P^2-SAS: Preserving Users' Privacy in Centralized Dynamic Spectrum Access Systems

Yanzhi Dou†, Kexiong (Curtis) Zeng†, He Li†, Yaling Yang†
Bo Gao‡, Chaowen Guan§, Kui Ren§, Shaoqian Li¶
†Virginia Tech, Blacksburg, VA, USA
‡Chinese Academy of Sciences, Beijing, China
§State University of New York at Buffalo, Buffalo, NY, USA
¶University of Electronic Science and Technology of China, Chengdu, China
†{yzdou,kexiong6,heli,yyang8}@vt.edu ‡gaobo@ict.ac.cn
§{chaoweng, kuiren}@buffalo.edu ¶lsq@uestc.edu.cn

ABSTRACT

Centralized spectrum management is one of the key dynamic spectrum access (DSA) mechanisms proposed to govern the spectrum sharing between government incumbent users (IUs) and commercial secondary users (SUs). In the current centralized DSA designs, the operation data of both government IUs and commercial SUs needs to be shared with a central server. However, the operation data of government IUs is often classified information and the SU operation data may also be commercial secret. The current system design dissatisfies the privacy requirement of both IUs and SUs since the central server is not necessarily trust-worthy for holding such sensitive operation data. To address the privacy issue, this paper presents a privacy-preserving centralized DSA system (P^2-SAS), which realizes the complex spectrum allocation process of DSA through efficient secure multi-party computation. In P^2-SAS, none of the IU or SU operation data would be exposed to any snooping party, including the central server itself. We formally prove the correctness and privacy-preserving property of P^2-SAS and 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.

CCS Concepts

•Networks → Mobile and wireless security; •Security and privacy → Privacy-preserving protocols;

Keywords

Dynamic spectrum access; secure multi-party computation; Paillier cryptosystem

1. INTRODUCTION

Dynamic spectrum access (DSA) technique has been widely accepted as a crucial solution to mitigate the potential spectrum scarcity problem. In the U.S., spectrum sharing between the government incumbents (i.e., federal or non-federal agencies) and commercial broadband operators/users is one of the key forms of DSA that are recommended by NTIA [28] and FCC [26]. Recommendations in the President's Council of Advisors on Science and Technology report (PCAST) have identified 1,000 MHz of federal spectrum to create "the first shared-use spectrum superhighways" [32].

The PCAST has also recommended to set up a centralized spectrum access system (SAS) to govern the spectrum sharing between incumbent users (IUs) and secondary users (SUs). This recommendation of SAS-driven design is also reflected in FCC's recent proposal for DSA in 3.5 GHz [17]. In Europe, a similar DSA scheme 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 [6]. Without loss of generality, we refer to the central DSA systems as SAS in the remainder of this paper.

One of the critical concerns in light of the increasing prospects of the SAS-driven spectrum sharing is the privacy issue [17]. For national security reasons, operation information of government IUs is often classified data. For example, the IUs in 3.5 GHz DSA band in the U.S. include military and fixed satellite service licensees [17]. In Europe, the IUs of 2.3-2.4 GHz LSA band include military aircraft services and police wireless communications [6]. 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 report [18, 32], 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 [5].

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

MobiHoc'16, July 04-08, 2016, Paderborn, Germany

DOI: http://dx.doi.org/10.1145/2942358.2942384

operation privacy from SAS becomes a critical challenge that can potentially deter the wide adoption of DSA technology. Unfortunately, there is very little existing research on this problem.

The goal of this paper is to fundamentally address the privacy challenge by developing a privacy-preserving SAS (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 nontrivial 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 [10, 27, 30], have been explored recently. These customized solutions are much more efficient than general-purpose MPC. Unfortunately, none of them are 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 interference management in DSA, SAS usually adopts complex radio propagation models for interference calculation, e.g., Longley-Rice (L-R) model [16]. 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 [33].

(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 paper, 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 [31]. 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 the 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. In addition, we propose practical acceleration methods to further improve P^2-SAS's efficiency.

The remainder of this paper is organized as follows. § 2 describes the system and adversary model, states our design goals, and introduces some preliminaries. § 3 describes the basic design of P^2-SAS. § 4 presents several refinement techniques on the basic design for better accuracy and efficiency. § 5 evaluates P^2-SAS. § 6 discusses some related work. § 7 concludes this paper.

2. PROBLEM STATEMENT

2.1 System Model

We consider a SAS involving three parties, as illustrated in Figure 1: IUs, SUs, and SAS Server. SAS Server refers to a central spectrum management infrastructure that allocates spectrum resources while considering incumbent operation protection 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 needs to send SAS Server a request for spectrum access along with its operation data. 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.

The above SAS model belongs to the protection zone approach for interference management, where an SU's operation is permitted as long as the sum of the SU's interference will not exceed any IU's interference sensitivity threshold. We have addressed the privacy issues of SAS for the exclusion zone approach in [14]. In exclusion zone scenario, an SU can only access the spectrum when it is out of the exclusion zones of all IUs. We focus on the protection zone approach in this paper because it can release more frequency opportunities than the exclusion zone approach [9], and thus is con-

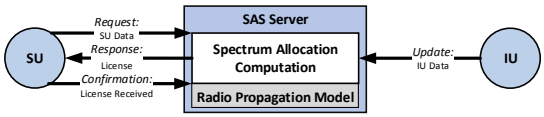


Figure 1: The system model of SAS

sidered as the future trend for interference management [12].

2.2 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 2.1 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 [23]. Specifically, we denote the computation of the SAS process in Section 2.1 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 [23].

Definition 1. (correctness) We say that π correctly computes f if

$$\{\mathsf{output}_{SUs}(x, y, z, n)\} \stackrel{c}{\equiv} \{f(x, y, z)\}, \quad (1)$$

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. $\stackrel{c}{\equiv}$ denotes computationally indistinguishability [23].

Definition 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

$$\{S(z, \mathsf{output}_{SAS}(x, y, z, n), n)\} \stackrel{c}{\equiv} \{\mathsf{view}^{\pi}(x, y, z, n)\}, \quad (2)$$

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. Loosely speaking, this privacy definition requires that SAS Server's view in an execution of π is simulatable given only its input and output, which then implies that SAS Server learns nothing from the protocol execution itself, as desired.

2.3 Preliminaries on Paillier Cryptosystem

The design of P^2-SAS heavily leverages the homomorphic properties of Paillier cryptosystem [31]. 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 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.

Table 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 the encryption of m with 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($\oplus$):	$\mathsf{Dec}_{\mathsf{sk}}([\![m_1]\!] \oplus [\![m_2]\!]) = m_1 + m_2$.
Scalar multiplication($\otimes$):	$\mathsf{Dec}_{\mathsf{sk}}(c \otimes [\![m]\!]) = c \cdot m$.
Subtraction($\ominus$):	$\mathsf{Dec}_{\mathsf{sk}}([\![m_1]\!] \ominus [\![m_2]\!]) = m_1 - m_2$.

3. BASIC P^2-SAS DESIGN

In this section, we present the basic design of P^2-SAS. In Section 4, we will introduce several refinement techniques on the basic design for better accuracy and efficiency.

3.1 P^2-SAS Design Overview

As shown in Figure 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; $\mathcal{K}$ can be operated by some neutral authority (e.g., FCC, NTIA) for security concern. Table 2 describes the notations to be used in the P^2-SAS scheme.

Table 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

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

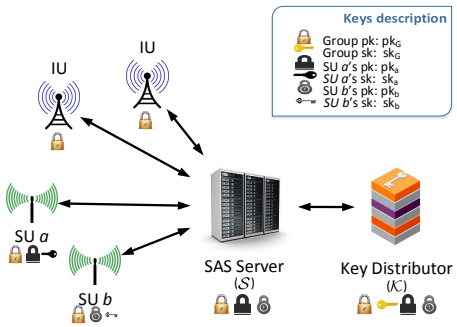


Figure 2: P^2-SAS overview.

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 are never exposed to any of the SAS components, including $\mathcal{S}$ and $\mathcal{K}$.

The remainder of this section presents the details of the above P^2-SAS design. First, we describe the content and format of the IU/SU 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 plaintext domain, and then we illustrate how each plaintext computation step can be carried out securely in ciphertext domain.

3.2 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 L-R model, which takes 13 parameters as input [16]. According to NTIA's guideline [24], among the 13 input parameters, only frequency, distance, antenna 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 [16]. Since DSA systems usually consider spectrum sharing in GHz band [32], 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 [1] and STRM3 [2]. With the above elimination of non-private parameters, all the privacy-related parameters for interference calculation can then be summarized in Table 3.

To reduce MPC's computation overhead in the later stages, we next need to represent these private IU/SU operation data of Table 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

Table 3: Privacy-related parameters

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

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 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:

$$T_i(l, h_I, f_I) \leftarrow T_i(l, h_I, f_I) + \phi, \tag{3}$$

where ϕ is the noise. Some preliminary noise generation techniques, such as introducing fake IU locations and fake interference thresholds, are proposed in [7] for traditional SAS. Moreover, we can use differential privacy [15] 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 [7], 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 in the future.

3.3 Spectrum Computation in Plaintext

In this subsection, we outline the spectrum computation steps in plaintext domain to ease the understanding of the secure spectrum computation steps in ciphertext domain in Section 3.4, 3.5 and 3.6.

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:

$$\mathbf{T}' := \sum_{i \in \text{all IUs}} \mathbf{T}_i, \tag{4}$$

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:

$$N(l, h_I, f_I) := \begin{cases} T'(l, h_I, f_I), & \text{if } T'(l, h_I, f_I) \neq 0 \\ \infty, & \text{if } T'(l, h_I, f_I) = 0 \end{cases}. \tag{5}$$

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:

$$F_b(l, h_I, f_I) := \sum_{j, h_S, f_S} I\,(l, j, h_I, h_S, f_I, f_S) \times R_b(j, h_S, f_S). \tag{6}$$

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

$$G_b(l, h_I, f_I) := N(l, h_I, f_I) - F_b(l, h_I, f_I), \forall (l, h_I, f_I). \tag{7}$$

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 $G_b(l^\star, h_I^\star, f_I^\star) \leq 0$:

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:

$$N(l, h_I, f_I) \leftarrow N(l, h_I, f_I) - F_b(l, h_I, f_I), \forall (l, h_I, f_I). \tag{8}$$

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 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 ciphertext domain.

3.4 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 $\mathcal{S}$.

3.5 Secure Computation of Step III

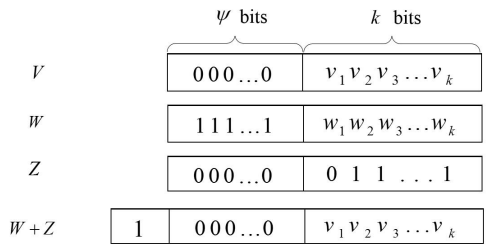


Figure 3: Integer encoding

Secure computation of Step III is tricky since formula (5) needs to determine the equality of $T'(l, h_I, f_I)$ and 0 in 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 [11, 13], 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 [8], 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 3.3'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 (5) without affecting the computation results.

With k fixed, the extra unused bits in plaintext domain can then be leveraged to realize Step III in 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 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 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',

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

Based on the above property, we realize Step III in 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

$$[\![\mathbf{T}']\!] := \oplus_{i \in \text{all IUs}} [\![\mathbf{T}_i]\!], \tag{9}$$

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

$$[\![\mathbf{N}]\!] := [\![\mathbf{T}']\!] \oplus [\![\mathbf{Z}]\!], \tag{10}$$

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.

3.6 Secure Computation of Step IV

Formula (6) and (7) in Step IV can be computed securely by straightforwardly applying homomorphic operations:

$$[\![F_b(l, h_I, f_I)]\!] := \oplus_{j, h_S, f_S} I(l, j, h_I, h_S, f_I, f_S) \otimes [\![R_b(j, h_S, f_S)]\!], \tag{11}$$

$$[\![G_b(l, h_I, f_I)]\!] := [\![N(l, h_I, f_I)]\!] \ominus [\![F_b(l, h_I, f_I)]\!]. \tag{12}$$

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 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 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 4, $\mathcal{S}$ creates $[\![Q_b(l, h_I, f_I)]\!]_{\mathsf{pk}_b}$, whose plaintext satisfies:

$$Q_b(l, h_I, f_I) = \begin{cases} 0, & \text{when } G_b(l, h_I, f_I) > 0 \\ -2, & \text{when } G_b(l, h_I, f_I) \leq 0 \end{cases}. \tag{13}$$

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 3.3, 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 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 (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 (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 (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 (13). Formal analysis of the relation between the value selection of the blinding factors and the effectiveness of the obfuscation can be found in the extended version of this paper [3].

Table 4: Key Conversion

$\mathcal{S}$:
(i) Generate $[\![\mathbf{X}_b]\!]$ by letting

$$[\![X_b(l, h_I, f_I)]\!] := \Big(\alpha(l, h_I, f_I) \otimes [\![G_b(l, h_I, f_I)]\!] \oplus [\![\tau(l, h_I, f_I)]\!] \ominus [\![\beta(l, h_I, f_I)]\!]\Big) \otimes \epsilon(l, h_I, f_I), \tag{14}$$

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

$$Y_b(l, h_I, f_I) := \begin{cases} 1, & \text{when } X_b(l, h_I, f_I) > 0 \\ -1, & \text{when } X_b(l, h_I, f_I) \leq 0 \end{cases}. \tag{15}$$

(v) Encrypt $\mathbf{Y}_b$ by pk_b and send $[\![\mathbf{Y}_b]\!]_{\mathsf{pk}_b}$ to $\mathcal{S}$.

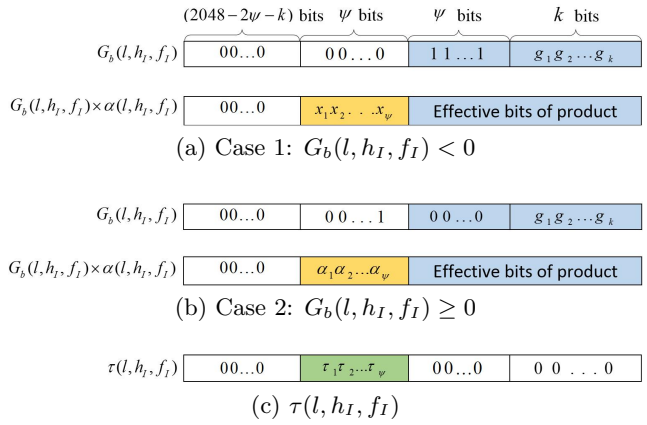
$\mathcal{S}$:
(vi) Generate $[\![\mathbf{Q}_b]\!]_{\mathsf{pk}_b}$ by letting

$$[\![Q_b(l, h_I, f_I)]\!]_{\mathsf{pk}_b} := (\epsilon(l, h_I, f_I) \otimes [\![Y_b(l, h_I, f_I)]\!]_{\mathsf{pk}_b}) \ominus [\![1]\!]_{\mathsf{pk}_b}, \tag{16}$$

where $\epsilon(l, h_I, f_I)$ is the same as the one in formula (14).

Careful readers may have noticed $\tau(l, h_I, f_I)$ in formula (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 4. The upper rows of subfigure 4a and 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 3.5, 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 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 blinding factor $\alpha(l, h_I, f_I)$. The overflowed bits are denoted as $x_1 x_2 ... x_\psi$ and $\alpha_1 \alpha_2 ... \alpha_\psi$ for case 1 and case 2, respectively, and $\alpha_1 \alpha_2 ... \alpha_\psi$ is the binary representation of $\alpha(l, h_I, f_I)$. Therefore, in case 2, the yellow part of the product holds an exact copy of $\alpha(l, h_I, f_I)$. 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 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,



(a) Case 1: $G_b(l, h_I, f_I) < 0$

(b) Case 2: $G_b(l, h_I, f_I) \geq 0$

(c) $\tau(l, h_I, f_I)$

Figure 4: Adding $\tau(l, h_I, f_I)$ for masking

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

$$[\![D_b]\!]_{\mathsf{pk}_b} := [\![C_b]\!]_{\mathsf{pk}_b} \oplus (\sigma \otimes (\oplus_{l,h_I,f_I} [\![Q_b\,(l, h_I, f_I)]\!]_{\mathsf{pk}_b})), \quad (17)$$

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 (8) securely by

$$[\![N(l, h_I, f_I)]\!] \leftarrow [\![N(l, h_I, f_I)]\!] - [\![U_b(l, h_I, f_I)]\!]. \quad (18)$$

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.

Firstly, $\mathcal{V}$ 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 provide $\mathbf{R}'_b$ since if SU b lies, the operation parameters provided will not match $\mathcal{V}$'s observation. 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. Using the Enc algorithm in Table 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 $[\![R_b{}'(j, h_S, f_S)]\!]$ does not equal $[\![R_b(j, h_S, f_S)]\!]$, $\mathcal{V}$ 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.

3.7 Security analysis

3.7.1 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 2.1. Detailed correctness proof of Paillier cryptosystem can be found in [31].

3.7.2 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 [23] 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 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 [31].

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 [25]. 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 the extended version of this paper [3]. □

4. 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.

4.1 Tuning of Quantization Granularity

As mentioned in Section 3.2, 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 derives smaller estimated attenuation value.

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 under-utilization 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.

$$\begin{aligned} \underset{A,B_s,B_i,B_a}{\text{minimize}} \quad & err(A, B_s, B_i, B_a) \\ \text{subject to} \quad & cost(A, B_s, B_i, B_a) \leq C, \\ & A \in \{100, 200, ..., 500\}, \\ & B_s \in \{1, 2, 3\}, \\ & B_i \in \{1, 2, 3, 4\}, \\ & B_a \in \{1, 2, ..., 30\} \end{aligned} \tag{19}$$

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 3. For example, formula (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 of a Paillier benchmark. The cost budget C is selected according to the response time required by specific application scenarios. Note that in (19), we assume $B_s \leq 3, B_i \leq 4$ 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 $B_a \leq 30$ since 30-bit space is more than enough to accommodate all the practical attenuation values.

To solve the optimization problem of (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 [29]. 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.

4.2 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.

4.2.1 Factoring

Through estimating the response time of $\mathcal{S}$ in Section 4.1, we observe that computing $[\![\mathbf{F}_b]\!]$ in formula (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 (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 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 (11) can be converted to the following more efficient form:

$$\begin{aligned} [\![F_b(l, h_I, f_I)]\!] := \oplus_{a=1}^{K} \Big(I_a \otimes \Big(\oplus_{I(l,j,h_I,h_S,f_I,f_S)\in a} \\ [\![R_b(j, h_S, f_S)]\!] \Big) \Big). \end{aligned} \tag{20}$$

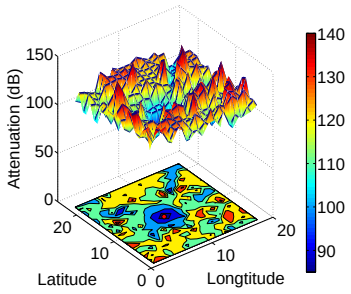


Figure 5: A sample of attenuation map in Washington D.C.

4.2.2 Precomputing if $\mathbf{I}$ is published

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 (11) in 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 (11) (or (20)), so that the most computationally intensive part in P^2-SAS is avoided.

4.2.3 Ciphertext Packing

Ciphertext packing technique [20] allows P^2-SAS to pack multiple integers into one ciphertext to increase the computation throughput, thus reducing computation and communication overhead. Recall the integer encoding policy specified in Section 3.6. All the integers involved in $\mathcal{S}$'s computation can be represented in $p := (k + 2\psi)$-bit space. 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. We apply the ciphertext packing technique to formulas (9), (10), (11) (or (20)), (12), (14), and the computation overhead of these formulas is reduced by a factor of q.

4.2.4 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.

5. EVALUATION

5.1 Experiment configurations

Implementation. We construct a Paillier cryptosystem of 112-bit security level by setting the security parameter n as 2048 bits. For parallelization, we divide the computation into 24 threads and evenly distribute them to three desktops with Intel i7-3770 CPU @ 3.40GHz and 12GB RAM.

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! [4] to calculate the attenuation map $\mathbf{I}$ in this area. Real terrain data from USGS [1] and SRTM3 [2] is fed to the L-R model. Important experiment parameter settings are presented in Table 5. The first four parameter values are obtained by solving the optimization problem (19) in Section 4.1.

Table 5: Experiment parameter settings

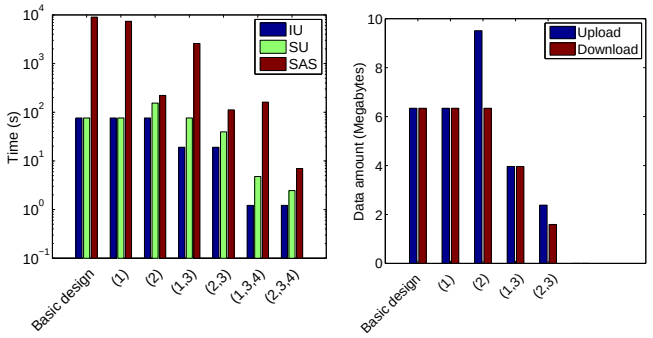
Side length of grid (A)	500
Bit length for SU height (B_s)	1
Bit length for IU height (B_i)	1
Bit length for attenuation (B_a)	20
Number of IUs	350
Number of SUs	20000
Number of grids (L)	650
Number of frequency channel (F)	5
Bit length of integer encoding (k)	60
Bit length of blinding factors (ψ)	220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

5.2 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 implementation, 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. The experiment 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 4.1.

5.3 Effectiveness of Acceleration Methods

In this subsection, we evaluate the performance improvement of different acceleration methods introduced in Section 4.2. 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. These metrics are critical to assess P^2-SAS's scalability for mobile SUs in highly dynamic environment.



(a) Computation overhead (b) Communication overhead

Figure 6: Comparison of different acceleration methods

The evaluation results of computation overhead and communication overhead are shown in Figure 6a and Figure 6b, respectively. For simplicity, we denote the acceleration method of Section 4.2.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 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 5.4.

5.4 Efficiency Comparison with Traditional SAS

For computation overhead, P^2-SAS's average processing time per SU spectrum request is 6.96 seconds as seen from Figure 6a. 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 6b. For traditional SAS, the uplink and downlink data amounts are 3.51 KB and 2.53 KB, respectively. We can 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.

6. RELATED WORK

Privacy in SAS-Driven Systems. In [19], private information retrieval (PIR) techniques are employed to protect SU's location privacy against untrusted SAS. However, in [19], 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 [7], 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 [7], it does not address the privacy threat from untrusted SAS.

Multi-party computation (MPC). 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 [22] or fully homomorphic encryption (FHE) [21]. While these approaches are appealing in the generality, they are still far from practical for most of the real-world applications. Therefore, as Goldreich pointed out in [22], 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 work is to find a customized solution that is much more efficient than the general theoretical solutions.

7. CONCLUSION & FUTURE WORK

In this paper, 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.

The current design of P^2-SAS only manages the spectrum sharing between IUs and SUs. The spectrum sharing among SUs is not yet considered. Some recent DSA proposals by FCC and the research community [17] also explore the possibility of using SAS to manage spectrum sharing among SUs. We plan to incorporate this new feature of SAS design into the future version of P^2-SAS. We also plan to extend the current design to handle some specific DSA scenarios, such as 3-tier model in 3.5 GHz. Finally, we seek to relax the semi-honest requirement of P^2-SAS and consider malicious adversary scenarios in the future.

8. REFERENCES

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

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

[3] https://www.dropbox.com/sh/2ln02adkukgqutz/AADOW0mb5feDe1URh2hnwGGOa?dl=0.

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

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

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

[7] 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.

[8] E. Barker et al. Recommendation for Key Management – Part 1: General (Revision 3). *NIST Special Publication*, 800(57):1–147, 2012.

[9] 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.

[10] P. Bogetoft et al. Secure multiparty computation goes live. In *Financial Cryptography and Data Security*, pages 325–343. Springer, 2009.

[11] O. Catrina et al. Improved primitives for secure multiparty integer computation. In *Security and Cryptography for Networks*, pages 182–199. 2010.

[12] 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*, 22, 2013.

[13] I. Damgård et al. Unconditionally secure constant-rounds multi-party computation for equality, comparison, bits and exponentiation. In *Theory of Cryptography*, pages 285–304. Springer, 2006.

[14] 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. *to appear in Proceedings of the 36th IEEE ICDCS*, 2016.

[15] C. Dwork. Differential privacy. In *Automata, languages and programming*, pages 1–12. Springer, 2006.

[16] FCC. Longley-rice methodology for evaluating TV coverage and interference. *Office of Engineering and Technology (OET) Bulletin*, (69), 2004.

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

[18] FCC. Shared Commercial Operations in the 3550–3650 MHz Band. *Federal Register*, 80(120), June 2015.

[19] 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*, pages 2751–2759. IEEE, 2013.

[20] T. Ge and S. Zdonik. Answering aggregation queries in a secure system model. In *VLDB*, pages 519–530, 2007.

[21] C. Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009.

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

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

[24] G. A. Hufford, A. G. Longley, W. A. Kissick, et al. *A guide to the use of the ITS irregular terrain model in the area prediction mode*. US Department of Commerce, National Telecommunications and Information Administration, 1982.

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

[26] P. Kolodzy and I. Avoidance. Spectrum policy task force. *Federal Commun. Comm., Washington, DC, Rep. ET Docket*, (02-135), 2002.

[27] Y. Lindell and B. Pinkas. Secure multiparty computation for privacy-preserving data mining. *Journal of Privacy and Confidentiality*, 1(1):5, 2009.

[28] 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.

[29] M. Minoux and H. Tuy. Discrete monotonic global optimization. Technical report, Citeseer, 2002.

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

[31] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT'99*, pages 223–238. Springer, 1999.

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

[33] 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*, pages 135–140. ACM, 2014.