ProCSA: Protecting Privacy in Crowdsourced Spectrum Allocation

Max Curran, Xiao Liang, Himanshu Gupta, Omkant Pandey, Samir R. Das

Stony Brook University, Stony Brook, USA
{mcurran,liang1,hgupta,omkant,samir}@cs.stonybrook.edu

Abstract. Sharing a spectrum is an emerging paradigm to increase spectrum utilization and thus address the unabated increase in mobile data consumption. The paradigm allows the "unused" spectrum bands of licensed primary users to be shared with secondary users, as long as the allocated spectrum to the secondary users does not cause any harmful interference to the primary users. However, such shared spectrum paradigms pose serious privacy risks to the participating entities, e.g., the secondary users may be sensitive about their locations and usage patterns. This paper presents a privacy-preserving protocol for the shared spectrum allocation problem in a crowdsourced architecture, wherein spectrum allocation to secondary users is done based on real-time sensing reports from geographically distributed and crowdsourced spectrum sensors. Such an architecture is highly desirable since it obviates the need to assume a propagation model, and facilitates estimation based on real-time propagation conditions and high granularity data via inexpensive means.

We design our protocol by leveraging the efficiency and generality of recently developed fast and secure two-party computation (S2PC) protocols. We show that this approach leads to practical solutions that outperform the state-of-the-art in terms of both efficiency as well as functionality. To achieve the desired computational efficiency, we optimize the spectrum allocation algorithm to select a small number of relevant reports based on certain parameters. This results in a faster RAM program for power allocation which, under suitable adjustments to underlying arithmetic operations, can be efficiently implemented using S2PC. We use the standard "ideal/real paradigm" to define the security of spectrum allocation and prove security of our protocol (in the semi-honest model). We also provide data from extensive simulations to demonstrate the accuracy, as well as computational and communication efficiency of our schemes.

1 Introduction

The RF spectrum is a natural resource in great demand due to the unabated increase in mobile (and hence, wireless) data consumption [4]. The research community has addressed this capacity crunch via development of *shared spectrum paradigms*, where the *unused* spectrum bands of a licensed primary user (PU)

can be allocated to an unlicensed secondary user (SU) as long as SU's usage does not cause harmful (wireless) interference to the PU. In a commonly used architecture for such shared spectrum systems, a centralized spectrum manager (SM) allocates available spectrum to SUs upon request, based on PUs' parameters and signal attenuation (path-loss) characteristics. In the *crowdsourced sensing* model we follow, the path-loss values are estimated from real-time sensing reports of geographically distributed and crowdsourced spectrum sensors (SS). Crowdsourcing allows high granularity spectrum data collection via relatively inexpensive means, and most importantly, obviates the need to *assume* a signal propagation model. However, presence of many independent entities makes the shared spectrum system particularly prone to leakage of private information (e.g., location of radar transmitter) [34,41,16]. As the viability of crowdsourced paradigm may depend upon privacy assurance of the crowdsourcing users (i.e., SS devices), it is critical to develop secured spectrum allocation protocols that preserve privacy of all entities. The goal of our work is to develop an efficient privacy-preserving spectrum allocation scheme in the context of the aforementioned shared spectrum architecture.

1.1 Spectrum Allocation Model, Security Challenges, Related Work

Crowdsourced Shared Spectrum Architecture. Spectrum allocation in shared spectrum systems has been studied extensively (see [59] for a survey). In the centralized SM architecture, it is generally assumed that the SM has complete knowledge of the PU parameters. Many prior works assume a propagation model which allows spectrum allocation power to be computed via simple techniques (see [59] survey). However, in practice, since even the best-known propagation models [48,25,22] have unsatisfactory accuracy, spectrum allocation must be done overly conservatively for correctness. Crowdsourced sensing has the potential to eliminate this limitation.

In a crowdsourced architecture, for a spectrum allocation *query* from the SU, the spectrum manager (SM) first estimates appropriate signal path-loss values from known PUs' parameters and real-time sensing reports of crowdsourced spectrum sensors (SS), and then use the estimated path-loss values to allocate spectrum to the SU. Allocation based on real-time channel conditions is important for accurate power allocation, as the conditions affecting signal attenuation (e.g., air, rain, vehicular traffic) may change over time. However, spectrum allocation based on sensing reports can be challenging, due to need for accurate path-loss estimation techniques from relatively inexpensive sensors – but the challenge is mitigated with the availability of a large number of sensing reports via crowdsourced spectrum sensing [13,40]. The practicality of crowdsourced sensing architectures has been demonstrated in research projects [13,67,11] as well as commercial ventures such as Flightaware [3]. Malicious behavior of some SS nodes (faulty sensing reports) can be handled by appropriate fault-tolerance strategies [19].

Spectrum Allocation Algorithm. For a given SU query, the goal of the spectrum allocation algorithm is to allocate maximum possible power to the SU such

that its transmission at the allocated power would not interfere with PU's reception at any of its receivers. There are many ways to model PU receivers, e.g., a coverage region around PU. As in [39], we assume a finite set of representative receivers called PURs around a PU. Each PUR is associated with an *initial threshold*, which is continually updated, to signify the maximum additional interference it can tolerate from the SUs. At a high-level, for a single SU request (we discuss multiple SUs in §3.1), the spectrum allocation algorithm consists of the following steps: (i) compute the path loss between the SU and each of the PURs, (ii) allocate spectrum as below, (iii) update the PURs' thresholds. See Figure 1. More formally, let us denote the path loss function by $P(,)$; we discuss estimation of this function in more detail in §3.1. If an SU $\mathrm{S_i}$ at location ℓ_i is allowed to transmit at power t_i, then the signal power received at PUR R_j at location ℓ_j is given by $p_{ij} = t_i \cdot P(\ell_i, \ell_j)$. To ensure that p_{ij} is less than each R_j's current threshold τ_j, the maximum power that can be allocated to $\mathrm{S_i}$ is:

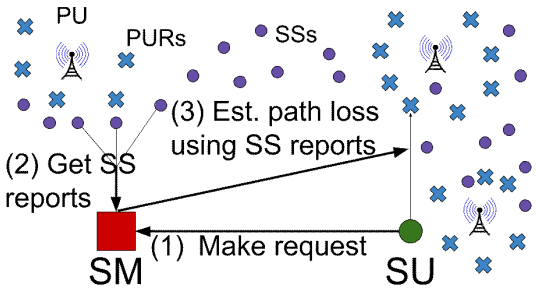

Fig. 1: Spectrum Allocation in a Shared Spectrum System

$$\min_j \frac{\tau_j}{P(\ell_i, \ell_j)}. \tag{1}$$

Once a certain transmit power t_i has been allocated to an SU $\mathrm{S_i}$, the threshold for a PUR at location ℓ_j is updated as:

$$\tau_j = \tau_j - t_i \times P(\ell_i, \ell_j). \tag{2}$$

Security Challenges. Despite the great potential of shared spectrum paradigms in improving spectrum utilization efficiency, these systems suffer from serious privacy and security risks – particularly, due to the presense of many independent entities. The data collected by SM from SU/SS/PU entities contains sensitive information such as the locations, transmit power, sensing reports, requested spectrum, etc. For example, a PU can be a military entity, an SU can be telecom operator, or an SS can be a private user. It is critical to protect the location, behavior and other information of such entities for personal privacy, corporate secrecy, and/or national security interests. Furthermore, the viability of crowdsourced paradigm may depend upon privacy assurance of the crowdsourcing users (i.e., SS devices).

In order to ensure privacy of participating entities, it is essential that the SM does not learn any information about them (including the allocated spectrum power since it can reveal approximate location of the requesting SU). Furthermore, the scheme must not introduce too much latency, to maintain system's prompt responsiveness to SU requests; moreover, a delayed response may render the spectrum availability information obsolete and thus useless. Such strong

privacy and efficiency requirements introduce several technical difficulties that are hard to resolve using basic cryptography. Indeed, the spectrum allocation function, which includes estimation of the path-loss values (as described in §3.1) computed by the SM, has a rather complex algorithm. While this can be handled using fully homomorphic encryption (FHE) [28], current FHE schemes are far from practical. Another option is to consider general-purpose secure multi-party computation (MPC) protocols [63,31]. While MPC would be impractical if all sensor nodes are involved in the computation, it can be quite efficient for smaller computations involving two (or three) parties. This is the approach we take since, in the setting of secure spectrum allocation, two semi-trusted non-colluding parties are naturally available: the SM and a key server (KS). The non-trivial part is to express the computation (at the time of SU request) as a small circuit or RAM program.

Related Works. The privacy and security issues in shared spectrum systems have received serious attention in the research community only in the last decade (see [35] for a survey). Due to the aforementioned difficulties, existing works focus on simpler versions of spectrum allocation. In particular, many privacy-preserving works have focused on the database-centric architecture, where the spectrum allocation is done based on a spectrum database, often maintained and controlled by a third party (e.g., Google, Spectrum Bridge, RadioSoft, etc.). Here, the security solutions have focused on protecting SU's location privacy by either anonymizing its location/identity [58,66,45], private retrieval from the database [15,24], or differential privacy or data obfuscation techniques [27,41] (also used to protect PU privacy [16,54]). Most works in the crowdsourced spectrum management have focused on protecting privacy of SS nodes only, e.g., location leakage of spectrum sensors from their sensing reports. These include encryption approaches to conceal the sensing reports [46,38,36] or using intermediate nodes to hide location [38,47,37], which incurs significant computation and communication overheads. Other approaches consider distributed architectures [42] or architectures involving multiple service providers [60]. In summary, most works have focused on privacy of SUs/SSs only, and either use data obfuscation techniques or incur substantial overheads.

State-of-the-Art. The state-of-the-art as well as closest to our work is the P^2-SAS system [21] which works in a simplified model where (a) rather than using SSs' real-time sensing reports, the SM pre-computes a signal attenuation map based on an *assumed* propagation model such as Longley-Rice [56]; (b) SM does not compute the actual allocation value; instead, the SM only provides a binary yes/no answer indicating whether the SU can transmit at the requested power v. Roughly speaking, these simplifications allow P^2-SAS to express the computation as a *linear* function which can be computed over encrypted values using the Paillier cryptosystem [52]. Since SM is not fully trusted, P^2-SAS also introduces a key server (KS) who is responsible for generating relevant keys but does not see the encrypted data held by the SM. P^2-SAS yields a solution in the semi-honest model where parties follow the protocol instructions and do not collude (but may analyze the data in their possession). Despite its limitations, P^2-SAS makes significant progress on this problem: it can serve yes/no answers to SU re-

quests with 97-98% accuracy under seven seconds, with appropriate acceleration strategies including parallelization of many computational steps.

1.2 Our Contributions

In this work, we present the first general solution to the problem of privacy-preserving spectrum allocation in the crowdsourced spectrum sensing model wherein a centralized spectrum manager orchestrates spectrum allocation using sensing reports from crowdsourced spectrum sensors. Our overall contributions are as follows:

- We present a new architecture for the problem of privacy preserving spectrum allocation based on fast and general-purpose S2PC protocols [8,43,17,44,18]. Our protocol computes the power allocation based on the *current* sensing reports by the SS nodes. Since the conditions affecting signal attenuation (e.g., air, rain, vehicular traffic) may change, path-loss estimation based on real-time sensing reports is important for accurate power allocation. In contrast, the state-of-the-art system P^2-SAS pre-computes a signal attenuation map over a grid based on an *assumed* propagation model, which then remains static and does not reflect the latest conditions. We remark that pre-computation of a attenuation map from sensing reports (i.e., without assuming a propagation model) in a *privacy-preserving* manner is also non-trivial.
- Our protocol is an order of magnitude faster that the the state-of-the-art systems. More specifically, our protocol can compute the *actual* power allocation in 2-2.5 seconds on average whereas P^2-SAS takes 7 seconds for a yes/no answer which must be iterated a few times to compute the actual allocation. See Table 1.
- As the spectrum allocation computation involving large number of sensing reports is computationally very expensive to be carried out over S2PC directly, we optimize the spectrum allocation algorithm to use only a small number of relevant sensing reports. We show experimentally that this optimization does not affect the quality of power allocation. Overall, this optimization results in a faster RAM program which can be efficiently implemented using fast S2PC protocols.
- To circumvent implementation issues in using available libraries for "S2PC for RAM program" (see §3.2), we build a custom solution that can be implemented in Ivory [55]. More specifically, we design a method for performing oblivious read/write operations, and use these routines with fast S2PC for (small) circuits to obtain a protocol that is proven secure in the semi-honest model under the standard ideal-world/real-world paradigm. We use additional optimizations such as moving arithmetic operations outside the S2PC framework whenever possible to extract further efficiency.
- The generality of our approach allows us to support *simultaneous allocation* queries in which several SUs simultaneously request for power allocation as opposed to just one. The knowledge of several SU requests at once allows the computation of power allocation for each one of them in a more fair and optimal manner. Ours is the first system to support such simultaneous allocation;

Algorithm	Time	Error wrt. Plaintext	Error wrt. Optimal	Comm. Cost
Two SMs	2 sec	2.10^{-4} dB	1 dB (`Log`), 4 dB (`L-R`)	0.15 MB
SM-KS	2.5 sec	2.10^{-4} dB	1 dB (`Log`), 4 dB (`L-R`)	5.35 MB
P^2-SAS [21]	7 secs	–	2.72%	5 MB

Table 1: Summary of Results

it is not possible in P^2-SAS or other known solutions since they commonly rely on some form of homomorphic encryption, which severely limits the type of functions they can compute within the encryption.

Results Summary. Table 1 shows the average time and accuracy of our designed schemes to serve each SU request in a large area with 400 PUs and 40,000 SSs in two propagation models (used to generate the ground truth), viz., Log-distance (`Log`) and Longley-Rice (`L-R`). Table shows results for two of our schemes: Two SMs (two spectrum managers) and SM-KS (SM and a key server). To handle the SU request, we select 10 PUs and SSs appropriately using a grid of 100×100 over the area. See §4 for further details. As the P^2-SAS [21] work outputs only `yes/no` answers, the P^2-SAS entry below shows accuracy as percentage of false positives and negatives.

2 Defining Semi-Honest Secure Spectrum Allocation

We define the functionality for spectrum allocation within the framework of secure multi-party computation. Informally, a MPC protocol is said to be *secure* if any information learned by an adversary can also be generated (or "simulated") by an ideal-world *simulator*. A formal treatment for MPC framework is given Appendix A. In the following, we define the ideal functionality for our spectrum allocation task. We focus on *semi-honest model with static corruption*, which means the set of corrupted parties is fixed before the execution of the protocol and all parts (including the corrupted ones) follow the protocol. We also assume authenticated communication channels between each pair of parties.

Ideal Functionality for Spectrum Allocation. The spectrum allocation functionality involves the following participants: the requesting SU S_i, PUs, PURs, SSs, and the two spectrum managers SM_0 and SM_1. We note that the roles of PUs, PURs, and SSs in the protocol are limited in that they only provide data for the computation but do not receive any output. For clarity of presentation, we will use PNs (acronym for Private Nodes) to represent all PURs, PUs and SSs. Also, even though PNs consist of many independent nodes, for ease of presentation, we will treat the entire set of PNs as one single party and use D to denote the concatenation of their data. The above simplifications are merely for clarity of presentation and do not affect the generality of our results.

The spectrum allocation functionality f^{SA} is described as follows (details are given in §3.1):

– **Input:** The requesting SU S_i sends its location ℓ_i to f^{SA}. SM_0 and SM_1 input nothing to f^{SA}, but we use the symbol $\perp$ as a placeholder for them. All the PNs (i.e., all the PURs, PUs and SSs, as mentioned above) send their data D to f^{SA}.

– **Computation:** Upon receiving the above input $(\ell_i, \perp, \perp, D)$, f^{SA} does the following (as described in §3.1):
  - For $j \neq i$, compute the path loss $P(\ell_i, \ell_j)$ between the $\mathrm{S_i}$ and $\mathrm{S_j}$
  - Calculate the proper transmit power t_i to $\mathrm{S_i}$ per Eqn. (1)
  - Update the thresholds for each PUR location ℓ_j per Eqn. (2)
– **Output:** $(t_i, \perp, \perp, \perp)$ are the outputs to participants $(\mathrm{S_i}, \mathrm{SM}_0, \mathrm{SM}_1, \mathrm{PNs})$ respectively.

We note that SM_0 and SM_1 neither send any input nor receive any output from f^{SA}. Even though the SMs are "dummy" within f^{SA} functionality, their existence is important to define and prove the security of our protocol.

Correctness and Security. For a protocol Π, we define its correctness and security w.r.t. f^{SA} in the following way.

Definition 1 (Correctness). *We say that Π correctly computes f^{SA} if the following holds except for negligible probability*

$$\mathsf{output}_{\Pi}(\ell_i, \perp, \perp, D) = f^{\mathsf{SA}}(\ell_i, \perp, \perp, D) \tag{3}$$

where the tuple $(\ell_i, \perp, \perp, D)$ denotes the input data from $(\mathrm{S_i}, \mathrm{SM}_0, \mathrm{SM}_1, \mathrm{PNs})$ and output_{Π} is the output function of protocol Π.

Definition 2 (Security). *We say that Π securely computes f^{SA} in a semi-honest model with static corruption if there exists a probabilistic polynomial-time algorithm $\mathcal{S}^{\Pi}$ such that for every $I \subseteq \{\mathrm{S_i}, \mathrm{SM}_0, \mathrm{SM}_1, \mathrm{PNs}\}$ that does not contain both SM_0 and SM_1,*

$$\{\mathcal{S}^{\Pi}(I, \mathsf{input}_I, f_I^{\mathsf{SA}}(\ell_i, \perp, \perp, D))\} \stackrel{c}{\equiv} \{\mathsf{view}_I^{\Pi}(\ell_i, \perp, \perp, D)\} \tag{4}$$

where the tuple $(\ell_i, \perp, \perp, D)$ denotes the input data from $(\mathrm{S_i}, \mathrm{SM}_0, \mathrm{SM}_1, \mathrm{PNs})$, input_I denotes the input of parties in set I and $\mathsf{view}_I^{\Pi}(\ell_i, \perp, \perp, D)$ is the views of parties in set I at Π's termination on input $(\ell_i, \perp, \perp, D)$.

We remark that, in our model, set I cannot simultaneously include both SM servers since they are non-colluding. The definitions are easy to modify to work with a single SM and a KS, or other equivalent setups.

3 Secure Spectrum Allocation

Our secured approach to spectrum allocation is based on on the S2PC technique, but we incorporate various optimizations to make the overall approach viable for our context. We start with describing the plaintext (unsecured) version of our spectrum allocation algorithm.

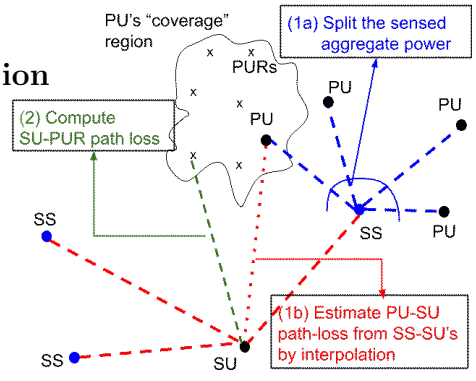

Fig. 2: Path Loss Estimation

3.1 Plaintext Algorithm

For a new SU request, the `Plaintext` algorithm can be described as a sequence of the following steps (as per §1.1): (i) compute the path loss between the SU and each of the PURs, (ii) allocate spectrum as per Eqn. (1), and (iii) update the thresholds of the PURs based on the allocation to the SU. We describe the first step in detail below; the other two steps are just straightforward assignment of values to appropriate variables. Later, we motivate and discuss selection of SSs and PUs to make the algorithm more computationally efficient, without much compromise in spectrum utilization.

Path Loss Estimation. As per Eqn. (1), we need to compute the path loss between the requesting SU S_i and each of the PUs' receivers (i.e., PURs). For a given PUR R_{jk} of a PU P_j, we compute the path loss $P(S_i, R_{jk})$ between R_{jk} and S_i as follows. See Figure 2.

1. First, we compute the path loss $P(S_i, P_j)$ between the SU S_i and PU P_j in two steps as follows:
   (a) Compute path loss $P(P_j, C_\ell)$ from PU P_j to each of the spectrum sensors C_ℓ. Since a spectrum sensor C_ℓ only senses the *aggregate* power received from all PUs, computing path loss from PU P_j to C_ℓ requires splitting the sensed power across the PUs (as described later).
   (b) Use interpolation to compute the path loss $P(S_i, P_j)$.
2. Then, we compute the desired path loss $P(S_i, R_{jk})$ from the above computed $P(S_i, P_j)$.

We now describe each of the above steps below.

(1a) Estimating $P(P_j, C_\ell)$ From Sensed Power at C_ℓ. As mentioned above, a spectrum sensor C_ℓ senses only the *aggregate* power received from all the PUs. Thus, we must first "split" the total received power of C_ℓ among the PUs; we do this splitting based on the weighted distance as follows. Let r_ℓ be the total power received at C_ℓ, and t_x be the transmit power of a PU P_x. Then, we estimate the power received r_{jl} at C_ℓ due to PU P_j as:

$$r_{jl} = \frac{t_j / d(C_\ell, P_j)^{\alpha_s}}{\sum_{P_x} t_x / d(C_\ell, P_x)^{\alpha_s}} \times r_\ell \tag{5}$$

Above, $d()$ is the distance function and α_s is an exponent parameter that is used to control the above splitting. Now, we can easily compute the path loss $P(P_j, C_\ell)$ as:

$$P(P_j, C_\ell) = r_{jl}/tj \tag{6}$$

Note that the above estimation of $P(P_j, C_\ell)$ does not depend upon S_i, and then can be precomputed.

(1b) Interpolation to Compute $P(S_i, P_j)$. Once we have estimated the path loss between a PU P_j and every SS C_ℓ, we use interpolation to estimate the path loss from P_j to the current SU S_i under consideration. Prior works [64,13] have used Ordinary Kriging (OK), k-nearest neighbors (k-NN) classifier, or inverse distance weighted (IDW) approaches for such interpolation—with k-NN and OK performing similarly [64]. Here, for simplicity, we start with the IDW approach, and later refine it to using IDW over k nearest neighbors (making the overall

scheme akin to a more sophisticated version of the traditional k-NN scheme [64]). Using IDW, we get (here, C_x is a SS node):

$$P(\mathrm{S_i}, P_j) = \frac{\sum_{C_x} P(C_x, P_j) / d(\mathrm{S_i}, C_x)^{\alpha_p}}{\sum_{C_x} 1/d(\mathrm{S_i}, C_x)^{\alpha_p}} \tag{7}$$

Above, α_p is an exponent parameter that is used to control the above interpolation.

(2) Compute Path Loss $P(\mathrm{S_i}, R_{jk})$ from SU to PUR. We now use the estimated path loss between the SU $\mathrm{S_i}$ and a PU P_j to estimate the path loss between the SU $\mathrm{S_i}$ and the PU P_j's PURs. Each PUR R_{jk} is represented by its location. To estimate the desired path loss $P(\mathrm{S_i}, R_{jk})$, we assume a uniform log-distance path loss model within the triangle of nodes P_j, $\mathrm{S_i}$ and R_{jk}. In particular, we use:

$$P(\mathrm{S_i}, R_{jk}) = \frac{P(\mathrm{S_i}, R_j)\,(d(\mathrm{S_i}, R_{jk}))^{\alpha_p}}{(d(\mathrm{S_i}, R_j))^{\alpha_p}} \tag{8}$$

Selection of PUs **and** SSs **for Computational Efficiency.** Involving all the PUs and SSs in the above path loss estimation is quite inefficient, as the number of PUs and especially SSs can be very large. This computational efficiency is particularly critical in the secured S2PC implementation. Thus, to improve computational efficiency, we devise a strategy to select only a small number of PUs and SSs—that are most pertinent to the SU $\mathrm{S_i}$ requesting spectrum. Note that the PUs that are very far away from $\mathrm{S_i}$ are unlikely to be affected by the spectrum allocated to $\mathrm{S_i}$, especially if there are sufficiently many PUs that are close enough to $\mathrm{S_i}$. Similarly, only the SSs that are close to the selected PUs (and thus to the SU $\mathrm{S_i}$) are going to be much useful in the above interpolation step. Note that in the interpolation step, the SSs are weighted by the inverse distance to $\mathrm{S_i}$; thus, SSs that are far away from $\mathrm{S_i}$ will have minimal impact. Based on the above arguments, for the sake of computational efficiency, we thus select PUs and SSs that are "close" to the given SU $\mathrm{S_i}$ and use only these PUs and SSs in the above computations. In particular, given an SU $\mathrm{S_i}$, we pick k_{ss} nearest SSs, and k_{pu} nearest PUs; here, the distance to SSs is unweighted, but the distance to a PU is weighted by the average of the thresholds of its PURs. The k_{ss} and k_{pu} values may be chosen based on the density of PUs and SSs. Our simulation results (see §4) show that only a small number of close-by SS and PU nodes suffice to obtain sufficiently accurate path-loss, if the density of SS nodes is sufficiently high.

Grid Based Implementation. To implement the selection of SSs and PUs efficiently, we employ a grid-based heuristic wherein we divide the given area into cells using horizontal and vertical grid lines, and associate with each cell the list of PUs and SSs that should be selected if the requesting SU is at the cell's center. When a request of SU $\mathrm{S_i}$ comes, we determine the cell C in which the $\mathrm{S_i}$ lies, and use the PUs and SSs associated with C for path-loss estimation steps. It is important to note that this grid-based heuristic is not exact, i.e., it may not return the nearest set of SSs and weighted PUs as it approximates the position of a requesting SU $\mathrm{S_i}$ by the center of C, the cell in which $\mathrm{S_i}$ lies. However, our simulation results show that the grid-based approach is computationally efficient and

sufficiently accurate for our purposes. Note that the set of PUs associated with some cell may need to be updated due to updates to the PUR thresholds after every spectrum allocation (recall that the distance to PUs are weighted by the average of their PUR thresholds); for efficiency, we only update the thresholds of the PURs of the selected PUs.

Handling Multiple SUs. The above describes the process to allocate spectrum to a single SU request. Multiple SU requests can be easily handled one at a time, except that in step (1a) above, we need to also account for the fact that a SS may sense power from SU transmissions. This can be handled easily by storing information about the active SUs with the containing cell, and incorporating it in the (1a) step. Multiple SU requests can also be handled *simultaneously*, to incorporate a given fairness constraint, by solving a system of linear equations (with one equation for each PUR) within our S2PC framework.

3.2 Secured Algorithm using Two SMs

In this section, we present the secured implementation of our plain algorithm between two spectrum managers. We first present the solution in the simpler setting where there are two semi-honest SMs, and then show how to replace the second SM with a key server. This allows us to focus on core issues related to security and efficiency first.

At a high-level, this secured algorithm works by having all the PNs secret-share their data to the two SMs, who will then run S2PC protocols between two SMs for each stage of our plain algorithm. Most of our spectrum allocation algorithm involves only simple arithmetic operations which can be implemented efficiently in S2PC; the only parts that require special attention are the following: in our grid-based interpolation, SMs need to read data from the selected PNs and update the threshold for PURs. These operations happen on the large data array secret-shared between the two SMs.

A direct S2PC implementation will be quite inefficient. One option is to use "S2PC for RAM program" [51,33]. However, the actual implementation using known libraries for efficient S2PC for RAM program [33,57,62,61,65,20,10] runs into several issues. While there are several available implementations that offer different features, these are maintained by individual researchers/teams and often incompatible with each other. Our spectrum allocation algorithm best operates as a RAM program often switching between arithmetic and boolean operations, and it becomes difficult to obtain a workable solution existing known implementations. Therefore, we design a novel oblivious read/write algorithms, which allow fast and secure access of the secret-shared data array. These algorithms can be easily incorporated into our secured protocol.

In the remainder of this subsection, we first give a formal description of our secured protocol, and then present our oblivious Read/Write algorithm in detail.

Protocol 1 (Secured Spectrum Allocation). *Our secured spectrum allocation protocol Π consists of the following stages (subprotocols):*
$\underline{\Pi_{\mathsf{off}}}$: All the PNs *secret-share their data D as $D_0 + D_1$ (using an additive secret sharing scheme), and send D_0 (resp. D_1) to* SM_0 *(resp.* SM_1*). These two* SMs

then run an S2PC *protocol implementing the functionality* f_{off}*, which denotes all the steps in §3.1 before the request of any* SU $\mathrm{S_i}$ *arrives. Specifically, it includes step (1a) of* **Path Loss Estimation** *and the construction of the grid system used to choose proper subset of* PUs *and SSs for efficient computation. The result is stored in an array data structure* A *for later use. At the end of this stage,* SM_0 *and* SM_1 *get* A_0 *and* A_1 *respectively, which are secret shares of* A *(i.e.* $A_0 + A_1 = A$*). We remark that the task of this stage should be done off-line (before any request of* SU *arrives) to improve efficiency.*

$\underline{\Pi_{\mathsf{slct}}}$*:* *This is the selecting stage to get the subset* J *of indices of array* A*, which indicates the data needed for pass loss estimation.* Π_{slct} *asks* $\mathrm{S_i}$ *to secret-share its location* $\ell_i = \ell_i^0 + \ell_i^1$ *to* SM_0 *and* SM_1*. Then the two* SMs *run an* S2PC *protocol implementing the functionality* $f_{\mathsf{slct}} : (\ell_i^0, \ell_i^1) \to (J_0, J_1)$ *described as follows.* f_{slct} *takes input* (ℓ_i^0, ℓ_i^1) *from* SM_0 *and* SM_1 *respectively. It first recovers* $\ell_i = \ell_i^0 + \ell_i^1$ *and then computes the indices as specified in* **Selection of** PUs **and** SSs *in §3.1, resulting in a set of indices* J*. Then the protocol secret shares* $J = J_0 + J_1$ *to* SM_0 *and* SM_1 *as the output of this stage.*

$\underline{\Pi_{\mathsf{read}}}$*:* SM_0 *and* SM_1 *use* J_0 *and* J_1 *respectively as input to read data from* A*, following our* Secured Array-Entry Read *algorithm (specified later). At the end of this sub-protocol,* $A[j]$ *will be secrete shared as* $A_0''[j] + A_1''[j]$ *for every* $j \in J$*. The output of this stage to* SM_0 *(resp.* SM_1*) is the sequence of secret shares* $\{A_0''[j]\}_{j\in J}$ *(resp.* $\{A_1''[j]\}_{j\in J}$*).*

$\underline{\Pi_{\mathsf{alloc}}}$*:* SM_0 *and* SM_1 *use* $\{A_0''[j]\}_{j\in J}$ *and* $\{A_1''[j]\}_{j\in J}$ *as input to calculate the allocated transit power* t_i*.* t_i *is then secret-shared to* $t_i^0 + t_i^1$*.* SM_0 *(resp.* SM_1*) gets* t_i^0 *(resp.* t_i^1*) as output. This sub-protocol again is implemented via 2PC.*

$\underline{\Pi_{\mathsf{update}}}$*:* SM_0 *and* SM_1 *run an* S2PC *protocol implementing the computation of the new threshold* τ_j *as per Eqn. (2). The results are again secret shared.* SM_0 *holds* $\{\tau_j^0\}_{j\in J}$*, and* SM_1 *holds* $\{\tau_j^1\}_{j\in J}$ *such that* $\tau_j^0 + \tau_j^1 = \tau_j$ *for all* $j \in J$*.*

$\underline{\Pi_{\mathsf{write}}}$*:* SM_0 *and* SM_1 *use* $\{\tau_j^0\}_{j\in J}$ *and* $\{\tau_j^1\}_{j\in J}$ *as input to update data in* $\{A[j]\}_{j\in J}$*. This sub-protocol is implemented as our* Secured Array-Entry Write *algorithm (specified later).*

$\underline{\Pi_{\mathsf{output}}}$*:* SM_0 *(resp.* SM_1*) sends* t_i^0 *(resp.* t_i^1*) as it received in* Π_{alloc} *to* $\mathrm{S_i}$*.* $\mathrm{S_i}$ *recovers* $t_i = t_i^0 + t_i^1$ *as the final output of the main protocol* Π*.* □

Secured Array-Entry Read. Consider an array $A[1..n]$. The secret sharing of array $A[1..n]$ entails that SM_0 and SM_1 store $A_0[1..n]$ and $A_1[1..n]$ respectively with $A_0[i]$ and $A_1[i]$ as two random numbers such that $A_0[i] + A_1[i] = A[i]$. Now, let's say we are given an index j (that has been "computed" in S2PC), and we wish to "load" the entry $A[j]$ into S2PC without either SM learning about either the index j or the entry $A[j]$ being accessed. We use the oblivious transfer (OT) technique [53,23] to implement our solution; the OT techniques allows two parties to exchange information securely.

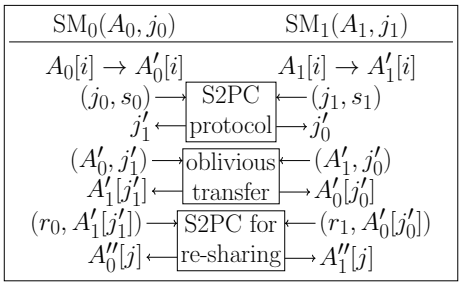

Fig. 3: Array-Entry Read Operation

In particular, if one party has the array and the other party has the index of interest, then OT allows the first party to transfer $A[j]$ to another without either party knowing the other party's input parameter. In our context, the additional challenge is that neither the index j nor the array A is known to either of the parties; these values are shared across the two SMs. We address this challenge by random "shifting" of the indexes and array values at each SM, and engage in S2PC appropriately as described below. Our solution to access an entry $A[j]$ into S2PC securely involves the following steps:

We start with assuming that, from earlier stages in the execution of S2PC, the target index j is shared across the two SMs. Thus, at the beginning of this stage, SM_0 holds j_0 and A_0 as input while SM_1 holds j_1 and A_1 as input; here, j_0 and j_1 are the secret shares of our target index j, and A_0 and A_1 are secret shares of data array A.

1. First, each SM creates new arrays by shifting the indices and entries of the given arrays by fixed random values. More formally, SM_0 and SM_1 create arrays $A'_0[1..n]$ and $A'_1[1..n]$ as:

$$A'_0[i] = A_0[(i + s_0)\%n] + r_0, \quad A'_1[i] = A_1[(i + s_1)\%n] + r_1$$

   where s_0 and r_0 (resp. s_1 and r_1) are random numbers chosen by SM_0 (resp. SM_1).
2. Now, S2PC protocol transfers appropriate indices to the SMs. In particular, SM_0 (resp. SM_1) holding j_0 and s_0 (resp. j_1 and s_1) as input run a S2PC protocol to implement the following functionality f: Upon receiving inputs from SM_0 and SM_1, f recovers $j = j_0 + j_1$ and sends $j'_1 \coloneqq (j + s_1)\%n$ (resp. $j'_0 \coloneqq (j + s_0)\%n$) to SM_0 (resp. SM_1) as the output.
3. Now, the SMs exchange array entries via OT. In particular, SM_0 fetches $A'_1[j'_1]$ from SM_1, and SM_1 fetches $A'_0[j'_0]$ from SM_0.
4. Then SM_0 (resp. SM_1) uses $A'_1[j'_1]$ and r_0 (resp. $A'_0[j'_0]$ and r_1) as input to run a S2PC protocol implementing the following functionality f: Upon receiving inputs from SM_0 and SM_1, f recovers $A[j]$ as

$$A[j] = A'_0[j'_0] + A'_1[j'_1] - r_0 - r_1$$

   and then secret shares the $A[j]$ as $A''_0[j] + A''_1[j]$, and sends $A''_0[j]$ (resp. $A''_1[j]$) to SM_0 (resp. SM_1) as the final output.

Secured Array-Entry Write. Consider an array $A[1..n]$ again as above, where the SM_0 and SM_1 store the secret shares $A_0[1..n]$ and $A_1[1..n]$ respectively of the array. Now, given two private values (secret-shared across the SMs) j and d, we wish to update the array entry $A[j]$ by adding d to it. We achieve the above update of $A[j]$ to $A[j] + d$ in a secured manner by adding `zero` to the remaining entries $A[i]$ (for $i \neq j$). One simple (but inefficient) way to achieve the above is as follows.

– At start, SM_0 (resp. SM_1) holds j_0 and d_0 (resp. j_1 and d_1) as input, where j_0 and j_1 are the secret shares of the target index j while d_0 and d_1 are the secret shares of value d to be added to $A[j]$.

– SM_0 creates an array $D_0[1..n]$ of random and private numbers.
– SM_0 holding D_0 and d_0 as input and SM_1 holding d_1 as input execute a S2PC protocol implementing the following functionality f: Upon receiving input from SMs, f computes the "complement" D_1 of D_0 such that $D_0[i]+D_1[i] = 0$ for $i \neq j$ and $D_0[j] + D_1[j] = d$. f sends D_1 to SM_1 as the output.
– Finally, each SM updates its array as: $A_0[i] = A_0[i] + D_0[i]$ and $A_1[i] = A_1[i] + D_1[i]$ for all i (including j).

The above approach however can be very inefficient due to a large number of operations ($O(n)$ additions) computed in S2PC. To circumvent this, we propose another approach that limits the number of arithmetic operations at S2PC to a small constant while pushing most of the arithmetic operations to the SMs. We achieve this by creating two arrays at each SM, shifting their indexes and exchanging them appropriately. We use the term *shifting* an array $B[1..n]$ by m to mean the operation $B[i] = B[(i + m)\%n]$. For $b \in \{0, 1\}$, our approach works as follows:

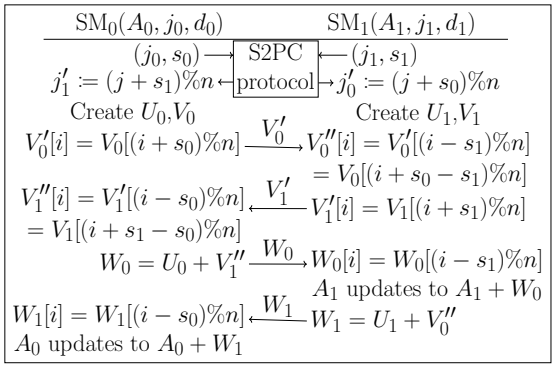

Fig. 4: Array-Entry Write Operation

– *Input.* Same as in the above approach, SM_b holds A_b, j_b and d_b as input.
– *Creating j_b'.* Each SM_b samples a random number s_b. Then each SM_b on input (j_b, s_b) execute a S2PC protocol implementing the following functionality f: Upon receiving SMs' input, f recovers $j = j_0 + j_1$ and sends $j_{1-b}' \coloneqq (j + s_{1-b})\%n$ to SM_b as the output.
– *Updating Arrays U_b and V_b.* Each SM_b creates two arrays U_b and V_b, such that $U_b[i] + V_b[i] = d_b$ for $i = j_{1-b}'$ and 0 otherwise. Here, the idea is that these arrays (after manipulation) will eventually be sent over to the other SM (i.e., SM_{1-b}) who will be able to shift these arrays by s_{1-b}, to get the share of d in the j^{th} index.
– *Manipulation and Exchange of V_b.* Now, each SM_b shifts V_b further by s_b (its private random number) and sends it over to the other SM (i.e., SM_{1-b}). The SM_b on receiving V_{1-b} shifts it by $-s_b$. Thus, each SM_b has V_{1-b} which has been shifted by $s_b + s_{1-b} - s_b = s_{1-b}$.
– *Addition of Local Update Arrays, and Exchange.* Each SM_b now adds the locally available U_b and V_{1-b} (each has a shift of s_{1-b}) to get W_b. At this point, W_0 and W_1 are such that, if we ignore their shifts, their respective entries add up to zero and d. SMs exchange their W_0 and W_1.
– *Final Updating.* Each SM_b now has W_{1-b} (with a shift of j_{1-b}). The array W_{1-b} is finally shifted by $-s_b$ and added to A_b array.

Correctness and Security. The correctness of Protocol 1 is obvious. The security proof is given in Appendix B.

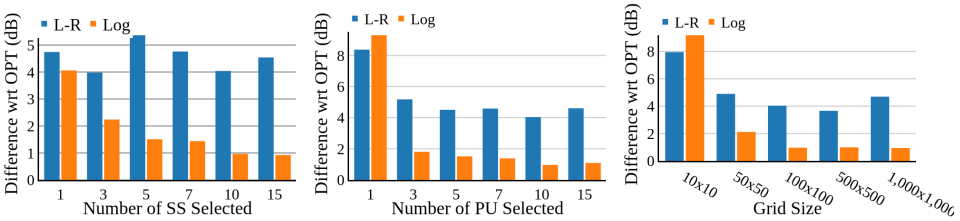

Fig. 5: Average difference in power allocated by Plaintext and Optimal schemes for varying number of (a) selected SSs, (b) selected PUs, and (c) grid size.

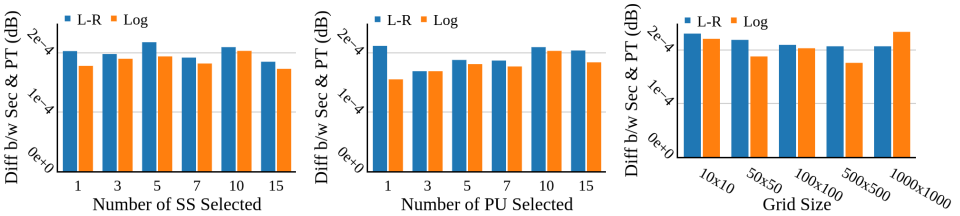

Fig. 6: Average difference in spectral power allocated by secured (i.e., `2-SMs` or `SM-KS`) and Plaintext schemes for varying parameter values.

3.3 Secure Allocation Using One SM and a Key Server

We now modify the secured algorithm from previous subsection to the case of a single SM and a key server. A key server (KS) is a semi-trusted entity in that it can use its persistent storage to only store the cryptographic keys and no other data. We can implement our secured approach over a single SM and a KS, with the following modification to the secret sharing mechanism.

An entity E with the data a_i that it wants to share (in our task, E could be a SS/PU/PUR node with its input) will ask KS for an AES key k_i. E secret shares a_i to $a_{i0} + a_{i1}$. It then sends a_{i0} and $\text{AES}_{k_i}(a_{i1})$ to SM, where $\text{AES}_{k_i}(a_{i1})$ is the AES-encrypted a_{i1} with key k_i. This finishes the secret sharing stage. After all the necessary data is shared in this way to SM, it can run our aforementioned protocol Π with KS playing the role of SM_1. More specifically, SM sends the encrypted shares to KS, who has the corresponding AES keys for decryption. Now we are in the setting where two parties hold the secret shares of input for the spectrum allocation task. They can then run protocol Π as if KS is SM_1.

The above mechanism enables secured two-party computation using S2PC protocol without requiring S2PC to perform any cryptographic operations. Also, it can be used easily to implement the secure read and write operations as described in the previous subsection.

4 Simulation Results

In this section, we evaluate our developed techniques for secured spectrum allocation, by demonstrating its accuracy and computational efficiency.

S2PC Implementation. The core component of our designed algorithm is the use of S2PC protocol to securely compute certain arithmetic operations. To aid our implementation, we use a pre-existing S2PC library Ivory [55]. Ivory provides

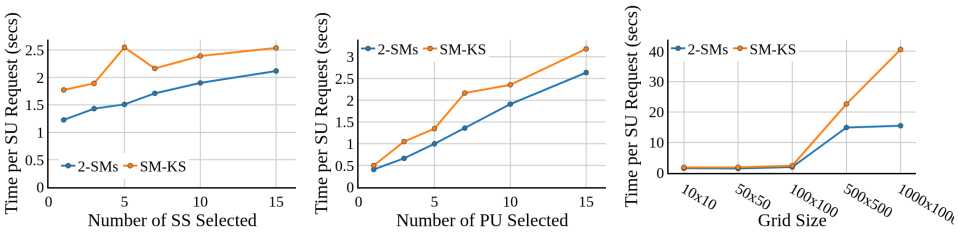

Fig. 7: Time taken by the secured algorithms for varying parameter values.

pre-built circuits for simple operations over integers; these circuits can be used to compute more complex functions using S2PC protocol. Limited by the Ivory library, we use a fixed-point representation for values in S2PC. In particular, we represent a real value v in terms of a standard `int` value x, such that $v = x \times 2^k$ where k is a (positive or negative) *constant* which determines the precision level of the fixed point value. We use 64 bits to represent real values.

Simulation Setup and Parameters. Similar to the settings in the most closely related work [21], we consider a geographic area of 10km × 10km, with 400 PUs and 40000 SSs randomly distributed in the area; we use a large number of spectrum sensors to demonstrate the scalability of our approach in high-density crowdsourced settings. We use 5 PURs for each PU located at 100m around the PU. In each of the plots below, we vary one of the parameter settings while keeping the other parameter settings to their default values. In particular, the default values for various parameter settings are as follows: number of selected PUs: 10, number of selected SSs: 10, grid of 100 × 100. We consider two signal propagation models, viz., log-distance (`Log`) and Longley-Rice (`L-R`) [56], to generate the "ground truth" data, i.e., the sensing reports at the SSs, based on the power and location of each PU. Log-distance (`Log`) model is a simple model, wherein the signal attenuation at a distance d is proportional to d^α where α is the path-loss exponent constant. In contrast, the Longley-Rice (`L-R`) is a complex model of wireless propagation, which takes multiple parameters, such as geolocation of transmitter (TX) and receiver (RX), their antenna configuration, terrain data, weather, and soil condition. In particular, we use the SPLAT! application [1] to generate path losses based on `L-R` model for desired pairs of points.

Accuracy of Plaintext (`PT`) Algorithm vs. Optimal (`OPT`). We start with evaluating the accuracy of our `PT` algorithm (§3.1) with respect to the optimal or "ground truth" (denoted by `OPT`) scheme which allocates maximum spectrum power possible based on the true path-loss values derived directly from the underlying propagation model. Recall that accuracy of `PT` algorithm is affected by three aspects of the algorithm: (i) path-loss estimation error, (ii) selection strategy, which selects only the nearest SSs and PUs, and the (iii) grid-based implementation which approximates a SU's location with the containing grid-cell's center and updates the PUR thresholds of only the PUs that are associated with the containing grid-cell. See Figure 5, which plots the average spectrum difference (in dB) between the spectrum power allocated by the `PT` algorithm and the optimal `OPT`, for varying number of selected SSs and PUs and grid size, for Log-distance and Longley-Rice propagation models. For the Log-distance model, as mentioned in §3.1, the first and the third steps of our path-loss estimation

process are provably 100% accurate if the chosen exponent is the same as that of the underlying model (as is the case in the simulations); thus, the path-loss estimation errors in the Log-distance model are solely due to the second (interpolation) step. In Figure 5, we observe that for the Log-distance (`Log`) model, the difference between the `PT` and `OPT` schemes on average is minimal (1-2 dB) when the number of selected PUs and SSs is 10, and the grid size is at least as 100×100. For the Longley-Rice (`L-R`) model, the average error is about 4-5 dB for similar parameter values; this is largely expected, as the complex `L-R` model depends on various terrain-specific factors and thus is more difficult to estimate accurately compared to the `Log` model which depends solely on distance between points. In summary, a small number (10-15) of SSs and PUs are sufficient to minimize the error, and choosing a larger number of SSs or PUs is not helpful. Also, a 100×100 grid seems fine enough. This justifies our selection strategy and its grid-based implementation, and facilitates computational efficiency of our secured schemes as described below.

Accuracy of Secured vs. Plaintext Algorithms. We now present statistics for the accuracy of our secured schemes (§3.2), as compared to the Plaintext (`PT`) algorithm. Note that the two secured schemes, viz., using two SMs (`2-SMs`) or an SM plus a key server KS (`SM-KS`), allocate the same spectrum power and thus will have the same accuracy—as they differ only in their implementation. In Figure 6, we plot the difference between the spectrum power allocated by our secured schemes and the Plaintext (`PT`) algorithm, for varying number of selected SSs or PUs or grid size, for Log-distance (`Log`) and Longley-Rice (`L-R`) propagation models. Here, the range of the values for number of SSs or PUs selected is partly dictated by the the results in Figure 5 which show that only a small number (10-15) of SSs of PUs are sufficient for minimizing the error of the Plaintext algorithm. Figure 6 shows that across all parameter values of interest the difference between the Secured and Plaintext is near zero dB.

Computation Time. Figure 7 plots computation time taken by the two secured schemes, viz., `2-SMs` (using two SMs) and `SM-KS` (one SM and a key server) for varying grid size, # of selected SSs, and # of selected PUs selected. We use a virtual machine with 48GB ram and 6 virtual CPUs— with each vCPU implemented as a single hardware hyper-thread on a Intel Xeon E5 v3 (Haswell) platform [2]. We observe that the computation time taken by either scheme is of the order of 2-3 seconds, except for grids larger than 500×500 (due to higher grid-table sizes). However, as shown in prior results, a 100×100 grid is fine enough for delivering high accuracy.

Communication Overhead. The communication overhead of our secured schemes was observed to be minimal. In particular, for the optimal parameters of a grid of 100×100 and 10 selected SSs and PUs each, the secured schemes (`2-SMs` as well as `SM-KS`) incur a communication overhead of about 150 KB in computing the arithmetic and access operations. The `SM-KS` scheme incurs an additional communication overhead of 5.2 MB to transfer the grid array. Thus, the total communication overhead for the `2-SMs` scheme is 150 KB, while that for the `SM-KS` scheme is 5.35 MB.

References

1. https://www.qsl.net/kd2bd/splat.html.
2. https://cloud.google.com/compute/docs/cpu-platforms.
3. FlightFeeder for Android, FlightAware. http://flightaware.com/adsb/android/.
4. J. Andrews et al. What will 5G be? *IEEE JSAC*, 2014.
5. T. Araki, A. Barak, J. Furukawa, T. Lichter, Y. Lindell, A. Nof, K. Ohara, A. Watzman, and O. Weinstein. Optimized honest-majority mpc for malicious adversaries-breaking the 1 billion-gate per second barrier. In *Security and Privacy (SP), 2017 IEEE Symposium on*, pages 843–862. IEEE, 2017.
6. T. Araki, J. Furukawa, Y. Lindell, A. Nof, and K. Ohara. High-throughput semi-honest secure three-party computation with an honest majority. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 805–817. ACM, 2016.
7. D. Beaver. Foundations of secure interactive computing. In *Annual International Cryptology Conference*, pages 377–391. Springer, 1991.
8. A. Ben-David, N. Nisan, and B. Pinkas. Fairplaymp: a system for secure multi-party computation. In *Proceedings of the 15th ACM conference on Computer and communications security*, pages 257–266. ACM, 2008.
9. M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 1–10. ACM, 1988.
10. N. Buescher, A. Weber, and S. Katzenbeisser. Towards practical ram based secure computation. In *European Symposium on Research in Computer Security*, pages 416–437. Springer, 2018.
11. R. Calvo-Palomino, D. Giustiniano, V. Lenders, and A. Fakhreddine. Crowdsourcing spectrum data decoding. In *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, 2017.
12. R. Canetti. Security and composition of multiparty cryptographic protocols. *Journal of CRYPTOLOGY*, 13(1):143–202, 2000.
13. A. Chakraborty, M. S. Rahman, H. Gupta, and S. R. Das. Specsense: Crowdsensing for efficient querying of spectrum occupancy. In *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, 2017.
14. D. Chaum, C. Crépeau, and I. Damgard. Multiparty unconditionally secure protocols. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 11–19. ACM, 1988.
15. B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private information retrieval. In *Foundations of Computer Science, 1995. Proceedings., 36th Annual Symposium on*, pages 41–50. IEEE, 1995.
16. M. A. Clark and K. Psounis. Trading utility for privacy in shared spectrum access systems. *IEEE/ACM Transactions on Networking*, 2017.
17. I. Damgård, V. Pastro, N. Smart, and S. Zakarias. Multiparty computation from somewhat homomorphic encryption. In *Advances in Cryptology–CRYPTO 2012*, pages 643–662. Springer, 2012.
18. D. Demmler, T. Schneider, and M. Zohner. Aby-a framework for efficient mixed-protocol secure two-party computation. In *NDSS*, 2015.
19. G. Ding, F. Song, Q. Wu, Y. Zou, L. Zhang, S. Feng, and J. Wang. Robust spectrum sensing with crowd sensors. In *IEEE VTC*, 2014.

20. J. Doerner and A. Shelat. Scaling oram for secure computation. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 523–535. ACM, 2017.
21. Y. Dou, K. C. 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*, 35(1):173–187, 2017.
22. E. Drocella, J. Richards, R. Sole, F. Najmy, A. Lundy, and P. McKenna. 3.5 GHz exclusion zone analyses and methodology. *Tech. Rep.*, 2015.
23. S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. *Communications of the ACM*, 28(6):637–647, 1985.
24. B. Fan, D. G. Andersen, M. Kaminsky, and M. D. Mitzenmacher. Cuckoo filter: Practically better than bloom. In *Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies*, pages 75–88. ACM, 2014.
25. U. FCC. Longley-rice methodology for evaluating TV coverage and interference. *OET Bulletin*, 69, 2004.
26. J. Furukawa, Y. Lindell, A. Nof, and O. Weinstein. High-throughput secure three-party computation for malicious adversaries and an honest majority. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 225–255. Springer, 2017.
27. 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.
28. C. Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing*, pages 169–178. ACM, 2009.
29. O. Goldreich. *Foundations of Cryptography: Basic Tools*. Cambridge University Press, New York, NY, USA, 2000.
30. O. Goldreich. *Foundations of cryptography: volume 2, basic applications*. Cambridge university press, 2009.
31. O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 218–229. ACM, 1987.
32. S. Goldwasser and L. Levin. Fair computation of general functions in presence of immoral majority. In *Conference on the Theory and Application of Cryptography*, pages 77–93. Springer, 1990.
33. S. D. Gordon, J. Katz, V. Kolesnikov, F. Krell, T. Malkin, M. Raykova, and Y. Vahlis. Secure two-party computation in sublinear (amortized) time. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 513–524. ACM, 2012.
34. M. Grissa, B. Hamdaoui, and A. A. Yavuz. Location privacy in cognitive radio networks: A survey. *IEEE Communications Surveys & Tutorials*, 2017.
35. M. Grissa, B. Hamdaoui, and A. A. Yavuza. Location privacy in cognitive radio networks: A survey. *IEEE Communications Surveys Tutorials*, 19(3), 2017.
36. M. Grissa, A. Yavuz, and B. Hamdaoui. Lpos: Location privacy for optimal sensing in cognitive radio networks. In *Global Communications Conference (GLOBECOM), 2015 IEEE*, pages 1–6. IEEE, 2015.
37. M. Grissa, A. Yavuz, and B. Hamdaoui. An efficient technique for protecting location privacy of cooperative spectrum sensing users. In *Computer Communications Workshops (INFOCOM WKSHPS), 2016 IEEE Conference on*, pages 915–920. IEEE, 2016.

38. M. Grissa, A. A. Yavuz, and B. Hamdaoui. Preserving the location privacy of secondary users in cooperative spectrum sensing. *IEEE Transactions on Information Forensics and Security*, 12(2):418–431, 2017.
39. A. T. Hoang, Y. Liang, and M. H. Islam. Power control and channel allocation in cognitive radio networks with primary users' cooperation. *IEEE Transactions on Mobile Computing*, 9, 2010.
40. P. Ishwar, A. Kumar, and K. Ramchandran. Distributed sampling for dense sensor networks: a bit-conservation principle. In *Information Processing in Sensor Networks*, pages 17–31. Springer, 2003.
41. X. Jin, R. Zhang, Y. Chen, T. Li, and Y. Zhang. DPSense: Differentially private crowdsourced spectrum sensing. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 296–307. ACM, 2016.
42. B. Kasiri, I. Lambadaris, F. R. Yu, and H. Tang. Privacy-preserving distributed cooperative spectrum sensing in multi-channel cognitive radio manets. In *Communications (ICC), 2015 IEEE International Conference on*, pages 7316–7321. IEEE, 2015.
43. V. Kolesnikov and T. Schneider. Improved garbled circuit: Free XOR gates and applications. In *Proceedings of 35th International Colloquium on Automata, Languages and Programming*, pages 486–498, 2008.
44. B. Kreuter, A. Shelat, and C.-H. Shen. Billion-gate secure computation with malicious adversaries. In *USENIX Security Symposium*, volume 12, pages 285–300, 2012.
45. H. Li, Q. Pei, and W. Zhang. Location privacy-preserving channel allocation scheme in cognitive radio networks. *International Journal of Distributed Sensor Networks*, 12(7):3794582, 2016.
46. S. Li, H. Zhu, Z. Gao, X. Guan, K. Xing, and X. Shen. Location privacy preservation in collaborative spectrum sensing. In *INFOCOM, 2012 Proceedings IEEE*, pages 729–737. IEEE, 2012.
47. Y. Mao, T. Chen, Y. Zhang, T. Wang, and S. Zhong. Protecting location information in collaborative sensing of cognitive radio networks. In *Proceedings of the 18th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pages 219–226. ACM, 2015.
48. A. Medeisis and A. Kajackas. On the use of the universal Okumura-Hata propagation prediction model in rural areas. In *Vehicular Technology Conference Proceedings, 2000. VTC 2000-Spring Tokyo. 2000 IEEE 51st*, volume 3, pages 1815–1818. IEEE, 2000.
49. S. Micali and P. Rogaway. Secure computation. In *Annual International Cryptology Conference*, pages 392–404. Springer, 1991.
50. P. Mohassel, M. Rosulek, and Y. Zhang. Fast and secure three-party computation: The garbled circuit approach. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 591–602. ACM, 2015.
51. R. Ostrovsky and V. Shoup. Private information storage. In *STOC*, volume 97, pages 294–303. Citeseer, 1997.
52. P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 223–238. Springer, 1999.
53. M. O. Rabin. How to exchange secrets with oblivious transfer. *IACR Cryptology ePrint Archive*, 2005:187, 2005.
54. N. Rajkarnikar, J. M. Peha, and A. Aguiar. Location privacy from dummy devices in database-coordinated spectrum sharing. In *Dynamic Spectrum Access Networks (DySPAN), 2017 IEEE International Symposium on*, pages 1–10. IEEE, 2017.

55. P. Rindal. Ivory. https://github.com/ladnir/Ivory-Runtime, 2018.
56. J. Seybold. *Introduction to RF Propagation*. Wiley, 2005.
57. E. Stefanov, M. Van Dijk, E. Shi, C. Fletcher, L. Ren, X. Yu, and S. Devadas. Path oram: an extremely simple oblivious ram protocol. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 299–310. ACM, 2013.
58. L. Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570, 2002.
59. E. Z. Tragos, S. Zeadally, A. G. Fragkiadakis, and V. A. Siris. Spectrum assignment in cognitive radio networks: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 15(3):1108–1135, 2013.
60. W. Wang and Q. Zhang. Privacy-preserving collaborative spectrum sensing with multiple service providers. *IEEE Transactions on Wireless Communications*, 14(2), 2015.
61. X. Wang, H. Chan, and E. Shi. Circuit oram: On tightness of the goldreich-ostrovsky lower bound. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 850–861. ACM, 2015.
62. X. S. Wang, Y. Huang, T. H. Chan, A. Shelat, and E. Shi. Scoram: oblivious ram for secure computation. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 191–202. ACM, 2014.
63. A. C.-C. Yao. How to generate and exchange secrets. In *Foundations of Computer Science, 1986., 27th Annual Symposium on*, pages 162–167. IEEE, 1986.
64. X. Ying, C. W. Kim, and S. Roy. Revisiting tv coverage estimation with measurement-based statistical interpolation. In *International Conference on Communication Systems and Networks (COMSNETS)*, 2015.
65. S. Zahur, X. Wang, M. Raykova, A. Gascón, J. Doerner, D. Evans, and J. Katz. Revisiting square-root oram: efficient random access in multi-party computation. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 218–234. IEEE, 2016.
66. L. Zhang, C. Fang, Y. Li, H. Zhu, and M. Dong. Optimal strategies for defending location inference attack in database-driven crns. In *Communications (ICC), 2015 IEEE International Conference on*, pages 7640–7645. IEEE, 2015.
67. T. Zhang, N. Leng, and S. Banerjee. A vehicle-based measurement framework for enhancing whitespace spectrum databases. In *Proc. ACM Mobicom*, 2014.

A Definition of MPC

The notion of secure multi-party computation (MPC) was first developed in [63,31,9,14] who also established initial feasibility results. It allows a set of mutually distrusting parties to compute a joint function over their private inputs so that each party learns nothing beyond its intended output. The efficiency of MPC protocols, particularly for the case of two and three parties, has seen tremendous improvements over the past few years, reaching the blazing-fast rates of up to a billion gates per second [8,17,44,18,50,6,26,5]. Our work takes advantage of this newfound efficiency of S2PC protocols, and benefits directly from further improvements to their efficiency.

We remark that we choose the MPC framework as opposed to the simpler *two party* framework even though the actual computation in our framework is

performed by two parties (either two SMs or a SM and a KS). This choice is important since before the two-party computation kicks in, other parties in the system are involved in security sensitive pre-processing and post-processing stages. The MPC framework captures all such activities and enables us to construct proofs for all parties in the system.

In the remainder of this section, we show the definition of MPC. We assume familiarity with standard cryptographic concepts such as Turing machines, probability ensembles, computational indistinguishability etc., and refer the reader to [29,30] for a detailed treatment.

The Multi-Party Functionality. In the n-party computation problem, there are n participants (parties). The computation task is cast by specifying a *functionality* and denote it as a function mapping n inputs to n outputs, namely, $f : (\{0,1\}^*)^n \rightarrow (\{0,1\}^*)^n$. We denote the input as a vector $\bar{x} = (x_1, \ldots, x_n)$. On input $\bar{x}$, $f_i(\bar{x})$ denotes the i-th output f. We say a n-party protocol Π computes f if, for $i \in \{1, \ldots, n\}$, party P_i with input x_i learns $f_i(\bar{x})$ as the result of the execution of Π.

The Ideal/Real Paradigm. Given a multi-party protocol Π that implements a functionality f, we define its security via the *simulation paradigm* [31,32,49,7,12]. Imagine an "ideal-world" execution, where all each party simply sends its input to the functionality f, and then receives the correct output. In this execution, there is no interactions between any pair of parties, and each party learns nothing more than its input and output (and the information that can be inferred). So this ideal-world execution is considered as the most secure scenario that one can hope for. Based on this mental experiment, we thus believe Π is secure if any adversary Adv (participating in a real execution of Π) can do no more harm than in an ideal-world execution of f. This intuition is formalized by consider a simulator in the ideal-world execution. If any information learned by Adv can be generated (or "simulated") by a *simulator* from the input and output of the parties corrupted by Adv, then Adv gains nothing in the real-world execution, thus the protocol is secure.

The Adversarial Model. We focus on *semi-honest model with static corruption*. By *semi-honest*, we mean that all the parties/entities (including the *corrupted* ones) follow the protocol; however, a corrupted party may attempt to infer private data of other parties from the information obtained during the protocol operation. By *static corruption*, we mean that the set of corrupted parties remains fixed throughout the execution of the protocol. We also assume authenticated communication channels between every pair of parties.

The correctness and security of an n-party protocol Π implementing a (deterministic) n-party functionality f is formalized as follows:

Definition 3 (Correctness [30]). *We say that Π correctly computes f if the following holds except for negligible probability: for any input vector $\bar{x}$, at the end of the execution of Π, party P_i gets $f_i(\bar{x})$ as its output.*

Definition 4 (Semi-Honest Security [30]). *For $I = \{i_1, \ldots, i_t\} \subseteq [n]$, let $f_I(\bar{x}) \coloneqq \big(f_{i_1}(\bar{x}), \ldots, f_{i_t}(\bar{x})\big)$, and let $\mathsf{view}_I^{\Pi}(\bar{x}) \coloneqq \big(I, \mathsf{view}_{i_1}^{\Pi}(\bar{x}), \ldots, \mathsf{view}_{i_t}^{\Pi}(\bar{x})\big)$. We say that protocol Π* securely computes *f under the semi-honest model with static corruption if there exists a probabilistic polynomial-time algorithm $\mathcal{S}$ (the simulator) such that for every I as above,*

$$\big\{\mathcal{S}\big(I, (x_{i_1}, \ldots, x_{i_t}), f_I(\bar{x})\big)\big\}_{\bar{x}\in(\{0,1\}^*)^n} \overset{c}{\equiv} \big\{\mathsf{view}_I^{\Pi}(\bar{x})\big\}_{\bar{x}\in(\{0,1\}^*)^n} \tag{9}$$

where $\overset{c}{\equiv}$ denotes computational indistinguishability [29], and $\mathsf{view}_i^{\Pi}(\bar{x})$ denotes the view of i-th party ($i \in [n]$) consisting of its input, random tape and received messages during the execution of Π on input $\bar{x}$.

B Security Proof

In this section, we give a proof of the following theorem, which shows the security of our protocol. After that, we also discuss how to extend the security proof to the scenario where one SM is replaced with a KS.

Theorem 1 (Security of Protocol 1). *Protocol 1 is a secure multi-party computation implementation of the plaintext algorithm shown in §3.1 with respect to semi-honest adversaries which do not corrupt* SM_0 *and* SM_1 *at the same time.*

B.1 Proof of Theorem 1 for Two-SM Setting

The details of our protocol Π are already given in Protocol 1. Here we summarize and illustrate its structure in Fig. 8 to ease our presentation.

According to Eqn. (4) in the security definition, we need to show a simulator for different combinations of views for all possible subset $I \subseteq \{\mathrm{S_i}, \mathrm{SM}_0, \mathrm{SM}_1, \mathrm{PNs}\}$ such that I does not contain SM_0 and SM_1 at the same time (Recall that we assume they do not collude). For our specific protocol Π, we claim that it will be sufficient if we can construct a simulator for each party separately (which is not necessarily true for general MPC protocols). To see why, recall that both $\mathrm{S_i}$ (except for its final output t_i) and PNs receive no message during the execution of Π. Simulators for them can be constructed in a "dummy" way by just outputting the input/output of $\mathrm{S_i}$ and PNs. So the essential part of Π is actually a S2PC protocol between SM_0 and SM_1. To prove the security of a S2PC protocol, it suffices to show two separate simulators for each participant. Also, it is not hard to verify that once SM_0 and SM_1 are not corrupted at the same time, the simulator for a spectrum manager can be composed with the aforementioned "dummy" simulators of $\mathrm{S_i}$ and PNs arbitrarily, to get a whole simulator for any corrupted set I that goes through the security proof. Therefore, we will conduct the security proof by constructing simulators for different parties in our protocol separately. In our protocol Π, we use $\mathcal{S}^{\mathsf{pty}}$ to denote the simulator for a party $\mathsf{pty} \in \{\mathrm{SU}, \mathrm{SM}_0, \mathrm{SM}_1, \mathrm{PNs}\}$.

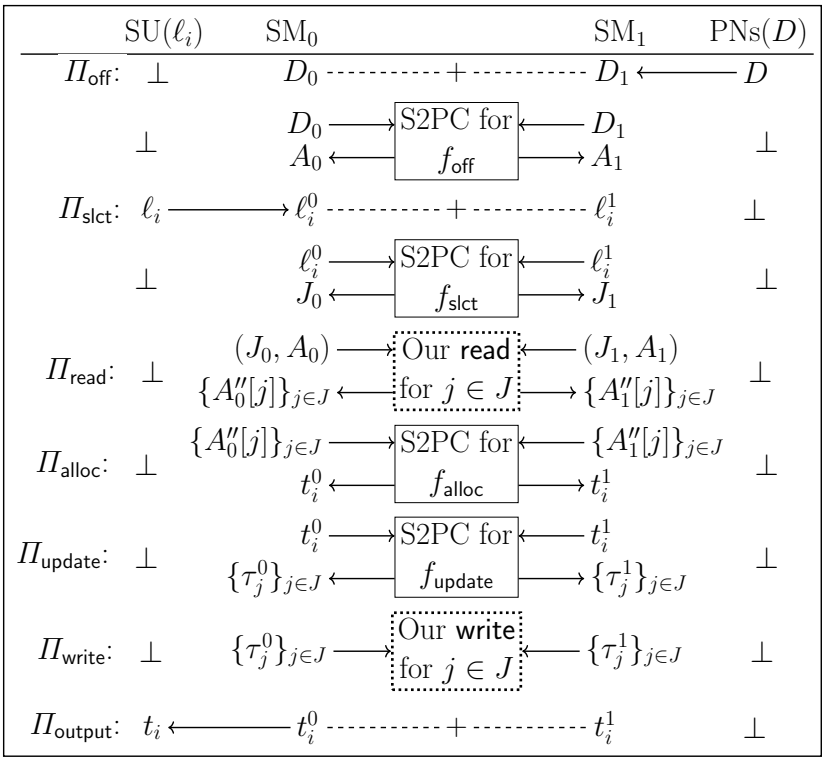

Fig. 8: Sub-protocols of Our Protocol Π

Simulator for SU**.** It is straightforward to simulate the view of requesting SU $\mathrm{S_i}$. Because all it does in protocol Π are conducting a secret sharing of its location ℓ_i to SMs, and receiving the final output t_i (as illustrated in Figure 8). So given ℓ_i and t_i, a simulator $\mathcal{S}^{\mathrm{SU}}$ can successfully simulate $\mathrm{S_i}$'s view by outputting a secret sharing of t_i. We remark that only messages *received by* (in contrast to "sent from") a party appear in its view. Since in secret sharing stage SU only sends out message, its view only contains t_i^0 and t_i^1 such that $t_i^0 + t_i^1 = t_i$ (besides its input and random tape). So the output of $\mathcal{S}^{\mathrm{SU}}(\ell_i, t_i)$ is identically distributed to the view of SU in a real execution.

Simulator for PNs**.** All the work PNs do in protocol Π is to secret share D to SMs. PNs receive no messages during the execution of Π. $\mathcal{S}^{\mathrm{PNs}}$ can easily finish his job by outputting D with the random tapes of PNs. This output is also identically distributed with the real-execution view.

Simulator for SMs**.** Since the roles of the two SMs are symmetric in Π, it suffices to show a simulator for SM_0 (or SM_1). At a high level, we will construct simulators for SM_0 in each sub-protocols of Π, which, if pulled together, will give us the final simulator for SM_0 in Π.

For $\mathsf{X} \in \{\mathsf{off}, \mathsf{slct}, \mathsf{read}, \mathsf{alloc}, \mathsf{update}, \mathsf{write}, \mathsf{output}\}$, denote $\mathcal{S}_{\mathsf{X}}^{\mathrm{SM}_0}$ as the simulator which can generate a view of SM_0 in sub-protocol Π_{X} on corresponding input and output. First, notice that $\mathcal{S}_{\mathsf{slct}}^{\mathrm{SM}_0}$, $\mathcal{S}_{\mathsf{alloc}}^{\mathrm{SM}_0}$ and $\mathcal{S}_{\mathsf{update}}^{\mathrm{SM}_0}$ must exist, because their corresponding sub-protocols are implemented via 2PC. Also, it is straightforward to see that $\mathcal{S}_{\mathsf{off}}^{\mathrm{SM}_0}$ and $\mathcal{S}_{\mathsf{output}}^{\mathrm{SM}_0}$ exist because participants of these two sub-protocols only conduct secret sharing. The existence of $\mathcal{S}_{\mathsf{read}}^{\mathrm{SM}_0}$ and $\mathcal{S}_{\mathsf{write}}^{\mathrm{SM}_0}$ for corresponding sub-protocols (shown as dashed box in Figure 8) is waiting to be shown. However, we will defer the proof to §B.2. Here we assume their existences and show how to construct simulator $\mathcal{S}_{\Pi}^{\mathrm{SM}_0}$ for Π based on these sub-protocol simulators.

The Full Simulator. $\mathcal{S}_{\Pi}^{\mathrm{SM}_0}$ runs these sub-protocols simulators in order, from $\mathcal{S}_{\mathsf{off}}^{\mathrm{SM}_0}$ to $\mathcal{S}_{\mathsf{output}}^{\mathrm{SM}_0}$. It always sets the simulated output of last sub-protocol as the input to next simulator, except for $\mathcal{S}_{\mathsf{slct}}^{\mathrm{SM}_0}$ and $\mathcal{S}_{\mathsf{output}}^{\mathrm{SM}_0}$ (see the following). For Π_{slct}, SM_0's input in Π_{slct} is from the secret sharing of ℓ_i, instead of the output of last stage (see Figure 8). In this case, $\mathcal{S}_{\Pi}^{\mathrm{SM}_0}$ will just input a random string to $\mathcal{S}_{\mathsf{slct}}^{\mathrm{SM}_0}$ to continue the simulation; for Π_{output}, SM_0 takes no input. Finally, $\mathcal{S}_{\Pi}^{\mathrm{SM}_0}$ combines and outputs (in order) the simulated view of all the sub-protocols as its output.

Proof for Indistinguishability. We then prove that the output of $\mathcal{S}_{\Pi}^{\mathrm{SM}_0}$ is indistinguishable from the view of SM_0 in a real execution of Π by hybrid argument. For X in the ordered list of stages $\{\mathsf{off}, \mathsf{slct}, \mathsf{read}, \mathsf{alloc}, \mathsf{update}, \mathsf{write}, \mathsf{output}\}$, denote $\mathsf{Hyb}_{\mathrm{SM}_0}^{\mathsf{X}}$ as the execution (also called "hybrid") of Π, where the simulator $\mathcal{S}_{\mathrm{SM}_0}^{\Pi}$ takes control until stage Π_{X} (included). The remaining stages are just executed as in the real protocol. By definition, we have $\mathsf{Hyb}_{\mathrm{SM}_0}^{\mathsf{output}} = \mathcal{S}_{\Pi}^{\mathrm{SM}_0}$.

First, notice that the view in real execution $\mathsf{view}_{\mathrm{SM}_0}^{\Pi}$ is identically distributed as $\mathsf{Hyb}_{\mathrm{SM}_0}^{\mathsf{off}}$. Because the only difference between them is that the view of SM_0 in the real execution of Π_{off} is replaced by the simulation results from $\mathcal{S}_{\mathsf{off}}^{\mathrm{SM}_0}$, and this simulation is perfect.

To show the indistinguishability between $\mathsf{Hyb}_{\mathrm{SM}_0}^{\mathsf{off}}$ and $\mathsf{Hyb}_{\mathrm{SM}_0}^{\mathsf{slct}}$, we introduce an intermediate hybrid $\mathsf{Hyb}_{\mathrm{SM}_0}^{\mathsf{off\text{-}slct}}$, which is the same as $\mathsf{Hyb}_{\mathrm{SM}_0}^{\mathsf{slct}}$ except that the input to SM_0 at the beginning of Π_{slct} is the real secret share ℓ_i^0. The only difference between $\mathsf{Hyb}_{\mathrm{SM}_0}^{\mathsf{off}}$ and $\mathsf{Hyb}_{\mathrm{SM}_0}^{\mathsf{off\text{-}slct}}$ is that Π_{slct} is executed in the former, but simulated by $\mathcal{S}_{\mathsf{slct}}^{\mathrm{SM}_0}$ in the latter (on the same "real" input ℓ_i^0). By the security of $\mathcal{S}_{\mathsf{slct}}^{\mathrm{SM}_0}$, we have $\mathsf{Hyb}_{\mathrm{SM}_0}^{\mathsf{off}} \overset{\mathrm{c}}{\equiv} \mathsf{Hyb}_{\mathrm{SM}_0}^{\mathsf{off\text{-}slct}}$. Also, it can be seen that $\mathsf{Hyb}_{\mathrm{SM}_0}^{\mathsf{off\text{-}slct}} \overset{\mathrm{id}}{\equiv} \mathsf{Hyb}_{\mathrm{SM}_0}^{\mathsf{slct}}$, where $\overset{\mathrm{id}}{\equiv}$ denotes *identical distribution* [29]. That is because the secret share ℓ_i^0 (in $\mathsf{Hyb}_{\mathrm{SM}_0}^{\mathsf{off\text{-}slct}}$) is identically distributed with a random string (in $\mathsf{Hyb}_{\mathrm{SM}_0}^{\mathsf{slct}}$). And since this is the only difference between these two hbyrids, we know that they are identical. In summary, we proved $\mathsf{Hyb}_{\mathrm{SM}_0}^{\mathsf{off}} \overset{\mathrm{c}}{\equiv} \mathsf{Hyb}_{\mathrm{SM}_0}^{\mathsf{off\text{-}slct}} \overset{\mathrm{id}}{\equiv} \mathsf{Hyb}_{\mathrm{SM}_0}^{\mathsf{slct}}$.

The indistinguishability of the remaining hybrids follows straightforwardly from the security of corresponding sub-protocol simulators, with the only excep-

tion of $\mathsf{Hyb}^{\mathsf{output}}_{\mathrm{SM}_0}$. But $\mathsf{Hyb}^{\mathsf{write}}_{\mathrm{SM}_0} \overset{\mathrm{id}}{\equiv} \mathsf{Hyb}^{\mathsf{output}}_{\mathrm{SM}_0}$ can also be shown following a similar argument as we did for $\mathsf{view}^{\Pi}_{\mathrm{SM}_0} \overset{\mathrm{id}}{\equiv} \mathsf{Hyb}^{\mathsf{off}}_{\mathrm{SM}_0}$.

In summary, we have the following relations, which proves $\mathcal{S}^{\mathrm{SM}_0}_{\Pi}$ successfully simulates the view of SM_0.

$$\mathsf{view}^{\Pi}_{\mathrm{SM}_0} \overset{\mathrm{id}}{\equiv} \mathsf{Hyb}^{\mathsf{off}}_{\mathrm{SM}_0} \overset{\mathrm{c}}{\equiv} \mathsf{Hyb}^{\mathsf{slct}}_{\mathrm{SM}_0} \overset{\mathrm{c}}{\equiv} \mathsf{Hyb}^{\mathsf{read}}_{\mathrm{SM}_0} \overset{\mathrm{c}}{\equiv} \mathsf{Hyb}^{\mathsf{alloc}}_{\mathrm{SM}_0} \overset{\mathrm{c}}{\equiv} \mathsf{Hyb}^{\mathsf{update}}_{\mathrm{SM}_0} \overset{\mathrm{c}}{\equiv} \mathsf{Hyb}^{\mathsf{write}}_{\mathrm{SM}_0} \overset{\mathrm{id}}{\equiv} \mathsf{Hyb}^{\mathsf{output}}_{\mathrm{SM}_0} = \mathcal{S}^{\mathrm{SM}_0}_{\Pi}$$

B.2 Simulators for Read and Write Sub-Protocols

Recall that the proof in §B.1 is based on our assumption that there exist simulators for our read and write protocol (namely, they are secure S2PC protocols). In this section, we prove the correctness of this assumption by constructing valid simulators for them. We remark that this will finally complete the security proof for our protocol Π.

Simulator for SM_0 in Π_{read}. In practice, Π_{read} is executed for $|J|$ times until all $\{A[j]\}_{j\in J}$ are retrieved. But it suffices to show a simulator for a single read operation (without loss of generality, the j-th one). As discussed in §3.2, the input of SM_0 is a secret share j_1, the output is $A''_0[j]$. To do the simulation, $\mathcal{S}^{\mathsf{read}}_{\mathrm{SM}_0}$ first samples randomness r_0, s_0 and set $A'_0[i] = A_0[(i+s_0)\%n] + r_0$ by himself. The remaining stages consist of a S2PC protocol to exchange j'_0 and j'_1 values, two OT protocols and a S2PC protocol to secret share $A[j] = A''_0[j] + A''_1[j]$. $\mathcal{S}^{\mathsf{read}}_{\mathrm{SM}_0}$ can finish the simulation for them simply by calling the corresponding simulators. The only place we need to be careful is the input to the S2PC protocol which implements the secret re-sharing of $A[j]$. In the real execution, SM_0 holds as input $(r_0, A'_1[j'_1])$. When $\mathcal{S}^{\mathsf{read}}_{\mathrm{SM}_0}$ replace the S2PC protocol with its simulator, it has no way to get the correct $A'_1[j'_1]$. But note that $A'_1[j'_1]$ is uniformly random. So we can just sample a random string to play the role of $A'_1[j'_1]$, helping us to finish the simulation.

Simulator for SM_0 in Π_{write}. In §3.2, we describe two versions of Π_{write}. The first one is simpler but inefficient. We call the it the *plain version*. We call the second one *efficient version*. For the plain version, the simulator can be constructed in a similar way as for $\mathcal{S}^{\mathsf{read}}_{\mathrm{SM}_0}$, which we need not repeat here. Next, we describe $\mathcal{S}^{\mathsf{write}}_{\mathrm{SM}_0}$ for the efficient version. (Similar as for simulating Π_{read}, we show a simulator for a single write operation).

The inputs for SM_0 consist of the secret share j_0 of the target index and the secret share τ_j of new threshold. SM_0 on input j_0 first runs a S2PC protocol with SM_1, after which SM_0 gets $j'_1 = (j+s_1)\%n$ (see §3.2 for details). $\mathcal{S}^{\mathsf{write}}_{\mathrm{SM}_0}$ can just call the simulator for this S2PC protocol to go through this stage. In the remaining stages, SM_0 receives two messages from SM_1: V_1 and W_1. Since both of them are secret shares created by SM_1, they look random to SM_0. So $\mathcal{S}^{\mathsf{write}}_{\mathrm{SM}_0}$ can replace them by sampling two random strings (a formal proof for the security of this replacement involves a standard reduction to the security of secret sharing scheme, which we omit here), which finishes the simulation successfully.

B.3 Replace SM_1 with a Key Server.

In §3.3, we discussed a variant of the protocol where SM_1 is replaced by a key server. To give a proof for that case, it suffices to show that the secret sharing mechanism happened among E, SM_0 and KS constitutes a secure 3-party computation (3PC) protocol, whose simulator is denoted as $\mathcal{S}_{3PC}$. This is because that our proof for the `2-SMs` setting is presented in a modular way. We can construct a simulator for this SM-KS setting by combining $\mathcal{S}_{3PC}$ with the simulator $\mathcal{S}^{\Pi}$ we constructed for `2-SMs` setting. (Of course, other parts of $\mathcal{S}^{\Pi}$ should be modified accordingly, but in a straightforward manner. Also, all the tasks of SM_1 now will be conducted by KS). Next, similar as we argued for the `2-SMs` scenario, we will prove the security of the 3PC protocol by showing separate simulators for each participants.

Simulator for KS**.** In the 3PC protocol, KS receives no output. All the work it does is to issue AES keys when E requests. So $\mathcal{S}^{KS}_{3PC}$ can be constructed trivially.

Simulator for E**.** E receives AES key K_i from KS. To construct the simulator $\mathcal{S}^{E}_{3PC}$ for him, we can sample a key from the AES key distribution and output it as K_i to E. Since both samplings conducted by $\mathcal{S}^{E}_{3PC}$ and KS are from the same distribution, this simulation is perfectly.

Simulator for SM_0**.** SM_0 receives a_{i0} and $\mathrm{AES}_{K_i}(a_{i1})$ in this 3PC protocol. Intuitively, a_{i1} looks uniform since it is a share of the secret sharing of a_i. It can be simulated easily by pick a uniform string r such that $|r| = |a_{i0}|$. To simulate $\mathrm{AES}_{K_i}(a_{i1})$, we can use $\mathrm{AES}_{K_i}(0^{|a_{i1}|})$, whose indistinguishability from $\mathrm{AES}_{K_i}(a_{i1})$ is guaranteed by the semantic security of AES scheme. A formal proof requires some caution to deal with the fact that a_{i0} and a_{i1} are not independent (they satisfy the constraint $a_{i0} + a_{i1} = a_i$). But this can be done by standard hybrid argument conducted in the following order:

$$\begin{aligned}&\big(a_i, a_{i0}, \mathrm{AES}_{k_i}(a_{i1})\big)\\ &\quad\overset{c}{\equiv}\big(a_i, a_{i0}, \mathrm{AES}_{k_i}(0^{|a_{i1}|})\big) \overset{c}{\equiv} \big(a_i, r, \mathrm{AES}_{k_i}(0^{|a_{i1}|})\big)\end{aligned}$$

which implies $\big(a_{i0}, \mathrm{AES}_{k_i}(a_{i1})\big) \overset{c}{\equiv} \big(r, \mathrm{AES}_{k_i}(0^{|a_{i1}|})\big)$ by simply hiding a_i from both sides.