the third generation of its 3.5GHz SAS prototype [15].

Even if the operator of SAS is trusted, it may be breached by adversaries (e.g., intrusions, malwares, insider attacks). In such cases, adversaries will have access to all IU and SU operation information. In essence, how to protect IU and SU operation privacy from SAS becomes a critical challenge that can potentially deter the wide adoption of DSA technology.

1.2.2 CR Security

A traditional radio can only affect its own network because its operation spectrum is tightly constrained by its fixed hardware design. The hardware design is typically approved by the FCC and is very difficult to compromise without physical access to the hardware. Thus, while malware may affect these traditional radios, the attack cost is high and the impact is limited to a single wireless system at limited spectrum bands. CR, however, has flexible software-oriented design that can be configured to opportunistically access available spectrum in an expansive spectrum range. Thus, by exploiting the inherent vulnerability of CR software, an adversary can remotely control CR devices to launch attacks on critical wireless systems at a wide range of spectrum bands, including health care and emergency communication systems. This threat is even more serious when CR becomes widely deployed. In such a case, an adversary can compromise and control a large number of CR devices to launch large-scale attacks on nationwide wireless infrastructures. This potential threat makes compromised CR systems much more destructive than compromised traditional radio systems. In addition, considering the increasing diversity of CR platforms and applications, the number of potential attack points is growing, making it relatively easy for an adversary to find new vulnerabilities and develop new attacks to compromise a CR system at low costs. Therefore, unlike traditional wireless systems, where it may be acceptable to consider security as an afterthought, security issues of CR need to be addressed proactively for this technology to pass regulation requirement and become widely deployed.

Existing work for CR security falls into three categories. The first category seeks to prevent downloading malicious software into CR systems through the use of digital signature and encryption [16, 17]. The second category leverages the ordinary protection measures for host computers to protect CR systems from malicious software [18, 19]. The third category of methods focuses on the networking aspect of CR security, where mechanisms are designed to ensure that CR networks can resist attacks from compromised CR nodes [20, 21, 22].

1.3 Contributions

This dissertation identifies critical emerging privacy and security issues of DSA and provides effective solutions to address these issues. The contributions are summarized as follows.

1 **Privacy-Preserving Protection-Zone-Based SAS:** To address the privacy issue in the protection-zone-based SAS, we design P^2-SAS, which realizes the complex spectrum allocation process of protection-zone-based DSA through efficient secure multi-party computation (MPC). In P^2-SAS, none of the IU or SU operation data would be exposed to any snooping party, including to SAS itself. To realize the complex computation of protection zone access enforcement policy accurately and efficiently in MPC, we separate the parameters of a radio propagation model that are privacy-related from those that are public knowledge. Only the part of radio propagation model that involves private parameters needs to be performed by MPC. Then, we disintegrate the

computation related to private parameters into a small set of basic operations that can be homomorphically computed using the Paillier cryptosystem [23]. We formally prove the correctness and privacy-preserving property of $\boldsymbol{P}^2$-SAS. To make $\boldsymbol{P}^2$-SAS practical in real-world scenarios, we explore various means to refine the system. We investigate the tradeoff between accuracy and efficiency of $\boldsymbol{P}^2$-SAS computation, and we formulate it into an optimization problem to search for the optimal $\boldsymbol{P}^2$-SAS parameter setting. We propose practical acceleration methods to improve $\boldsymbol{P}^2$-SAS's efficiency and demonstrate the scalability and practicality of $\boldsymbol{P}^2$-SAS using experiments based on real-world data. Experiment results show that $\boldsymbol{P}^2$-SAS can respond an SU's spectrum request in 6.96 seconds with communication overhead of less than 4 MB.

2 **Privacy-Preserving Exclusion-Zone-Based SAS:** To address the privacy issue in the exclusion-zone-based SAS, we design IP-SAS based on MPC. Our IP-SAS realizes exclusion-zone-based spectrum allocation in SAS without exposing any information that can potentially lead to privacy violation. To prevent malicious parties from compromising the IP-SAS system, we extend the basic design to make IP-SAS robust against more complex and realistic malicious attacks. The task of countering malicious attacks is challenging due to IP-SAS's privacy-preserving design, since one cannot easily tell cheating behaviors from information discrepancy. To tackle the challenge, we manipulate the plaintext space to accommodate additive zero-knowledge proof in the secure SAS computation. We leverage the unique properties of IP-SAS computation to develop acceleration mechanisms, which significantly reduce IP-SAS's computation and communication overhead. We conduct extensive experiments based on real-world data to evaluate IP-SAS, and the results show that IP-SAS can respond an SU's spectrum request in 1.25 seconds with communication overhead of 17.8 KB.

3 **Privacy-Preserving SAS Without Key Distributor:** The design of the $\boldsymbol{P}^2$-SAS

and IP-SAS leverage an online somewhat-trusted third party, Key Distributor, to provide a weak proxy re-encryption service in the spectrum allocation process. The downsides of the design is that it brings the risk of single point vulnerability and also requires extra infrastructure for Key Distributor. To remove Key Distributor, we redesign the privacy-preserving SAS and propose RE-SAS. RE-SAS adopts the AFGH cryptosystem [24] to realize a strong notion of proxy re-encryption. And it also supports homomorphic encryption to enable secure spectrum allocation computations. In fact, to the best of our knowledge, we are the first to discover the homomorphic property of AFGH scheme, and we are also the first to use the homomorphic proxy re-encryption scheme to achieve a privacy preserving system design. We rigorously define and analyze the security of ER-SAS using standard security experiments. Evaluation results show that RE-SAS is able to handle a spectrum request in 2.59 ms on average, which is more than two orders of magnitude faster than IP-SAS.

4 **Malware Detection for CR:** To proactively mitigate the potentially serious CR security issue, we design GuardCR, a scalable and accurate zero-day malware detection system customized for CR software. GuardCR detects malware by first learning the normal behaviors of a CR application through training, and then detecting anomaly behaviors deviating from the normal behaviors. Based on the unique properties of CR systems, we integrate proper domain knowledge for better detection performance. To the best of our knowledge, this is the first work that enhances the security of CR networks at the device level through dynamic behavior monitoring. To evaluate GuardCR, we propose a general experimental method to automatically generate artificial malware cases with varied malicious behaviors based on mutation testing. We implement a prototype of GuardCR for CR applications on GNU Radio and USRP. Evaluation results show that GuardCR is able to detect malicious behaviors within 1.10 second at an

accuracy of 94.9%.

1.4 Dissertation Organization

The remainder of this dissertation is organized as follows.

- Chapter 2 presents $\boldsymbol{P}^2$-SAS, a privacy-preserving protection-zone-based SAS design, which guarantees that no snooping parties can obtain the private operation data of IUs and SUs during the spectrum allocation process.
- Chapter 3 introduces IP-SAS, a privacy-preserving design for the exclusion-zone-based SAS. IP-SAS is even robust in the more complex and realistic malicious attack scenarios.
- Chapter 4 presents ER-SAS, which is a privacy-preserving SAS design without the involvement of the somewhat-trusted third party Key Distributor.
- Chapter 5 presents GuardCR, a scalable and accurate malware detection system to enhance the CR network security at device level.
- Chapter 6 draws the conclusions and points out the future research directions.

Chapter 2

P^2-SAS: Privacy-Preserving Protection-Zone-Based SAS

In this chapter, we present the design of our privacy-preserving protection-zone-based SAS, a.k.a. P^2-SAS [25, 21, 26].

2.1 Challenges and Contributions

The goal of this chapter is to fundamentally address the privacy challenges in the protection-zone-based SAS by developing a privacy-preserving SAS system named P^2-SAS. Through an efficient multi-party computation (MPC) design, P^2-SAS guarantees that no snooping entities, including SAS itself, can obtain any information about SU and IU operation data during the entire DSA process.

MPC is a well-known technique for secure computation. It allows multiple parties to jointly compute a function over their inputs, while keeping these inputs, the intermediate computation results and the outputs private. However, designing an MPC-based SAS is a non-trivial task. MPC for general function computation is currently not practical due to its huge computation complexity. In the security community, customized MPC solutions for specific applications, such as distributed voting, private bidding and auctions, and private information retrieval [27, 28, 29], have been explored recently. These customized solutions are much more efficient than general-purpose MPC. Unfortunately, none of them is able to realize MPC for SAS. This is because SAS demands very complex computations that are significantly different from those applications explored in existing literatures. Specifically, designing MPC for SAS encounters the following challenges:

(1) To ensure accurate access enforcement process in DSA, protection-zone-based SAS usually adopts complex radio propagation models for interference calculation, e.g., Longley-Rice (L-R) model [30]. These models take multiple parameters, such as IU/SU's geolocation, IU/SU's antenna configuration, terrain data, weather, and soil condition, into some sophisticated math computations. These math computations involve multiple trigonometric, logarithmic, exponential, comparative, multiplicative and additive operations. Realizing such complex computations in MPC incurs huge computation and communication overhead.

(2) SAS needs to ensure that SUs' operation will not disturb any IU. Specifically, to decide whether to approve or deny an SU's spectrum access request, SAS needs to compute whether the accumulated interference from all the SUs will exceed any IU's interference threshold when this SU starts to operate. Achieving the above procedure in MPC needs to compare integers in a secure way. Yet, secure integer comparison is a known complex problem and no existing solution is fast and practical enough for large scale computation [31].

(3) If an SU's spectrum access request is approved, SAS needs to issue a license that permits the SU to access the spectrum in a certain pattern (e.g., location, antenna height, transmit power, etc). To ensure privacy, the above procedures, including the response of permitting

the SU or not, and the permissible SU operation parameters in the license, must all remain private in MPC. Yet, the yes/no response and the license's content all need to be digitally signed to prevent forging attempts at the SU side. However, no existing literature has ever provided efficient digital signature generation using MPC.

At first glance, the above challenges for designing an MPC-based SAS seem daunting. In this chapter, we make the following contributions towards completing this difficult mission.

• On a high level, we separate the parameters in a radio propagation model that need privacy protection from those that are public knowledge. Only the part of radio propagation model that involves private parameters needs to be performed by MPC. Then, we disintegrate the computation related to private parameters into a small set of basic operations that can be homomorphically computed using Paillier cryptosystem [23]. In addition, we precompute the non-private part, which transforms this part of computation into simple lookup for further efficiency improvement.

• By investigating the properties of integers involved in SAS's computation, we employ a clever way for integer encoding to circumvent the complex secure integer comparison problem, yet we can still obtain the integer comparison results securely.

• We realize the computation of spectrum license and its digital signature generation in MPC by an innovative combination of radio operation's observable nature with digital signature's integrity property.

• We explore various means to make P^2-SAS practical. We investigate the tradeoff between accuracy and efficiency of P^2-SAS in interference computation, and we formulate it into an optimization problem to search for the optimal P^2-SAS parameter setting. We propose practical acceleration methods to improve P^2-SAS's efficiency, and we demonstrate the scalability and practicality of P^2-SAS using experiments based on real-world data.

The remainder of this chapter is organized as follows. Section 2.2 discusses some related work. Section 2.3 describes the system and adversary model, states our design goals, and introduces some preliminaries. Section 2.4 describes the basic design of P^2-SAS. Section 2.5 presents several refinement techniques on the basic design for better accuracy and efficiency. Section 2.6 evaluates P^2-SAS. Section 2.7 summarizes this chapter.

2.2 Related Work

2.2.1 Privacy in SAS-Driven Systems

In [32], private information retrieval (PIR) techniques are employed to protect SU's location privacy against untrusted SAS. However, in [32], only SU's privacy protection is considered, while in many DSA cases involving government-commercial sharing, IU's privacy is a more critical concern that needs to be preferentially addressed.

In [33], an inference attack is identified where a malicious SU can derive IUs' operation information by examining the returned spectrum access permissions from SAS. To counter this attack, obfuscation techniques are adopted to introduce noises in the response to the SU's spectrum query, so that IUs' operation information can be somewhat protected from the malicious SU. Although the malicious SUs' threat to IUs' privacy is mitigated in [33], it does not address the privacy threat from untrusted SAS. Other related work about security of SAS-driven DSA systems aim at location spoofing [34] and secondary user faking [35].

2.2.2 Multi-party computation (MPC)

The secure MPC problem was introduced by Yao in [36]. In general, secure MPC allows a set of n parties, each with a private input, to securely and jointly compute the output of an n-party function f over their inputs. Theoretically, the general secure multi-party computation problem is solvable using circuit evaluation protocol [37]. While this approach seems appealing in its generality, the communication complexity of the protocol depends on the size of the circuit that describes the function f, and in addition, involves large constant factors in their complexity. In recent years, fully homomorphic encryption (FHE), which can homomorphically perform both additive and multiplicative operations, are also being proposed for general MPC. The first FHE scheme was proposed in 2009 in [38]. Although research in FHE has made tremendous progress in improving efficiency [39], it is still far from practical for most of the real-world applications. Therefore, as Goldreich pointed out in [37], since these general solutions can be impractical for many MPC problems, special solutions should be developed for special cases for efficiency reasons. DSA problem, due to its complex nature, belongs to the type that cannot be efficiently solved by the general method. The purpose of this chapter is to find a customized solution that is much more efficient than the general theoretical solutions.

2.3 Problem Statement

2.3.1 System Model

We consider a protection-zone-based SAS involving three parties, as illustrated in Figure 2.1: IUs, SUs, and SAS Server. SAS Server refers to a central spectrum management infrastructure that allocates spectrum resources while considering incumbent operation protection

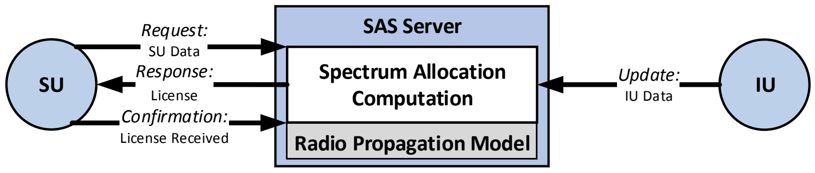


Figure 2.1: The system model of protection-zone-based SAS

from interference. A typical scenario of the system is described as follows. Firstly, IUs update SAS Server with their operation data, such as location, interference sensitivity threshold, and antenna height. Then, any SU that needs the spectrum must send SAS Server a request for spectrum access along with its operation data. SAS Server employs the protection zone access enforcement method to determine if the SU is allowed to operate. Specifically, based on some accurate radio propagation model, SAS Server computes whether the accumulated interference from all the SUs will exceed any IU's interference sensitivity threshold when this SU starts to operate. If the answer is yes, a response to the SU denies its request. If the answer is no, a response to the SU permits its spectrum access with a license. Finally, the SU sends a confirmation message to SAS Server to confirm the reception of the response. We focus on the protection zone approach in this chapter because it can release more frequency opportunities than the exclusion zone approach [40], and thus is considered as the future trend for interference management [41].

2.3.2 Adversary Model & Design Goals

We assume SAS Server is semi-honest (a.k.a. honest-but-curious), which means that it acts in an "honest" fashion and exactly follows the protocol design for spectrum allocation, but it is also "curious" and attempts to infer private IU/SU operation data from the information communicated to it.

The goal of P^2-SAS is to realize the SAS process described in Section 2.3.1 correctly, while preserving the IU/SU data privacy from the semi-honest SAS Server. In the following, we formally define correctness and privacy of a SAS scheme in the semi-honest model using the simulation paradigm [42]. Specifically, we denote the computation of the SAS process in Section 2.3.1 as a functionality f, and denote any SAS scheme for computing f as π. Since f is deterministic, correctness and privacy can be defined separately as follows.

Definition 2.3.1. (correctness) We say that π correctly computes f if

$$\{\mathsf{output}_{SUs}(x,y,z,n)\} \overset{c}{\equiv} \{f(x,y,z)\}, \tag{2.1}$$

where x,y,z are the input data from IUs, SUs, and SAS Server respectively, n is the security parameter, and output_{SUs} is the output of SUs during an execution of π. In our scenario, it is the approval/deny information SUs finally obtain. $\overset{c}{\equiv}$ denotes computationally indistinguishability. Intuitively, in this definition, we say a privacy-preserving SAS scheme π is correct if the output of π, which is defined as the approval/deny response sent to an SU, is the same as the original SAS process defined in Section 2.3.1. Basically, this means that π will not alter the spectrum allocation results.

Definition 2.3.2. (privacy) We say that π securely computes f in the presence of semi-honest SAS Server if there exist probablistic polynomial-time algorithms, denoted as S, such that

$$\{S(z,\mathsf{output}_{SAS}(x,y,z,n),n)\} \overset{c}{\equiv} \{\mathsf{view}^{\pi}(x,y,z,n)\}, \tag{2.2}$$

where view is the view of SAS Server during an execution of π, i.e., the transcript of messages that it receives and its internal states. $\mathsf{output}_{SAS}(x,y,z,n)$ is the output of SAS Server during an execution of π, i.e., the data it finally sends to SUs. The definition essentially says that π is secure in the presence of semi-honest SAS Server if SAS Server cannot gain additional

Table 2.1: Paillier cryptosystem

Key generation: $(\mathsf{pk},\mathsf{sk}) = \mathsf{KeyGen}(n)$
1. Choose two large random prime numbers p and q, ensuring that $\gcd(pq,(p-1)(q-1)) = 1$.
2. Compute $n = pq$, $\lambda = \mathrm{lcm}(p-1,q-1)$.
3. Choose random integer g where $g \in \mathbb{Z}^*_{n^2}$. Compute $\mu = \left(L\left(g^\lambda \mod n^2\right)\right)^{-1} \mod n$ if exists, where $L(x) = \frac{x-1}{n}$.
4. The public key pk is (n,g), and the secret key sk is (λ,μ). n is security parameter.

Encryption: $[\![m]\!] = \mathsf{Enc}_{\mathsf{pk}}(m,r) = g^{(m+nr)} \mod n^2$
$[\![m]\!]$ is an encryption of m with a one-time random number r.
Decryption: $m = \mathsf{Dec}_{\mathsf{sk}}([\![m]\!]) = L\left([\![m]\!]^\lambda \mod n^2\right) \cdot \mu \mod n$

Homomorphic properties:
Addition($\oplus$): $\mathsf{Dec}_{\mathsf{sk}}\left([\![m_1]\!] \oplus [\![m_2]\!]\right) = m_1 + m_2$.
Scalar multiplication($\otimes$): $\mathsf{Dec}_{\mathsf{sk}}\left(c \otimes [\![m]\!]\right) = c \cdot m$.
Subtraction($\ominus$): $\mathsf{Dec}_{\mathsf{sk}}\left([\![m_1]\!] \ominus [\![m_2]\!]\right) = m_1 - m_2$.

information in the execution of π beyond the information of its input and output data.

2.3.3 Preliminaries on Paillier Cryptosystem

The design of P^2-SAS heavily leverages the homomorphic properties of Paillier cryptosystem [23]. Paillier cryptosystem efficiently supports homomorphic addition, subtraction and scalar multiplication operations on the ciphertexts, and the generated results, when decrypted, match the results of the operations on the plaintexts. The details are shown in Table 2.1. Note that Enc is a probabilistic algorithm due to the introduction of the random number r while Dec is deterministic, so one plaintext can be encrypted to different ciphertexts while one ciphertext can only be decrypted to one plaintext.

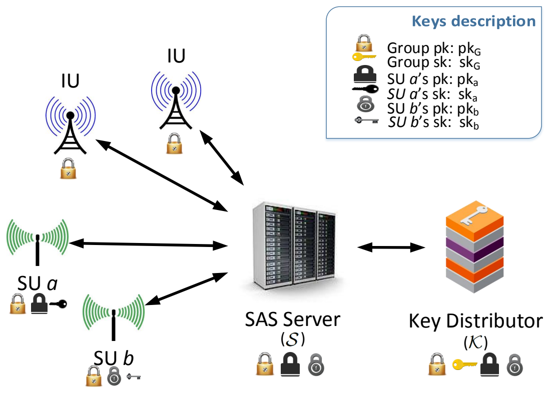


Figure 2.2: P^2-SAS overview.

2.4 Basic P^2-SAS Design

In this section, we present the basic design of P^2-SAS. In Section 2.5, we will introduce several refinement techniques on the basic design to accelerate its execution speed.

2.4.1 P^2-SAS Design Overview

As shown in Figure 2.2, our P^2-SAS design involves four parties: (1) a SAS Server $\mathcal{S}$ for computing spectrum allocation, (2) IUs, (3) SUs, and (4) a Key Distributor $\mathcal{K}$. $\mathcal{K}$ creates a group Paillier public/private key pair $(\mathsf{pk}_G,\mathsf{sk}_G)$. pk_G is distributed to $\mathcal{S}$ and all the users, while sk_G is kept as a secret only known to $\mathcal{K}$. In addition, each SU b has his own pair of Paillier public/private keys $(\mathsf{pk}_b,\mathsf{sk}_b)$, and pk_b is sent to $\mathcal{K}$. Using sk_G and pk_b, $\mathcal{K}$ can provide key conversion service, where it converts a ciphertext encrypted by pk_G to a ciphertext encrypted by pk_b for SU b to decrypt. We assume $\mathcal{K}$ is trusted in keeping sk_G secret only to itself, and $\mathcal{K}$ will not collude with $\mathcal{S}$ to compromise IU/SU operation data. In the real world, $\mathcal{S}$ can be operated by some commercial third party (e.g., Google) for enhanced efficiency and scalability; $\mathcal{K}$ is operated by IUs.

P^2-SAS works as follows. Both IUs and SUs encrypt their operation data using the group public key pk_G before sending to $\mathcal{S}$. Upon receiving SU b's spectrum access request, $\mathcal{S}$ performs secure computation on the encrypted operation data to calculate whether the addition of SU b's interference will exceed any IU's interference threshold. Based on the private spectrum computation results, $\mathcal{S}$ is able to generate a response to SU b. The response includes a ciphertext message encrypted by SU b's individual public key pk_b, which is created by leveraging $\mathcal{K}$'s key conversion service. Upon decrypting the ciphertext message, SU b finds out whether its spectrum request is approved or not. If the request gets approved, SU b also obtains a spectrum access license. The license contains SU b's operation parameter specification and is properly signed by $\mathcal{S}$ to prevent any potential forging or tampering attempt. Finally, SU b sends a confirmation message to $\mathcal{S}$ to confirm its reception of the response. In the entire process, all the IU/SU operation data, and the intermediate and final computation results stay private and are never exposed to any of the SAS components, including $\mathcal{S}$ and $\mathcal{K}$. Table 2.2 gives the description of notation to be used in our scheme.

It is worth mentioning that in some cases, for example 3.5 GHz band, IUs will not provide operation information directly to $\mathcal{S}$ due to policy reasons. Instead, FCC allows one or more environmental sensing capabilities (ESCs) to sense federal IU activity and provide the sensing data to $\mathcal{S}$ for spectrum allocation computation [43]. FCC requires that ESCs should be managed and maintained by a non-governmental entity. And to protect the privacy of federal operations, FCC also requires that ESC does not store or disclose any information on the locations or movements of IUs. We believe that P^2-SAS is readily to support such ESC-based system. Specifically, under P^2-SAS, ESC service can derive the IU operation data from its sensing result, encrypt the operation data and send the ciphertext to $\mathcal{S}$. All these computations can be done in memory and then all sensing data and intermediate results can be deleted from memory immediately after the ciphertext is transmitted. In this

Table 2.2: Notations

Notation	Description
$\mathcal{S}$	SAS Server
$\mathcal{K}$	Key Distributor
$(\mathsf{pk}_G, \mathsf{sk}_G)$	Group key pair
$(\mathsf{pk}_b, \mathsf{sk}_b)$	SU b's individual key pair
$[\![\cdot]\!]$	Encryption by pk_G
$[\![\cdot]\!]_{\mathsf{pk}_b}$	Encryption by pk_b

way, location data of IUs is never stored and retained in ESC. The transmitted ciphertext is encrypted by the group public key pk_G and hence will never be exposed to any entities, satisfying the non-disclosure requirement of FCC as well. From $\mathcal{S}$'s point of view, the same secure computation will be performed regardless of whether the source of encrypted IU operation data is from IU itself or from the ESC services. Thus, P^2-SAS can work well in an ESC-based system. Without loss of generality, we will discuss the design of P^2-SAS assuming the source of IU operation data is IU itself in the remainder of this chapter.

The remainder of this section presents the details of the above P^2-SAS design. First, we describe the content and format of the input data to $\mathcal{S}$. Then, we show how we realize the secure spectrum computation over the input data. Specifically, we firstly introduce the spectrum computation in the plaintext domain, and then we illustrate how each plaintext computation step can be carried out securely in the ciphertext domain.

2.4.2 Private Input Data to SAS

To reduce secure computation overhead, we need to separate the privacy-related parameters in interference calculation from those that are public knowledge. For interference calculation, we adopt the highly-sophisticated Longley-Rice (L-R) model, which is the default model for calculating radio propagation selected by FCC [30] and NTIA [44]. It takes 13 parameters as input, and among these input parameters, only frequency, distance, antenna

Table 2.3: Privacy-related parameters

Parameter	Notation	Quantization level
IU, SU location	l, j	L
IU, SU antenna height	h_I, h_S	H_I, H_S
IU, SU operating frequency	f_I, f_S	F
IU interference threshold	ζ	–
SU maximum transmit power	η	–

height, transmit power, polarization, and terrain data need to be specifically considered. The other parameters are environmental parameters, such as earth dielectric constant, earth conductivity, atmospheric bending constant, climate, etc., which describe the statistics of the environment in which the L-R system is to operate. These environmental parameters are independent of individual users' operation settings and hence need no privacy protection. In addition, polarization only affects the reflectivity of the ground, which is also a known constant when frequency is above 100MHz [30]. Since DSA systems usually consider spectrum sharing in GHz band [7], polarization is also considered to be non-private. Finally, terrain information is public knowledge, which can be easily found in government terrain database, such as USGS [45] and STRM3 [46]. With the above elimination of non-private parameters, all the privacy-related parameters for interference calculation can then be summarized in Table 2.3.

To reduce MPC's computation overhead in the later stages, we next need to represent these private IU/SU operation data of Table 2.3 in proper form. Specifically, we quantize the service area of $\mathcal{S}$ into L equal-sized grids, and express users' location using the grid number. In addition, we quantize IU antenna height into H_I levels and SU antenna height into H_S levels, and we assume there are F frequency bands for spectrum sharing. Based on the above quantization, an IU i's operation data can be represented by a three-dimensional matrix $\mathbf{T}_i := \{T_i(l, h_I, f_I)\}_{L\times H_I\times F}$. If IU i is located in grid l with antenna height of h_I and operating frequency of f_I (a.k.a. operation position of (l, h_I, f_I) in the spatial and spectrum domains), $T_i(l, h_I, f_I)$ is set to IU i's interference threshold ζ. The rest entries of $\mathbf{T}_i$ are set to 0. Similarly, a three-dimensional matrix $\mathbf{R}_b := \{R_b(j, h_S, f_S)\}_{L\times H_S\times F}$ is used to represent an SU b's operation data. If SU b's operation position is (j, h_S, f_S), $R_b(j, h_S, f_S)$ is set to SU b's maximum transmit power η. The rest entries of $\mathbf{R}_b$ are set to 0, indicating that no active transmission exists in these operation positions.

It is worth to note that if an IU worries that malicious SUs may infer its operation data by analyzing multiple SAS's spectrum responses, the IU can add obfuscation noises to its operation data $\mathbf{T}_i$ as follows:

$$T_i(l, h_I, f_I) \leftarrow T_i(l, h_I, f_I) + \phi, \tag{2.3}$$

where ϕ is the noise. Some preliminary noise generation techniques, such as introducing fake IU locations and fake interference thresholds, are proposed in [33] for traditional SAS. Moreover, we can use differential privacy [47] to generate Laplace noise for more rigorous privacy guarantee. Note that these obfuscation techniques for traditional SAS are fully compatible with our P^2-SAS design since they only affect $\mathbf{T}_i$ by noise addition, and the following process of P^2-SAS stays the same. As pointed out by the existing work [33], the potential downside of such obfuscation techniques is the lowered spectrum utilization efficiency due to the added noise. We leave the work of finding the right balance between obfuscation effectiveness and spectrum efficiency for a more comprehensive privacy-preserving solution in the future.

2.4.3 Spectrum Computation in Plaintext

In this subsection, we outline the spectrum computation steps in the plaintext domain to ease the understanding of the secure spectrum computation steps in the ciphertext domain in Section 2.4.4, 2.4.5 and 2.4.6.

I. Initialization:

$\mathcal{S}$ precomputes an attenuation map $\mathbf{I} := \{I\,(l, j, h_I, h_S, f_I, f_S)\}_{L^2\times H_I\times H_S\times F^2}$ based on the public terrain data and L-R model. Entry $I\,(l, j, h_I, h_S, f_I, f_S)$ is set to the path attenuation from an SU with operation position (j, h_S, f_S) to an IU with operation position (l, h_I, f_I).

II. IUs update SAS with their operation data $\mathbf{T}_i$:

To reduce the amount of transmitted information, if multiple IUs have the same operation position in both the spatial and spectrum domains, we assume they will locally coordinate with one another so that only the IU with the smallest interference threshold sends its $\mathbf{T}_i$ to $\mathcal{S}$. Note that the need for coordination does not occur frequently in the real world since the grid size is usually very small (e.g., a couple hundreds of meters in length). Even if IU co-location happens, the IUs that are so densely packed in the same small grid and frequency band likely belong to the same organization. Thus, the coordination is fairly easy.

III. $\mathcal{S}$ initializes the interference budget matrix N:

Upon receiving all the IUs' input $\mathbf{T}_i$, $\mathcal{S}$ aggregates $\mathbf{T}_i$ to create $\mathbf{T}'$ by:

$$\mathbf{T}' := \sum_{i\in \text{all IUs}} \mathbf{T}_i, \tag{2.4}$$

such that $T'(l, h_I, f_I) := \sum_{i\in\text{all IUs}} T_i(l, h_I, f_I)$, $\forall(l, h_I, f_I)$. Note that if there exists an IU with operation position (l, h_I, f_I), $T'(l, h_I, f_I)$ equals the interference threshold of the IU. If no such IU exists, $T'(l, h_I, f_I)$ equals 0.

$\mathcal{S}$ then initializes an interference budget matrix $\mathbf{N} := \{N(l,h_I,f_I)\}_{L\times H_I\times F}$ as follows:

$$N(l,h_I,f_I) := \begin{cases} T'(l,h_I,f_I), & \text{if } T'(l,h_I,f_I) \neq 0 \\ \infty, & \text{if } T'(l,h_I,f_I) = 0 \end{cases}. \tag{2.5}$$

IV. $\mathcal{S}$ makes spectrum allocation decision based on SU operation data $\mathbf{R}_b$ and N:

SU b transmits a spectrum access request along with its operation data $\mathbf{R}_b$ to $\mathcal{S}$. Upon receiving the request, $\mathcal{S}$ computes SU b's interference to each IU operation position (l,h_I,f_I) by:

$$F_b(l,h_I,f_I) := \sum_{j,h_S,f_S} I\,(l,j,h_I,h_S,f_I,f_S) \times R_b(j,h_S,f_S). \tag{2.6}$$

$\mathcal{S}$ subtracts SU b's interference from $\mathbf{N}$ to create an interference indicator matrix $\mathbf{G}_b$ by:

$$G_b(l,h_I,f_I) := N(l,h_I,f_I) - F_b(l,h_I,f_I), \forall(l,h_I,f_I). \tag{2.7}$$

Based on $\mathbf{G}_b$, S will take one of the following two sets of actions.

IV.A. If $\exists(l^\star,h_I^\star,f_I^\star)$ such that $G_b(l^\star,h_I^\star,f_I^\star) \le 0$:

In this case, the interference budget of the IU with operation position $(l^\star,h_I^\star,f_I^\star)$ is exceeded if SU b is allowed to operate. Thus, $\mathcal{S}$ denies SU b's spectrum access request.

IV.B. Otherwise (i.e., $G_b(l,h_I,f_I) > 0, \forall(l,h_I,f_I)$):

In this case, all the IUs are still safe even if SU b is allowed to operate. Thus, $\mathcal{S}$ permits SU b's spectrum access request and provides it with a valid license. $\mathcal{S}$ then lowers the interference budget matrix $\mathbf{N}$ by deducting SU b's interference from it:

$$N(l,h_I,f_I) \leftarrow N(l,h_I,f_I) - F_b(l,h_I,f_I), \forall (l,h_I,f_I). \tag{2.8}$$

From Step IV.A. and IV.B, we can see that, when SU b obtains the spectrum license, the interference budgets are reduced by SU b's interference. Otherwise, the budgets stay the same. In such manner, $\mathbf{N}$ is gradually reduced as more SUs obtain spectrum licenses.

Note that Step I does not require any private input data. Hence, it can be carried out in the plaintext domain. In contrast, Step II to IV involves the private IU/SU input data. Therefore, as shown in the next three subsections, they will be carefully realized in the ciphertext domain.

2.4.4 Secure Computation of Step II

To preserve the privacy of IU operation data, IU i encrypts each entry of $\mathbf{T}_i$ by pk_G, and sends the encrypted operation data $[\![\mathbf{T}_i]\!]$ to $\mathcal{S}$.

2.4.5 Secure Computation of Step III

Secure computation of Step III is tricky since formula (2.5) needs to determine the equality of $T'(l,h_I,f_I)$ and 0 in the ciphertext domain. This is a secure integer comparison problem, which is known as a hard problem. Even though the existing literatures have provided methods to do secure integer comparison [48, 49], they require that the integers involved in the secure computation are bit-wise encrypted. $T'(l,h_I,f_I)$, unfortunately, cannot be encrypted in this way since bit-wise encryption would make the rest of interference computation extremely complex and time-consuming. Secure integer comparison also needs multiple rounds of communications, which is also undesirable. We completely avoid the overhead of secure integer comparison by the following novel integer encoding scheme.

Our method leverages the fact that to perform homomorphic computation on both positive and negative integers in the spectrum computation, integers should be encoded in two's-complement scheme. Under this encoding scheme, representing practical radio signal strength requires only a small number of bits. For example, representing 1000W in the extremely small unit fW (i.e., 10^{-15} watt) requires only 60 bits. On the other hand, to ensure the recommended 112-bit security strength by NIST [50], the security parameter n of Paillier cryptosystem needs to be 2048 bits long. This means that the Paillier plaintext space is also 2048 bits long. Thus, we can easily identify a number k that is much smaller than 2048 but large enough so that (i) all the integer values involved in Section 2.4.3's spectrum computation can be safely encoded in k-bit two's-complement form without the risk of overflow; (ii) $Z = 2^{k-1} - 1$, which is the largest positive value for k-bit two's-complement integers, can be used to represent ∞ in formula (2.5) without affecting the computation results.

With k fixed, the extra unused bits in the plaintext domain can then be leveraged to realize Step III in the ciphertext domain. Assume ψ is a small positive number. Using two's-complement encoding scheme, the representation of a k-bit positive integer V in $(k+\psi)$ bits is shown in Figure 2.3. Also in the context of $(k+\psi)$-bit two's-complement representation, we create two integers $W = V + 1 - 2^{k-1}$ and $Z = 2^{k-1} - 1$. We calculate $W + Z$ and the result is shown in Figure 2.3. Constrained by $(k+\psi)$-bit representation, the leftmost bit of $(W+Z)$ is ignored. Thus $(W + Z)$ equals V. Essentially, the above property of two's-complement encoding means that: For an $(k+\psi)$-bit integer T',

$$T' + Z = \begin{cases} V, & \text{if } T' = W = V + 1 - 2^{k-1} \\ Z = \infty, & \text{if } T' = 0 \end{cases}.$$

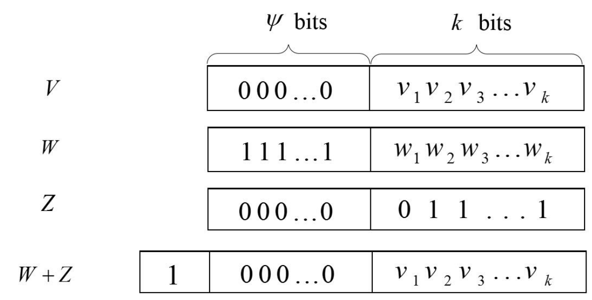


Figure 2.3: Integer encoding

Based on the above property, we realize Step III in the ciphertext domain as follows. We slightly adjust an IU i's way of computing $\mathbf{T}_i$. Assume IU i's interference threshold is a k-bit positive integer V. IU i creates a $(k+\psi)$-bit integer $W = V + 1 - 2^{k-1}$ in two's-complement form. If IU i's operation position is (l, h_I, f_I), $T_i(l, h_I, f_I)$ holds W. The rest entries of $\mathbf{T}_i$ are set to 0. IU i encrypts $\mathbf{T}_i$ and submits $[\![\mathbf{T}_i]\!]$ to $\mathcal{S}$. When $\mathcal{S}$ receives all the IUs' input $[\![\mathbf{T}_i]\!]$, it executes

$$[\![\mathbf{T}']\!] := \oplus_{i\in\text{all IUs}}[\![\mathbf{T}_i]\!], \tag{2.9}$$

where $\oplus_{i\in\text{all IUs}}$ is the homomorphic version of $\sum_{i\in\text{all IUs}}$. Then, $[\![\mathbf{N}]\!]$ is computed by

$$[\![\mathbf{N}]\!] := [\![\mathbf{T}']\!] \oplus [\![\mathbf{Z}]\!], \tag{2.10}$$

where $\mathbf{Z}$ is a $(L \times H_I \times F)$-sized matrix whose entries are all set to $2^{k-1} - 1$, and we enforce a policy that the bits higher than the $(k+\psi)$th position in decrypted plaintexts should be ignored.

2.4.6 Secure Computation of Step IV

Formula (2.6) and (2.7) in Step IV can be computed securely by straightforwardly applying homomorphic operations:

$$[\![F_b(l,h_I,f_I)]\!] := \oplus_{j,h_S,f_S} I(l,j,h_I,h_S,f_I,f_S) \otimes [\![R_b(j,h_S,f_S)]\!], \tag{2.11}$$

$$[\![G_b(l,h_I,f_I)]\!] := [\![N(l,h_I,f_I)]\!] \ominus [\![F_b(l,h_I,f_I)]\!]. \tag{2.12}$$

Securely computing Step IV.A and IV.B is nontrivial. First, to decide which step to take, $\mathcal{S}$ has to determine the sign of $G_b(l,h_I,f_I)$ given only its ciphertext $[\![G_b(l,h_I,f_I)]\!]$. This is again a secure integer comparison problem. Second, in Step IV.B, a spectrum license specifying the operation parameters needs to be generated if an SU is allowed to operate. To preserve the privacy of the SU, these parameter specifications need to remain in the ciphertext domain and should only be revealed to the SU. In addition, to prevent the SU from forging the operation parameters, these parameter specifications also need be digitally signed. Yet, creating a digital signature in the ciphertext domain is still an open problem.

We solve the above challenges by a two-step approach as follows.

Step (1): Using the key conversion algorithm in Table 2.4, $\mathcal{S}$ creates $[\![Q_b(l,h_I,f_I)]\!]_{\mathsf{pk}_b}$, whose plaintext satisfies:

$$Q_b(l,h_I,f_I) = \begin{cases} 0, & \text{when } G_b(l,h_I,f_I) > 0 \\ -2, & \text{when } G_b(l,h_I,f_I) \leq 0 \end{cases}. \tag{2.13}$$

Here, $[\![Q_b(l,h_I,f_I)]\!]_{\mathsf{pk}_b}$ denotes the encryption of $Q_b(l,h_I,f_I)$ by SU b's individual public key pk_b. Based on the description in Section 2.4.3, we know that the sign of $G_b(l,h_I,f_I)$ holds the information that whether the IU with operation position (l,h_I,f_I) will be disturbed if

SU b is allowed to operate. Now this information is also reflected in $Q_b(l, h_I, f_I)$. It will later be used to generate a pk_b-encrypted license in Step (2) that can be eventually decrypted by SU b.

In the algorithm of Table 2.4, $\mathcal{S}$ uses the blinding factors $\alpha(l, h_I, f_I)$, $\beta(l, h_I, f_I)$ and $\epsilon(l, h_I, f_I)$ in formula (2.14) to obfuscate the true value and sign of $G_b(l, h_I, f_I)$ respectively before sending $[\![G_b(l, h_I, f_I)]\!]_{\mathsf{pk}_b}$ to $\mathcal{K}$ for key conversion. By doing this, the intermediate computation results $G_b(l, h_I, f_I)$ will not be leaked to $\mathcal{K}$. Specifically, given $X_b(l, h_I, f_I)$ by decrypting $[\![X_b(l, h_I, f_I)]\!]$ with sk_G, $\mathcal{K}$ cannot infer the true value or sign of $G_b(l, h_I, f_I)$. $\mathcal{K}$ generates $Y_b(l, h_I, f_I)$ based on the sign of $X_b(l, h_I, f_I)$ in formula (2.15), and encrypts it using pk_b before sending it back to $\mathcal{S}$. Finally, using the recorded sign blinding factor $\epsilon(l, h_I, f_I)$, $\mathcal{S}$ creates $[\![Q_b(l, h_I, f_I)]\!]_{\mathsf{pk}_b}$ by firstly recovering the sign information of $G_b(l, h_I, f_I)$ through multiplying $[\![Y_b(l, h_I, f_I)]\!]_{\mathsf{pk}_b}$ by $\epsilon(l, h_I, f_I)$, and then subtracting $[\![1]\!]_{\mathsf{pk}_b}$ in formula (2.16). It is easy to verify that the plaintext of $[\![Q_b(l, h_I, f_I)]\!]_{\mathsf{pk}_b}$ generated in this way conforms formula (2.13). Formal analysis of the relation between the value selection of the blinding factors and the effectiveness of the obfuscation can be found in Appendix 1.

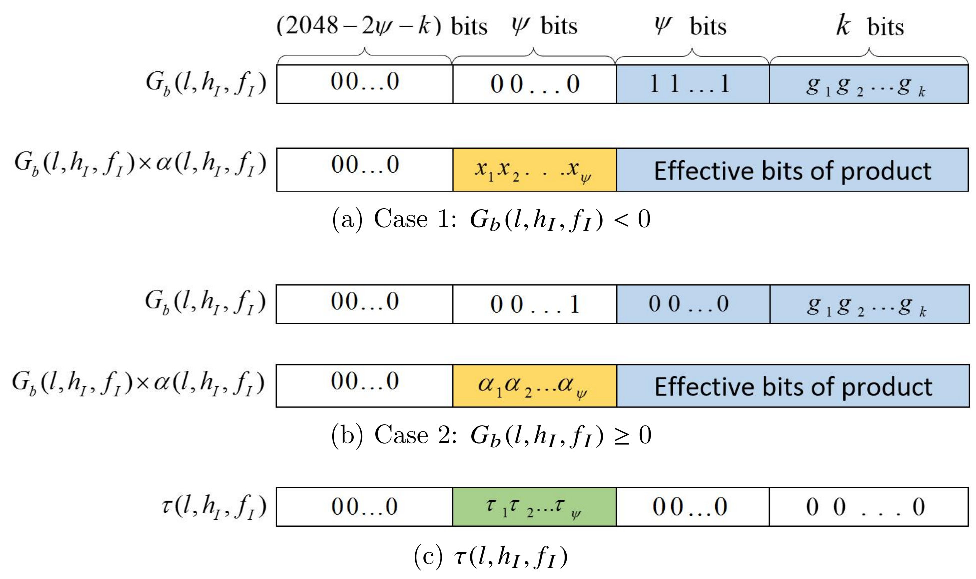
Careful readers may have noticed $\tau(l, h_I, f_I)$ in formula (2.14). To understand the purpose of $\tau(l, h_I, f_I)$, consider the plaintext message format of $G_b(l, h_I, f_I)$ as shown in Figure 2.4. The upper rows of subfigure 2.4a and 2.4b illustrate how the plaintext looks like when $G_b(l, h_I, f_I)$ is negative and positive, respectively. The lower rows show the plaintext of $\alpha(l, h_I, f_I) \times G_b(l, h_i, f_I)$. As described in Section 2.4.5, our secure computation steps assume the lower $(\psi + k)$ bits in the plaintext hold the effective data in two's-complement form, which are marked by blue color in Figure 2.4. Note that $\alpha(l, h_I, f_I)$ is a ψ-bit integer, so the product of $G_b(l, h_I, f_I)$ and $\alpha(l, h_I, f_I)$ takes $(k + 2\psi)$-bit space. While the lower $(\psi + k)$ bits still hold the effective computation results under two's-complement encoding, the yellow part that holds the overflowed bits from the product can potentially leak information of the

Table 2.4: Key Conversion

$\mathcal{S}$: **(i)** Generate $[\![\mathbf{X}_b]\!]$ by letting $[\![X_b(l,h_I,f_I)]\!] := \Big(\alpha(l,h_I,f_I) \otimes [\![G_b(l,h_I,f_I)]\!] \oplus [\![\tau(l,h_I,f_I)]\!] \ominus [\![\beta(l,h_I,f_I)]\!]\Big) \otimes \epsilon(l,h_I,f_I), \quad (2.14)$ where $\alpha(l,h_I,f_I)$, $\beta(l,h_I,f_I)$ are ψ-bit random numbers, and $\alpha(l,h_I,f_I) > \beta(l,h_I,f_I) > 0$. They are used to blind the true value of $G_b(l,h_I,f_I)$ without affecting its sign. $\epsilon(l,h_I,f_I)$ is chosen in $\{-1,1\}$ uniformly at random to obfuscate the sign of $G_b(l,h_I,f_I)$. $\tau(l,h_I,f_I)$ is used to avoid the undesirable reveal of $\alpha(l,h_I,f_I)$ value in the product of $\alpha(l,h_I,f_I)$ and $G_b(l,h_I,f_I)$. **(ii)** Send $[\![\mathbf{X}_b]\!]$ to $\mathcal{K}$.
$\mathcal{K}$: **(iii)** Decrypt $[\![\mathbf{X}_b]\!]$. **(iv)** Generate $[\![\mathbf{Y}_b]\!]$ by letting $Y_b(l,h_I,f_I) := \begin{cases} 1, & \text{when } X_b(l,h_I,f_I) > 0 \\ -1, & \text{when } X_b(l,h_I,f_I) \leq 0 \end{cases}. \quad (2.15)$ **(v)** Encrypt $\mathbf{Y}_b$ by pk_b and send $[\![\mathbf{X}_b]\!]_{\mathsf{pk}_b}$ to $\mathcal{S}$.
$\mathcal{S}$: **(vi)** Generate $[\![\mathbf{Q}_b]\!]_{\mathsf{pk}_b}$ by letting $[\![Q_b(l,h_I,f_I)]\!]_{\mathsf{pk}_b} := \Big(\epsilon(l,h_I,f_I) \otimes [\![Y_b(l,h_I,f_I)]\!]_{\mathsf{pk}_b}\Big) \ominus [\![1]\!]_{\mathsf{pk}_b}, \quad (2.16)$ where $\epsilon(l,h_I,f_I)$ is the same as the one in formula (2.14).

blinding factor $\alpha(l,h_I,f_I)$. The overflowed bits are denoted as $x_1x_2...x_\psi$ and $\alpha_1\alpha_2...\alpha_\psi$ for case 1 and case 2 in Figure 2.4, respectively, and $\alpha_1\alpha_2...\alpha_\psi$ is the binary representation of $\alpha(l,h_I,f_I)$. Therefore, this threat is especially acute for case 2, where the yellow part holds an exact copy of $\alpha(l,h_I,f_I)$. Therefore, to avoid the undesirable leakage of $\alpha(l,h_I,f_I)$, we add a mask $\tau(l,h_I,f_I)$ shown in subfigure 2.4c to the product, where $\tau_1,\tau_2,\tau_3,...,\tau_\psi$ are random bits.

Step (2): In this step, we show how to approve/deny SU b's spectrum request by generating valid/invalid signature for spectrum license. Specifically, $\mathcal{S}$ first creates a spectrum license for SU b. The license includes the identity of SU b, the identity of license issuer $\mathcal{S}$, and $[\![\mathbf{R}_b]\!]$, which is the encrypted operation parameters of SU b that are submitted in the spectrum



(2048 − 2ψ − k) bits
ψ bits
ψ bits
k bits
$G_b(l,h_I,f_I)$
00...0
00...0
11...1
$g_1g_2\cdots g_k$
$G_b(l,h_I,f_I)\times\alpha(l,h_I,f_I)$
00...0
$x_1x_2...x_\psi$
Effective bits of product
(a) Case 1: $G_b(l,h_I,f_I) < 0$
$G_b(l,h_I,f_I)$
00...0
00...1
00...0
$g_1g_2\cdots g_k$
$G_b(l,h_I,f_I)\times\alpha(l,h_I,f_I)$
00...0
$\alpha_1\alpha_2...\alpha_\psi$
Effective bits of product
(b) Case 2: $G_b(l,h_I,f_I) \geq 0$
$\tau(l,h_I,f_I)$
00...0
$\tau_1\tau_2...\tau_\psi$
00...0
00...0
(c) $\tau(l,h_I,f_I)$

Figure 2.4: Adding $\tau(l,h_I,f_I)$ for masking

request. Then, $\mathcal{S}$ uses some digital signature system (e.g., Digital Signature Algorithm) to generate a signature C_b of the license. SAS encrypts C_b by SU b's public key pk_b and obtains $[\![C_b]\!]_{\mathsf{pk}_b}$. It then computes

$$[\![D_b]\!]_{\mathsf{pk}_b} := [\![C_b]\!]_{\mathsf{pk}_b} \oplus \left(\sigma \otimes \left(\oplus_{l,h_I,f_I} [\![Q_b\,(l,h_I,f_I)]\!]_{\mathsf{pk}_b}\right)\right), \tag{2.17}$$

where σ is a random integer. In essence, D_b holds the valid license signature C_b if $Q_b\,(l,h_I,f_I) = 0, \forall(l,h_I,f_I)$. If for some (l^*,h_I^*,f_I^*), $Q_b(l^*,h_I^*,f_I^*) = -2$, D_b is equal to C_b+*some random number*, which is an invalid signature. This process guarantees that only the SU whose spectrum request is approved by $\mathcal{S}$ can get valid license signature.

Finally, $\mathcal{S}$ sends the spectrum license along with $[\![D_b]\!]_{\mathsf{pk}_b}$ and $[\![\mathbf{F}_b]\!]$ back to SU b. Upon decrypting $[\![D_b]\!]_{\mathsf{pk}_b}$, SU b finds out whether it is allowed to access the spectrum by examining the validity of D_b using the verification algorithm of the digital signature system. If its request is approved, SU b generates $[\![U_b(l,h_I,f_I)]\!] := [\![F_b(l,h_I,f_I)]\!] \oplus [\![0]\!]$. Otherwise, SU b generates $[\![U_b(l,h_I,f_I)]\!] := [\![0]\!]$. $[\![\mathbf{U}_b]\!]$ is then sent back to $\mathcal{S}$ along with SU b's confirmation message. $\mathcal{S}$ executes formula (2.8) securely by

$$[\![N(l,h_I,f_I)]\!] \leftarrow [\![N(l,h_I,f_I)]\!] - [\![U_b(l,h_I,f_I)]\!]. \tag{2.18}$$

Preventing forging attempts: The above two-step approach ensures that SU b can obtain a properly signed spectrum license if and only if its operation does not disturb any IU. Note that the license only holds $[\![\mathbf{R}_b]\!]$, which can only be decrypted with sk_G. In the following, we show that a verifier $\mathcal{V}$ can still easily detect an SU b that attempts to deviate its operation parameters from the licensed $\mathbf{R}_b$ to some other values (denoted as $\mathbf{R}'_b$) without sk_G.

First, $\mathcal{V}$ requests SU b to provide its operation parameters, which include SU b's physical location. $\mathcal{V}$ can observe SU b's operation near its physical location, so SU b can only submit $\mathbf{R}'_b$ since if SU b lies, the operation parameters provided will not match $\mathcal{V}$'s observation. Note that when we say $\mathcal{V}$ observes an SU's operation, we mean it is close enough to measure the radio signal of the SU so that the SU's antenna height, transmit power, transmit frequency and location can be estimated. Techniques such as radio localization and signal spectrum analyzing can be used to derive such operation data from the emitted radio signal of the SU. Next, $\mathcal{V}$ requests SU b to provide a copy of the signed spectrum license. Note that the license includes $[\![\mathbf{R}_b]\!]$. Then, $\mathcal{V}$ requests SU b to provide the the random number r used to create $[\![R_b(j,h_S,f_S)]\!]$ from $R_b(j,h_S,f_S)$ in the encryption process of Paillier cryptosystem. Using the Enc algorithm in Table 2.1, $\mathcal{V}$ attempts to re-encrypt $R_b(j,h_S,f_S)$ using pk_G and r. Since pk_G and r are the same used to create $[\![R_b(j,h_S,f_S)]\!]$ from $R_b(j,h_S,f_S)$, $[\![{R_b}'(j,h_S,f_S)]\!]$ should be the same as $[\![R_b(j,h_S,f_S)]\!]$ if $R_b(j,h_S,f_S)$ equals $R'_b(j,h_S,f_S)$. If $[\![R_b(j,h_S,f_S)]\!] \neq [\![{R_b}'(j,h_S,f_S)]\!]$, $\mathcal{V}$ claims that SU b has forged its operation parameter specification. The above verification process is performed for all the operation positions. Note that SU b cannot find another random number r' that can be used to re-encrypt $R'_b(j,h_S,f_S)$ to get

$[\![R_b(j,h_S,f_S)]\!]$. This is because one ciphertext can only be decrypted to one plaintext.

2.4.7 Security analysis

Correctness

It is straightforward to see that, if the underlying Paillier cryptosystem is correct, P^2-SAS correctly performs the SAS process described in Section 2.3.1. Detailed correctness proof of Paillier cryptosystem can be found in [23].

Privacy

Theorem 1. P^2-SAS securely performs the SAS process in the presence of semi-honest $\mathcal{S}$ and $\mathcal{K}$ as long as Paillier cryptosystem is semantically secure, blinding factors are properly generated, and $\mathcal{S}$ and $\mathcal{K}$ are non-colluding.

Proof. This theorem can be proved using the composition theorem [42] under the semi-honest model by analyzing the security of each step in P^2-SAS. Specifically, the computation steps in $\mathcal{S}$ are all performed in the ciphertext domain, so if $\mathcal{K}$ is not colluding with $\mathcal{S}$, $\mathcal{S}$ cannot infer any private IU/SU operation information from these computation steps due to the semantic security of Paillier cryptosystem [23].

In the key conversion service, $\mathcal{K}$ can obtain the plaintext of $X_b(l,h_I,f_I)$. The privacy is still preserved if knowing $X_b(l,h_I,f_I)$ gives $\mathcal{K}$ negligible advantage in distinguishing $G_b(l,h_I,f_I)$ compared with random guesses [51]. This requirement can be fulfilled by properly generating blinding factors $\alpha(l,h_I,f_I)$, $\beta(l,h_I,f_I)$, and $\epsilon(l,h_I,f_I)$ to obfuscate the true value and sign of $G_b(l,h_I,f_I)$. The guidelines for proper generation of blinding factors and the formal security proof can be found in Appendix A.1. □

2.5 Tuning & Acceleration

In this section, we investigate the tradeoff between accuracy and computation overhead in interference calculation by tuning the quantization granularity parameters. We also propose effective acceleration methods to improve P^2-SAS's efficiency.

2.5.1 Tuning of Quantization Granularity

As mentioned in Section 2.4.2, to reduce computation overhead, we quantize location and antenna height values. Quantization introduces error in attenuation estimation, which may lead to either interference underestimation or interference overestimation.

Interference underestimation happens when the estimated attenuation value is larger than the true value. It should be strictly forbidden since it may cause the accumulative interference to IUs exceeds the interference threshold. The key to eliminate interference underestimation is to always choose the quantized values that make the estimated attenuation smaller. For example, the antenna height should always be rounded up since the higher antenna makes the estimated attenuation value smaller.

Interference overestimation happens when the estimated attenuation value is smaller than the true value, which causes more consumption of interference budget in $\mathcal{S}$'s computation than necessary. Interference overestimation leads to underutilization of spectrum, which is undesirable yet tolerable. More fine-grained quantization yields more precise attenuation map, which reduces interference overestimation error at the cost of larger computation overhead. We quantitatively study the impact of quantization granularity on the tradeoff between interference overestimation error and computation overhead. Specifically, we use 4 integer metrics A, B_s, B_i, B_a to denote the quantization granularity. A is the side length of grid. SU height, IU height, and attenuation are quantized into 2^{B_s} , 2^{B_i} , 2^{B_a} levels, respectively. We formulate the following optimization problem to seek for the optimal quantization granularity setting that minimizes the interference overestimation error under constrained computation overhead.

$$\begin{aligned} \underset{A,B_s,B_i,B_a}{\text{minimize}} \quad & err(A,B_s,B_i,B_a) \\ \text{subject to} \quad & cost(A,B_s,B_i,B_a) \le C, \\ & A \in \{100,200,...,500\}, \\ & B_s \in \{1,2,3\}, \\ & B_i \in \{1,2,3,4\}, \\ & B_a \in \{1,2,...,30\} \end{aligned} \tag{2.19}$$

Function $err(A,B_s,B_i,B_a)$ is defined as the root-mean-square error (RMSE) between the estimated attenuation values under the granularity setting (A,B_s,B_i,B_a) and the true attenuation values. Function $cost(A,B_s,B_i,B_a)$ is measured as the estimated response time to an SU's spectrum request under the granularity setting (A,B_s,B_i,B_a). Specifically, given the values of A,B_s,B_i,B_a, we can count the number of different Paillier operations executed in $\mathcal{S}$ by analyzing the formulas in Section 2.4. For example, formula (2.12) needs $L*2^{B_i}*F$ homomorphic subtraction operations. The estimation of response time is obtained by multiplying the number of different Paillier operations with the average execution time per operation. A benchmark of Paillier cryptosystem is given in Table 2.5. The cost budget C is selected according to the response time required in specific application scenarios. Note that in (2.19), we assume $B_s \le 3, B_i \le 4$ since 3 and 4 have been shown to be usually larger than our optimal solution of B_s and B_i and hence are safe boundary values. We set $B_a \le 30$ since 30-bit space is more than enough to accommodate all the practical attenuation values.

To solve the optimization problem of (2.19), we observe that it is a monotonic integer optimization problem. This type of optimization problem can be efficiently solved by the branch-and-bound method [52]. Notably, the optimization problem only needs to be solved once at the initialization stage of P^2-SAS and is not recomputed for runtime decision making.

2.5.2 Improving Efficiency

Since P^2-SAS potentially needs to serve a large number of IUs and SUs, its efficiency is critical for the scalability and practicality in the real-world deployment. In this subsection, we present some acceleration methods to improve P^2-SAS's efficiency.

Factoring

Through estimating the response time of $\mathcal{S}$ in Section 2.5.1, we observe that computing $[\![\mathbf{F}_b]\!]$ in formula (2.11) dominates the computation overhead of the whole system. It needs $L^2 * 2^{B_i} * 2^{B_s} * F^2$ number of $\otimes$ operations and the same number of $\oplus$ operations. By reducing the computation complexity of formula (2.11), we can greatly improve P^2-SAS's efficiency.

Our method is based on factoring, i.e., $\oplus_{i=1}^{K}([\![a_i]\!] \otimes b) = \left(\oplus_{i=1}^{K}[\![a_i]\!]\right) \otimes b$. Note that the transformation from the left side of the equation to the right side can reduce the number of $\otimes$ operation from K to 1, while keeping the number of $\oplus$ operation the same. We also observe that in the real world, large areas often share the same attenuation values, as demonstrated by the sample attenuation map in Figure 2.5. Thus, given (l, h_I, f_I), we group the $I(l, j, h_I, h_S, f_I, f_S)$ entries that have the same value. Assume there are totally K groups and all the entries in group a equal I_a, $a = \{1, 2, ..., K\}$. Then, formula (2.11) can be converted to the following more efficient form:

$$[\![F_b(l, h_I, f_I)]\!] := \oplus_{a=1}^{K}\Big(I_a \otimes \Big(\oplus_{I(l,j,h_I,h_S,f_I,f_S)\in a}[\![R_b(j, h_S, f_S)]\!]\Big)\Big). \qquad (2.20)$$

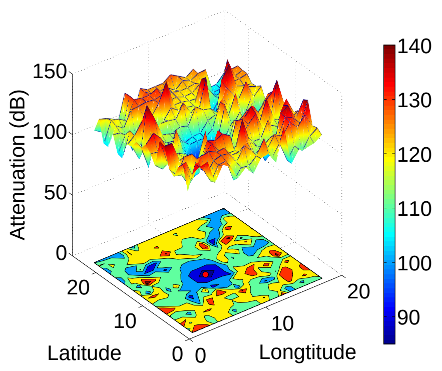


Figure 2.5: A sample of attenuation map in Washington D.C. The attenuation value is measured from each coordinate point to the center.

Precomputing

In the case where $\mathcal{S}$ does not consider the interference map $\mathbf{I}$ as proprietary data, we can further reduce $\mathcal{S}$'s computation overhead by letting $\mathcal{S}$ publish $\mathbf{I}$. Leveraging the published interference map $\mathbf{I}$, SU b can directly compute $F_b(l, h_I, f_I)$ of formula (2.11) in the plaintext domain, encrypt the result and send $[\![\mathbf{F}_b]\!]$ to $\mathcal{S}$ as part of the spectrum request. In this way, $\mathcal{S}$ does not need to execute formula (2.11) (or (2.20)), so that the most computationally intensive part in P^2-SAS is avoided.

This acceleration method is based on the assumption that SUs are trusted to make the correct interference calculation. The assumption might not necessarily be true because there are incentives of SUs to cheat in the interference calculation process to gain illegal access to spectrum. We propose a verification mechanism to discover such cheating behavior to ensure that the interference calculation on the SU side is faithfully performed. Specifically, a verifier can firstly get SU b's real operation parameters $\mathbf{R}_b$ by estimation near SU adopting the similar technique in Section 2.4.6. Then, the verifier requires $\mathcal{K}$ to provide the decryption of $[\![\mathbf{F}_b]\!]$. By checking whether the equation (2.6) holds, the verifier can find whether the interference calculation is faithfully performed by SU b.

Ciphertext Packing

Ciphertext packing technique [53] allows P^2-SAS to pack multiple integers into one ciphertext to reduce computation and communication overhead. Specifically, the ciphertext packing technique is applied to the formulas (2.9), (2.10), (2.20), (2.12), (2.14) as follows.

Recall the integer encoding policy specified in Section 2.4.6. All the integers involved in $\mathcal{S}$'s computation can be represented in $p := (k + 2\psi)$-bit space and the lower $(k + \psi)$ bits are effective. Therefore, we can divide the 2048-bit plaintext space into $q := \lfloor 2048/p \rfloor$ segments, and each segment holds one p-bit integer. This means that q p-bit integers can be packed into one 2048-bit plaintext message. After encryption, the integers packed in one ciphertext can be homomorphically operated simultaneously.

Leveraging the ciphertext packing technique, an IU i packs q $\mathbf{T}_i$ entries together before encryption. For an SU b, if the interference map $\mathbf{I}$ is shared among SUs (Section 2.5.2's scheme is used), it packs q $\mathbf{F}_b$ entries together before encryption. If $\mathbf{I}$ is kept secret in $\mathcal{S}$ (Section 2.5.2's scheme is used), SU b can only submits the unpacked $[\![\mathbf{R}_b]\!]$ in the spectrum request to $\mathcal{S}$. The packing of $\mathbf{R}_b$ happens on $\mathcal{S}$ side. Specifically, $\mathcal{S}$ packs $[\![\mathbf{R}_b]\!]$ entries directly in the ciphertext domain by first shifting each entry using homomorphic scalar multiplication to the right bit position, and then combining these entries using homomorphic addition. With the packed $[\![\mathbf{T}_i]\!]$ and $[\![\mathbf{R}_b]\!]$, the computation overhead in formulas (2.9), (2.10), (2.20), (2.12), (2.14) is reduced by a factor of q.

It is worth mentioning that when applying the packing technique to formula (2.14), the sign-bit blinding factor ϵ cannot be packed and one ϵ will be used to scale all the $\mathbf{G}_b$ entries that are packed together, which can potentially weaken the protection on $\mathbf{G}_b$' sign. To address this problem, note that for a $(k + \psi)$-bit integer, adding $\Delta = (1\underbrace{000...0}_{k+\psi-1})_2$ only changes the sign bit. Thus, we can modify formula (2.14) as:

$$[\![X_b(l,h_I,f_I)]\!] := \Big(\alpha(l,h_I,f_I) \otimes [\![G_b(l,h_I,f_I)]\!] \oplus [\![\tau(l,h_I,f_I)]\!] \ominus [\![\beta(l,h_I,f_I)]\!]\Big) \oplus [\![\delta(l,h_I,f_I)]\!], \tag{2.21}$$

where $\delta(l,h_I,f_I)$ is uniformly chosen in $\{0,\Delta\}$. With formula (2.21), the packing technique can be applied to $\delta(l,h_I,f_I)$ so that the signs of the $\mathbf{G}_b$ entries packed in the same plaintext can be individually perturbed.

Careful readers may also have noticed that in formula (2.21), the $\mathbf{G}_b$ entries that are packed in the same plaintext are scaled by the same blinding factor α since α cannot be packed. This can cause certain loss of α's blinding effectiveness. However, it can be shown that the loss of blinding effectiveness is slight and the blinding technique is still secure. The formal proof can be found in Appendix A.2.

Parallelization

The computation tasks of P^2-SAS are readily to be parallelized. Specifically. the entries of a matrix can be divided into subsets and the computation tasks of different subsets can be distributed to different threads and servers.

2.6 Evaluation

2.6.1 Implementation

We construct the Paillier cryptosystem based on GMP library [54]. Security parameter n is set to be 2048 bits long. Table 2.5 shows a benchmark of the Paillier implementation. For parallelization implementation, we divide the computation into 24 threads and evenly

Table 2.5: Benchmark of Paillier cryptosystem

Encryption	11.73 ms
Decryption	6.69 ms
Homomorphic addition	0.008 ms
Homomorphic scale (20 bits)	0.200 ms
Homomorphic scale (112 bits)	0.685 ms
Homomorphic subtraction	0.058 ms
Public key size	4096 bits
Secret key size	4096 bits
Plaintext message size	2048 bits
Ciphertext size	4096 bits

Table 2.6: Experiment parameter settings

Side length of grid (A)	500 m
Bit length for SU height (B_s)	1
Bit length for IU height (B_i)	1
Bit length for attenuation (B_a)	20
Number of IUs	350
Number of SUs	20000
Number of grids (L)	650
Number of frequency channel (F)	5
Bit length of integer encoding (k)	60
Bit length of blinding factors (ψ)	220
Packing block size (p)	502
Number of packing blocks (q)	4

distribute them to three desktops with Intel i7-3770 CPU @ 3.40GHz and 12GB RAM.

2.6.2 Evaluation Settings

To evaluate P^2-SAS, we set the service area to be a 154.82 km^2 area in Washington D.C. We employ L-R model provided by SPLAT! [55] to calculate the attenuation map $\mathbf{I}$ in this area. Real terrain data from USGS [45] and SRTM3 [46] is used in L-R model. Important experiment parameter settings are presented in Table 2.6. The first four parameter values are obtained by solving the optimization problem (2.19) in Section 2.5.1. There are 20000 SUs in the experiment, and we assume that each SU is scheduled to send an SU request to $\mathcal{S}$ to renew its SU license in a dedicated time slot, which is determined through some central scheduler in $\mathcal{S}$. The interference thresholds of IUs are randomly assigned between -80 dBm and -130 dBm to simulate the interference thresholds of typical radar receivers.

2.6.3 Accuracy

In this subsection, we evaluate the accuracy of P^2-SAS in spectrum allocation. Specifically, we measure the error rates of the spectrum allocation decisions made by P^2-SAS compared with the traditional SAS implementation as ground truth. In the traditional SAS implemented, no privacy-preserving feature is adopted, and no quantization process in the interference computation is employed. The errors we focus on include false positive error and false negative error. False positive error refers to the situation that an SU's request that should be denied in traditional SAS gets approved in P^2-SAS. This error is usually caused by interference underestimation. In contrast, false negative error refers to the situation that an SU's request that should be approved in traditional SAS gets denied in P^2-SAS. This is usually caused by interference overestimation. We run P^2-SAS for 1000 times. In each run, the operation parameters of IUs and SUs are all randomly generated. We measure the average false positive rate and false negative rate in the experiment, and the results show that the false positive rate and false negative rate are 0 and 2.72%, respectively. Therefore, P^2-SAS incurs no false positive error and very small false negative error due to the optimal tuning of quantization in Section 2.5.1.

2.6.4 Effectiveness of Acceleration Methods

In this subsection, we evaluate the performance improvement of different acceleration methods introduced in Section 2.5.2. In terms of the performance metrics, we focus on the computation overhead of each party and the communication overhead of SU in one spectrum access process.

The evaluation results of computation overhead and communication overhead are shown in Figure 2.6 and Figure 2.7, respectively. For simplicity, we denote the acceleration method of Section 2.5.2.x as (x). As can be seen in the figures, the combination of acceleration methods (2,3,4) produces the largest performance improvement with respect to both computation overhead and communication overhead. Note that the acceleration method (4) is

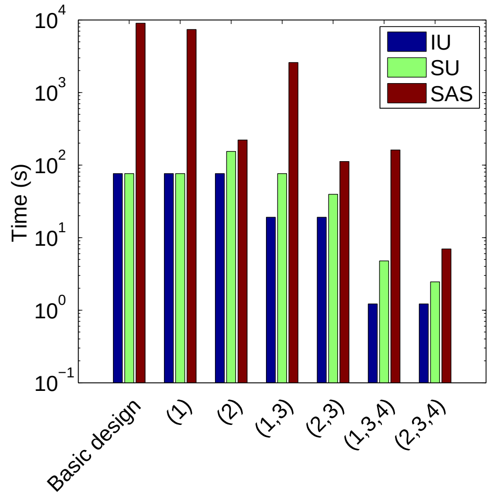


Figure 2.6: Computation overhead of P^2-SAS

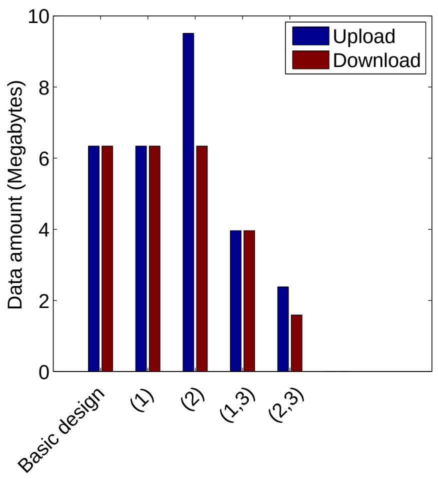


Figure 2.7: Communication overhead of P^2-SAS

parallelization, which will not affect the communication overhead. We will use (2,3,4) as the default acceleration setting, and compare the accelerated P^2-SAS with the traditional SAS in Section 2.6.5.

2.6.5 Comparison with Traditional SAS

In this subsection, we compare the performance of the accelerated P^2-SAS with the traditional SAS that has no privacy protection. For computation overhead, P^2-SAS's average processing time per SU spectrum request is 6.96 seconds as seen from Figure 2.6, while the average processing time of traditional SAS implementation is 0.13 seconds. For the communication overhead of SU, the uplink and downlink data amounts are 2.38 MB and 1.59 MB respectively as seen from Figure 2.7. For traditional SAS, the uplink and downlink data amounts are 3.51 KB and 2.53 KB, respectively. Based on the experiment result, we can compute that all the 20000 SUs' licenses can be renewed in 38.7 hours. This means that a license issued for each SU must be valid for at least 38.7 hours. We can also see that while P^2-SAS's performance is still not as good as the traditional SAS service due to the overhead of privacy protection, its performance is already good enough for large scale DSA applications. In the real-world deployment, P^2-SAS can be hosted on advanced computing clusters to further increase its performance.

2.7 Summary

In this chapter, we build P^2-SAS for privacy-preserving centralized DSA by converting complex spectrum allocation computation and certification procedures into the limited homomorphic computation types. Combining the unique characteristics of spectrum allocation computation with the nature of Paillier cryptosystem, we are able to significantly reduce the computation overhead of P^2-SAS. We evaluate its scalability and practicality using experiments based on real-world data. Experiment results show that P^2-SAS can respond an SU's spectrum request in 6.96 seconds with communication overhead of less than 4 MB.

Chapter 3

IP-SAS: Privacy-Preserving Exclusion-Zone-Based SAS

In this chapter, we present the design of IP-SAS for privacy-preserving exclusion-zone-based SAS [56, 22, 57].

3.1 Challenges and Contributions

The aim of this chapter is to build a privacy-preserving SAS systems that adopt exclusion zone (E-Zone) access enforcement method. E-Zone method is the one adopted by most of the current SAS proposals, such as FCC's recent proposal for DSA in 3.5 GHz [8]. In a typical scenario of E-Zone-based SAS systems, IUs first compute their E-Zones and send the E-Zone data to SAS in the initialization phase. When an SU wants to access the spectrum, it needs to provide its operation parameters and geolocation to SAS. SAS checks whether the SU is within the E-Zone of any IU. For a given channel, if the answer is yes, SAS denies the SU's spectrum access to this channel. If the answer is no, the SU's spectrum access request is permitted for this channel. This chapter focuses on preserving IUs' privacy against untrusted SAS. The privacy issues of SUs in untrusted SAS has been addressed using private information retrieval (PIR) techniques in [32]. It is worth mentioning that the PIR scheme is readily to be integrated in our design to achieve a more comprehensive privacy-preserving solution for exclusion-zone-based SAS.

Our contribution can be summarized at the following four aspects:

• To the best of our knowledge, we are the first to identify and address the privacy issues of IUs in E-Zone-based SAS. Through an efficient secure-multiparty-computation design, our IP-SAS realizes E-Zone-based spectrum allocation in SAS without exposing any information that can potentially lead to IU privacy violation. These protected IU-privacy-related information includes IU E-Zone information, intermediate spectrum allocation computation results, and final spectrum allocation decisions.

• To prevent malicious parties from compromising the IP-SAS system, we extend the basic design to make IP-SAS robust against more complex and realistic malicious attacks. The task of countering malicious attacks is challenging due to IP-SAS's privacy-preserving design, since one cannot easily tell cheating behaviors from information discrepancy. To tackle the challenge, we manipulate the plaintext space to accommodate additive zero-knowledge proof in the secure SAS computation. We also use the deterministic property of decryption algorithm for zero-knowledge proof.

• We leverage the unique properties of IP-SAS computation to develop acceleration mechanisms, which significantly reduce IP-SAS's computation and communication overhead.

• We conduct extensive experiments based on real-world data to demonstrate the scalability and practicality of IP-SAS.

The rest of this chapter is organized as follows. Section 3.2 overviews the related work. Section 3.3 introduces the system and adversary model, our design goals, and some preliminaries of homomorphic encryption. In Section 3.4, we present the basic IP-SAS design against semi-honest adversary model. After that, we extend the basic design to defend against malicious adversaries in Section 3.5. In Section 3.6, we propose acceleration methods to improve IP-SAS's efficiency. We run experiments to evaluate the system and the evaluation results are shown in Section 3.7. Finally, we summarize this chapter in Section 3.8.

3.2 Related Work

For efficient MPC solutions, a two-party computation models is proposed in [58, 59]. The main idea is to use two non-colluding central parties to perform some computational functionality jointly instead of just one central party. The advantage of this distributed design is that instead of using some heavyweight cryptosystems in one central party to provide some functionality in a privacy-preserving way, some lightweight cryptosystems may be sufficient to realize the same privacy-preserving functionality across two interacting central parties. IP-SAS in this chapter also involves two central parties (SAS Server $\mathcal{S}$ and Key Distributor $\mathcal{K}$), however, it is different from the two-party computation model in [58, 59]. In [58, 59], only semi-honest attack model is considered. In contrast, IP-SAS is able to counter the sophisticated malicious attacks. In addition, [58] is designed for a distributed database environment. Thus, it needs $2k + 4$ message exchanges per query, where k is the number of databases in the system. IP-SAS only needs 4 message exchanges per queries regardless of the number of IUs in the system and hence is much more scalable. [59] is designed for cloud computing scenarios where the computing server (corresponding to SAS Server $\mathcal{S}$ in IP-SAS) must communicate with a key conversion server to convert the group-key-encrypted computation results to be encrypted by individual querier's public key. This design is heavier than IP-SAS in terms of key management overhead since it requires the key conversion server to maintain the public keys of all potential queriers while IP-SAS does not require any individual keys from the queriers. In addition, the two-party model in [59] requires its two central servers are both online at the same time and interact with each other to serve a single user query. In contrast, IP-SAS does not have this requirement since $\mathcal{S}$ and $\mathcal{K}$ are not directly communicating with each other. The advantage of the non-interactive design of two servers is that even if one server is down, the other server is still able to perform partial duty. For example, even if $\mathcal{K}$ is offline, SU can still send spectrum request and get the response from $\mathcal{S}$. The only change is that SU needs to wait for $\mathcal{K}$ to be online again to eventually interpret the encrypted spectrum results, but it does not need to query $\mathcal{S}$ again.

3.3 Problem Statement

3.3.1 System Model

We consider an E-Zone-based SAS involving three parties, as illustrated in Figure 3.1: IUs, SUs, and SAS Server. SAS Server refers to a cloud-based spectrum management infrastructure that allocates spectrum resources while considering incumbent and secondary operation protection. In the initialization phase, IUs calculate their E-Zones by employing appropriate radio propagation models, and update SAS with the generated E-Zone information. In spectrum computation phase, SAS Server receives SUs' spectrum access request, determines the availability of the spectrum in SUs' location, and returns the corresponding request response to SUs. A spectrum access request contains the identity of the requester, its location information, and operation parameters. A response contains the availability of spectrum channels in SU's location. If for a given spectrum channel, the requesting SU is in the E-Zone of any

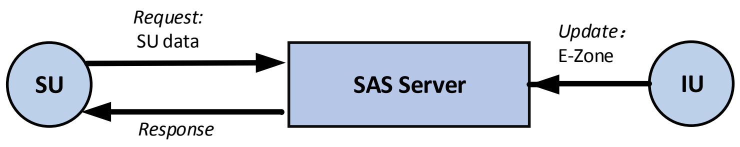


Figure 3.1: The architecture of E-Zone-based SAS

IU, SAS Server responds the SU with a denial of access to that channel. Otherwise, SAS Server responds with an operation permission.

3.3.2 Adversary Models

We consider two adversary models. The basic semi-honest adversary model assumes all the parties in the system are semi-honest (a.k.a. honest-but-curious), which means that all the parties act in an "honest" fashion and exactly follow the protocol as prescribed in Section 3.3.1, but they are also "curious" and attempt to infer private operation data of IUs from the information communicated to them.

Although the assumption of semi-honest adversary model has been widely adopted in related research areas such as cloud security [60] [59], it is often too strong in real world scenarios since it is highly likely that the corrupted parties would deviate from the prescribed protocols by following adversaries' instructions [61]. Therefore, beyond the basic semi-honest model, we also consider a more sophisticated malicious adversary model, where corrupted parties can behave maliciously by deviating from the protocol.

3.3.3 Design Goals

To enable IU-privacy-preserving SAS under the aforementioned adversary models, our IP-SAS design should simultaneously achieve the following goals:

• **Correctness**. The SAS process described in Section 3.3.1 should be performed correctly, i.e. the spectrum should be allocated to SUs correctly based on the IUs' E-Zone information.

• **Privacy**. Minimum IU operation information is leaked from either the IU input data or intermediate and final spectrum computation results.

• **Resistance to malicious behaviors**. IP-SAS should be able to defend against malicious behaviors of corrupted parties.

• **Efficiency**. Above goals of functionality and security should be achieved with small computation and communication overhead.

3.4 Semi-honest System Design

In this section, we present the basic design of IP-SAS in semi-honest adversary model.

3.4.1 Design Overview

As shown in Figure 3.2, IP-SAS involves four parties: (1) a SAS Server $\mathcal{S}$ for spectrum allocation, (2) IUs, (3) SUs, and (4) a Key Distributor $\mathcal{K}$. $\mathcal{K}$ creates a Paillier public/private key pair $(\mathsf{pk},\mathsf{sk})$ and is trusted for keeping sk a secret only known to itself. In the real world, the role of $\mathcal{K}$ can be played by some authorities such as FCC and NTIA, or be outsourced to a cloud service provider under the authorities' authorization, or even IUs *per se*. The high-level idea of IP-SAS is that IUs first encrypt their E-Zone data using pk before sending

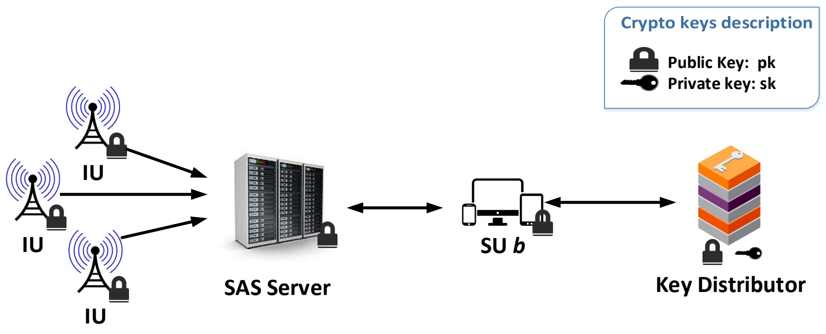


Figure 3.2: IP-SAS overview

it to $\mathcal{S}$, and then $\mathcal{S}$ performs the spectrum allocation computation on the encrypted E-Zone data based on the homomorphic properties of Paillier cryptosystem. The protocol of IP-SAS is shown in Table 3.1. In the following, we describe the details of each step.

3.4.2 E-Zone Information Generation & Representation

To reduce unnecessary wastes of spectrum resources, IUs have to compute E-Zone boundaries accurately. Following the recommendations in [62], we assume that the E-Zones computed by an IU can have multiple tiers. Each tier corresponds to the E-Zone for SUs with a specific operation parameter setting. An SU operation parameter setting is a tuple $(f_s, h_s, p_{ts}, g_{rs}, i_s)$. Similarly, an IU operation parameter setting is also a tuple $(f_i, h_i, p_{ti}, g_{ri}, i_i)$. Descriptions of the operation parameters are presented in Table 3.2.

Plug SU and IU's operation parameter settings into a sophisticated radio propagation model that incorporates terrain details, IU can accurately compute its E-Zone for SU of the specified setting. Specifically, denoting SU's location by l and assume $f_s = f_i$, the E-Zone with respect to SU setting $(f_s, h_s, p_{ts}, g_{rs}, i_s)$ is defined by:

$$EZ(f_s, h_s, p_{ts}, g_{rs}, i_s) := \{l | p_{ti} a_{is} g_{rs} \geq i_s \text{ or } p_{ts} a_{is} g_{ri} \geq i_i\}$$

Table 3.1: The protocol of basic IP-SAS system under semi-honest model

I. Initialization Phase:
$\mathcal{K}$:
(1) $\mathcal{K}$ runs KeyGen and generates a Paillier key pair $(\mathsf{pk},\mathsf{sk})$. pk is distributed to $\mathcal{S}$ and IUs, and sk is kept secret.
IUs (numbered as $1,2,...,k,...,K$):
(2) IU k calculates its E-Zone map $\mathbf{T}_k$.
(3) IU k encrypts $\mathbf{T}_k$ with pk and gets $\widehat{\mathbf{T}}_k$.
(4) IU k uploads $\widehat{\mathbf{T}}_k$ to $\mathcal{S}$.
$\mathcal{S}$*:*
(5) Upon all IUs having uploaded their E-Zone map, $\mathcal{S}$ computes $\widehat{\mathbf{M}} := \oplus_{k\in\{1,2,...,K\}}\widehat{\mathbf{T}}_k$ to aggregate the E-Zone map of all IUs and generates a global E-Zone map $\widehat{\mathbf{M}}$.
II. Spectrum Computation Phase:
SU b:
(6) SU b submits spectrum request containing its operation parameters (h_s,p_{ts},g_{rs},i_s) and location l to $\mathcal{S}$.
$\mathcal{S}$*:*
(7) $\mathcal{S}$ retrieves the corresponding entry in the global E-Zone map $\widehat{\mathbf{M}}$ and obtains $\widehat{\mathbf{X}}_b$.
(8) $\mathcal{S}$ adds random blinding factor $\widehat{\boldsymbol{\beta}}$ to $\widehat{\mathbf{X}}_b$ to generate $\widehat{\mathbf{Y}}_b$.
(9) $\mathcal{S}$ returns $\widehat{\mathbf{Y}}_b$ and $\boldsymbol{\beta}$ to SU b.
III. Recovery Phase:
SU b:
(10) SU b relays $\widehat{\mathbf{Y}}_b$ to $\mathcal{K}$ for decryption.
$\mathcal{K}$*:*
(11) $\mathcal{K}$ decrypts $\widehat{\mathbf{Y}}_b$ with sk and returns $\mathbf{Y}_b$ to SU b.
SU b:
(12) SU b recovers $\mathbf{X}_b$ by removing the blinding factor $\boldsymbol{\beta}$ from $\mathbf{Y}_b$.

Table 3.2: SU/IU operation parameter descriptions

f_s, f_i	Operation frequency
h_s, h_i	Antenna height
p_{ts}, p_{ti}	Transmitter effective radiated power
g_{rs}, g_{ri}	Receiver antenna gain
i_s, i_i	Receiver interference tolerance threshold

where a_{is} is the path attenuation between IU and SU and is a function of SU and IU's locations, SU and IU antenna heights (h_s,h_i), SU/IU shared frequency f_s, and terrain data. The E-Zone information that an IU provides SAS includes the set of $EZ(f_s,h_s,p_{ts},g_{rs},i_s)$ for all considered SU settings.

To reduce computation complexity, we divide the service area of $\mathcal{S}$ into L small grids of equal size. In addition, we assume there are F frequency channels for spectrum sharing, H_s levels of SU antenna height, P_{ts} levels of SU effective radiation power, G_{rs} levels of SU receiver antenna gain, and I_s levels of SU interference tolerance threshold. With these assumptions, each IU k in IP-SAS captures its multi-tier E-Zone information using a multidimensional E-Zone map matrix $\mathbf{T}_k := \{T_k(l,f,h_s,p_{ts},g_{rs},i_s)\}_{L\times F\times H_s\times P_{ts}\times G_{rs}\times I_s}$, where

$$T_k(l,f,h_s,p_{ts},g_{rs},i_s) := \begin{cases} \epsilon, & \text{grid } l \in EZ(f,h_s,p_{ts},g_{rs},i_s) \\ 0, & \text{grid } l \notin EZ(f,h_s,p_{ts},g_{rs},i_s) \end{cases}. \tag{3.1}$$

where ϵ is a positive random number used to denote that SU is in the E-Zone of IU and should not be allowed to transmit, and 0 means SU is out of the E-Zone of this IU and should be allowed to transmit. The reason to use a positive random number here is to guarantee no extra IU information leakage to SU, which will be further explained in Section 3.4.6. Note that if IU k does not operate in a frequency f', then the $EZ(f',*,*,*,*)$ function will be an empty set and the corresponding $T_k(*,f',*,*,*,*)$ values will be all 0.

To protect IU privacy against untrusted $\mathcal{S}$, each entry of $\mathbf{T}_k$ is encrypted by pk before sending to $\mathcal{S}$. For notation simplicity, given any plaintext m, we denote $\widehat{m}$ as its ciphertext created using pk. In essence, IU k transmits $\widehat{\mathbf{T}}_k := \{\widehat{T}_k(l,f,h_s,p_{ts},g_{rs},i_s)\}_{L\times F\times H_s\times P_{ts}\times G_{rs}\times I_s}$ when updating $\mathcal{S}$ with its E-Zone map.

3.4.3 E-Zone Map Aggregation in $\mathcal{S}$

Assume that there are altogether K IUs registered in $\mathcal{S}$ and all of them have sent in their E-Zone map. The first step of $\mathcal{S}$ is to aggregate each IU's E-Zone map to create a global

E-Zone map $\widehat{\mathbf{M}} := \{\widehat{M}(l,f,h_s,p_{ts},g_{rs},i_s)\}_{L\times F\times H_s\times P_{ts}\times G_{rs}\times I_s}$ by $\widehat{\mathbf{M}} := \oplus_{k\in\{1,2,\dots,K\}}\widehat{\mathbf{T}}_k$, which means each entry of $\widehat{\mathbf{M}}$ is:

$$\widehat{M}(l,f,h_s,p_{ts},g_{rs},i_s) := \oplus_{k\in\{1,2,\dots,K\}}\widehat{T}_k(l,f,h_s,p_{ts},g_{rs},i_s), \tag{3.2}$$

where $\oplus$ is the homomorphic version of summation symbol $\sum$. From formula (3.1), it is easy to see that $M(l,f,h_s,p_{ts},g_{rs},i_s) = 0$ means that the grid l is out of E-Zones of all IUs, and $M(l,f,h_s,p_{ts},g_{rs},i_s) > 0$ means that the grid l is within the E-Zone of at least one IU. Formally, for an SU at location l with operation parameter setting (h_s,p_{ts},g_{rs},i_s),

$$M(l,f,h_s,p_{ts},g_{rs},i_s)\begin{cases} = 0, & \text{Spectrum } f \text{ is available.} \\ > 0, & \text{Spectrum } f \text{ is unavailable.}\end{cases}. \tag{3.3}$$

3.4.4 Spectrum Computation Phase & Recovery Phase

When an SU b needs to access the spectrum, it submits a spectrum request containing its operation parameters (h_s,p_{ts},g_{rs},i_s) and location l in plaintext to $\mathcal{S}$. $\mathcal{S}$ retrieves the corresponding entries in the global E-Zone map $\widehat{\mathbf{M}}$ and generates $\widehat{\mathbf{X}}_b := \{\widehat{X}_b(f)\}_F$, where

$$\widehat{X}_b(f) := \widehat{M}(l,f,h_s,p_{ts},g_{rs},i_s). \tag{3.4}$$

Essentially, $\widehat{\mathbf{X}}_b$ holds the information of spectrum availability in SU b's location. $\mathcal{S}$ then homomorphically adds some random blinding factors $\widehat{\boldsymbol{\beta}} := \{\widehat{\beta}(f)\}_F$ to obfuscate the results as follows:

$$\widehat{Y}_b(f) := \mathsf{Add}_{\mathsf{pk}}(\widehat{X}_b(f),\widehat{\beta}(f)), \tag{3.5}$$

where the plaintext of the blinding factor $\beta(f)$ is a one-time random number. $\widehat{\mathbf{Y}}_b$ and the plaintext $\boldsymbol{\beta}$ are both sent back to SU b. SU b needs to decrypt $\widehat{\mathbf{Y}}_b$ for the channel availability information, so it sends $\widehat{\mathbf{Y}}_b$ to $\mathcal{K}$ for decryption. $\mathcal{K}$ decrypts every entry of $\widehat{\mathbf{Y}}_b$ using sk and gets $\mathbf{Y}_b$. $\mathcal{K}$ cannot infer the channel availability information from $\mathbf{Y}_b$ since it does not have the values of the blinding factors $\boldsymbol{\beta}$ to recover $\mathbf{X}_b$. In this way, SU b's spectrum allocation result is kept secret from $\mathcal{K}$.

$\mathcal{K}$ sends $\mathbf{Y}_b$ to SU b, which uses $\boldsymbol{\beta}$ to recover the correct spectrum computation results $\mathbf{X}_b$ as follows:

$$X_b(f) = Y_b(f) - \beta(f). \tag{3.6}$$

From formulas (3.3) and (3.4), it can be seen that if $X_b(f)$ is a positive number, it means that SU b cannot access channel f. If $X_b(f)$ is 0, SU b is permitted to access channel f.

3.4.5 Correctness and security analysis

Correctness

It is straightforward to see that, if the underlying Paillier cryptosystem is correct, the protocol in Table 3.1 correctly performs the SAS process described in Section 3.3.1.

Security

We now analyze the security of IP-SAS under semi-honest adversary model.

Theorem 2. In IP-SAS, it is computationally infeasible for $\mathcal{S}$ to distinguish an IU's operation information as long as Paillier cryptosystem is semantically secure, blinding factors are properly generated.

Due to the limited space, we only provide a sketch of proof. Claim 1 can be proved using composition theorem (under semi-honest adversary model) [42] by evaluating the security of each step of IP-SAS. Basically, according to composition theorem, if each step of IP-SAS is secure, then the whole IP-SAS system is secure. The operation steps of IP-SAS in Table 3.1 are based on the operations $\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Add}$ of Paillier cryptosystem, thus these operations are secure since Paillier cryptosystem is semantically secure [23]. For the blinding factor scheme in step (8)-(10), which are independent of the Paillier operations, the security is guaranteed as long as for all probabilistic polynomial-time adversaries, the probability to distinguish the blinded message is negligible. The detailed proof of blinding factor scheme can be found in [59, 58].

3.4.6 Discussion

IP-SAS not only protects IUs' privacy, it also guarantees no extra information exposure to SUs in the final spectrum computation results other than the spectrum availability in SUs' location. This is because in formula (3.1), we use a positive random number ϵ. If we instead had used some fixed number like 1 to denote that SU is in the E-Zone of IU, the decryption result in formula (3.6) would contain the information of the number of IUs whose E-Zone covers SU's location, which could be leveraged to derive the identity and location of IUs around SU.

It is worth mentioning that if an IU worries that malicious SUs may infer its operation data by analyzing multiple SAS's spectrum responses, the IU can add obfuscation noises to its E-Zone data $\mathbf{T}_k$ as follows:

$$T_k(l, f, h_s, p_{ts}, g_{rs}, i_s) \leftarrow T_k(l, f, h_s, p_{ts}, g_{rs}, i_s) + \phi, \tag{3.7}$$

where ϕ is the noise. Some preliminary noise generation techniques for E-Zone are proposed in [33] for traditional SAS. Note that these obfuscation techniques for traditional SAS are fully compatible with our IP-SAS design since they only affect $\mathbf{T}_k$ by noise addition, and the following process of IP-SAS stays the same. As pointed out by the existing work [33], the potential downside of such obfuscation techniques is the lowered spectrum utilization efficiency due to the added noise. We will investigate the right balance between obfuscation effectiveness and spectrum efficiency in the future.

Finally, although the basic IP-SAS design does not protect SUs' privacy against untrusted SAS, IP-SAS is readily to apply the similar PIR techniques as [32] to address the issue. Specifically, when an SU submits the spectrum access request, it needs to indicate its location information and operation parameters to retrieve the right E-Zone entry from $\mathcal{S}$. By adopting PIR, the SU can still retrieve the right E-Zone entry without revealing its location information and operation parameters to $\mathcal{S}$. In this way, the SU's privacy is preserved.

3.5 Countering Malicious Attacks

In this section, we extend the basic IP-SAS design to defend against the malicious attacks. Defending against the malicious attacks is non-trivial because IP-SAS keeps information known by each party to the minimum to protect IU privacy. Thus, one cannot easily tell cheating behaviors from information discrepancy. In this section, we will show how an attacker can compromise the basic IP-SAS under the malicious adversary model and what are the practical countermeasures. Note that IUs and $\mathcal{K}$ are assumed as trusted parties and are not malicious. Table 3.3 presents the improved protocol of IP-SAS under the malicious adversary model.

Table 3.3: The protocol of IP-SAS system under malicious adversary model

I. Initialization Phase:

$\mathcal{K}$:

(1) $\mathcal{K}$ runs KeyGen and generates a Paillier key pair $(\mathsf{pk},\mathsf{sk})$. pk is distributed to $\mathcal{S}$ and IUs, and sk is kept secret.

IUs (numbered as $1,2,...,k,...,K$):

(2) IU k calculates its E-Zone map $\mathbf{T}_k$.

(3) For each entry e_k of $\mathbf{T}_k$, IU k computes the commitment c_{ek} to it and records the random factor r_{ek}. e_k and r_{ek} are put into one Paillier plaintext to generate w_k. The commitment c_{ek} is published. When all the entries of $\mathbf{T}_k$ is processed, IU k obtains $\mathbf{W}_k$.

(4) IU k encrypts $\mathbf{W}_k$ with pk and gets $\widehat{\mathbf{W}}_k$.

(5) IU k sends $\widehat{\mathbf{W}}_k$ to $\mathcal{S}$.

$\mathcal{S}$:

(6) Upon all IUs having uploaded their $\widehat{\mathbf{W}}_k$, $\mathcal{S}$ computes
$\widehat{\mathbf{M}} := \oplus_{k\in\{1,2,...,K\}}\widehat{\mathbf{W}}_k$.

II. Spectrum Computation Phase:

SU b:

(7) SU b generates spectrum request containing its operation parameters (h_s,p_{ts},g_{rs},i_s) and location l and signs it. SU b submits the spectrum request and the signature to $\mathcal{S}$.

$\mathcal{S}$:

(8) $\mathcal{S}$ retrieves the corresponding entries in the global E-Zone map $\widehat{\mathbf{M}}$ and obtains $\widehat{\mathbf{X}}_b$.

(9) $\mathcal{S}$ adds random blinding factor $\widehat{\boldsymbol{\beta}}$ to $\widehat{\mathbf{X}}_b$ and generates $\widehat{\mathbf{Y}}_b$.

(10) $\mathcal{S}$ signs $\widehat{\mathbf{Y}}_b$ and $\boldsymbol{\beta}$, and returns $\widehat{\mathbf{Y}}_b$, $\boldsymbol{\beta}$, and the signature to SU b.

III. Recovery Phase:

SU b:

(11) SU b relays $\widehat{\mathbf{Y}}_b$ to $\mathcal{K}$ for decryption.

$\mathcal{K}$:

(12) $\mathcal{K}$ decrypts $\widehat{\mathbf{Y}}_b$ and gets $\mathbf{Y}_b$.

(13) $\mathcal{K}$ computes the random numbers $\{\gamma\}$ to generate $\{\widehat{Y_b}(f)\}$ from $\{Y_b(f)\}$ as a proof.

(14) $\mathcal{K}$ sends $\mathbf{Y}_b$ and $\{\gamma\}$ to SU b.

SU b:

(15) SU b recovers $\mathbf{X}_b$ by removing the blinding factor $\boldsymbol{\beta}$ from $\mathbf{Y}_b$.

(16) SU b verifies whether $\mathbf{X}_b$ is correctly computed in $\mathcal{S}$ using formula (3.8).

3.5.1 Malicious SUs

Attack: SU $\boldsymbol{b}$ is involved in step (6) and (10) of Table 3.1. By deviating from step (6), a malicious SU $\boldsymbol{b}$ can obtain a fake $\mathbf{X}'_b$ that does not correspond to its true operation parameters and location through submitting faked data in the spectrum request. A malicious SU $\boldsymbol{b}$ can also directly claim an $\mathbf{X}'_b$ that is different from the spectrum computation result calculated by $\mathcal{S}$ by deviating from step (10).

These malicious behaviors can be leveraged by SU $\boldsymbol{b}$ to gain illegal benefits. For example, even if SU $\boldsymbol{b}$ is denied access by $\mathcal{S}$, it can claim the opposite by tempering the spectrum computation result. Without proper extension to IP-SAS, this cheating SU cannot be caught since SU is the only party that knows the true value of $\mathbf{X}_b$. $\mathcal{S}$ and $\mathcal{K}$ are both unable to know $\mathbf{X}_b$'s value due to IP-SAS's IU-privacy-preserving design.

Countermeasure (Step (7), (10), and (13) in Table 3.3): The countermeasure is based on digital signature system.

To verify whether SU $\boldsymbol{b}$ submits faked data in the spectrum request, a verifier requests SU $\boldsymbol{b}$ to sign its spectrum request before submitting it to $\mathcal{S}$. The verifier can verify the spectrum request by measuring the operation parameters and location of SU $\boldsymbol{b}$ in the field and comparing them with those contained in the spectrum request. The non-repudiation property of digital signature ensures that SU $\boldsymbol{b}$ cannot deny its cheating behavior after been caught.

To verify whether SU $\boldsymbol{b}$ claims a different $\mathbf{X}'_b$ in the final spectrum allocation results, $\mathcal{S}$ is requested to sign $\widehat{\mathbf{Y}}_b$ and $\boldsymbol{\beta}$ and send the signature to SU $\boldsymbol{b}$ as well. The signature ensures that SU $\boldsymbol{b}$ cannot modify $\widehat{\mathbf{Y}}_b$ and $\boldsymbol{\beta}$. Then, based on formula (3.6), the verifier can compute a $\mathbf{Y}'_b$ by adding $\mathbf{X}'_b$ to $\boldsymbol{\beta}$. Finally, if the verifier is able to prove that $\mathbf{Y}'_b$ is not the decryption of $\widehat{\mathbf{Y}}_b$, then it can expose SU $\boldsymbol{b}$'s cheating behavior. Note that the verifier cannot simply decrypt $\widehat{Y_b}(f)$ and compare the decryption with $Y_b'(f)$, since it does not hold the secret key sk. Instead, we propose a zero-knowledge proof for $Y_b'(f) \neq \mathsf{Dec}_{\mathsf{sk}}(\widehat{Y_b}(f))$. Zero-knowledge proof is a method for a prover to prove a statement to the verifier without revealing anything except the veracity of the statement. The zero-knowledge proof works as follows.

1) The verifier requests $\mathcal{K}$ to provide the random number γ for generating $\widehat{Y_b}(f)$ (refer to the Enc operation in Table 2.1). Note that in Paillier cryptosystem, with the secret key $\mathsf{sk} = (\lambda, \mu)$, one can recover the random number $\gamma \in \mathbb{Z}_n$ that is used to create ciphertext $\widehat{m}$ from plaintext m by: (a) Compute $d = g^{-m} \cdot \widehat{m} \mod n$. (b) $\gamma = d^{n^{-1} \bmod \lambda} \mod n$. Refer to Section 5 in [23] for proof.

2) The verifier verifies whether $Y_b'(f)$ is the decryption of $\widehat{Y_b}(f)$ by re-encrypting $Y_b'(f)$ using pk and the random number γ provided by $\mathcal{K}$. If $\mathsf{Enc}_{\mathsf{pk}}(Y_b'(f), \gamma) \neq \widehat{Y_b}(f)$, then the verifier can claim that SU b has cheated about the value of $\mathbf{X}_b$.

3.5.2 Malicious $\mathcal{S}$

Attack: $\mathcal{S}$ is involved in step (5), (7), (8), (9) of the protocol described in Table 3.1. By deviating from these steps, a malicious $\mathcal{S}$ can create a wrong output $\mathbf{Y}_b'$ and/or $\boldsymbol{\beta}'$, which will cause SU b to recover a wrong spectrum computation result $\mathbf{X}_b'$. For example, the malicious $\mathcal{S}$ may alter IU k' E-Zone map by modifying the entries of $\widehat{\mathbf{T}}_k$. Alternatively, the malicious $\mathcal{S}$ may incorrectly execute the aggregation operation in formula (3.2), such as intentionally omitting the E-Zone map of some IU, or involving the E-Zone map of some IU in the aggregation computation for multiple times. The malicious $\mathcal{S}$ may also retrieve wrong entries in $\widehat{\mathbf{M}}$ for SU, i.e. entries that are not associated with SU's operation parameters or location.

Countermeasure (step (3) and (16) in Table 3.3): To guard against malicious $\mathcal{S}$,

Random factor bits (r_{e_k})	E-Zone bits (e_k)

Figure 3.3: The format of an entry w_k in $\mathbf{W}_k$

SUs should be able to verify whether the computation is correctly performed in $\mathcal{S}$ based on the true SU and IU input data. To make the computation in $\mathcal{S}$ verifiable, we propose a novel method based on Pederson commitment scheme [63]. Pederson commitment scheme consists of three phases, commitment setup phase (**Setup**), commit phase (**Commit**), and open phase (**Open**). **Setup** generates the parameters *par* of the commitment scheme. **Commit**(par, r_x, x) outputs a commitment c_x to x, where r_x is a random input factor. **Open**(par, c_x, x, r_x) outputs *accept* if c_x is the commitment to x, otherwise outputs *deny*. Pederson commitment scheme is additive-homomorphic, which means that given two commitments c_{x1} and c_{x2} to x_1 and x_2 with the corresponding random factors r_{x1} and r_{x2} respectively, **Open**($par, c_{x1} * c_{x2}, x_1 + x_2, r_{x1} + r_{x2}$) outputs *accept*. In the following, we employ Pederson commitment scheme to verify the computation in $\mathcal{S}$.

The commitment computation is performed by an IU k after it finishes calculating its E-Zone map $\mathbf{T}_k$ (step (2) in Table 3.1) and before encrypts it using pk (step (3) in Table 3.1). For each entry e_k in $\mathbf{T}_k$, IU k computes a commitment c_{ek} to e_k, records the corresponding random factor r_{ek}, and publishes c_{ek}. Then, as shown in Figure 3.3, IU k partitions the space of a Paillier plaintext message w_k into two segments and puts the random factor r_{ek} in the first segment and $\mathbf{T}_k$ entry e_k in the second segment, essentially making $w_k = \langle r_{ek} || e_k \rangle$. These Paillier plaintext messages created in this way form a new matrix $\mathbf{W}_k$ that carries both $\mathbf{T}_k$ and the random factors of $\mathbf{T}_k$ entries' commitments. Note that to ensure 112-bit security level, Paillier cryptosystem has to use a large 2048-bit plaintext size, which is more than enough to accommodate both r_{ek} and e_k without overlapping. All the IUs use the same way to partition the plaintext message. Then, IU k encrypts the plaintext entries of $\mathbf{W}_k$ and sends $\widehat{\mathbf{W}}_k$ to $\mathcal{S}$. When all IUs finish uploading their $\widehat{\mathbf{W}}_k$, $\mathcal{S}$ homomorphically aggregates $\widehat{\mathbf{W}}_k$

from all IUs to generate the global E-Zone map $\widehat{\mathbf{M}}$, where the random factors of different IUs are also aggregated and stored in $\widehat{\mathbf{M}}$. Due to the very large size of Paillier plaintext message, the bit length reserved of both the r_{ek} and e_k segments in the Paillier plaintext is long enough so that the aggregation in each segment will not cause overflow.

At the end of the recovery phase, an SU b obtains the final spectrum computation results $\mathbf{X}_b$. Consider the entry $X_b(f)$ in $\mathbf{X}_b$ for a given frequency channel f. The first segment of $X_b(f)$ is the aggregation of random factor bits from IU's input matrix $\widehat{\mathbf{W}}_k$, denoted as $R := \sum_{k \in \{1,2,...,K\}} r_{ek}$, and the second part stores the aggregation of E-Zone bits, denoted as $E := \sum_{k \in \{1,2,...,K\}} e_k$. We use a subscript k to denote that r_{ek} and e_k are from IU k. SU b then finds out all the commitments published by IUs that are corresponding to the index $(l, f, h_s, p_{ts}, g_{rs}, i_s)$, where $(l, h_s, p_{ts}, g_{rs}, i_s)$ are the operation parameters provided by SU b in its spectrum request. These commitments are denoted as $c_{e1}, c_{e2}, ..., c_{eK}$. According to additive-homomorphic property of Pederson commitment scheme,

$$\mathbf{Open}(par, c_{e1} * c_{e2} * ... * c_{eK}, E, R) \tag{3.8}$$

should output *accept* if all the computation in $\mathcal{S}$ is correct. Otherwise, the computation in $\mathcal{S}$ is incorrect.

3.6 Improving SAS Efficiency

For a SAS system design with privacy protection, the computation and communication overhead is critical for its scalability and practicality in real world. In this section, we explore two techniques, ciphertext packing and parellel computing, to reduce IP-SAS's overhead and accelerate the processing speed.

3.6.1 Ciphertext Packing

Instead of encrypting every entry of E-Zone map individually, the packing technique packs multiple entries into one ciphertext message, so that we not only reduce the amount of bits that need to be exchanged between different parties, but also achieve homomorphic addition and encryption of multiple entries with only one addition/encryption operation. Ciphertext packing technique [53] is feasible for Paillier crytosystem since the plaintext space of Paillier cryptosystem is significantly larger than the space that is needed for E-Zone entries. Assume we set Paillier's plaintext to be 2048-bit long to achieve 112-bit security strength. The space requirement of the E-Zone bits is determined by the length of random numbers in formula (3.1), which does not need to be very long (e.g. 10-bit is a reasonable choice). Considering all the follow-up addition operations, 50-bit space will be more than enough concerning overflow. Thus, half of the Paillier plaintext space (1024-bit) can hold at least 20 of such 50-bit E-Zone entries. With this observation, we can modify step (3) in Table 3.3 as follows. As shown in Figure 3.4, IU k packs V entries of $\mathbf{T}_k$ into one Paillier plaintext message w_k. Then, IU k computes the Pederson commitment to the entire packed entries $e_{k1}||e_{k2}||e_{k3}||...||e_{kV}$ and puts the random factor of the commitment in the leftmost segment of w_k. Using this packing technique, we can reduce the dimension of $\widehat{\mathbf{W}}_k$ and the number of homomorphic addition/encryption operations by a factor of V.

One undesirable side effect of adopting the packing technique is that $\mathbf{X}_b$ contains E-Zone entries that are not related to SU's spectrum request. This may cause unnecessary leakage of IU information when SU obtains $\mathbf{X}_b$ in step (15) of Table 3.3. To solve this problem, $\mathcal{S}$ needs to perform an extra masking step before step (9) in Table 3.3. In the masking step, $\mathcal{S}$ first homomorphically adds a random mask to hide irrelevant entries in $\mathbf{X}_b$. For example, assume in $X_b(f) = \langle r_e||e_1||e_2||e_3||...||e_V\rangle$, only e_2 is relevant to SU request. $\mathcal{S}$ creates a random mask $Z := \langle r_z||z_1||0||z_3||...||z_V\rangle$. Here $z_1, z_3, ..., z_V$ are random numbers for masking $e_1, e_3, ..., e_V$,

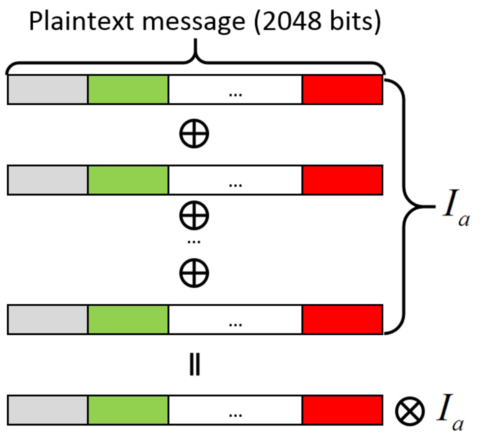


Figure 3.4: Packing V entries of E-Zone map into a plaintext message w_k

and r_z is the random factor of the commitment c_z to $z_1||0||z_3||,...,||z_V$. $\mathcal{S}$ encrypts Z using pk and computes $\widehat{X_b}(f) \leftarrow \mathsf{Add}_{\mathsf{pk}}(\widehat{Z}, \widehat{X_b}(f))$ to hide the irrelevant E-Zone entries in $X_b(f)$ while preserving the desired entry e_2. Since the masking bits are kept secret, SU cannot recover the irreverent E-Zone entries in $\mathbf{X}_b$. For the purpose of r_z, note that the random masking bits change the value of E in formula (3.8), thus making $\mathbf{Open}(par, c_{e1} * c_{e2} * ... * c_{eK}, E, R)$ no longer output *accept* even if the computation in $\mathcal{S}$ is correct. To correct this, $\mathcal{S}$ needs to create the commitment c_z to the random masking bits $z_1||0||z_3||,...,||z_V$, so that the random factor r_z of masking bits is also counted in R. $\mathcal{S}$ publishes the commitment c_z. The revised version of formula (3.8) becomes $\mathbf{Open}(par, c_{e1} * c_{e2} * ... * c_{eK} * c_z, E, R)$.

3.6.2 Parallel Computing

In the initialization phase of IP-SAS, all the computation tasks of IUs and $\mathcal{S}$ are readily to be parallelly computed to further reduce the computation overhead. Specifically, in Table 3.3, E-Zone map generation (step (2)), commitments computation (step (3)), encryption (step (4)), and aggregation (step (6)) can be parallelly computed by splitting the matrices into subsets. The computation tasks of the subsets are distributed to multiple threads and servers. Moreover, for the spectrum computation phase and recovery phase, $\mathcal{S}$ and $\mathcal{K}$ can handle multiple SUs' request concurrently.

3.7 Experiments and Evaluation

3.7.1 Implementation and Experiment Settings

We set the service area of SAS to be a large 154.82 km^2 area in Washington DC, and all the IUs need to generate the E-Zone maps on this area. We employ the sophisticated Longley-Rice (L-R) radio propagation model provided by SPLAT! [55] to calculate the point-to-point attenuation values for E-Zone computation. L-R model takes in terrain data, and the granularity of terrain data determines the accuracy of attenuation computation. In order to obtain accurate attenuation values, we feed high resolution terrain elevation data SRTM3 [46] to SPLAT!.

We implement a Paillier cryptosystem of 112-bit security level by setting the security parameter n as 2048 bits. We build IP-SAS on the Paillier implementation and execute IP-SAS over two Dell desktops, both with Intel Core i7-3770 CPU @3.40GHz and 12GB RAM. The plaintext size the Paillier cryptosystem is 2048-bit long, and the first 1024-bit space is used to store random factor bits of commitment scheme. In the rest 1024-bit space, we pack 20 50-bit E-Zone entries. We distribute the computation tasks to 16 concurrent threads, and each of the two desktops runs 8 threads. For digital signature scheme, we employ the RSA-2048 signature system. Table 3.4 shows the experiment parameter settings.

Table 3.4: Experiment parameter settings

Number of IUs (K)	500
Number of grids (L)	15482
Number of frequency channels (F)	10
Number of SU antenna heights (H_s)	5
Number of SU effective radiated power values (P_{ts})	3
Number of SU receiver antenna gain values (G_{rs})	3
Number of SU interference tolerance thresholds (I_s)	3

3.7.2 Computation and Communication Overhead

Table 3.5 shows the computation overhead of each step in Table 3.3 before/after applying the acceleration methods presented in Section 3.6. After adopting the acceleration methods, an IU can generate an encrypted E-Zone map in 2 hours, compared with 101 hours without acceleration. Note that more than 80% of the 2-hour time is actually spent on the plaintext computation (e.g. E-Zone map calculation using L-R model), and the extra overhead brought by IP-SAS's privacy-preserving operations takes less than 20% of the total time. Also note that E-Zone map calculation does not need to be repeated frequently since IUs' operation parameters are often static. After $\mathcal{S}$ receives the E-Zone data from all IUs, it needs about 5.2 minutes for aggregation with acceleration. In the spectrum computation phase and recovery phase, it takes a short response time of around 1.25 second for an SU to get the final spectrum allocation result after submitting its request.

Table 3.5: Computation Overhead of IP-SAS

	Before Acceleration	After Acceleration
(2) E-Zone map calculation	21.2 hours	1.65 hours
(3) Commitment	11.7 hours	3.21 minutes
(4) Encryption	68.5 hours	17.9 minutes
(6) Aggregation	29.0 hours	5.2 minutes
(8)-(10) $\mathcal{S}$ Response	1.12 seconds	1.11 seconds
(12)(13) Decryption	0.134 seconds	0.134 seconds
(15) Recovery	-	-
(16) Verification	0.118 seconds	0.118 seconds

Table 3.6 shows the communication overhead of IP-SAS before/after applying the ciphertext packing technique. In the initialization step, the ciphertext packing technique reduces the amount of encrypted E-Zone data transmitted from an IU to $\mathcal{S}$ by 95% from nearly 10 GB to 510 MB. Note that IUs do not need to provide its E-Zone data frequently since their E-Zones are often static. Moreover, IUs can send the E-Zone data through wired backbone with high data transmission speed, thus the E-Zone data transmission can be finished in short time. In the spectrum computation phase and recovery phase, the communication overhead is 17.8 KB for an SU, which is small enough to satisfy the requirement of both static and mobile SUs in real world.

Table 3.6: Communication Overhead of IP-SAS

	Before Packing	After Packing
(4) IU$\rightarrow \mathcal{S}$	9.97 GB	510 MB
(6) SU$\rightarrow \mathcal{S}$	25 B	25 B
(9) $\mathcal{S}$ $\rightarrow$SU	7.75 KB	7.75 KB
(10) SU$\rightarrow \mathcal{K}$	5 KB	5 KB
(13) $\mathcal{K}$ $\rightarrow$SU	5 KB	5 KB

3.8 Summary

In this chapter, we build IP-SAS to perform efficient SAS process while preserving IUs' privacy. This system leverages additive-homomorphic encryption to allow secure spectrum allocation computation. Moreover, we design mechanisms to prevent malicious parties from compromising IP-SAS. Finally, we implement acceleration methods to increase IP-SAS's efficiency. Experiments based on real-world data demonstrate the scalability and practicality of IP-SAS in real-world deployment. Evaluation results show that IP-SAS can respond an SU's spectrum request in only 1.25 seconds with communication overhead of 17.8 KB.

Chapter 4

RE-SAS: Privacy-Preserving SAS Without Key Distributor

In this chapter, we present RE-SAS, a privacy-preserving SAS without Key Distributor.

4.1 Challenges and Contributions

RE-SAS adopts the Exclusion-Zone-based spectrum management scheme. Compared with IP-SAS, RE-SAS does not need an online somewhat-trusted third party Key Distributor and hence eliminate the single point of vulnerability issue. In addition, RE-SAS can handle a spectrum request in 2.59 ms on average, which is more than two orders of magnitude faster than IP-SAS. These advantages of RE-SAS are achieved through leveraging the homomorphic property of an efficient proxy re-encryption scheme, called "AFGH" scheme [24]. The main contributions of this chapter are summarized below.

1 We proposed a novel privacy preserving framework for SAS, which is computationally

efficient, provably secure, and free of online trusted third party.

2 To the best of our knowledge, we are the first to discover the homomorphic property of a proxy re-encryption scheme, and we are also the first to use homomorphic proxy re-encryption scheme to achieve privacy preserving system design.

3 We rigorously define and analyze IU privacy for database driven system using standard security experiments.

The rest of this chapter is organized as follows. Section 4.2 introduces the system model and the design objective. Section 4.3 provides background information and Section 4.4 presents technical details. Formal security definitions and corresponding analysis are presented in Section 4.5. In 4.6, we discuss techniques which addresses the resource allocation issue in ER-SAS. Evaluations are provided in section 4.7. Summary of the chapter is offered in Section 4.8.

4.2 System Model and Security Properties

4.2.1 System Model

We consider an E-Zone-based server-driven DSA framework consisting of SAS, IUs and SUs. An SU requests an assignment from the access system by sending information on its present location and expected transmission channel. SAS needs to ensure SU not located at any E-Zone before approving the request. Meanwhile, IUs constantly send E-Zone information to SAS so that it can set up a real time database on E-Zones of IUs.

The system model in this chapter is a simplified yet general one. There exists hierarchy among SUs sometimes. For example, in 3.5 GHz band FCC proposed two types of SUs in its proposal [64] priority access and general access, where the framework is referred as "three-tiered". This affects the resource allocation strategy, yet it is out of the scope of our major focus, i.e. preserving IU privacy. We will discuss the resource allocation issue in section 4.6.

4.2.2 Security properties

In the following discussions, we provide an overview on security properties of RE-SAS. Formal definitions will be presented after introducing details of the system.

Correctness: This property requires that when an SU would cause interference greater than any IU's sensitivity threshold, its spectrum request can not be approved. i.e. SU can not receive a valid spectrum license in this case.

Privacy for IUs: For privacy property, we assume SAS as the adversary. We assume that the SAS may misbehave the protocol and collude with some SUs. IU privacy requires that SAS is not able to identify the E-Zone information of a single IU so as not to infer its operational data.

Meanwhile, we also expect that in non-collusion scenario IUs have stronger privacy guarantee. This requires that if SAS is not colluding with any SUs, then SAS is not able to identify the any E-Zone information, which reflects the operational feature of IU groups.

Soundness: Since SAS does not own the spectrum, for any SU who desn not get spectrum access, it may require confirmation to ensure SAS faithfully executing the protocol. This issue is brought by privacy preserving SAS, since without privacy preserving feature, SAS can simply present E-Zone info that is digitally signed by IUs to a trusted third party. Soundness property requires that if an SU is not located in E-Zone, it should never get a negative response from SAS which can not be disputed.

4.3 Cryptosystem Preliminaries

In this section, we introduce the features of the crypto-systems that are leveraged in RE-SAS design.

4.3.1 Blinear Pairing on DDH groups

Definition 4.3.1. (Bilinear Groups): $(\mathbb{G}_1, \mathbb{G}_2)$ is called a bilinear group pair, if there exists a group $\mathbb{G}_T$ and a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ with the following properties:

1. $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are multiplicative cyclic groups of prime order p;
2. g_1 is a generator of $\mathbb{G}_1$, and g_2 is a generator of $\mathbb{G}_2$;
3. ψ is an efficiently computable isomorphism from $\mathbb{G}_2$ to $\mathbb{G}_1$, with $\psi(g_2) = g_1$;
4. e is an efficiently computable bilinear map with the following properties:
    - Bilinear: $e(u^a, v^b) = e(u, v)^{ab}$, $\forall u \in \mathbb{G}_1$, $v \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}_p^*$; and
    - Non-degenerate: $e(g_1, g_2) \neq 1$.

We utilize the Type 1 pairing, which means that $\mathbb{G}_1 = \mathbb{G}_2$, and ψ is an identity map [65]. Also, we assume that the elements generated in all the algorithms are efficiently checked and verified for their membership of the specified groups to thwart small-subgroup attacks [66].

4.3.2 Homomorphic proxy re-encryption scheme

RE-SAS uses AFGH scheme, which is a widely used proxy re-encryption scheme proposed by G. Ateniese et al. [24]. The encryption of AFGH scheme additionally allows for homomorphic multiplication and exponentiation. Specifically homomorphic property for multiplication means that, if two messages M and M' are encrypted to C and C' respectively, $C \otimes C'$ can be decrypted as $M \cdot M'$. Note that such an "$\otimes$" operation is not specified in [24], and in this chapter we divulge this property and define the "multiplication" operation.

The following describes the construction of the AFGH scheme, which is the third attempt in [24]. The difference are mostly at the presentation level. We stick to one option for first level encryption so as to properly define operation between cipher text and further exploit the homomorphic property.

- **System parameters**: A Type I bilinear pairing system, g as the generator of $\mathbb{G}_1$, and $Z = e(g,g)$.
- **Key Generation($KG(a_1,a_2)$)**: For two inputs $a_1, a_2 \in \mathbb{Z}_p^*$, set secret key $sk = (a_1,a_2)$, compute public key $pk = (Z^{a_1}, g^{a_2})$. This algorithm can also run without input parameters; in that case we randomly sample parameters a_1 and a_2 from $\mathbb{Z}_p^*$, which is denoted as $a_1, a_2 \leftarrow_\$ \mathbb{Z}_p^*$.
- **Re-Encryption Key Generation ($RG(sk_a, pk_b)$)**: taking private key $sk_a = (a_1,a_2)$ of user A and public key $pk_b = (Z^{b_1}, g^{b_2})$ for user B, the re-encrypting key is computed by $rk_{A\to B} := (g^{b_2})^{a_1} = g^{a_1 b_2}$.
- **First-Level Encryption ($E_1(M, pk_a)$)**: for a message $M \in \mathbb{G}_T$ and public key $pk_a = (Z^{a_1}, g^{a_2})$, select $r \leftarrow_\$ \mathbb{Z}_p^*$, and compute $c_1 = Z^r \cdot M$, $c_2 = Z^{ra_2}$. The ciphertext is $C = (c_1, c_2)$.
- **Second-Level Encryption ($E_2(M, pk_a)$)**: for a message $M \in \mathbb{G}_T$ and public key $pk_a = (Z^{a_1}, g^{a_2})$, select $r \leftarrow_\$ \mathbb{Z}_p^*$, and compute $c_1 = Z^{ra_1} \cdot M$, $c_2 = g^r$. The ciphertext is $C = (c_1, c_2)$.

- **First-Level Decryption** ($D_1(C_r, sk_b)$): for a first-level ciphertext $C_r = (c_1, c_2)$ and its corresponding private key $sk_b = (b_1, b_2)$, the plaintext is obtained by computing $M^* = \frac{c_1}{c_2^{1/b_2}}$.

- **Second-Level Decryption** ($D_2(C_r, sk_a)$): for a second-level ciphertext $C_r = (c_1, c_2)$ and its corresponding private key $sk_a = (a_1, a_2)$, the plaintext is obtained by computing $M^* = \frac{c_1}{e(g^{a_1}, c_2)}$.

- **Re-Encryption** ($R(C, rk_{A \to B})$): 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 key g^b by computing $c_2^* := e(c_2, rk_{A \to B}) = Z^{rb}$. The re-encrypted first level ciphertext is $C_r = (c_1, c_2^*)$.

Homomorphic property of AFGH scheme

We divulge the homomorphic feature of AFGH scheme and propose two algorithms to define multiplication and exponentiation operation on cipher text space:

- **Homomorphic multiplication** $mul(C, C')$, for an input of two cipher texts $C = (c_1, c_2)$ and $C' = (c_1', c_2')$ which are encrypted using the same public key and the same level of encryption, the output is $C_m = (c_1 c_1', c_2 c_2')$.

- **Homomorphic exponentiation** $exp(C, \alpha)$, for an input of first level or second level ciphertext $C = (c_1, c_2)$ and power $\alpha \in \mathbb{Z}_p^*$, the output is $C_e = (c_1^{\alpha}, c_2^{\alpha})$.

In the rest of this chapter, we'll use operator "$\otimes$" to express algorithm mul. i.e. $C \otimes C' := mul(C', C')$.

For the correctness of proposed homomorphic operation algorithm, we have the following propositions.

Proposition 1. Algorithm $mul(\cdot,\cdot)$ is a correct homomorphic multiplication algorithm, i.e., given:

$$\begin{aligned}
&(sk,pk) \leftarrow KG \\
&M, M' \leftarrow_{\$} \mathbb{G}_T \\
&C_1 \leftarrow E_1(M,pk), C_2 \leftarrow E_1(M,pk) \\
&C_1' \leftarrow E_1(M',pk), C_2' \leftarrow E_2(M',pk) \\
&C_{m1} \leftarrow mul(C_1, C_1'), C_{m2} \leftarrow mul(C_2, C_2'),
\end{aligned}$$

We have $D_1(C_{m1}, sk) = D_2(C_{m2}, sk) = MM'$.

Proof. Let $sk = (a_1, a_2)$ and $pk = (Z^{a_1}, g^{a_2})$. Followed by the details of encryption algorithms, let $C_1 = (MZ^{r_1}, Z^{r_1 a_2})$, $C_1' = (M'Z^{s_1}, Z^{s_1 a_2})$, $C_2 = (MZ^{r_2 a_1}, g^{r_2})$ and $C_2' = (M'Z^{s_2 a_1}, g^{s_2})$. Then followed by the the details of $mul(\cdot,\cdot)$ algorithm, we have:

$$\begin{aligned}
C_{m1} &= (MZ^{r_1} \cdot M'Z^{s_1}, Z^{r_1 a_2} Z^{s_1 a_2}) \\
&= (MM' \cdot Z^{r_1+s_1}, Z^{(r_1+s_1)a_2}), \\
C_{m2} &= (MZ^{r_2 a_1} \cdot M'Z^{s_2 a_1}, g^{r_2} g^{s_2}) \\
&= (MM' \cdot Z^{(r_2+s_2)a_1}, g^{r_2+s_2})
\end{aligned}$$

Therefore we have:

$$D_1(C_{m1}, sk) = \frac{MM' \cdot Z^{r_1+s_1}}{(Z^{(r_1+s_1)a_2})^{1/a_2}} = MM' \tag{4.1}$$

$$D_1(C_{m2}, sk) = \frac{MM' \cdot Z^{(r_2+s_2)a_1}}{e(g^{a_1}, g^{r_2+s_2})} = MM' \tag{4.2}$$

$$. \tag{4.3}$$

Note that the equation (4.2) holds since $Z^{r_2+s_2}a_1 = e(g,g)^{r_2+s_2}a_1 = e(g^{a_1}, g^{r_2+s_2})$. □

Proposition 2. Algorithm $exp(\cdot,\cdot)$ is a correct homomorphic exponentiation algorithm, i.e., given:

$$\begin{aligned}
&(sk, pk) \leftarrow KG\\
&M \leftarrow_{\$} \mathbb{G}_T, \alpha \leftarrow_{\$} \mathbb{Z}_p^*\\
&C_1 \leftarrow E_1(M, pk), C_2 \leftarrow E_2(M, pk)\\
&C_{e1} \leftarrow exp(C_1, \alpha), C_{e2} \leftarrow exp(C_2, \alpha)
\end{aligned}$$

We have $D_1(C_{e1}, sk) = D_2(C_{e2}, sk) = M^\alpha$.

Proof. The proof is similar as the proof for Proposition 1. Let $sk = (a_1, a_2)$ and $pk = (Z^{a_1}, g^{a_2})$. Followed by the details of encryption algorithms, let $C_1 = (MZ^{r_1}, Z^{r_1 a_2})$, and $C_2 = (MZ^{r_2 a_1}, g^{r_2})$.

Then followed by the the details of $exp(\cdot,\cdot)$ algorithm, we have:

$$\begin{aligned}
C_{e1} &= ((MZ^{r_1})^\alpha, (Z^{r_1 a_2}))^\alpha)\\
&= (M^\alpha Z^{\alpha r_1}, Z^{\alpha r_1 a_2})\\
C_{e2} &= ((MZ^{r_2 a_1})^\alpha, (g^{r_2})^\alpha)\\
&= (M^\alpha Z^{\alpha r_2 a_1}, g^{\alpha r_2}).
\end{aligned}$$

Therefore we have:

$$D_1(C_{e1}, sk) = \frac{M^\alpha Z^{\alpha r_1}}{(Z^{\alpha r_1 a_2})^{1/a_2}} = M^\alpha \tag{4.4}$$

$$D_1(C_{e2}, sk) = \frac{M^\alpha Z^{\alpha r_2 a_1}}{e(g^{a_1}, g^{\alpha r_2})} = M^\alpha. \tag{4.5}$$

$$\tag{4.6}$$

Equation (4.5) holds since $Z^{\alpha r_2 a_1} = e(g,g)^{\alpha r_2 a_1} = e(g^{a_1}, g^{\alpha r_2})$. □

In addition, we also claim that the our additional algorithm does not break the security of AFGH scheme itself. This is because the proof of Theorem 3.1 in [24] is not affected.

4.3.3 Cryptographic Assumptions

The security of AFGH scheme is preserved under extended decisional bilinear Diffie-Hellman (EDBDH) assumption, Decision Linear (DLIN) and discret logarithm (DL) assumption [24].

4.4 System framework

In the following subsections, we present the structure as well as technical details of the algorithms and the protocols that make up RE-SAS.

4.4.1 Overview

As shown in Figure. 4.1, there are four parties in a RE-SAS system: (1) IUs, (2) a SAS server for spectrum management, (3) SUs, and (4) a IU tracker for maintaining the commitments of all encrypted blinded E-Zone data.

Two functionss of SAS are expected to be realized: maintaining the database of E-Zone map, and spectrum allocation based on the encypted map. To ensure their operational security, IUs send the input of E-Zone information encrypted and blinded, to the SAS, which is denoted as $[\![\mathbf{X}]\!]$. After collecting all encrypted and blinded E-Zone data for one time period, SAS will try to release the overall blinding by taking the aid of some IU. This process will remove the blinding of joint exclusion zone. The resulting unblinded yet encrypted E-Zone map is denoted as $[\![\mathbf{D}]\!]_{II}$.

To handle a spectrum request, SAS computes a potential spectrum license `cred`, based on the encrypted E-Zone map $[\![\mathbf{D}]\!]_{II}$. 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 SU is not located in E-Zone.

When an SU recovers an invalid spectrum license, it may contact spectrum enforcer to further confirm this request. The SU sends its original spectrum request and the accumulated commitments associated with that request to the SAS. The enforcer will fetch the corresponding encrypted data entries from SAS, and fetch accumulated helper values from IU trackers. With all these inputs, the enforcer will be able to decide whether SU is located in E-Zone and dispute the result from SAS associated with this spectrum request.

In the following, we will describe the details of each step in RE-SAS design.

4.4.2 System Setup

To initialize the system, the following three steps are executed:

- 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 $(\texttt{sk}, \texttt{pk}) = \mathsf{KG}(a_1, a_2)$.

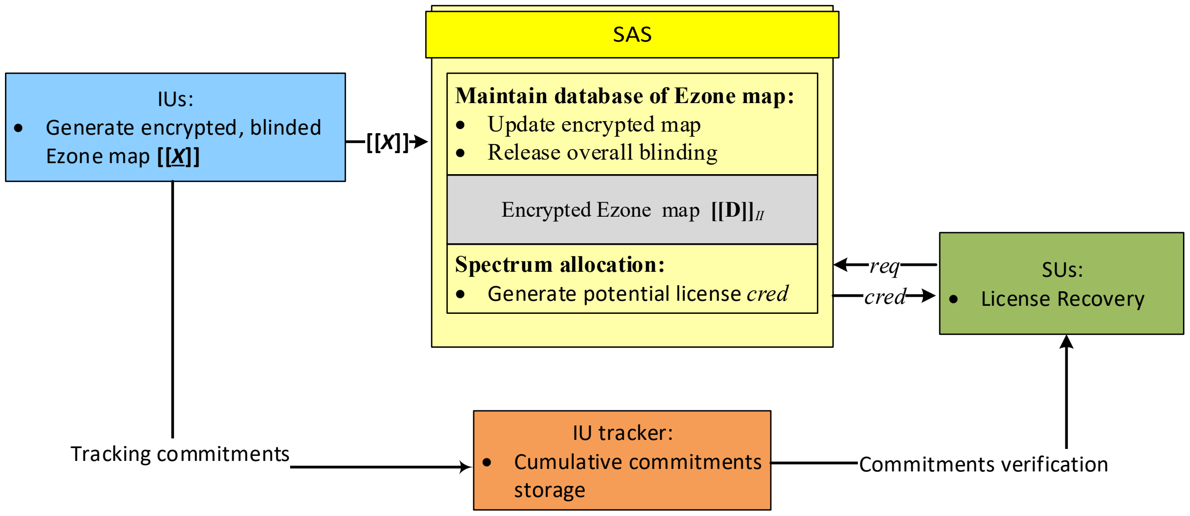

SAS
IUs:
Generate encrypted, blinded Ezone map [[X]]
[[X]]
Maintain database of Ezone map:
Update encrypted map
Release overall blinding
Encrypted Ezone map [[D]]_II
Spectrum allocation:
Generate potential license cred
req
cred
SUs:
License Recovery
IU tracker:
Cumulative commitments storage
Tracking commitments
Commitments verification

Figure 4.1: System Framework

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 $\mathtt{params} = (\mathbb{G}_1, \mathbb{G}_T, p, e, g, Z, f_1, h, Y)$.

- Step 3: Publish $\mathtt{ipk}$ and $\mathtt{params}$.

In RE-SAS , an SU b registers its identity before sending any spectrum request. It generates an randomized AFGH key pair $(\mathtt{sk}_b, \mathtt{pk}_b)$ and sends $\mathtt{pk}_b$ to obtain 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 RE-SAS moves to the normal operation state. In this state, one function of SAS is to maintain the database of E-Zone map, and the other is to process spectrum allocation based on the encypted map. In the following, we describe the details of these two SAS functions.

4.4.3 Details of maintaining the database of E-Zone map

Input data types

The input data from an IU to SAS indicates its exclusion zone information. To limit the computation overhead, all input data are discretized into a limited set of possible values. The discretization process divides RE-SAS 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. The whole spectrum owned by IU is divided into F channels.

Generate encrypted, blinded E-Zone map report

The E-Zone information is represented as a matrix $\mathbf{T}$ of dimension $L \times F$. If channel f at grid l is regarded as E-Zone by the IU, then $\mathbf{T}$'s entry $\mathbf{T}_{l,f}$ is a random non-identity element picked from $\mathbb{Z}_p$; formally,

$$\mathbf{T}_{l,f} \leftarrow_{\$} \mathbb{Z}_p \backslash \{1\}\,. \tag{4.7}$$

For the other entries, we set $\mathbf{T}_{l,f}$ to be 1; formally,

$$\mathbf{T}_{l,f} \leftarrow 1 \tag{4.8}$$

To blind the E-Zone data at each location, the IU pick a random nonce $\mathbf{A}_{l,f}$ in $\mathbb{Z}_p$, add this nonce to $\mathbf{T}_{l,f}$ and compute the the exponentiation of the result at base Y to obtain entries of blinded E-Zone data map $\mathbf{X}$; formally,

$$\mathbf{X}_{l,f} \leftarrow Y^{\mathbf{T}_{l,f}+\mathbf{A}_{l,f}} \tag{4.9}$$

for all entries of $\mathbf{T}_{l,f}$ and $\mathbf{A}_{l,f}$.

Afterwards, IU encryptes every entry of $\mathbf{X}$ using level 2 encryption and public key `ipk`. We denote the encrypted results as $[\![\mathbf{X}]\!]_{II}$, which is sent to SAS.

Meanwhile, IU computes the Pedersen commitment of blinded e-zone data. That is:

$$\mathbf{r}_{l,f} \leftarrow_{\$} \mathbb{Z}_p, \mathbf{C}_{l,f} \leftarrow Y^{\mathbf{T}_{l,f}+\mathbf{A}_{l,f}} H^{\mathbf{r}_{l,f}} \tag{4.10}$$

for all entries of $\mathbf{T}_{l,f}$ and $\mathbf{A}_{l,f}$. Then $\mathbf{C}_{l,f}$ and $H^{\mathbf{r}_{l,f}}$ are sent to the IU tracker. The helper value $\mathbf{B}$ is computed by:

$$\mathbf{B}_{l,f} \leftarrow Y^{\mathbf{A}_{l,f}} \mathbf{C}_{l,f} \tag{4.11}$$

for all entries of $\mathbf{A}_{l,f}$ and $\mathbf{C}_{l,f}$. Then the helper value is encrypted using level 2 encryption and the result $[\![\mathbf{B}]\!]_{II}$ is also sent to SAS.

Update the encrypted, blinded map

Upon receiving all encrypted, blinded E-Zone map $\{[\![\mathbf{X}(i)]\!]_{II}\}_{i=1}^{N}$ for one time interval, SAS firstly integrates them together to form an aggregated map $[\![\mathbf{X}'(i)]\!]_{II}$ by computing element-wise homomorphic product of all input maps. That is:

$$[\![\mathbf{X'}_{l,f}]\!]_{II} \leftarrow \otimes_{i=1}^{N} [\![\mathbf{X}(i)_{l,f}]\!]_{II}. \tag{4.12}$$

Then SAS proceed to release the overall blinding.

4.4.4 Release the overall blinding

Note that SAS also receives all encrypted helper value map $\{[\![\mathbf{B}(i)]\!]_{II}\}_{i=1}^{N}$ for same time interval. At first SAS aggregates them together to form an aggregated helper $[\![\mathbf{B}'(i)]\!]_{II}$ by computing element-wise homomorphic product of all input maps. That is:

$$[\![\mathbf{B}'_{l,f}]\!]_{II} \leftarrow \otimes_{i=1}^{N}[\![\mathbf{B}(i)_{l,f}]\!]_{II}. \tag{4.13}$$

Then SAS queries the IU tracker to obtain the current cummulative commitment value $\mathbf{C}'$ where $\mathbf{C}'_{l,f} = \prod_{i=1}^{N}\mathbf{C}(i)_{l,f}$ for all locations l. Then SAS release the overall blinding by element-wise homomorphically dividing $[\![\mathbf{X}']\!]_{II}$ by $[\![\mathbf{B}']\!]_{II}$, then element-wise homomorphically multiplies the result to $[\![\mathbf{C}']\!]_{II}$. That is,

$$[\![\mathbf{D}_{l,f}]\!]_{II} \leftarrow [\![\mathbf{X}'_{l,f}]\!]_{II} \otimes \mathsf{inv}\left([\![\mathbf{B}'_{l,f}]\!]_{II}\right) \otimes [\![\mathbf{C}'_{l,f}]\!]_{II}. \tag{4.14}$$

4.4.5 Spectrum computation

Spectrum computation functionality requires SAS to generate a spectrum license and send it to the SU if and only if it is not located in any exclusion zone set by an IU. This is not likely to be achieved directly, since SAS is not assumed to have knowledge on the overall E-Zone information if it is not colluding with SU(s). To achieve this, we apply another layer of encryption on a valid spectrum license using a randomized key; then SAS mixes the E-Zone data with this key and send it to the SU. In the end, SU is capable of recovering this key if and only if the corresponding E-Zone data indicates "non-E-Zone".

Let l be the location of the SU. SAS firstly generates the valid license `cred`. Then SAS picks random element α in $\mathbb{G}_T$ and hashes it to random bit string `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 $[\![\mathtt{cred}]\!]_{AES,\mathtt{k}}$. Meanwhile, SAS encrypts α to level 2 ciphertext using key $\mathtt{ipk}$ and homomorphically multiplies it with the $[\![\mathbf{D}_{l,f}]\!]_{II}$ returned by the algorithm. That is, set $\hat{K} \leftarrow \mathsf{E}_{II}(\alpha, \mathtt{ipk}) \otimes [\![\mathbf{D}_{l,f}]\!]_{II}$. Note that if the SU is not located in an E-Zone, then $\mathbf{D}_{l,f} = 1_{\mathbb{G}_T}$ and hence $\hat{K}$ can be decrypted to α. Otherwise, $[\![\mathbf{D}_{l,f}]\!]_{II}$ is the ciphertext of some random number and decryption of $\hat{K}$ just result in a random number.

Finally, SAS and sends $\hat{K}$ and encrypted license $[\![\mathtt{cred}]\!]_{AES,\mathtt{k}}$ to the SU.

4.4.6 Operations at SU: recover license

Upon receiving the encrypted, blinded key $\hat{K}$ and the encrypted license $[\![\mathtt{cred}]\!]_{AES,\mathtt{k}}$. The SU attempts to recover the encrypted, blinded key by re-encryption and decryption of AFGH cryptosystem, using corresponding keys. Then SU attempts to recover a valid license. This proceeds as follows:

- Using the reencryption key $\mathtt{rk}_b$ obtained during its registration process, SU b re-encrypts $\hat{K}$ to be a first level cipher text encrypted by $\mathtt{pk}_b$: $\hat{K}^* \leftarrow \mathsf{R}(\hat{K}, \mathtt{rk}_b)$.
- SU b decrypts C^* and recovers an AES key:

$$\mathtt{k}^* \leftarrow H(\mathsf{D}_I(\hat{K}^*, \mathtt{sk}_b)).$$

- SU b obtains the recovered spectrum license:

$$\mathtt{cred}^* \leftarrow \mathsf{D}_{AES}([\![\mathtt{cred}]\!]_{AES,\mathtt{k}}, \mathtt{k}^*).$$

Afterwards, the SU checks the validity of `cred`*. Only when `cred`* has been successfully validated, can the SU access the spectrum. Note that if the validation fails, then it means either SAS doesn't honestly execute the protocol or SU is not using the correct key to recover the encrypted license, which means $\mathbf{D}_{l,f} \neq 1_{\mathbb{G}_T}$ and this SU is located in E-Zone.

4.4.7 Operations at SU: confirmation of negative response

When SU recover an invalid response, it may send its original request and the full response from SAS to the enforcer to confirm this negative response. The enforcer proceeds as follows to verify the integrity of SAS:

- Verifiy the integrity of SU: the enforcer will content and the signature of the response from SAS to ensure SU is sending the original response and confirm this implies a negative response.
- The enforcer extract the location and time from the response and get the corresponding accumulated commitments $\mathbf{C}'_{l,f}$, helper value $\mathbf{B}'_{l,f}$ and nonce $\mathbf{H}'_{l,f}$.
- If $\mathbf{H}'_{l,f} = (\mathbf{C}'_{l,f})^2\mathbf{B}'_{l,f}$, dispute the execution.

4.5 Formal Security Definitions and Analysis

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

4.5.1 Correctness

Correctness property requires that when an SU is located in the E-Zone of any of IUs, its spectrum request cannot be approved. i.e. SU cannot receive a valid or useful spectrum license in this case. Specifically, we denote the whole RE-SAS functionality as a function f:

$$\texttt{cred}^* := f(\mathcal{X}, \mathcal{B}, \texttt{req}), \tag{4.15}$$

where the $\mathcal{X}$ is the set of all received IU E-Zone reports and $\mathcal{B}$ is the set of all helper value maps. The formal definition of correctness is given as follows:

Definition 4.5.1. RE-SAS is correct if for any input $(\mathcal{X}, \mathcal{B}, \texttt{req})$ to RE-SAS functionality with location l in the exclusion zone, the recovered license $\texttt{cred}^* := f(\mathcal{X}, \mathcal{B}, \texttt{req})$ is invalid.

Theorem 3. The probability with which RE-SAS is NOT correct is negligible.

Proof. Let l be the location in the spectrum request $\texttt{req}$. Since it is located in E-Zone, we have $\mathbf{T}_{l,f}(i) = Q \neq 1$ for some $Q \leftarrow_{\$} \mathbb{Z}_p \backslash \{1_{\mathbb{Z}_p}\}$ for some i. Note that

$$[\![\mathbf{D}_{l,f}]\!]_{II} = \bigotimes_{\mathbf{X}(i) \in \mathcal{X}} [\![Y^{\sum_{\mathbf{X}(i) \in \mathcal{X}} \mathbf{T}_{l,f}(i)}]\!]_{II}, \tag{4.16}$$

Hence, I is essentially homormophic multiplication of $[\![Y^Q]\!]_{II} \neq [\![1_{G_T}]\!]_{II}$ with several items which are either $[\![1_{\mathbb{G}_T}]\!]_{II}$ or some other random elements. Hence, the probability with which $[\![\mathbf{D}_{l,f}]\!]_{II} = [\![1_{\mathbb{G}_T}]\!]$, 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. $\texttt{k}^* = \texttt{k}$) and the signature of $\texttt{cred}^*$ can be verified as valid is negligible. This demonstrate that the chance that RE-SAS is incorrect is negligible. □

4.5.2 IU individual privacy

To formally define IU individual privacy property, we setup 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 E-Zone information of an arbitrary IU at one arbitrary overall E-Zone location at its own will. To setup the experiment, the challenger C generates randomized (yet unknown) IUs' E-Zone data and send inputs to $\mathcal{A}$ assuming C is IU group and adv 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 formal definition of individual privacy experiment is shown in Figure. 4.2. Note that the procedure of generating E-Zone map report described in section V.C.2 is denoted as algorithm $([\![\mathbf{X}]\!]_{II}, [\![\mathbf{B}]\!]_{II}) = \mathsf{GenRep}(\mathbf{T})$.

```
$\mathbf{Exp}_{\mathcal{A}}^{\mathrm{ind-Priv}}(\lambda)$
$(\mathtt{msk}, \mathtt{ipk}, \mathtt{params}) \leftarrow \mathsf{Setup}(2^{\lambda})$.
for $k = 1, \cdots, M$
    $\mathbf{T}(k) \leftarrow_{\$} \{0,1\}^{L}$ .
    $([\![\mathbf{X}(k)]\!]_{II}, [\![\mathbf{B}(k)]\!]_{II}) = \mathsf{GenRep}(\mathbf{T}(k))$.
endfor
$(k^*, l) \leftarrow \mathcal{A}^{\mathsf{Reg}(\cdot)}(\mathtt{ipk}, \mathtt{params}, \{[\![\mathbf{X}(k)]\!]_{II}, [\![\mathbf{B}(k)]\!]_{II}\}_{k=1}^{M})$.
return 1 if $\mathbf{T}_{l,f}(k^*) = 1$; otherwise return 0.

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

Figure 4.2: Definition of privacy experiment

The formal definition of IU privacy is shown as follows:

Definition 4.5.2. RE-SAS preserves individual privacy for IUs 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

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

Theorem 4. RE-SAS preserves individual privacy for IUs under DLIN assumption.

Proof. To prove RE-SAS 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 RE-SAS 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.

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 $\mathbf{T}_{l,f}(i_0) = z - (a+b)$, select $A^* \leftarrow_{\$} \mathbb{Z}_p$ and set $\mathbf{A}_{l,f}(i_0) = A^* + b$. In this way, $\mathbf{T}_{l,f}(i_0) + \mathbf{A}_{l,f}(i_0) = A^* + c - a$ and $\mathbf{X}_{l,f}(i_0) = Y^{A^*+c-a} = Y^{A^*} e(f_1, h^c) e(f_1^a, h)^{-1}$. Select $\mathbf{B}_{l,f}(i_0) \leftarrow_{\$} \mathbb{G}_1$.
- For i_1: select $T', A' \leftarrow_{\$} \mathbb{Z}_p$, set $\mathbf{T}_{l,f}(i_1) = T' + b$, and set $\mathbf{A}_{l,f}(i_1) = A' - b$. In this way, $\mathbf{T}_{l,f}(i_1) + \mathbf{A}_{l,f}(i_1) = A' + T'$ and $\mathbf{X}_{l,f}(i_1) = Y^{A'+T'}$. Set $\mathbf{B}_{l,f}(i_1) \leftarrow \frac{Y^{A^*+A'} \mathbf{C}_{l,f}(i_0) \mathbf{C}_{l,f}(i_1)}{\mathbf{B}_{l,f}(i_0)}$.

The other parts of E-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 $\mathbf{B}_{l,f}(i_0)$ and $\mathbf{B}_{l,f}(i_1)$ are still randomly distributed, and the product of all helper values (i.e. $\mathbf{B}_{l,f}(i)$) is the same as what is generated faithfully. Therefore, as long as $\mathcal{A}$ breaks individual privacy and outputs (i_0, l) as E-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 RE-SAS preserves individual privacy. $\square$

4.5.3 IU semantic security

IU semantic security requires that when SAS is not colluding with other entities in the scheme, if it already gets encrypted, blinded E-Zone reports, it cannot determine any partial information on any report. To formally define semantic security, we setup an security experiment where an adversary who tries to distinguish two groups of encrypted blinded E-Zone report, which is shown in Figure 4.3.

The formal definition of IU privacy is shown as follows:

Definition 4.5.3. RE-SAS is semantically secure for IUs if for all $\lambda \in \mathbb{N}$, the advantage $\mathsf{Adv}_{\mathcal{A}}^{\text{sem-Sec}}(\lambda)$ is negligible in λ for all Probabilistic Polynomial-Time (PPT) Adversaries $\mathcal{A}$,

```
$\mathbf{Exp}_{\mathcal{A}}^{\text{sem-Sec}}(\lambda)$
$(\texttt{msk}, \texttt{ipk}, \texttt{params}) \leftarrow \mathsf{Setup}(2^{\lambda})$.
$(\mathbf{T}^{(0)}, \mathbf{T}^{(1)}) \leftarrow \mathcal{A}(\texttt{ipk}, \texttt{params})$, where
    $\mathbf{T}^{(0)} := \left\{\mathbf{R}^{(0)}(k)\right\}_{k=1}^{N}$;
    $\mathbf{T}^{(1)} := \left\{\mathbf{R}^{(1)}(k)\right\}_{k=1}^{N}$;
$b \leftarrow_{\$} \{0, 1\}$.
for $k = 1, \cdots, N$
    $([\![\mathbf{X}^{(b)}(k)]\!]_{II}, [\![\mathbf{B}^{(b)}(k)]\!]_{II}) = \mathsf{GenRep}(\mathbf{T}^{(b)}(k))$.
endfor
$b' \leftarrow \mathcal{A}\left(\texttt{ipk}, \texttt{params}, \left\{[\![\mathbf{X}^{(b)}(k)]\!]_{II}, [\![\mathbf{B}^{(b)}(k)]\!]_{II}\right\}_{k=1}^{N}\right)$.
return 1 if $b = b'$; otherwise return 0.
```

Figure 4.3: Definition of semantic privacy experiment

where

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

Theorem 5. If AFGH scheme is semantically secure, RE-SAS is semantically secure for IUs.

Proof. To prove RE-SAS is semantically secure for IUs, we assume that there exists an adversary $\mathcal{A}$ which can break IU semantic 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.

We recall the definition A.2 in [24], i.e. definition of semantic security, which is shown in

Figure. 4.4. AFGH scheme is semantically secure if:

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

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 4.4: Definition of semantic security of AFGH scheme

Before setting up the simulated RE-SAS semantic security experiment, we assume that $\mathbf{T}^{(0)}$ and $\mathbf{T}^{(1)}$ 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 RE-SAS 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 $\mathbf{T}^{(0)}$ and $\mathbf{T}^{(1)}$ 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 $\mathbf{R}^{(0)}(k)_{l,f} = \mathbf{R}^{(1)}(k)_{l,f}$, 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 $\mathbf{R}^{(0)}(k)_{l,f} = 0$, $\mathcal{S}$ sets $[\![\mathbf{X}^*(k)_{l,f}]\!]_{II} \leftarrow \left(\otimes_{i=1}^{r} C\right) \otimes [\![Y^{\mathbf{A}_{l,f}}]\!]_{II}$.

  If $\mathbf{R}^{(0)}(k)_{l,f} = 1$, $\mathcal{S}$ 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 $\mathbf{R}^{(0)}(k)_{l,f} = \mathbf{R}^{(1)}(k)_{l,f}$, 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 $\mathbf{R}^{(0)}(k)_{l,f} \neq \mathbf{R}^{(1)}(k)_{l,f}$, then $\mathcal{S}$ perfectly simulates the data reports in a manner where b in RE-SAS 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 $\mathbf{R}^{(0)}(k)_{l,f} = 0$; $[\![\mathbf{X}^*(k)_{l,f}]\!]_{II} = ((Z^{sB_1}m_b/m_1)^r, (g^s)^r)) \otimes [\![Y^{\mathbf{A}_{l,f}}]\!]_{II} = (Z^{(sr)B_1}m_1^{-r}, g^{sr}) \otimes [\![Y^{\mathbf{A}_{l,f}}]\!]_{II}$ when $\mathbf{R}^{(0)}(k)_{l,h,f,\gamma} = 1$. Thus in this case $\mathcal{S}$ simulates $[\![\mathbf{X}^*(k)]\!]_{II}$ perfectly in the case of setting b as 0 in the RE-SAS 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 $\mathbf{R}^{(1)}(k)_{l,f} = 0$; 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 $\mathbf{R}^{(1)}(k)_{l,f} = 1$. Thus $\mathcal{S}$ also simulates $\mathbf{X}^*(k)$ perfectly in the case of setting b as 1 in the RE-SAS 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 RE-SAS semantic security experiment and the advantage for breaking AFGH semantic security is the same as the advantage for breaking RE-SAS semantic security.

$\square$

Claim 1. Under EDBDH assumption, RE-SAS is semantically secure for IUs.

Proof. According to theorem 3.1 in [24], AFGH is semantically secure under EDBDH assumption, so RE-SAS is semantically secure for IUs under EDBDH assuption. $\square$

4.5.4 Soundness

Recall that soundness feature prevents SAS from diverting the protocol to intentionally reject SU's request regardless of E-Zone information. Here we claim that RE-SAS preserves the soundness feature since that if any request where SU is not located in E-Zone gets responded with invalid license, this execution will be disputed. This is true followed by the specification of protocol.

4.5.5 Privacy of SUs

Though the major focus of this chapter is IU privacy, SU privacy is also identified as important in literature [67][68], and the solution in [67] preserves the privacy for SUs. Yet different from IU privacy, SU privacy can actually be preserved though directly applying some common techniques, for example, differential privacy. SU can directly applying geo-indistinguishablity [69] to obtain an fuzzy location and put the fuzzy one in its spectrum request. SAS can draw a range of locations and process several spectrum requests corresponding to all locations in the range. Finally, multiply allocation results are responded to the SU. This can raise accuracy loss of the system, and we will evaluate it in section 4.7.

4.6 Accomodating multiple SUs

In this section discuss techniques which addresses the resource allocation issue in RE-SAS. Most literature discussing DSA architecture and resource allocation are assuming that SAS has the knowledge of available channels [70][71][72], where the privacy of IUs are neglected. In an IU privacy preserving DSA system, it is not likely to model the resource allocation in the same manner, since SAS has no knowledge, or inaccurate knowledge on available channels. For RE-SAS, we are proposing distributed channel access as well as centralized SUs accommodation as examples of resource allocation scheme.

4.6.1 Distributed resource allocation

In RE-SAS an SU finally recovers a spectrum license indicating it can transmit as a specific location, channel, and time interval. Meanwhile, other SUs located in the same grid may also get their licenses granting access to the same location, channel, and overlapped time interval.

In this case a collision avoidance mechanism can solve the resource allocation problem. If all the signals are authenticated with spectrum license, then the vastly used CSMA/CA protocol works in this scenario.

We can apply PHY-layer authentication techniques [73] to append authentication signals to packets for CSMA/CA protocol. In the scenario where multiple tiers of SUs exist (such as 2-tiers SUs in 3.5GHz), priority schemes can be applied [74], which has already been widely studied.

4.6.2 Centralized SU accommodation

In some literature [71][72] SAS is specified to be responsible for channel allocation, yet this is not likely to be feasible if IUs' privacy is preserved through cryptographic schemes, since the design goals include not letting SAS get knowledge on available channels. We propose an SU accomodation scheme instead, where the an SU can recover an extra occupancy based license if and only if its location, frequency, and time interval are not occupied by other SUs. The hierarchy of SUs group can be implemented by defining the eligibility of registering spectrum occupation.

In the accommodation scheme SAS sets up an encrypted SU occupancy map $[\![\mathbf{O}]\!]_{II}$, which is a $L \times F$ matrix. $\mathbf{O}$ is originally set as $[\![1_{\mathbb{G}_T}]\!]_{II}$ for all entries, indicating no channel are occupied.

At spectrum computation phase, SAS executes the following extra procedures to generate occupancy based license:

- Picks a random element β in $\mathbb{G}_T$, and hashes it to a random bit string $k^{(o)}$.
- Encrypts the valid occupancy based license using AES assuming $k^{(o)}$ as the secret key.

This is denoted as $[\![\mathtt{cred}^{(o)}]\!]_{AES,\mathtt{k}^{(o)}}$.

- Encrypts β to level 2 ciphertext in AFGH cryptosystem and mixes it with E-Zone information and occupancy information. That is, set

$$\hat{K}^{(o)} \leftarrow \mathsf{E}_{II}(\beta, \mathtt{ipk}) \otimes [\![\mathbf{D}_{l,f}]\!]_{II} \otimes [\![\mathbf{O}_{l,f}]\!]_{II}.$$

- Finally, sends encrypted license $[\![\mathtt{cred}^{(o)}]\!]_{AES,\mathtt{k}^{(o)}}$ and $\hat{K}^{(o)}$ to the SU.

This is similar to the procedures of generating encrypted spectrum license, and the recovery procedure is the same.

To register an occupation of spectrum, an eligible SU will set an occupation report $[\![\mathbf{R}]\!]_{II}$, which is a $L \times F$ matrix. It selects $Q \leftarrow_{\$} \mathbb{G}_T$ and sets $[\![\mathbf{R}_{l,f}]\!]_{II} \leftarrow [\![Q]\!]_{II}$ if it registers on location l and channel f. It sets the rest entries as $[\![1_{\mathbb{G}_T}]\!]_{II}$. Upon receiving the report, SAS conducts element-wise homomorphic multiplication between $[\![\mathbf{O}]\!]_{II}$ and $[\![\mathbf{R}]\!]_{II}$ to update its occupancy map.

When an SU cannot recover a valid occupancy based license, it will either contact SAS again trying to acquire a valid one through changing channel, or trying to opportunistically access the channel claimed by another SU.

4.7 Preliminary Evaluation Results

We construct the AFGH cryptosystem based on the pairing-based cryptography (PBC) library available at [75]. To evaluate RE-SAS, we set the service area to be a 405 km^2 area in Washington D.C. We employ L-R model provided by SPLAT! to calculate the attenuation map in this area, where real terrain data obtained from USGS and SRTM3 is used. The IU interference sensitivity level is set as -80 dBm, which is the sensitivity for a general military radar. We split this urban area of Washington D.C. into 36 × 45 grids with a side length of 500m. We assume there are 10 IUs and 10 frequency channels. All the experiments are conducted on a laptop with Intel i7-4770 CPU @ 3.4GHz.

The computation overhead is shown in 4.1. As can be seen, it takes only about 2.59ms to process one spectrum request, which is more than 2 orders of magnitude faster than IP-SAS.

Table 4.1: Computational overhead

System routine	Entity	Computation overhead
Maintaining database	IU	54.36s
	SAS	2.183s
SU Registration	SU	1.574ms
Spectrum Allocation	SAS	0.5081ms
	SU	2.085ms
Negative response confirmation	enforcer	0.004ms

4.8 Summary

In this chapter, we focus on preserving IU privacy under untrusted SAS scenario with provable guarantee; meanwhile the quality of SAS service is not expected to be degraded. Firstly, we explore the concept of individual privacy under untrusted SAS environment. Intuitively it implies that even an adversary obtains the knowledge of overall E-Zone data, it is not capable of obtaining the E-Zone data for any individual IU. We formulate the notion of individual privacy as well as its formal definition using standard security experiments. Moreover, we propose a framework of E-Zone based DSA that preserves individual security. This is achieved through special blinding technique based on AFGH cryptosystem and a tailored ciphertext packaging technique. Preliminary evaluation results show that ER-SAS is able to handle one spectrum request in 2.59ms, which is more than 2 orders of magnitude faster than IP-SAS.

Chapter 5

GuardCR: Malware Detection for CR

In this chapter, we present the design of GuardCR for malware detection in CR [76].

5.1 Challenges and Contributions

The aim of this chapter is to enhance the security of CR device itself by designing a scalable and accurate malware detection system named GuardCR. Due to the necessity of proactive defense for CR security, GuardCR needs to be built and evaluated even before CR malware appears. This means that GuardCR needs to be able to effectively detect zero-day attacks. GuardCR achieves accurate detection by taking advantages of the strong correlations among CR's execution actions. Specifically, a normal-operating CR exhibits strong correlations among its execution actions. A CR that is infected by malware, however, is unlikely to exhibit the similar correlations. Leveraging the host-based anomaly detection techniques [77], GuardCR builds normal behavior model by extracting the correlations among benign CR application's execution actions, and detects malware by measuring the behavior deviations from the normal model. To the best of our knowledge, this is the first work that enhances the security of CR networks at device level through dynamic behavior monitoring.

Although the host-based anomaly detection has been widely studied in intrusion detection systems (IDS) for securing operating systems [78, 79, 80], it encounters the following unique challenges when first applied to CR systems.

(1) Anomaly detection for CR needs to involve data flow analysis, which is generally considered expensive due to the occurrence of continuous numeric values. Traditional data flow analysis attempting to enumerate all the samples in continuous numeric space would lead to state explosion [81]. Additionally, data flow analysis provides limited context information of the execution states of operating systems, and thus is often not considered in the previous IDSs [78, 79, 82]. Monitoring data flow of a CR application, however, is important and cannot be ignored. In CR, data from both external environments (e.g., IU's signals) and internal resources (e.g., waveform capabilities) is collected and then sent to the cognitive engine, which generates proper radio parameters accordingly. These configuration parameters are finally delivered to the software radio platform to alter the radio's behavior. In the process, one small data discrepancy may change a CR's behavior drastically.

(2) Anomaly detection for CR needs to capture suppression attacks, which are not considered in existing IDSs. We define a suppression attack in CR as an attack that suppresses a CR node's desirable behaviors, making some functionalities of the CR nodes become futile. For example, in normal situation, a secondary CR user needs to periodically check if any IU appears in its current channel, and if so, the secondary CR user would switch its channel to avoid interfering with the IU. A malware may suppress such checking action so that the SU will never switch channel even when an IU appears in its current channel. Traditional IDSs cannot detect suppression attacks effectively because they focus on detecting undesirable behaviors that deviate from normal behavior set, while the suppression attacks causes harm not by introducing undesirable behaviors, but rather by suppressing desirable behaviors.

Suppression attacks are possible when a vulnerable CR node is hijacked and its control flow is altered.

(3) Traditional IDSs are evaluated based on the malware captured in the real environment. Since CR has not reached a widely-deployed state, there is a lack of real malware cases in the field that can be used to evaluate GuardCR. How to make meaningful evaluation of GuardCR is the third challenge.

To solve the challenges, we make the following contributions in this chapter:

• We realize efficient data flow analysis by reducing the very large data space into a small number of clusters. The rationale behind is that while the continuous numeric values may be infinite, similar values lead to the same operation context, which creates the same execution behavior.

• We address suppression attacks by designing new computational methods for security verification against them. Specifically, we regard the absence of desirable behaviors as a special anomalous signal, and we measure the signal strength and compare it with some predefined threshold to determine if suppression attack occurs.

• We propose an approach to automatically generate artificial malware cases with varied malicious behaviors used for evaluation. Specifically, our approach is based on mutation testing, which injects various mutations into a program to create mutants. The mutants exhibiting malicious behaviors are selected for testing the capability of GuardCR.

• We increase the detection accuracy and the detection speed of GuardCR by enabling GuardCR to automatically identify the security-critical components of a CR application, and *GuardCR* only monitors these components while ignoring the security-irrelevant components.

The rest of this chapter is organized as follows. We provide the related work in Section

5.2. We state the problem in Section 5.3. We describe the design of GuardCR in detail in Section 5.4. We present the approach for artificial malware generation in Section 5.5 and the evaluation results are given in Section 5.6. Finally, we summarize this chapter in Section 5.7.

5.2 Related work

The analysis of function call sequences has been widely used in anomaly detection [77]. In [78], Hofmeyr et al. modeled system call sequences using $N-$grams and scored anomaly based on the mismatches of $N-$grams. In [79], Warrender et al. proposed an HMM-based anomaly detection approach, which scored anomaly based on the likelihood of testing sequences generated by the learnt HMM and the average state transition probability of testing sequences. In addition, they compared the performance of the HMM-based approach and the $N-$gram-based approach, and found out that the HMM-based approach produce higher detection accuracy on average at the cost of higher computation overhead. [83] and [84] used specialized HMM for program anomaly detection on Linux server and utility programs. Both techniques involved static program analysis when initializing the HMM models, with [84] being context insensitive and an improved model in [83] with 1-level calling context sensitive. In [85], Shu et al. presented an outlier detection technique with low false positives. In [80], Hao et al. analyzed traffic causality and scalable triggering relation to detect stealthy malware activities. However, none of these approaches were capable of detecting suppression attacks that are critical for CR security.

In [81], Mutz et al. identified anomalous occurrences in system calls by characterizing string arguments based on string length, character distribution, string structure and token information. An anomaly score was calculated for each individual system call based on its string arguments. However, the important correlations among system calls were not considered. In [86], Maggi et al. extended [81] by building a behavioral Markov model that incorporates both arguments and correlations of system calls. However, both [81] and [86] only considered strings in arguments and the analytical method proposed for strings were not readily applicable to continuous numbers in CR applications. In this chapter, we propose an efficient data flow analysis method to address the continuous numeric values in arguments based on clustering and classification algorithms.

In [87], Wagner and Dean discovered that ignoring security-irrelevant system calls in execution traces could reduce the computation overhead and even increase the detection accuracy. However, how to efficiently identify the security-irrelevant system calls was not addressed. In this chapter, we propose a practical approach to automatically identify the security-irrelevant function calls and the security-critical function calls.

5.3 Problem Statement

5.3.1 Architecture of CR

Figure 5.1 shows a typical architecture of CR. The cognitive engine analyses radio operation environment, application requirements and FCC regulations to determine proper radio parameters. The radio configuration parameters are then sent to the radio functional modules, which is defined in the CR software on host computer and the Radio Configuration File (R-CFG) on FPGA. The radio functional modules controls the majority of radio components on hardware, such as mixers, filters, amplifiers, modulators/demodulators, and digital down/up converter etc. The intelligence of cognitive engine and the flexibility of the software-oriented design of radio functional modules enable a CR to adapt itself dynamically to use different

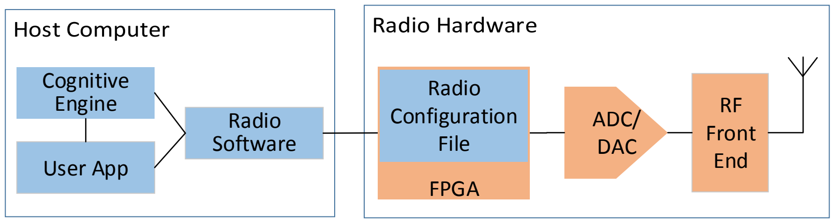


Figure 5.1: CR Architecture

spectrum bands with potentially different wireless communication technologies. As a result, the spectrum utilization can be significantly improved.

5.3.2 Attack Model

In CR, the radio functional modules can be reconfigured to generate a wide range of authorized or unauthorized waveforms. Adversaries can exploit the inherent vulnerabilities of CR software, non-CR software that shares the same host computer, and the underlying operating system to gain remote control of a CR device. For example, in [88], it has been shown that an external malware on a CR's host computer can easily modify the frequency value stored in a processor register at runtime, which causes the CR to transmit on a wrong frequency band. Note that since the CR's software files remain unchanged during the attack, that static software integrity inspection cannot detect such a malware. In the following, we give two attack scenarios as examples and discuss the possible attack impacts.

Scenario 1: Attacker Mallet configures a CR node with improper radio parameters by the similar method as [88]. For example, Mallet can modify the center frequency of an IU to migrate it from the current channel with good quality to a poor channel. Mallet can also impact the IU indirectly by increasing the transmit power of a secondary CR user that shares the same channel with the IU to cause more interference. In both cases, Mallet can worsen channel condition of the IU, causing poor data rate and high latency in communication. The damage is particularly severe when Mallet targets on time-sensitive wireless infrastructures, such as health care and emergency organizations.

Scenario 2: Attacker Trudy suppresses some critical functionalities of a CR node by exploiting the inherent software vulnerabilities, such as buffer overflow [89], to skip the execution of the corresponding code segments. For example, assume Trudy wants to suppress a function f, which is responsible for sensing the existence of IUs. To achieve this goal, Trudy can overwrite the return address of the previous function before f is executed in a stack frame. Once that function returns, execution will resume at the return address as specified by Trudy, such as the return address of f itself, or simply the entry of an infinite loop. In this way, Trudy can manipulate the program to suppress f.

5.3.3 Security Goal

The goal of GuardCR is to detect run-time malicious modifications to CR execution. The design of GuardCR leverages the host-based anomaly detection techniques. In order to realize the full potential of GuardCR in malware detection, the following three assumptions need to be satisfied.

Assumption 1: To cause harm, a compromised CR application needs to conduct different behaviors from those normally seen, and these behaviors should be readily monitored.

For example, consider a CR that should only attempt to switch its operation channel under two triggering events: an IU becomes active in its current operation channel or a user application's traffic demand exceeds the current channel's capacity. Each triggering event is marked by very distinctive traces of events. If the appearance of IU is detected through spectrum sensing, then it must be led by a series of actions. Specifically, the CR must first be switched to sensing mode, and then detect strong enough signal with coherent signatures of IU for such triggering event to happen. If a switching of channel is needed due to lack of bandwidth, then there must present either a series of events that mark the degradation of channel capacity, or a series of events from the application and operating system that mark the increase of traffic demand from the application. In all of the above described cases, there are clear causal and temporal relationships between events. A malware-infected CR that attempts to jam a target channel, obviously, will not have similar event traces since it is likely that its switching to the target channel is not triggered by any valid external event and thus will not have the same event traces.

Assumption 2: The training data is nearly ”complete” with regard to all possible ”normal” behaviors of a CR application.

This is a key assumption for learning-based anomaly detection schemes [82]. If the assumption is not satisfied, the learned detection model can not confidently classify an unmatched data as ”anomalous” since it is possible that the unmatched data is just an unseen ”normal” data. We satisfy this assumption in the design of GuardCR by feeding varied user inputs, application demands and operation environment to a CR application to capture as many the application's normal behaviors as possible (see Section 5.4.1).

Assumption 3: GuardCR itself can not be compromised.

This is an implicit assumption for nearly all IDSs, since if an IDS is compromised, it is no longer safe to make guarantee of the IDS's capability of intrusion detection. For better detection performance, GuardCR chooses to reside on the host computer to obtain a more comprehensive view of the internal activities of a CR node, which also makes GuardCR more vulnerable to attacks. We want to retain the excellent visibility into the state of the monitored CR node and also provide strong isolation for GuardCR from the host computer to lend significant resistance to attacks. To achieve the two seemingly contradictory goals, we put GuardCR in a more secure environment on the host computer by adopting the virtual machine monitoring (VMM) approach proposed in [90]. This approach allows us to pull GuardCR ”outside” of the host computer into a completely different protection domain, which provides high-confidence protection to GuardCR from malicious attacks. Yet, it also provides the ability to directly inspect the internal state of the CR node.

5.4 System Design

The overview of GuardCR is shown in Figure 5.2. GuardCR is composed of several components, including *Application Driver*, *Operation Tracer*, *Argument Processor*, *Normal Profile Database*, *Anomaly Detector*, and *Anomaly Feature Database.* *Argument Processor* and *Anomaly Detector* are the most important security components in our design. *Argument Processor* is able to conduct data flow analysis efficiently by adopting the clustering techniques. *Anomaly Detector* employs the host-based anomaly detection method to detect anomalous behaviors. Specifically, we choose two popular models, including $N-$gram model [78] and Hidden Markov Model (HMM) [79], for behavior modeling. In order to capture the suppression attacks, we introduce new computational methods for security verification to $N-$gram model and HMM, respectively. In the following, we present a brief sketch of how GuardCR works.

GuardCR’s workflow consists of two phases, the offline training phase and the online detection phase. In the offline training phase, GuardCR learns a model of the normal behavior of a CR application. And then in the online detection phase, GuardCR detects malicious behaivors of the runtime CR application by measuring the behavior deviation from the learnt model. Specifically, in the offline training phase, the clean-state version of a CR application is loaded into *Application Driver*, which executes the application in varied experimental

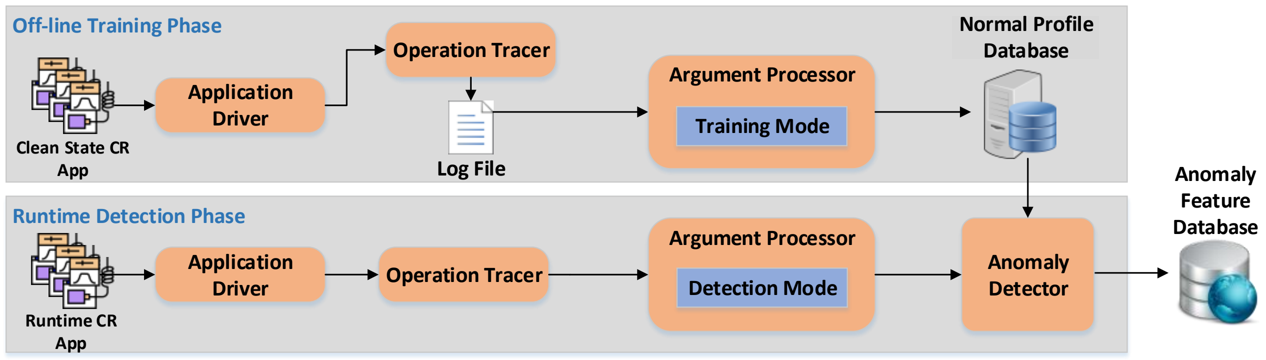


Figure 5.2: Overview of GuardCR. The most important security components are *Argument Processor* and *Anomaly Detector*. *Argument Processor* is responsible for conducting data flow analysis based on the clustering techniques. *Anomaly Detector* is responsible for detecting anomalous behaviors by employing the host-based anomaly detection techniques.

settings to expose as many normal behaviors as possible. During the execution, *Operation Tracer* collects the necessary operation logging data containing the CR application's normal behavior profiles, and stores it into a log file. This log file is digitally signed to resist tampering attempts. *Argument Processor* extracts the correlations of data flow by analyzing the logging data. The analytic results are sent to *Normal Profile Database*, which builds $N-$gram model or HMM to represent the normal behaviors of the CR application.

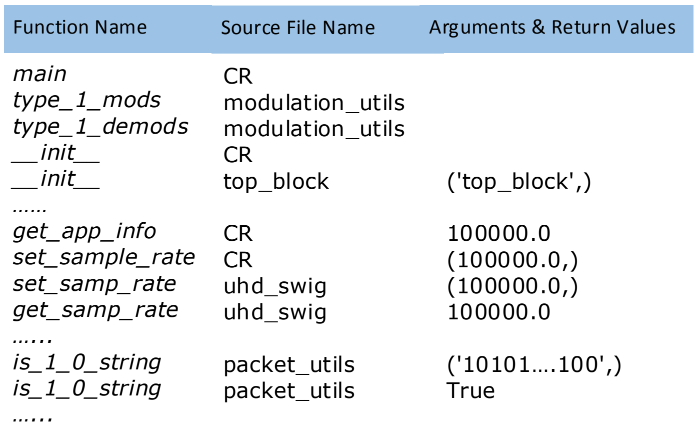
In the online detection phase, the runtime CR application is also executed in *Application Driver* and the runtime operation data is also recorded. *Argument Processor* analyzes the data flow of the operation data and sends the results to *Anomaly Detector*, which measures the deviation of the runtime CR application's behaviors from the normal behavior model stored in *Normal Profile Database*. If the deviation is larger than a predefined threshold, GuardCR raises an alarm, indicating that the CR application is infected. The anomalies captured are sent to a remote *Anomaly Feature Database*, which collects the anomalies from different CR nodes, and extracts the features of the anomalies for future signature-based malware detection. In the following, we will explain the details of each component.

5.4.1 Application Driver

In the offline training phase, *Application Driver* needs to traverse as many program branches as possible while executing a CR application, so that a more comprehensive description of the application's behaviors can be obtained. Therefore, the *Application Driver* executes the CR application for a large number of times. At each time, the application is stimulated by varied user inputs, application demands, and operation environment.

5.4.2 Operation Tracer

In the current version of GuardCR, *Operation Tracer* focuses on monitoring the function calls because they can precisely describe CR's internal operations. Specifically, a CR application needs to call functions to access the system resources and reconfigure the radio settings, and these function call traces can provide excellent view into how the CR application operates. *Operation Tracer* mainly focuses on two levels of function calls, including application level and radio-related software level. Typically, a CR application calls the signal processing functions provided by a software radio platform (e.g., GNU Radio [91], Matlab [92], LabVIEW [93]) to control the reconfigurable radio hardware platform (e.g., USRP [94]). In this work, we consider CR applications developed on the most popular CR development combination, i.e., GNU Radio and USRP. As CR applications on GNU Radio are mainly written in Python programming language, *Operation Tracer* leverages Python's build-in tracing modules to monitor function calls. For each function being called, we record the function name, the source file name, the argument values, and the return values. Specifically, the function name, the source file name, and the return values are intercepted by Python's `trace` module [95]. The arguments are intercepted by Python's `inspect` module [96]. We use the function name and the source file name to uniquely identify a function since we found that different



Function Name	Source File Name	Arguments & Return Values
main	CR	
type_1_mods	modulation_utils	
type_1_demods	modulation_utils	
__init__	CR	
__init__	top_block	('top_block',)
……		
get_app_info	CR	100000.0
set_sample_rate	CR	(100000.0,)
set_samp_rate	uhd_swig	(100000.0,)
get_samp_rate	uhd_swig	100000.0
……		
is_1_0_string	packet_utils	('10101….100',)
is_1_0_string	packet_utils	True
……		

Figure 5.3: An example of function call trace

source files might contain functions with the same function name, and using the function name solely as identifier may cause ambiguity. We record the arguments and the return values since they are essential for data flow analysis.

Figure 5.3 shows an example of intercepted function call trace. The source code of the CR application is stored in the file "CR", and the rest source files belong to GNU Radio. Note that there are two functions named *__init__*, but they belong to different source files. This means that they are actually different functions. We record all the types of argument values and return values, including strings, numbers, boolean etc. Note that the function *is_1_0_string* has both input arguments and return values. For this type of function with both input arguments and return values, we disintegrate it into two elements, one with only the input arguments and one with only the return values. The purpose of this process is to expose the correlations between the input arguments and the return values of each function. Without loss of generality, we use argument as the general term for both input argument and return value in the rest of this chapter.

5.4.3 Argument Processor

The complexity of data flow analysis is known to be very high due to the occurrence of continuous numeric values. To solve this problem, *Argument Processor* reduces the very large data space into a small number of clusters.

Overall Design

Argument Processor divides the continuous argument values into clusters to reduce the state space in data flow analysis. Finding the right clusters is a non-trivial problem. In CR, some functions do not have one single normal behavior, but a variety of behaviors naturally occurring in different operation contexts. Specifically, these functions may take different argument values as input under different operation contexts, then act differently according to these inputs, and finally output different return values. *Argument Processor* needs to use a small number of states to describe all the possible behaviors for scalability purpose.

To achieve this goal, *Argument Processor* leverages the fact that while the number of arguments may be infinite, similar arguments lead to the same operation context, which creates the same function behavior. Based on this observation, in the training mode, *Argument Processor* clusters the elements of a function's innovations based on these elements' argument values. The elements belonging to the same cluster have similar argument values, and hence are likely in the same operation context and exhibit same function behavior. Each cluster is given a unique name, and thus the sequences of function invocation elements are transformed to the sequences of cluster labels. Figure 5.4 shows an example of the clustering process in *Argument Processor*. In the online detection mode, *Argument Processor* classifies each function invocation element captured in the runtime trace into the most similar cluster according to its argument values. Figure 5.5 shows an example of the classification process

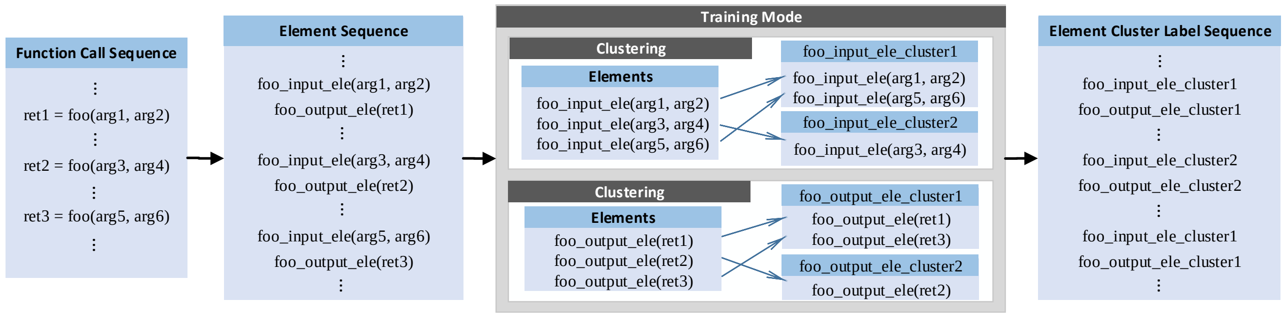


Figure 5.4: An example of the clustering process. *Argument Processor* clusters the three elements of foo_input_ele based on these elements' argument values. Since the argument vectors (arg1, arg2) and (arg5, arg6) are similar, so foo_input_ele(arg1, arg2) and foo_input_ele(arg5, arg6) are put into one cluster, and they share the same name, which is the cluster name foo_input_ele_cluster1, in the cluster label sequence. foo_input_ele(arg3, arg4) has different argument values, so it is put into another cluster foo_input_ele_cluster2. The same process is performed for foo_output_ele elements.

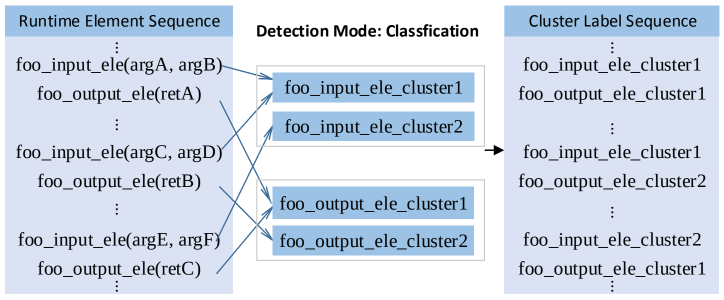


Figure 5.5: An example of the classification process. *Argument Processor* classifies the three elements of foo_input_ele based on these elements' argument values. Since the argument vectors (argA, argB) and (argC, argD) are similar to the argument vectors in cluster foo_input_ele_cluster1, so foo_input_ele(argA, argB) and foo_input_ele(argC, argD) are classified into foo_input_ele_cluster1, and they use their cluster name in the cluster label sequence. The argument vectors (argE, argF) is similar to the argument vectors in cluster foo_input_ele_cluster2, so foo_input_ele(argE, argF) is classified into cluster foo_input_ele_cluster2. The same process is performed for foo_output_ele elements.

in *Argument Processor*.

Training Mode

Firstly, we determine what types of elements need to be clustered. We observe that an element's argument is either countable or uncountable depending on if its values can be enumerated from a small set of possible choices. Examples of countable argument in a CR application are indexes, tokens, and enumerations (e.g., modulation schemes), etc. Examples of uncountable argument in a CR application are channel signal strength acquired by sensors, choices of sampling rate, and level of noise floor, etc. The values of a countable argument are naturally clustered by regarding each value as an individual cluster. Uncountable arguments include uncountable strings and uncountable numbers. Uncountable strings are neglected in data flow analysis because they are usually text messages for display purpose, and do not affect execution flow. The elements with uncountable numbers in arguments need to be clustered carefully. In the following, we use the term "uncountable argument" to denote the argument whose values are uncountable numbers.

We adopt a single-linkage and bottom-up hierarchical agglomerative clustering algorithm [97] to cluster the elements with uncountable arguments. Specifically, the clustering algorithm initially treats each element as a singleton cluster. It then successively agglomerates pairs of clusters based on some rule until termination criterion is hit. Since an element may contain multiple uncountable arguments, we aggregate these arguments as an argument vector. Assume the total number of elements of a function is M_d, and we want to have N_s clusters in the end. Algorithm 1 shows the detailed procedures of the elements clustering algorithm.

In the elements clustering algorithm, the distance $d_{i,j}$ between element i and element j is

Algorithm 1: Elements clustering algorithm.

```
1 Assign each of the $M_d$ elements to an individual cluster.
2 Compute a $M_d \times M_d$ distance matrix $D$. $D(i,j) \leftarrow d_{i,j}$, where $d_{i,j}$ is the distance
  between element $i$ and element $j$ defined in Equation (5.1).
3 for $N = M_d$ to $N_s$ do
4     Merge the two clusters with the smallest distance $D_N$.
5     Update $D$ by recomputing the distance between the newly merged cluster and the
      remaining.
6     $N \leftarrow N - 1$.
7 end
```

defined by standardized euclidean distance of their argument vectors:

$$d_{i,j} = \sqrt{\sum_{m=1}^{M} \left(\frac{a_i^{(m)} - a_j^{(m)}}{\sigma_a^{(m)}} \right)^2}, \tag{5.1}$$

where $a^{(1)}, ... a^{(M)}$ are the entries in the argument vector, and $\sigma_a^{(m)}$ is the standard deviation of $a^{(m)}$ over all the possible values in the training data. The distance between two clusters is defined as the minimum distance between two elements chosen from each cluster.

We adopt "L method" [98] to determine the value of N_s in the termination criterion. Specifically, in step 4 of Algorithm 1, we record D_N, which is called merge distance in [98]. We plot an evaluation graph by taking the number of clusters N as x axis and the merge distance D_N as y axis. The "L method" attempts to find the knee point in the evaluation graph, which is the point of maximum curvature. The knee point represents a balance of clusters that are both highly homogeneous, and also dissimilar to each others. N_s is set as the knee point's x-axis value.

Argument Processor automatically determines whether an argument is countable or uncountable by picking out all the values of the argument from the training data. It counts the total number of the argument value as q_1 and the number of unique argument value as q_2. If q_2/q_1

is very small, the argument is said to be countable; If it is close to 1, then the argument is said to be uncountable. Considering that the training data is gathered from a large number of program executions, q_1 is generally large enough to make the quotient q_2/q_1 very small if the argument is countable.

Detection Mode

In the detection mode, *Argument Processor* applies the k-nearest neighbours (k-NN) algorithm [99] to classify each runtime element into an element cluster that is built in the training mode. Then, *Argument Processor* assigns the cluster's label to the runtime element. The high-level description of the k-NN classification algorithm is as follows. Based on Equation (5.1), *Argument Processor* computes the distances between the runtime element and all the training-time elements of the same function. Then, it picks out the k training-time elements that are closest to the runtime element, which are also known as the k-nearest neighbors of the runtime element. Finally, it classifies the runtime element into the cluster most common among the k-nearest neighbors.

Time Complexity

In the training mode, the time complexity of the agglomerative clustering approach and the "L method" is $O({M_d}^2)$ [97] and $O(M_d)$, respectively. So the total time complexity of the training mode is $O({M_d}^2)$. In the detection mode, the time complexity of the k-NN algorithm is $O(M_d)$.

5.4.4 Anomaly Detector

Anomaly Detector aims at recognizing anomalous sequences whose composition and variety substantially deviate from normal execution sequences. Two popular models, including $N-$gram model and HMM are explored for this purpose. In addition, we design new computational methods based on statistic features of the training data sets for security verification against the unique suppression attacks in CR. We illustrate how the new security rules can be integrated into the existing $N-$gram model and HMM.

$N-$gram model

Given a cluster label sequence, all the unique substrings of fixed length N in the sequence are called $N-$grams. They can be obtained by sliding an $N-$sized window along the sequence to record every substring. $N-$grams are capable of grasping the function elements' execution contexts, which are the correlations among the function elements within a certain span. The set of all the $N-$grams in a sequence is called $N-$gram set. In the offline training phase, the $N-$gram sets are extracted from all the training sequences and are stored in *Normal Profile Database*; While in the online detection phase, *Anomaly Detector* calculates the deviation of a runtime $N-$gram set from the training $N-$gram sets in *Normal Profile Database.* The $N-$gram model is appropriate for describing CR applications' behavior because of its discriminability. In CR applications, malicious behaviors often occur in local burst because adjacent components are closely related by data flow between them, and one malicious modification will largely affect the execution in the local span. $N-$gram model is capable of capturing the local deviations by high-resolution local inspection.

We use four metrics, including N_o, S_A, N_{oc}, and S_{Ac}, to measure the deviation of a runtime $N-$gram set from the $N-$gram sets stored in *Normal Profile Database.* N_o and S_A focus on capturing undesirable behaviors by detecting the $N-$grams that are not in *Normal Profile Database* but are in the runtime $N-$gram set. They have been used in the existing work to detect anomalous behaviors in operating systems [78]. However, they are not able to detect suppression attacks in CR since suppression attacks cause harm not by introducing new undesirable behaviors, but rather by suppressing a CR application's desirable behaviors. To address suppression attacks, we design the other two metrics, N_{oc} and S_{Ac}, to capture the lack of desirable behaviors by detecting the $N-$grams that are in *Normal Profile Database* but are missing in the runtime $N-$gram set. The detailed description of each metric is as follows.

• N_o is defined as the number of outliers, which are the $N-$grams appear in the runtime $N-$gram set but not in *Normal Profile Database.* Ideally, we expect N_o to be 0 for normal execution traces, and to be some positive number that can be easily distinguished from 0 when anomalous behavior occurs.

• N_{oc} is defined as the number of $N-$grams that are in *Normal Profile Database* but are missing in the runtime $N-$gram set. Ideally, we expect N_{oc} to be some small positive value for new normal execution traces, and to be some distinguishably large positive value when suppression attack occurs.

• S_A is a local measure that is not dependent on the length of cluster label sequences. It represents the strength of the anomalous signal, i.e. how much it deviates from a known normal pattern. Specifically, we use the Hamming distance between two $N-$grams to measure the difference. The Hamming distance $d_H(i,j)$ between two $N-$gram i and j is defined as the number of positions where the two $N-$grams differ, and $0 \leq d_H(i,j) \leq N$. Based on that, S_A is defined as follows:

$$S_A = \max_{i \in I} \min_{j \in D} d_H(i,j) \tag{5.2}$$

where D is the set of all the $N-$grams in *Normal Profile Database* and I is the runtime $N-$gram set.

• S_{Ac} is also a local measure, which represents the strength of the anomalous signal due to suppression attacks. Recall that we run a clean state CR application for many times in *Application Driver* in the training phase. Assume the application is run for n times and n training $N-$gram sets, $J_{s1}, J_{s2}, ..., J_{sn}$, are obtained. For each training $N-$gram set, we record the number of $N-$grams that are in the training set but not in the runtime $N-$grams set I. The training $N-$gram set that has the smallest number, denoted as J_{sk}, is regarded as the most similar training set to I. S_{Ac} is defined as follows:

$$S_{Ac} = \max_{j \in J_{sk}} \min_{i \in I} d_H(j, i). \tag{5.3}$$

We propose the following criterion to determine whether a runtime sequence is malicious or not based on the four metrics as follows:

$$R = (N_o \geq t_o \wedge S_A \geq t_A) \vee (N_{oc} \geq t_{oc} \wedge S_{Ac} \geq t_{Ac}) \tag{5.4}$$

The runtime sequence is marked as malicious if $R = True$. t_{oc}, t_{Ac}, t_o, t_A are the corresponding thresholds with respect to each metric. These thresholds should be set appropriately to reduce false positives in malware detection. Specifically, to find a proper t_{oc}, we need to capture the largest possible suppressing signal of a normal execution sequence. So for $i \in \{1, 2, ..., n\}$, we count the number of $N-$grams that are in $D - J_{si}$ but not in J_{si}, and t_{oc} is set as the largest among the numbers. If N_{oc} is larger than t_{oc}, it is very likely that some functionality of the CR application is maliciously suppressed. In addition, we set t_{Ac} as N to prevent oversensitivity in suppression attack detection to reduce false positives. We set t_o as 1 and t_A as $\lceil \frac{N}{3} \rceil$ based on both the instructions in [78] and the empirical results in CR applications.

Hidden Markov Model

Assuming that the execution of a CR application follows some state transitions, the normal CR behavior can be modeled by constructing a HMM in *Normal Profile Database*. The states of the HMM are not observable, but each state can produce observations following certain emission probability distribution. A state can transit to other states following certain transition probability distribution. We construct an HMM using the sequences of element cluster labels produced by *Argument Processor* as follows.

First, we set the number of states as the number of unique element cluster labels in the sequences. Transitions are allowed between any pair of states. We adopt the well-known Baum-Welch algorithm [100] to train the HMM. The high-level idea of the Baum-Welch algorithm is to find the maximum likelihood estimate of the transition and emission probability distributions of an HMM given a set of observations. Specifically, the transition and emission probability distributions are randomly initialized at the beginning of the algorithm. Then during training, the two distributions are iteratively adjusted to increase the likelihood that the HMM generates the training sequences based on the expectation-maximization (EM) algorithm [100].

With the well-trained HMM, *Anomaly Detector* detects anomalous behaviors as follows. Firstly, it computes the likelihood $\boldsymbol{P}$ of a runtime sequence given the HMM model using the forward algorithm [100]. Then, it compares this likelihood with a predefined threshold $\boldsymbol{P_T}$. If $\boldsymbol{P} < \boldsymbol{P_T}$, the sequence is marked as malicious. Otherwise, the sequence is marked as benign.

For suppression attacks, *Anomaly Detector* firstly adopts Viterbi algorithm [100] to calculate the optimal state transition sequence of a runtime sequence. Then, it counts the number $\boldsymbol{Q}$

of unique state transitions in the state transition sequence and compares Q with a predefined threshold Q_T. If $Q < Q_T$, the runtime sequence is marked as suppressed. Otherwise, the runtime sequence is marked as not suppressed. The detection method is based on the observation that the state transitions of a suppressed runtime sequence are usually suppressed as well. This is because the state transitions corresponding to the suppressed functionalities are also suppressed.

We propose the following criterion to determine whether a runtime sequence is malicious or not as follows:

$$R = (P < P_T) \vee (Q < Q_T) \tag{5.5}$$

If $R = True$, the sequence is marked as malicious.

Comparison of Storage Space and Time Complexity

• HMM needs smaller storage space than $N-$gram model. Specifically, assume the training data contains S unique element cluster labels. HMM needs to store the $2S^2$ values for the transition matrix and the emission matrix. $N-$gram model needs to store $\frac{S!}{(S-N)!}$ $N-$grams in the worst case.

• In the training phase, the time complexity of $N-$gram model is lower than HMM; While in the detection phase, the time complexity of $N-$gram model is higher than HMM. Specifically, in the training phase, for HMM, the Baum-Welch algorithm's complexity for training a new training sequence is $O(TS^2)$ [100], where T is the length of the sequence; While for $N-$gram, training a new training sequence only needs $O(T)$. In the detection phase, the complexity of HMM is $O(2TS^2)$ since the complexity of the Viterbi algorithm and the forward algorithm are both $O(TS^2)$ [100]. While for $N-$gram model, it takes $O(N)$ comparisons to determine whether an $N-$gram is an "outlier" when the normal $N-$grams are stored in a forest. Computing the distance between an $N-$gram and the normal $N-$grams requires $O(DN)$ comparisons, where D is the number of normal $N-$grams. If $D \approx \frac{S!}{(S-N)!} \approx S^N$, the complexity would be $O(NS^N)$. In total, the complexity of $N-$gram model in the detection phase is $O(NS^N + N) = O(NS^N)$.

5.4.5 Anomaly Feature Database

All the anomalous behaviors captured during the execution of CR applications are recorded and sent to a remote *Anomaly Feature Database*, which collects the anomaly reports from all the CR nodes and mines the malware patterns from the vast data for further signature-based detection. Moreover, the anomaly data exposes the vulnerable components of the software, which aids the software maintainers in enhancing the security of the CR software itself.

5.4.6 System Refinement

We improve $GuardCR$'s detection performance by refining the system. Specifically, we design a mechanism to automatically select a set of security-critical function calls. Only these function calls are monitored in $GuardCR$ while the rest security-irrelevant function calls are neglected. The refinement process is able to reduce GuardCR's false positives in malware detection by reducing the execution ambiguity caused by the security-irrelevant function calls. Additionally, by concentrating on the most crucial function calls relevant to security, the refinement process can increase both the detection accuracy and the detection speed.

The refinement method is based on the observation that many intercepted function calls provide little context information of the execution states of CR applications. For example, some function calls are only involved in printing messages on screen, and some functions are only related to thread scheduling in the Python programming language. What's worse, the latter group of the function calls often leads to false positives since these function calls often occur at unexpected locations in the execution traces. On the other hand, since the operation fidelity of CR applications is determined by the integrity of its RF parameters, function calls that can alter the values of the RF parameters directly or indirectly are critical for CR security. Based on the above observations, in the refined version of GuardCR, only these security-critical function calls that can impact the RF parameters are logged by *Operation Tracer* and processed by the following components of GuardCR. How to automatically identify these security-critical function calls will be addressed in Section 5.5.

5.5 Artificial Malware Generation

Most of current IDS solutions are passive and reactive, focusing on defending against known attacks [101]. The IDSs to protect a system are typically built after the system has been widely deployed and successful attacks to the system have been witnessed. Thus, the effectiveness of the proposed IDSs can be evaluated using real malwares captured in the field. However, the reactive defense strategy is not appropriate for CR since compromised CR poses serious threats to not only CR systems but also other critical wireless infrastructures. GuardCR, thus, is proactively developed before the appearance of real malwares. However, the proactive defense strategy poses a challenge on how to evaluate GuardCR since the traditional evaluation methods using the real malwares are no longer feasible due to a lack of malwares in the new field. To fill the gap between the proactive defense solution and its evaluation, we propose a method to automatically generate artificial malwares for meaningful evaluation.

Observation

The artificial malware cases are generated by adopting mutation testing technique [102], which is initially developed for software testing. Specifically, it injects various mutations into a program's source code or byte code to create mutants, and a software test's quality is evaluated by the percentage of mutants that are caught by the test.

We use the mutation testing technique to generate artificial malwares based on the observation that malwares can be viewed as a special type of mutants in a certain sense. A malware usually needs to inject malicious modifications into an application to alter the application's variable values and instruction execution sequences. Hence, if a malware detection scheme is effective on detecting mutants at runtime, it should also be effective on detecting real malwares.

Method

As shown in Table 5.1, we apply six mutation operators to change the original program in different ways, thus generating a wide range of malware-alike behaviors. For example,"VAR" modifies the values of variables, such as center frequency, bandwidth, and transmit power. "JMP" modifies the branch conditions, which may cause that a secondary CR node never leaves its current channel even when the condition that an IU appears in its channel is met. "ARI" and "CMP" alter the arithmetic operations and the comparison operations respectively, which can lead to data calculation error. "JPT" can redirect a program's execution to a random position to stimulate unexpected behaviors. "IIL" can insert an infinite loop at an arbitrary position of a program, which can impede the program's execution and suppress its normal functionalities.

Not every mutant generated by the above mutation operators can be used for evaluation since some of the mutants do not exhibit malicious behaviors. This can happen since the mutation testing is not originally designed for malware generation. We find that some mutants do not change the original program's semantics. For example, a mutation that changes "$for(i = 0; i < 5; i++)$" to "$for(i = 0; i \neq 5; i++)$" in a program will not change the program's behavior. Some mutants only change the unimportant variables and branches of a program, such as displaying information on screen or writing logs to files. The mutants that do not change the radio behaviors are regarded as benign mutants. To accurately evaluate GuardCR's performance, these benign mutants are excluded from the testing set, and only the mutants that exhibit malicious behaviors are used for evaluation.

Table 5.1: Mutation Operators

Mutation operator	Description	# of all mutants	# of malicious mutants
JMP	Alter Branch Condition	92	55
JPT	Randomize Jump Target	491	393
VAR	Change Variable Value	597	417
ARI	Alter Arithmetic Operator	799	598
CMP	Alter Comparison Operator	308	144
IIL	Insert Infinite Loop	468	339
Total		2755	1946

Identifying security-critical functions calls

Besides serving for evaluation, the mutants can also be used to automatically identify the security-critical function calls for system refinement in Section 5.4.6 as follows.

We construct a database $\mathcal{D}$ to store all the security-critical function calls. Firstly, we initialize $\mathcal{D}$ by bringing in all the USRP Hardware Driver (UHD) APIs. These APIs take the RF parameters as input, and output the corresponding waveforms configurations. The UHD is the final boundary from software side to hardware side, and all the GNU Radio applications need to invocate the UHD APIs to reconfigure a radio. If these APIs are fed with anomalous inputs, the radio will be reconfigured to generate anomalous waveforms. Secondly, we send all the mutants (including the benign ones) into GuardCR and obtain the mismatched $N-$grams in *Anomaly Feature Database*. If an anomaly feature of some mutant contains an UHD API, all the other function calls in the anomaly feature are inserted to $\mathscr{D}$. This step is to ensure that the function calls that may indirectly cause anomalies to UHD APIs are included in $\mathscr{D}$. After $\mathscr{D}$ is completed, only the function calls in $\mathscr{D}$ need to be monitored by GuardCR.

5.6 Prototype & Evaluation

We set up a CR testbed on GNU Radio 3.7.1 and three N210 USRPs, as shown in Figure 5.6. Each USRP is individually connected with a host computer. Two USRPs are used as secondary transceivers and the rest one is used as incumbent transmitter. The secondary transceivers execute a CR application obtained from CORNET [103], which is an open-access cognitive radio testbed developed by Virginia Tech. In this application, each secondary transceiver senses both the external radio environment and the internal application's requirements to adjust its radio parameters accordingly. Specifically, for the external environment, each secondary transceiver will switch its current channel to a white channel if it perceives the existence of the incumbent transmitter on its current channel; For the internal application's requirements, each secondary transceiver will change its transmission bandwidth according to the application's required data rate. The CR application has 703 lines source code, which are converted to 2114 lines byte code. As shown in Table 5.1, after applying the mutation operators, 2755 mutants are generated, and 1946 of them are selected as testing malware cases.

We implement a prototype of GuardCR on a host computer connected with a secondary transceiver. GuardCR is equipped with two anomaly detection models, $N-$gram model

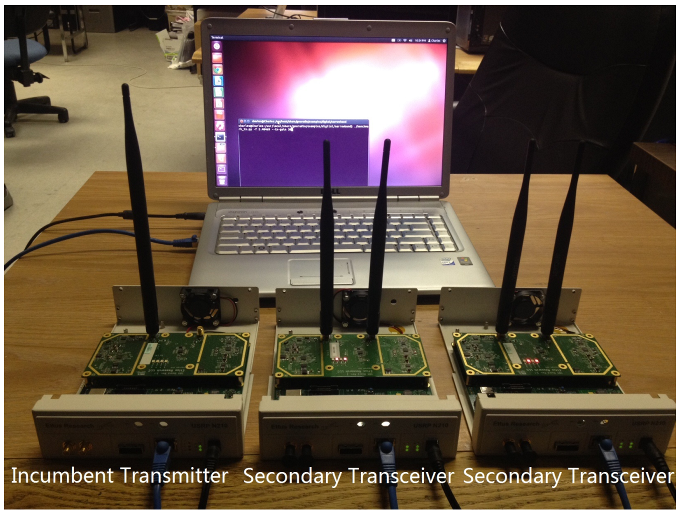


Figure 5.6: CR testbed setup. Two USRPs are used as secondary transceivers and the rest one is used as incumbent transmitter.

and HMM, for comparison purpose. For each model, we build the basic implementation that analyzes all the intercepted function calls as a ground case, and we compare the performance of the refined implementation that only analyzes security-critical function calls. We focus on two key metrics: detection accuracy and detection speed. We measure the detection accuracy based on the true positive (TP) rate and the false positive (FP) rate. The true positive happens when a malicious trace is detected, while the false positive happens when a benign trace is mistakenly classified as malicious. In general, better detection accuracy means higher TP rate and lower FP rate.

To generate normal traces, the CR application is run for 2000 times in *Application Driver* and each run lasts for 22 seconds. We feed different initialization inputs (e.g., frequency band, required data rate) at the beginning of each run. We also generate different incumbent transmitter behavior patterns and different data rate requirement to stimulate the monitored secondary transceiver at runtime. We randomly select 60% of the generated normal traces for training, and the rest 40% for evaluating the FP rate. To obtain the malicious testing traces for evaluating the TP rate, every malicious mutant is also executed for 22 seconds in

Application Driver.

5.6.1 Detection Accuracy

As previously mentioned, we evaluate four implementations of GuardCR, including $N-$gram, $N-$gram refined, HMM, and HMM refined. For $N-$gram and $N-$gram refined, we also study the impact of the sliding window size N to the detection accuracy. For HMM and HMM refined, we study the impact of the threshold P_T, which is the threshold to differentiate the likelihood of a malicious trace from that of normal traces. Since likelihoods tend to be very small when traces are very long, we present the likelihoods in logarithmic form.

Figure 5.7 shows the TP rate and FP rate of different GuardCR implementations with respect to different parameter settings. As we can see, for the two $N-$gram-based implementations, the TP rate and the FP rate both increase with N. For a fixed N, $N-$gram refined draws higher TP rate and lower FP rate compared with $N-$gram. When $N = 5$, the $N-$gram refined reaches the peak TP rate 93.94% while only incurring 0.25% FP rate. For the two HMM-based implementations, the TP rate and the FP rate both increase with P_T. For a fixed P_T, HMM refined draws no lower TP rate and lower FP rate compared with HMM. The peak TP rate of the two HMM-based implementations are both 94.86%, which is better than the two $N-$gram-based implementations. The evaluation results demonstrate that the refinement process can increase the TP rate and reduce the FP rate simultaneously.

We also evaluate GuardCR in detecting suppression attacks. We use the mutants generated by the IIL mutation operator in Table 5.1 to mimic suppression attacks. GuardCR achieves the TP rate of 96.76% employing $N-$gram refined and 96.82% employing HMM refined in detecting the suppression attacks.

In order to further illustrate the detection capability of GuardCR against real-world mal-

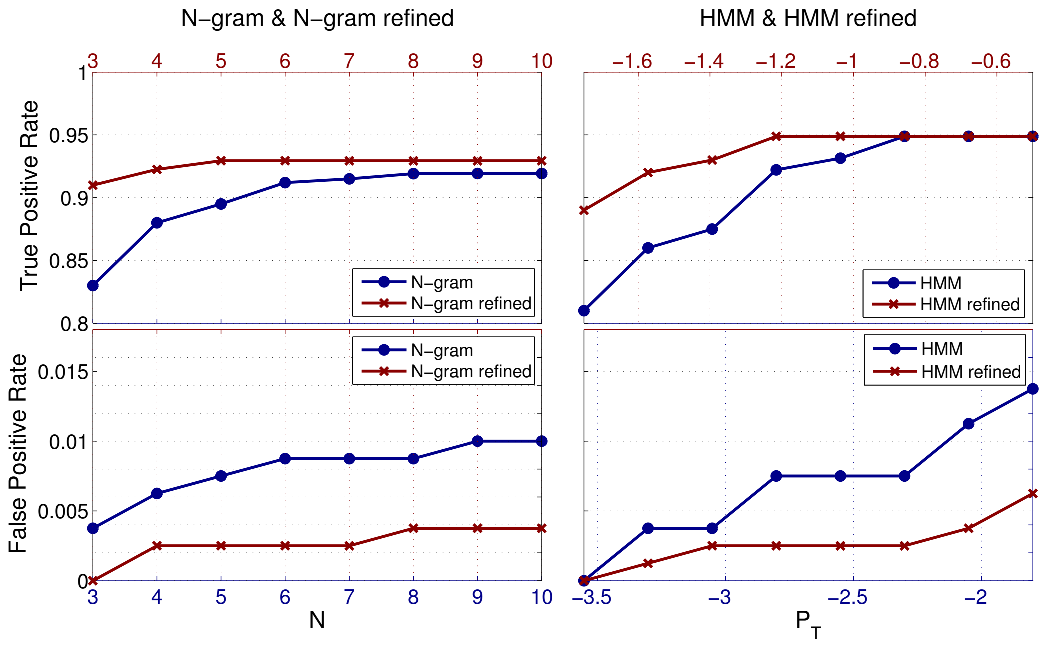


Figure 5.7: TP/FP rate versus different configuration parameters. N is the sliding window size in $N-$gram, P_T is the likelihood threshold in HMM and is in $\log_{10}$ scale

wares, we generate high-order mutants by inserting more than one mutations into the original program. The rationale is that in real world, the compromised software is usually modified at more than one positions to bring in more significant changes to the execution flow for specific malicious goals. We can simulate such process by applying more mutation operators to the program. Table 5.2 shows the TP rate of GuardCR against mutants of different orders. The order of a mutant is defined as the number of mutation operators applied to generate the mutant. We generate 542 2-order mutants and 542 3-order mutants as testing cases. As can be seen from Table 5.2, GuardCR achieves even higher detection accuracy for the high-order mutants.

Table 5.2: TP rate of GuardCR against high-order mutants

Mutation order	1	2	3
$N-$gram refined	93.94%	98.52%	99.63%
HMM refined	94.86%	99.08%	99.81%

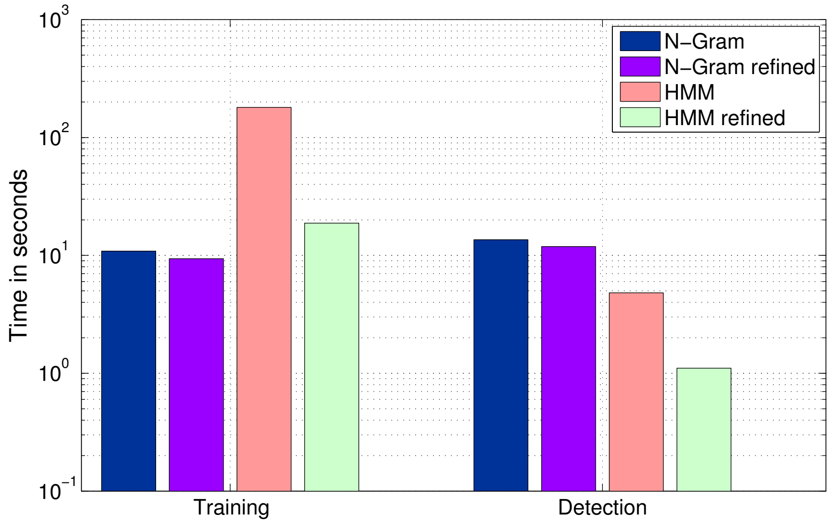


Figure 5.8: Detection speed of different GuardCR implementations.

5.6.2 Detection Speed

We evaluate the detection speed of different GuardCR implementations. Figure 5.8 shows the average time to process one trace in the training phase and the detection phase, respectively. As can be seen, HMM-based implementations take longer time in training while shorter time in detection compared with $N-$gram-based implementations, which confirms the theoretical complexity analysis in Section 5.4.4. The refinement approach increases the detection speed of both $N-$gram model and HMM. The improvement is larger for HMM: The training time is reduced from 182.28s to 18.38s, while the detection time is reduced from 4.81s to 1.10s.

Table 5.3 shows overhead reduction brought by the refinement process. Of all the 97 unique functions captured in the training phase, 57 functions are selected as security-critical functions. After the refinement process, the average trace length is reduced from 1683 to 983, and the number of states in HMM is reduced from 246 to 109.

Table 5.3: Overhead Reduction Brought by Refinement process

	Basic	Refined	Percentage change
# of unique functions	97	57	41.42%
Average trace length	1683	983	41.59%
# of states in HMM	246	109	55.70%

5.7 Summary

This chapter presents GuardCR, the first scalable and accurate zero-day malware detection system for CR. GuardCR monitors both control flow and data flow of CR execution to profile CR software's normal behaviors, and detects malwares by measuring the behavior deviations from the normal profiles. GuardCR is able to detect suppression attacks, which are significant threats to CR networks. To reduce false positives and accelerate detection speed, we propose a refinement approach to prune away the security-irrelevant function calls from being analyzed. To evaluate GuardCR in a meaningful way, we propose an artificial malware generation scheme based on mutation testing. Evaluation results demonstrate the scalability and the accuracy of GuardCR.

Chapter 6

Conclusions and Future Work

In this chapter, we conclude the dissertation and point out the future research directions.

6.1 Conclusions

In this dissertation, we have identified crucial privacy and security issues in the existing DSA systems that may severely hinder the progress of DSA's deployment in the real world. To address these issues, we have designed effective and practical defending systems based on the recent advances in cryptography, proxy re-encryption, machine learning, anomaly detection, and software testing. We refine our defending systems in terms of practicality and accuracy through careful customization and integration of proper domain knowledge of DSA. We build solid prototypes on prevalent platforms such as GNU Radio and USRP, and conduct experiments based on real world data. Evaluation results demonstrate that our proposed defending systems are practical and scalable.

In chapter 2, we identify the serious user privacy issues in the existing protection-zone-based

SAS designs. To address the issue, we design P^2-SAS, which realizes the complex spectrum allocation process of protection-zone-based DSA through secure MPC. In P^2-SAS, sensitive operation data of IUs and SUs is encrypted by Paillier cryptosystem before submitting SAS Server. SAS Server performs the protection-zone access enforcement computations on the encrypted data based on the homomorphic properties of Paillier encryption, and thus is oblivious to the plaintext of the operation data. Finally SUs can decrypt the results and unveil the spectrum access response. We formally prove the correctness and privacy-preserving property of P^2-SAS. To make P^2-SAS practical in the real-world scenario, we explore various means to refine the system. We investigate the tradeoff between accuracy and efficiency of P^2-SAS computation, and we formulate it into an optimization problem to search for the optimal P^2-SAS parameter setting. We propose practical acceleration methods to improve P^2-SAS's efficiency, including factoring, precomputing, ciphertext packing, parallelization. We demonstrate the scalability and practicality of P^2-SAS using experiments based on real-world data. Experiment results show that P^2-SAS can respond an SU's spectrum request in 6.96 seconds with communication overhead of less than 4 MB.

In chapter 3, we develop a privacy-preserving design named IP-SAS for the exclusion-zone-based SAS. IP-SAS is also based on secure MPC, nevertheless, in a different way since the core part of computation, which is the access enforcement method, is different. We extend the basic design that only considers semi-honest adversaries to consider malicious adversaries. We leverage the unique properties of IP-SAS computation to develop acceleration mechanisms, which significantly reduce IP-SAS's computation and communication overhead. We conduct extensive experiments based on real-world data to evaluate IP-SAS, and the results show that IP-SAS can respond an SU's spectrum request in 1.25 seconds with communication overhead of 17.8 KB.

In chapter 4, we present RE-SAS for privacy-preserving SAS in exclusion-zone scenario. RE-

SAS is based on AFGH cryptosystem, which supports both proxy re-encryption and homomorphic encryption. Compared with IP-SAS, RE-SAS does not need an online somewhat-trusted third party Key Distributor and hence eliminate the single point of vulnerability issue. In addition, RE-SAS can handle a spectrum request in 2.59 ms on average, which is more than two orders of magnitude faster than IP-SAS.

In chapter 5, we identify the potentially serious threat of compromised CR devices to the ambient wireless infrastructures and believe that CR security should be carefully taken care of before CR become widespread. Even worse, the software-oriented design of CR makes it easier for adversaries to take control through exploiting the inherent and ever-present vulnerabilities of CR software. To address this severe zero-day security threat, we propose a defending system called GuardCR, which is a scalable and accurate malware detection system customized for CR software. GuardCR leverages the host-based anomaly detection technique driven by machine learning, which makes it autonomous in malicious behavior recognition. We refine the GuardCR system to improve its detection accuracy and speed by inspecting and leveraging the characteristics of CR software. To evaluate GuardCR before the existence of real malwares, we propose a general experimental method to automatically generate artificial malware cases with varied malicious behaviors based on mutation testing. We implement a prototype of GuardCR for CR applications on GNU Radio and USRP. Evaluation results show that GuardCR is able to detect malicious behaviors within 1.10 second at an accuracy of 94.9%.

6.2 Future Work

The current SAS designs only manage the spectrum sharing between IUs and SUs. The coexistence problem of SUs is not yet considered. Some recent DSA proposals by FCC and the research community [8] also explore the possibility of using SAS to manage the coexistence problem of SUs. We plan to incorporate this new feature into the future version of SAS. We also plan to apply AFGH cryptosytem to protection-zone-based SAS system.

For GuardCR, we will extend GuardCR from only monitoring CR software' operation events to monitoring hardware-level, OS-level, and network-level events. A multi-layer detection strategy will be more effective to capture malware.

Bibliography

[1] FCC, “Et docket no 03-222 notice of proposed rule making and order,” 2003.

[2] I. F. Akyildiz, W.-Y. Lee, M. C. Vuran, and S. Mohanty, “Next generation/dynamic spectrum access/cognitive radio wireless networks: a survey,” *Computer networks*, vol. 50, no. 13, pp. 2127–2159, 2006.

[3] Q. Zhao and B. M. Sadler, “A survey of dynamic spectrum access,” *IEEE signal processing magazine*, vol. 24, no. 3, pp. 79–89, 2007.

[4] P. Kolodzy and I. Avoidance, “Spectrum policy task force,” *Federal Commun. Comm., Washington, DC, Rep. ET Docket*, no. 02-135, 2002.

[5] G. Locke and L. Strickling, “Plan and timetable to make available 500 megahertz of spectrum for wireless broadband,” *US Department of Commerce, Washington, DC, USA*, 2010.

[6] ——, “An assessment of the near term viability of accommodating wireless broadband systems in the 1675-1710 mhz, 1755-1780 mhz, 3500-3650 mhz, 4200-4220 mhz, 4380-4400 mhz bands,” *U. S. Department of Commerce, Washington, DC, October*, vol. 1, 2010.

[7] PCAST, "Report to the president realizing the full potential of government-held spectrum to spur economic growth," 2012.

[8] FCC, "Amendment of the commission's rules with regard to commercial operations in the 3550-3650 MHz band," *Notice of Proposed Rulemaking and Order*, pp. 12–148, 2012.

[9] M. Altamimi, M. B. Weiss, and M. McHenry, "Enforcement and spectrum sharing: Case studies of federal-commercial sharing," *Available at SSRN 2310883*, 2013.

[10] FCC, "Amendment of the commission's rules with regard to commercial operations in the 1695-1710 MHz, 1755-1780 MHz, and 2155-2180 MHz bands," *Notice of Proposed Rulemaking and Order, FCC*, 2014.

[11] S. Haykin, "Cognitive radio: brain-empowered wireless communications," *IEEE journal on selected areas in communications*, vol. 23, no. 2, pp. 201–220, 2005.

[12] F. C. Commission, "Connecting America: The national broadband plan," 2010.

[13] "Mobile broadband services in the 2300 MHz - 2400 MHz frequency band under Licensed Shared Access regime," *ETSI TR 103 113 V1.1.1*, 2013.

[14] FCC, "Shared Commercial Operations in the 3550–3650 MHz Band," *Federal Register*, vol. 80, no. 120, June 2015.

[15] "Google's Spectrum Access System Allows Spectrum Sharing," http://www.androidheadlines.com/2015/05/googles-spectrum-access-system-allows-spectrum-sharing.html.

[16] M. Kurdziel, J. Beane, and J. Fitton, "An SCA security supplement compliant radio architecture," in *Military Communications Conference, MILCOM 2005. IEEE*, pp. 2244–2250.

[17] W. Scott, A. Houle, and A. Martin, "Information assurance issues for an SDR operating in a manet network," in *SDR Forum, November*, 2006.

[18] R. Falk, J. Esfahani, and M. Dillinger, "Reconfigurable radio terminals—Threats and security objectives," in *SDR Forum Input Document, SDRF-02-I-0056*, 2002.

[19] T. Ulversoy, "Software defined radio: Challenges and opportunities," *Communications Surveys & Tutorials, IEEE*, vol. 12, no. 4, pp. 531–550, 2010.

[20] R. Chen, J.-M. Park, Y. T. Hou, and J. H. Reed, "Toward secure distributed spectrum sensing in cognitive radio networks," *Communications Magazine, IEEE*, vol. 46, no. 4, pp. 50–55, 2008.

[21] Y. Dou, K. C. Zeng, H. Li, Y. Yang, B. Gao, C. Guan, and S. Li, "P2-SAS: Preserving Users' Privacy in Centralized Dynamic Spectrum Access Systems," *in Proceedings of the 17th ACM MobiHoc*, 2016.

[22] Y. Dou, H. Li, K. C. Zeng, J. Liu, Y. Yang, B. Gao, and S. Li, "Poster: Preserving Incumbent Users' Privacy in Server-Driven Dynamic Spectrum Access Systems," *in Proceedings of the 36th IEEE ICDCS*, 2016.

[23] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *EUROCRYPT'99*. Springer, 1999, pp. 223–238.

[24] G. Ateniese, K. Fu, M. Green, and S. Hohenberger, "Improved proxy re-encryption schemes with applications to secure distributed storage," *ACM Trans. Inf. Syst. Secur.*, vol. 9, no. 1, pp. 1–30, Feb. 2006. [Online]. Available: http://doi.acm.org/10.1145/1127345.1127346

[25] Y. Dou, K. Zeng, H. Li, Y. Yang, B. Gao, K. Ren, and S. Li, "P2-sas: Privacy-preserving centralized dynamic spectrum access system," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 1, pp. 173–187, 2017.

[26] Y. Dou, K. C. Zeng, and Y. Yang, "Poster: Privacy-preserving server-driven dynamic spectrum access system," in *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking (MobiCom)*. ACM, 2015, pp. 218–220.

[27] P. Bogetoft *et al.*, "Secure multiparty computation goes live," in *Financial Cryptography and Data Security*. Springer, 2009, pp. 325–343.

[28] Y. Lindell and B. Pinkas, "Secure multiparty computation for privacy-preserving data mining," *Journal of Privacy and Confidentiality*, vol. 1, no. 1, p. 5, 2009.

[29] M. Naehrig, K. Lauter, and V. Vaikuntanathan, "Can homomorphic encryption be practical?" in *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*. ACM, 2011, pp. 113–124.

[30] FCC, "Longley-rice methodology for evaluating TV coverage and interference," *Office of Engineering and Technology (OET) Bulletin*, no. 69, 2004.

[31] M. Schneider and T. Schneider, "Notes on non-interactive secure comparison in image feature extraction in the encrypted domain with privacy-preserving sift," in *Proceedings of the 2nd ACM workshop on Information hiding and multimedia security*. ACM, 2014, pp. 135–140.

[32] Z. Gao, H. Zhu, Y. Liu, M. Li, and Z. Cao, "Location privacy in database-driven cognitive radio networks: Attacks and countermeasures," in *INFOCOM, 2013 Proceedings IEEE*. IEEE, 2013, pp. 2751–2759.

[33] B. Bahrak, S. Bhattarai, A. Ullah, J.-M. Park, J. Reed, and D. Gurney, "Protecting the primary users' operational privacy in spectrum sharing," in *DYSPAN*. IEEE, 2014.

[34] K. Zeng, S. K. Ramesh, and Y. Yang, "Location spoofing attack and its countermeasures in database-driven cognitive radio networks," in *Communications and Network Security (CNS), 2014 IEEE Conference on*. IEEE, 2014, pp. 202–210.

[35] X. Jin, J. Sun, R. Zhang, and Y. Zhang, "Safedsa: Safeguard dynamic spectrum access against fake secondary users," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2015, pp. 304–315.

[36] A. C.-C. Yao, "How to generate and exchange secrets," in *Proceedings of FOCS*, 1986, pp. 162–167.

[37] O. Goldreich, "Secure multi-party computation," *Manuscript. Preliminary version*, 1998.

[38] C. Gentry, "A fully homomorphic encryption scheme," Ph.D. dissertation, Stanford University, 2009.

[39] Z. Brakerski, "Fully homomorphic encryption without modulus switching from classical gapsvp," in *Advances in Cryptology–CRYPTO 2012*. Springer, 2012, pp. 868–886.

[40] C. Bazelon, "The economic basis of spectrum value: Pairing aws-3 with the 1755 mhz band is more valuable than pairing it with frequencies from the 1690 mhz band," *The Brattle Group, Washington DC*, 2011.

[41] C. S. M. A. Committee *et al.*, "Final report of working group 1–1695-1710 mhz meteorological-satellite," *National Telecommunications and Information Agency, Washington DC*, vol. 22, 2013.

[42] O. Goldreich, *Foundations of cryptography: volume 2, basic applications.* Cambridge university press, 2004.

[43] FCC, “Small Entity Compliance Guide: Amendment of the commission’s rules with regard to commercial operations in the 3550-3650 MHz band,” *FCC Gaussian noise Docket 12-354*, 2015.

[44] E. Drocella, J. Richards, R. Sole, F. Najmy, A. Lundy, and P. McKenna, “3.5 ghz exclusion zone analyses and methodology,” *Tech. Rep.*, 2015.

[45] http://dds.cr.usgs.gov/pub/data/DEM/250/.

[46] http://dds.cr.usgs.gov/srtm/version2_1/SRTM3/.

[47] C. Dwork, “Differential privacy,” in *Automata, languages and programming.* Springer, 2006, pp. 1–12.

[48] O. Catrina *et al.*, “Improved primitives for secure multiparty integer computation,” in *Security and Cryptography for Networks*, 2010, pp. 182–199.

[49] I. Damgård *et al.*, “Unconditionally secure constant-rounds multi-party computation for equality, comparison, bits and exponentiation,” in *Theory of Cryptography.* Springer, 2006, pp. 285–304.

[50] E. Barker *et al.*, “Recommendation for Key Management – Part 1: General (Revision 3),” *NIST Special Publication*, vol. 800, no. 57, pp. 1–147, 2012.

[51] J. Katz and Y. Lindell, *Introduction to modern cryptography.* CRC Press, 2014.

[52] M. Minoux and H. Tuy, “Discrete monotonic global optimization,” Citeseer, Tech. Rep., 2002.

[53] T. Ge and S. Zdonik, "Answering aggregation queries in a secure system model," in *VLDB*, 2007, pp. 519–530.

[54] T. Granlund and the GMP development team, *GNU MP: The GNU Multiple Precision Arithmetic Library*, 6th ed., 2014, http://gmplib.org/.

[55] http://www.qsl.net/kd2bd/splat.html.

[56] Y. Dou, H. Li, K. C. Zeng, J. Liu, Y. Yang, B. Gao, and K. Ren, "Preserving incumbent users privacy in exclusion-zone-based spectrum access systems," in *Distributed Computing Systems (ICDCS), 2017 IEEE 37th International Conference on.* IEEE, 2017, pp. 2486–2493.

[57] Y. Dou, K. C. Zeng, Y. Yang, and K. Ren, "Poster: Preserving incumbent users' privacy in exclusion-zone-based spectrum access systems," in *Proceedings of the 22st Annual International Conference on Mobile Computing and Networking (MobiCom)*, 2016.

[58] S. S. Chow, J.-H. Lee, and L. Subramanian, "Two-Party Computation Model for Privacy-Preserving Queries over Distributed Databases." in *NDSS*, 2009.

[59] B. Wang, M. Li, S. S. Chow, and H. Li, "Computing encrypted cloud data efficiently under multiple keys," in *Communications and Network Security (CNS), 2013 IEEE Conference on.* IEEE, 2013, pp. 504–513.

[60] C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for data storage security in cloud computing," in *INFOCOM, 2010 Proceedings IEEE.* Ieee, 2010, pp. 1–9.

[61] C. Hazay and Y. Lindell, *Efficient secure two-party protocols: Techniques and constructions.* Springer Science & Business Media, 2010.

[62] A. Ullah, S. Bhattarai, J.-M. Park, J. Reed, D. Gurney, and B. Bahrak, "Multi-Tier Exclusion Zones for Dynamic Spectrum Sharing," in *Proceedings of IEEE International Conference in Communications (ICC)*, 2015.

[63] T. P. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," in *Advances in Cryptology—CRYPTO'91*. Springer, 1992, pp. 129–140.

[64] F. C. Commission *et al.*, "Report and order and second further notice of proposed rulemaking," *Amendment of the Commissions Rules with Regard to Commercial Operations in the*, pp. 3550–3650, 2015.

[65] S. D. Galbraith, K. G. Paterson, and N. P. Smart, "Pairings for cryptographers," *Discrete Applied Mathematics*, vol. 156, no. 16, pp. 3113 – 3121, 2008, applications of Algebra to Cryptography. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0166218X08000449

[66] L. Chen, Z. Cheng, and N. P. Smart, "Identity-based key agreement protocols from pairings," *International Journal of Information Security*, vol. 6, no. 4, pp. 213–241, 2007. [Online]. Available: http://dx.doi.org/10.1007/s10207-006-0011-9

[67] Y. Dou, K. C. Zeng, H. Li, Y. Yang, B. Gao, C. Guan, K. Ren, and S. 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*. ACM, 2016, pp. 321–330.

[68] L. Rainie, S. Kiesler, R. Kang, M. Madden, M. Duggan, S. Brown, and L. Dabbish, "Anonymity, privacy, and security online," *Pew Research Center*, vol. 5, 2013.

[69] M. E. Andrés, N. E. Bordenabe, K. Chatzikokolakis, and C. Palamidessi, "Geo-indistinguishability: Differential privacy for location-based systems," in *Proceedings*

of the 2013 ACM SIGSAC Conference on Computer &#38; Communications Security, ser. CCS '13. New York, NY, USA: ACM, 2013, pp. 901–914. [Online]. Available: http://doi.acm.org/10.1145/2508859.2516735

[70] S. S. C. . W. Security, "Cbrs operational security," *Wireless Innovation Forum*, no. WINNF-15-S-0017, 2016.

[71] M. M. Sohul, M. Yao, T. Yang, and J. H. Reed, "Spectrum access system for the citizen broadband radio service," *IEEE Communications Magazine*, vol. 53, no. 7, pp. 18–25, 2015.

[72] C. W. Kim, J. Ryoo, and M. M. Buddhikot, "Design and implementation of an end-to-end architecture for 3.5 ghz shared spectrum," in *Dynamic Spectrum Access Networks (DySPAN), 2015 IEEE International Symposium on.* IEEE, 2015, pp. 23–34.

[73] V. Kumar, J.-M. Park, and K. Bian, "Blind transmitter authentication for spectrum security and enforcement," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '14. New York, NY, USA: ACM, 2014, pp. 787–798. [Online]. Available: http://doi.acm.org/10.1145/2660267.2660318

[74] Y. Xiao, "Performance analysis of priority schemes for ieee 802.11 and ieee 802.11 e wireless lans," *IEEE transactions on wireless communications*, vol. 4, no. 4, pp. 1506–1515, 2005.

[75] A. De Caro and V. Iovino, "jpbc: Java pairing based cryptography," in *Proceedings of the 16th IEEE Symposium on Computers and Communications, ISCC 2011*, Kerkyra, Corfu, Greece, June 28 - July 1, 2011, pp. 850–855.

[76] Y. Dou, K. C. Zeng, Y. Yang, and D. D. Yao, "Madecr: correlation-based malware detection for cognitive radio," in *2015 IEEE Conference on Computer Communications (INFOCOM)*. IEEE, 2015, pp. 639–647.

[77] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Computing Surveys (CSUR)*, vol. 41, no. 3, p. 15, 2009.

[78] S. A. Hofmeyr, S. Forrest, and A. Somayaji, "Intrusion detection using sequences of system calls," *Journal of computer security*, vol. 6, no. 3, pp. 151–180, 1998.

[79] C. Warrender, S. Forrest, and B. Pearlmutter, "Detecting intrusions using system calls: Alternative data models," in *Proceedings of the 1999 IEEE Symposium on Security and Privacy, S&P 1999*. IEEE, pp. 133–145.

[80] H. Zhang, D. D. Yao, and N. Ramakrishnan, "Detection of stealthy malware activities with traffic causality and scalable triggering relation discovery," in *Proceedings of the 9th ACM symposium on Information, computer and communications security*. ACM, 2014, pp. 39–50.

[81] D. Mutz, F. Valeur, G. Vigna, and C. Kruegel, "Anomalous system call detection," *ACM Transactions on Information and System Security (TISSEC)*, vol. 9, no. 1, pp. 61–93, 2006.

[82] W. Lee and S. J. Stolfo, "Data mining approaches for intrusion detection," in *Usenix Security*, 1998.

[83] K. Xu, K. Tian, D. Yao, and B. Ryder, "A sharper sense of self: Probabilistic reasoning of program behaviors for anomaly detection with context sensitivity," in *Proceedings of the 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2016.

[84] K. Xu, D. D. Yao, B. G. Ryder, and K. Tian, "Probabilistic program modeling for high-precision anomaly classification," in *Proceedings of the 28th IEEE Computer Security Foundations Symposium (CSF).* IEEE, 2015, pp. 497–511.

[85] X. Shu, D. Yao, and N. Ramakrishnan, "Unearthing stealthy program attacks buried in extremely long execution paths," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security.* ACM, 2015, pp. 401–413.

[86] F. Maggi, M. Matteucci, and S. Zanero, "Detecting intrusions through system call sequence and argument analysis," *IEEE Transactions on Dependable and Secure Computing*, vol. 7, no. 4, pp. 381–395.

[87] D. Wagner and D. Dean, "Intrusion detection via static analysis," in *Proceedings of the 2001 IEEE Symposium on Security and Privacy, S&P 2001.* IEEE, pp. 156–168.

[88] J.-M. Park, T. Jatin, and T. Alexandru, "Malicious modification of software defined radios using hardware breakpoints," in *In 2009 Virginia Tech Symposium on Wireless Personal Communications*, 2009.

[89] C. Cowan, P. Wagle, C. Pu, S. Beattie, and J. Walpole, "Buffer overflows: Attacks and defenses for the vulnerability of the decade," in *DARPA Information Survivability Conference and Exposition, 2000. DISCEX'00. Proceedings*, vol. 2. IEEE, 2000, pp. 119–129.

[90] T. Garfinkel, M. Rosenblum *et al.*, "A Virtual Machine Introspection Based Architecture for Intrusion Detection." in *NDSS*, vol. 3, 2003, pp. 191–206.

[91] "Gnuradio project," http://gnuradio.org/redmine/projects/gnuradio/wiki.

[92] http://www.mathworks.com/discovery/sdr.html.

[93] http://www.ni.com/sdr/.

[94] "The universal software radio peripheral," http://www.ettus.com/.

[95] https://docs.python.org/2/library/trace.html.

[96] https://docs.python.org/2/library/inspect.html.

[97] R. Sibson, "SLINK: an optimally efficient algorithm for the single-link cluster method," *The Computer Journal*, vol. 16, no. 1, pp. 30–34, 1973.

[98] S. Salvador and P. Chan, "Determining the number of clusters/segments in hierarchical clustering/segmentation algorithms," in *16th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2004.* IEEE, pp. 576–584.

[99] E. Alpaydin, *Introduction to machine learning.* MIT press, 2004.

[100] L. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[101] Z. Xu, L. Chen, G. Gu, and C. Kruegel, "Peerpress: utilizing enemies' p2p strength against them," in *Proceedings of the 2012 ACM conference on Computer and communications security.* ACM, 2012, pp. 581–592.

[102] Y. Jia and M. Harman, "An analysis and survey of the development of mutation testing," *Software Engineering, IEEE Transactions on*, vol. 37, no. 5, pp. 649–678, 2011.

[103] T. R. Newman, A. He, J. Gaeddert, B. Hilburn, T. Bose, and J. H. Reed, "Virginia tech cognitive radio network testbed and open source cognitive radio framework," in *5th International Conference on Testbeds and Research Infrastructures for the Development of Networks & Communities and Workshops, TridentCom 2009.* IEEE, pp. 1–3.

Appendix A

Security Proof

A.1 Security Analysis on the blinding factor in formula (2.14)

We use blinding factors $\alpha(l, h_I, f_I)$, $\beta(l, h_I, f_I)$, and $\epsilon(l, h_I, f_I)$ in formula (2.14) to obfuscate the sensitive intermediate results $G_b(l, h_I, f_I)$ to Key Distributor. By carefully choosing these blinding factors, the security can be guaranteed by making the probability of distinguishing $G_b(l, h_I, f_I)$ given $X_b(l, h_I, f_I)$ at Key Distributor negligible compared with random guesses. Formally, it means that we need to carefully determine the distribution of the random integers α and β so that, given that one knows X and the fact that $X = \epsilon(\alpha G - \beta), \epsilon \overset{\$}{\leftarrow} \{1, -1\}$ ($\overset{\$}{\leftarrow}$ denotes that ϵ is drawn uniformly at random from $\{1, -1\}$) for some integer G, the value of G can have ρ different uniformly distributed choices and the probability of distinguishing the right G value is only $1/\rho$. When ρ is large enough, we can say the obfuscating technique is secure enough. Let us first consider the case where $\epsilon = 1$ and $G > 0$, that is to say: $X + \beta = \alpha \times G$ and $G > 0$. Since $\alpha > \beta \geq 0$, we also have $X > 0$ in this case. Under this case, we have the following definition and theorem.

Definition A.1.1. Given that one knows X, $X > 0$, we define the set $S_G(X)$

$$S_G(X) = \{G_1, G_2, ..., G_n\} \tag{A.1}$$

be the set of possible G values, where (1) $\exists \alpha_i > \beta_i \geq 0, s.t. X + \beta_i = \alpha_i \times G_i$. (2) $G_i \neq G_j, \forall i \neq j$. (3) $G_i > 0, \forall i$.

Theorem 6. If $X = a^2$ or $X = a^2 \pm a$, for some $a \geq 2, a \in \mathbb{Z}$, $|S_G(X)| + 1 = |S_G(X+1)|$. Otherwise, $|S_G(X)| = |S_G(X+1)|$.

Proof of Theorem 6. Assuming that $G_i \in S_G(X)$, by Definition A.1.1, there exists $\alpha_i > \beta_i \geq 0$ that makes $X + \beta_i = \alpha_i \times G_i$.

Case 1.1: $\beta_i \neq 0$

In this case, rearranging $X + \beta_i = \alpha_i \times G_i$, we get $(X+1) + (\beta_i - 1) = \alpha_i \times G_i, \alpha_i > \beta_i - 1 \geq 0$, which means that $G_i \in S_G(X+1)$. In essence, it means that in this case, an element $G_i \in S_G(X)$ is mapped to the same element $G_i \in S_G(X+1)$.

Case 1.2: $\beta_i = 0$ and $G_i - \alpha_i < -1$

In this case, $X = \alpha_i \times G_i$. So we also have $X = G_i \times \alpha_i$ and $G_i > 0$, which means $\alpha_i \in S_G(X)$. Rearrange $X = \alpha_i \times G_i$ as $(X+1) + (G_i - 1) = (\alpha_i + 1) \times G_i$. Since $\alpha_i + 1 > G_i - 1$, so $G_i \in S_G(X+1)$. Since $G_i > G_i - 1$, so $(\alpha_i + 1) \in S_G(X+1)$.

Next, we prove $\alpha_i + 1 \notin S_G(X)$. The proof is based on considering the contradiction. Assume there exists Δ satisfying $X + (G_i + \Delta + \Delta \times \alpha_i) = (G_i + \Delta)(\alpha_i + 1)$. To make $\alpha_i + 1 \in S_G(X)$, we should have $G_i + \Delta > G_i + \Delta + \Delta \times \alpha_i \geq 0$. Equivalently, $G_i + \Delta > G_i + \Delta + \Delta \times \alpha_i \Leftrightarrow \Delta \times \alpha_i < 0 \Leftrightarrow \Delta < 0$. $G_i + \Delta + \Delta \times \alpha_i \geq 0$ & $\Delta < 0 \Leftrightarrow G_i \geq -(1 + \alpha_i)\Delta \geq 1 + \alpha_i$, which contradicts with $G_i - \alpha_i < -1$. So $\alpha_i + 1 \notin S_G(X)$.

We also can prove that $\alpha_i \notin S_G(X+1)$. The proof is also based on contradiction.

In summary, in case 1.2, element $G_i \in S_G(X)$ and its companion element $\alpha_i \in S_G(X)$ uniquely maps to two entries $G_i \in S_G(X+1)$ and $(\alpha_i + 1) \in S_G(x+1)$.

Case 1.3: $\beta_i = 0$ and $G_i - \alpha_i > 1$

In this case, we have $X = \alpha_i \times G_i$ and $\alpha_i - G_i < -1$. This case is the same as case 1.2 except that G_i and α_i is exchanged. Thus, using the same proof of case 1.2, we can get $G_i \in S_G(X), \alpha_i \in S_G(X), \alpha_i \in S_G(X+1), G_i+1 \in S_G(x+1), G_i \notin S_G(X+1)$ and $G_i+1 \notin S_G(X)$.

In summary, in case 1.3, element $G_i \in S_G(X)$ and its companion element $\alpha_i \in S_G(X)$ uniquely maps to two entries $G_i + 1 \in S_G(X+1)$ and $\alpha_i \in S_G(x+1)$.

With the above case studies, we can draw the following conclusions. In case 1.1, one element in $S_G(X)$ uniquely maps to one element in $S_G(X+1)$. In case 1.2 and 1.3, two companion entries in $S_G(X)$ uniquely map to two entries in $S_G(X+1)$. Similarly, we can also prove that the map holds from the opposite direction, i.e. $S_G(X+1)$ to $S_G(X)$. Specifically, given $\forall G_j \in S_G(X+1)$, there exists $(X+1) + \beta_j = \alpha_j * G_j$. $\alpha_j > \beta_j \geq 0$. When $\alpha_j > \beta_j + 1$, we can prove one element in $S_G(X+1)$ uniquely maps to one element in $S_G(X)$. This condition corresponds to case 1.1. When $\alpha_j = \beta_j + 1$ & $G_j - \beta_j < 1$, or $\alpha_j = \beta_j + 1$ & $G_j - \beta_j > 3$, we can also find two companion entries in $S_G(X+1)$ uniquely map to two entries in $S_G(X)$. These conditions corresponds to case 1.2 and 1.3.

Case 2: $\beta_i = 0$ and $|G_i - \alpha_i| \leq 1$, equivalently $X = G_i^2$ or $X = G_i * (G_i - 1)$

Similar to Case 1.2, from $X = \alpha_i \times G_i$, we have $\alpha_i \in S_G(x)$. Rearrange $X = \alpha_i \times G_i$ as follows:

$(X+1) + (G_i - 1) = (\alpha_i + 1) \times G_i$. Since $|G_i - \alpha_i| \leq 1 \Leftrightarrow (\alpha_i + 1) > (G_i - 1)$, so $G_i \in S_G(X+1)$. Since $G_i > (G_i - 1)$, $\alpha_i + 1 \in S_G(X+1)$.

$(X+1) + (\alpha_i - 1) = (G_i + 1) \times \alpha_i$. Since $|G_i - \alpha_i| \leq 1 \Leftrightarrow (G_i + 1) > (\alpha_i - 1)$, $\alpha_i \in S_G(X+1)$.

Since $\alpha_i > (\alpha_i - 1)$, $G_i + 1 \in S_G(X+1)$.

When $G_i - \alpha_i = 0$, $X = G_i^2$. We have $G_i \in S_G(X+1)$ and $G_i + 1 \in S_G(X+1)$. Using proof by contradiction (similar with the proof in case 2), we can show that $G_i + 1 = \alpha_i + 1 \notin S_G(X+1)$. In this case, we have $G_i = \alpha_i \in S_G(X), G_i = \alpha_i \in S_G(X+1), G_i + 1 = \alpha_i + 1 \in S_G(X+1)$ and $G_i + 1 = \alpha_i + 1 \notin S_G(X)$.

When $G_i - \alpha_i = -1$, $X = G_i \times (G_i - 1)$, we have $G_i \in S_G(X), G_i + 1 \in S_G(x), G_i \in S_G(X+1), G_i + 1 \in S_G(X+1), G_i + 2 \in S_G(X+1)$. In addition, using contradiction, we can also show that $G_i + 2 \notin S_G(X)$.

In Case 2, two companion entries in $S_G(X)$ map to three entries in $S_G(X+1)$. Since Case 2 only happens for one G_i value in $S_G(X)$ when $X = a^2$ or $X = a^2 - a$, we know that $|S_G(X)| + 1 = |S_G(X+1)|$ in this case. For other X values, only case 1.1, 1.2 and 1.3 can happen, which makes $|S_G(X)| = |S_G(X+1)|$. □

From Theorem 6, it is not hard to see that $|S_G(X)|$ can be expressed by the following fomula:

$$|S_G(X)| = \begin{cases} 2a - 1 & \text{if } a(a-1) + 1 \le X \le a^2 \\ 2a & \text{if } a^2 + 1 \le X \le a(a+1) \end{cases}, \tag{A.2}$$

where $a \ge 2, a \in \mathbb{Z}$. Formula (A.2) also completely matches the numerical analysis results shown in Figure A.1.

When ϵ is randomly picked in $\{-1,1\}$ and G can be both positive and negative integers, it is easy to see that the possible number of choices of G values to a given X will be $2|S_G(X)|$. Then, we can use $2|S_G(X)|$ to determine the parameter settings of $\alpha(l, h_I, f_I)$ and $\beta(l, h_I, f_I)$ in formula (2.14). Specifically, if we want the time/advantage ratio to be 2^w, we can tune the blinding factors $\alpha(l, h_I, f_I)$ and $\beta(l, h_I, f_I)$ so that in the distribution of $X_b(l, h_I, f_I)$, the

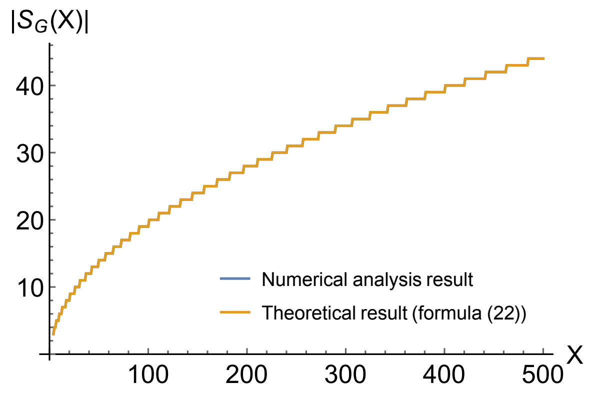


Figure A.1: Theoretical result in formula (A.2) and numerical analysis result

probability of $X(l,h_I,f_I) < 2^{2*w-4}$ is negligible. In this way, each $X_b(l,h_I,f_I)$ is mapped to at least 2^w choices of $G_b(l,h_I,f_I)$, so the time/advantage ratio to guess the real one is at least 2^w. In P^2-SAS, we set $\alpha(l,h_I,f_I)$ and $\beta(l,h_I,f_I)$ as 220 bits random positive integers so that the blinding factor scheme has security strength of at least 112 bits, which is consistent with the 2048 bit Paillier cryptosystem that we have implemented.

A.2 Security Analysis of the packing technique in formula (2.14)

As discussed in Section 2.5.2, while using the packing technique, the same α will be used to scale the $G_b(l,h_I,f_I)$ entries packed in the same 2048-bit plaintext. We have the following theorem to address the security issue.

Theorem 7. In formula (2.14), using the same α to scale the $G_b(l,h_I,f_I)$ entries packed in the one plaintext message will reduced the security level by $2q$-2 bits, where q is the number of $G_b(l,h_I,f_I)$ entries packed together in one plaintext.

Proof of Theorem 7. Consider the whole plaintext denoted as G. Given X which is calculated by G following (2.14), the number of different G satisfying $X = \epsilon(\alpha G - \beta)$ is $2 * 2\sqrt{2^{2048}} = 2^2 * 2^{1024}$. Now we consider each $G_b(l, h_I, f_I)$ entry packed in the plaintext G. Assume we are using different α for each entry, so the total number of $G_b(l, h_I, f_I)$ combinations to get X is $(2 * 2\sqrt{2^{2048/q}})^q = 2^{2q} * 2^{1024}$. So the time/advantage ratio is reduce from $2^{2q} * 2^{1024}$ to $2^2 * 2^{1024}$ and the security level is reduced by $2q - 2$ bit. $\square$

Since we only pack $q = 4$ segments in one 2048-bit plaintext, we only lose 6-bit security level.