Preserving the Location Privacy of Secondary Users in Cooperative Spectrum Sensing

Mohamed Grissa, Student Member, IEEE, Attila A. Yavuz, Member, IEEE, and Bechir Hamdaoui, Senior Member, IEEE

Abstract—Cooperative spectrum sensing, despite its effectiveness in enabling dynamic spectrum access, suffers from location privacy threats, merely because Secondary Users (SUs)' sensing reports that need to be shared with a fusion center to make spectrum availability decisions are highly correlated to the users' locations. It is therefore important that cooperative spectrum sensing schemes be empowered with privacy preserving capabilities so as to provide SUs with incentives for participating in the sensing task. In this paper, we propose privacy preserving protocols that make use of various cryptographic mechanisms to preserve the location privacy of SUs while performing reliable and efficient spectrum sensing. We also present cost-performance tradeoffs. The first consists on using an additional architectural entity at the benefit of incurring lower computation overhead by relying only on symmetric cryptography. The second consists on using an additional secure comparison protocol at the benefit of incurring lesser architectural cost by not requiring extra entities. Our schemes can also adapt to the case of a malicious Fusion Center (FC) as we discuss in this paper. We also show that not only are our proposed schemes secure and more efficient than existing alternatives, but also achieve fault tolerance and are robust against sporadic network topological changes.

Index Terms—Location privacy, secure cooperative spectrum sensing, order preserving encryption, cognitive radio networks.

I. INTRODUCTION

Cooperative spectrum sensing is a key component of cognitive radio networks (CRNs) essential for enabling dynamic and opportunistic spectrum access [1]–[3]. It consists of having secondary users (SUs) sense the licensed channels on a regular basis and collaboratively decide whether a channel is available prior to using it so as to avoid harming primary users (PUs). One of the most popular spectrum sensing techniques is energy detection, thanks to its simplicity and ease of implementation, which essentially detects the presence of PU’s signal by measuring and relying on the energy strength of the sensed signal, commonly known as the received signal strength (RSS) [4].

Broadly speaking, cooperative spectrum sensing techniques can be classified into two categories: centralized and distributed [1]. In centralized techniques, a central entity called fusion center (FC) orchestrates the sensing operations as follows. It selects one channel for sensing and, through a control channel, requests that each SU perform local sensing on that channel and send its sensing report (e.g., the observed RSS value) back to it. It then combines the received sensing reports, makes a decision about the channel availability, and diffuses the decision back to the SUs. In distributed sensing techniques, SUs do not rely on a FC for making channel availability decisions. They instead exchange sensing information among one another to come to a unified decision [1].

Despite its usefulness and effectiveness in promoting dynamic spectrum access, cooperative spectrum sensing suffers from serious security and privacy threats. One big threat to SUs, which we tackle in this work, is location privacy, which can easily be compromised due to the wireless nature of the signals communicated by SUs during the cooperative sensing process. In fact, it has been shown that RSS values of SUs are highly correlated to their physical locations [5], thus making it easy to compromise the location privacy of SUs when sending out their sensing reports. The fine-grained location, when combined with other publicly available information, could easily be exploited to infer private information about users [6]. Examples of such private information are shopping patterns, user preferences, and user beliefs, just to name a few [6]. With such privacy threats and concerns, SUs may refuse to participate in the cooperative sensing tasks. It is therefore imperative that cooperative sensing schemes be enabled with privacy preserving capabilities that protect the location privacy of SUs, thereby encouraging them to participate in such a key CRN function, the spectrum sensing.

In this paper, we propose two efficient privacy-preserving schemes with several variants for cooperative spectrum sensing. These schemes exploit various cryptographic mechanisms to preserve the location privacy of SUs while performing the cooperative sensing task reliably and efficiently.

In addition, we study the cost-performance tradeoffs of the proposed schemes, and show that higher privacy and better performance can be achieved, but at the cost of deploying an additional architectural entity in the system. We show that our proposed schemes are secure and more efficient than their existing counterparts, and are robust against sporadic topological changes and network dynamism.

A. Related Work

Location privacy, though well studied in the context of location-based services (LBS) [11], [12], has received little attention in the context of CRNs [5], [13], [14]. Some works focused on location privacy but not in the context of cooperative spectrum sensing (e.g., database-driven spectrum sensing [13], [15] and dynamic spectrum auction [14]) and are skipped since they are not within this paper’s scope.

In the context of cooperative spectrum sensing, Shuai et al. [5] showed that SUs’ locations can easily be inferred from their RSS reports, and called this the SRLP (single report location privacy) attack. They also identified the DLP (differential location privacy) attack, where a malicious entity can estimate the RSS (and hence the location) of a leaving/joining user from the variations in the final aggregated RSS measurements before and after user’s joining/leaving of the network. They finally proposed PPSS to address these two attacks. Despite its merits, PPSS has several limitations: (i) It needs to collect all the sensing reports in order to decode the aggregated result. This is not fault tolerant, since some reports may be missing due, for example, to the unreliable nature of wireless channels. (ii) It cannot handle dynamism if multiple users join or leave the network simultaneously. (iii) The pairwise secret sharing requirement incurs extra communication overhead and delay. (iv) The underlying encryption scheme requires solving the \textit{Discrete Logarithm Problem}, which is possible only for very small plaintext space and can be extremely costly. Chen et al. [16] proposed \textit{PDAFT}, a fault-tolerant and privacy-preserving data aggregation scheme for smart grid communications. \textit{PDAFT}, though proposed in the context of smart grids, is suitable for cooperative sensing schemes. But unlike PPSS, \textit{PDAFT} relies on an additional semi-trusted entity, called gateway, and like other aggregation based methods, is prone to the DLP attack. In our previous work [17] we proposed an efficient scheme called \textit{LPOS} to overcome the limitations that existent approaches suffer from. \textit{LPOS} combines order preserving encryption and Yao’s millionaire protocol to provide a high location privacy to the users while enabling an efficient sensing performance.

\subsection*{B. Our Contribution}

In this paper, we propose two location privacy-preserving schemes for cooperative spectrum sensing that achieve:

- Location privacy of secondary users while performing the cooperative spectrum sensing effectively and reliably.
- Fault tolerance and robustness against network dynamism (e.g., multiple SUs join/leave the network) and failures (e.g., missed sensing reports).
- Reliability and resiliency against malicious users via an efficient reputation mechanism.
- Accurate spectrum availability decisions via half-voting rules while incurring minimum communication and computation overhead.

Compared to our preliminary works [18] and [17], this paper provides a more efficient version of \textit{LPOS} [17], referred to as \textit{LP-2PSS} in this paper, that is also robust against malicious users and adapted to a stronger threat model for FC. Besides, this paper provides another variant of \textit{LP-3PSS} [18] that improves the cryptographic end-to-end delay. Finally, this paper provides an improved security analysis and more comprehensive performance analysis.

The reason why we present two variants is to give more options to system designers to decide which topology and which approach is more suitable to their specific requirements. There are tradeoffs between the two options. While \textit{LP-2PSS} provides location privacy guarantees without needing to introduce an extra architectural entity, it requires relatively high computational overhead due to the use of the Yao’s millionaires’ protocol. On the other hand, \textit{LP-3PSS} provides stronger privacy guarantees (as the private inputs are shared among 3 non-colluding entities) and reduces the computational overhead substantially when compared to \textit{LP-2PSS}, but at the cost of introducing an extra architectural entity.

The remainder of this paper is organized as follows. Section II provides our system and security threat models. Section III presents our preliminary concepts and definitions. Section IV and V provide an extensive explanation of the proposed schemes. Section VI gives the security analysis of these schemes. Section VII presents their performance analysis and a comparison with existent approaches. Finally, Section VIII concludes this work.

\section*{II. System and Security Threat Models}

\subsection*{A. System Model}

We consider a cooperative spectrum sensing architecture that consists of a FC and a set of SUs. Each SU is assumed to be capable of measuring RSS on any channel by means of an energy detection method [4]. In this cooperative sensing architecture, the FC combines the sensing observations collected from the SUs, decides about the spectrum availability, and broadcasts the decision back to the SUs through a control channel. This could typically be done via either hard or soft decision rules. The most common soft decision rule is aggregation, where FC collects the RSS values from the SUs and compares their average to a predefined threshold, \( \tau \), to decide on the channel availability.

In hard decision rules, e.g. voting, FC combines votes instead of RSS values. Here, each SU compares its RSS value with \( \tau \), makes a local decision (available or not), and then sends to the FC its one-bit local decision/vote instead of sending its RSS value. FC applies then a voting rule on the collected votes to make a channel availability decision. However, for security reasons to be discussed shortly, it may not be desirable to share \( \tau \) with SUs. In this case, FC can instead collect the RSS values from the SUs, make a vote for each SU separately, and then combine all votes to decide about the availability of the channel.

In this work, we opted for the voting-based decision rule, with \( \tau \) not to be shared with the SUs, over the aggregation-based rule. Two reasons for why choosing voting over aggregation: One, aggregation methods are more prone to sensing errors; for example, receiving some erroneous measurements that are far off from the average of the RSS values can skew the computed RSS average, thus leading to wrong decision. Two, voting does not expose users to the DLP attack [5]
(which was identified earlier in Section I-A). We chose not to share \( \tau \) with the SU's because doing so limits the action scope of malicious users that may want to report falsified RSS values for malicious and/or selfish purposes.

In this paper, in addition to this 2-party (i.e., FC and SU's) cooperative sensing architecture that we just described above, we investigate a 3-party cooperative sensing architecture, where a third entity, called gateway (GW), is incorporated along with the FC and SU's to cooperate with them in performing the sensing task. As we show later, this gateway allows to achieve higher privacy and lesser computational overhead, but of course at its cost.

B. Security Threat Models and Objectives

We make the following security assumptions:

**Security Assumptions 1.** (i) FC may modify the value of \( \tau \) in different sensing periods to extract information about the RSS values of SU's; (ii) GW executes the protocol honestly but shows interest in learning information about the other parties; (iii) FC does not collude with SU's; and (iv) GW does not collude with SU's or FC.

We aim to achieve the following security objectives:

**Security Objectives 1.** (i) Keep RSS value of each SU confidential; and (ii) Keep \( \tau \) confidential. This should hold during all sensing periods and for any network membership change.

III. PRELIMINARIES

A. Half-Voting Availability Decision Rule

Our proposed schemes use the half-voting decision rule, shown to be optimal in [19], and for completeness, we here highlight its main ideas. Details can be found in [19].

Let \( h_0 \) and \( h_1 \) be the spectrum sensing hypothesis that \( PU \) is absent and present, respectively. Let \( P_f, P_d \) and \( P_m \) denote the probabilities of false alarm, detection, and missed detection, respectively, of one SU; i.e., \( P_f = Pr(\text{RSS} > \tau | h_0), P_d = Pr(\text{RSS} > \tau | h_1), \) and \( P_m = 1 - P_d. \)

FC collects the 1-bit decision \( D_i \) from each SU \( i \) and fuses them together according to the following fusion rule [19]:

\[
\text{dec}(t) = \begin{cases} 
H_1, & \sum_{i=1}^{n} D_i \geq \lambda \\
H_0, & \sum_{i=1}^{n} D_i < \lambda 
\end{cases}
\]

FC infers that \( PU \) is present, i.e. \( H_1 \), when at least \( \lambda \) SU's are inferring \( h_1 \). Otherwise, FC decides that \( PU \) is absent, i.e. \( H_0 \). Note that the OR fusion rule, in which FC decides \( H_1 \) if at least one of the decisions from the SU's is \( h_1 \), corresponds to the case where \( \lambda = 1 \). The AND fusion rule, in which FC decides \( H_1 \) if and only if all decisions from the SU's are \( h_1 \), corresponds to the case where \( \lambda = n \). The cooperative spectrum sensing false alarm probability, \( Q_f \), and missed detection probability, \( Q_m \), are: \( Q_f = Pr(H_1 | h_0) \) and \( Q_m = Pr(H_0 | h_1) \).

Letting \( n \) be the number of SU's, the optimal value of \( \lambda \) that minimizes \( Q_f + Q_m \) is \( \lambda_{opt} = \min(n, \lceil n/(1+\alpha) \rceil) \), where \( \alpha = \ln(\frac{P_f}{1-P_f})/\ln(\frac{P_d}{1-P_d}) \) and \( \lceil \cdot \rceil \) denotes the ceiling function. The value of \( \lambda_{opt} \) comes from the half-voting rule presented in [19]. We use it since it was proven in [19] to provide the best sensing performance in voting based cooperative sensing. For simplicity, \( \lambda_{opt} \) is denoted as \( \lambda \) throughout this paper.

B. Reputation Mechanism

To make the voting rule more reliable, we incorporate a reputation mechanism that allows FC to progressively eliminate faulty and malicious SU's. It does so by updating and maintaining a reputation score for each SU that reflects its level of reliability. Our proposed schemes incorporate the Beta Reputation mechanism [20]. For completeness, we highlight its key features next; more details can be found in [20] from which all computations in this subsection are based.

At the end of each sensing period \( t \), FC obtains a decision vector, \( b(t) = [b_1(t), b_2(t), \ldots, b_n(t)]^T \) with \( b_i(t) \in \{0, 1\} \), where \( b_i(t) = 0 \) (resp. \( b_i(t) = 1 \)) means that the spectrum is reported to be free (resp. busy) by user SU \( i \). FC then makes a global decision using the fusion rule \( f \) as follows:

\[
dec(t) = f(w(t), b(t)) = \begin{cases} 
1 & \text{if } \sum_{i=1}^{n} w_i(t)b_i(t) \geq \lambda \\
0 & \text{otherwise}
\end{cases}
\]

where \( w(t) = [w_1(t), w_2(t), \ldots, w_n(t)]^T \) is the weight vector calculated by FC based on the credibility score of each user, as will be shown shortly, and \( \lambda \) is the voting threshold determined by the Half-voting rule [19], as presented in Section III-A.

For each SU \( i \), FC maintains positive and negative rating coefficients, \( \varrho_i(t) \) and \( \eta_i(t) \), that are updated every sensing period \( t \) as: \( \varrho_i(t) = \varrho_i(t-1) + \nu_i(t) \) and \( \eta_i(t) = \eta_i(t-1) + \nu_i(t) \), where \( \nu_i(t) \) and \( \nu_i(t) \) are calculated as

\[
\begin{align*}
\nu_i(t) &= \begin{cases} 
1 & b_i(t) = \text{dec}(t) \\
0 & \text{otherwise}
\end{cases} \\
\nu_i(t) &= \begin{cases} 
1 & b_i(t) \neq \text{dec}(t) \\
0 & \text{otherwise}
\end{cases}
\end{align*}
\]

Here, \( \varrho_i(t) \) (resp. \( \eta_i(t) \)) reflects the number of times SU \( i \)’s observation, \( b_i(t) \), agrees (resp. disagrees) with the FC’s global decision, \( \text{dec}(t) \).

FC computes then SU \( i \)’s credibility score, \( \varphi_i(t) \), and contribution weight, \( w_i(t) \), at sensing period \( t \) as suggested in [20]:

\[
\begin{align*}
\varphi_i(t) &= \frac{\varrho_i(t) + \nu_i(t)}{\varrho_i(t) + \eta_i(t) + 2} \\
w_i(t) &= \varphi_i(t) / \sum_{j=1}^{n} \varphi_j(t)
\end{align*}
\]

C. Cryptographic Building Blocks

Our schemes use a few known cryptographic building blocks, which we define next before using them in the next sections when describing our schemes so as to ease the presentation.

**Definition 1.** Order Preserving Encryption (OPE): is a deterministic symmetric encryption scheme whose encryption preserves the numerical ordering of the plaintexts, i.e. for any two messages \( m_1 \) and \( m_2 \) s.t. \( m_1 \leq m_2 \), we have \( c_1 \leftarrow \text{OPE.EK}(m_1) \leq c_2 \leftarrow \text{OPE.EK}(m_2) \) [21], with
\[c \leftarrow OPE.E_K(m)\] is order preserving encryption of a message \(m \in \{0,1\}^d\) under key \(K\), where \(d\) is the block size of \(OPE\).

**Definition 2. Yao’s Millionaires’ (YM) Protocol [22]:** is a Secure Comparison protocol that enables two parties to execute “the greater-than” function, \(GT(x, y) = [x > y]\), without disclosing any other information apart from the outcome.

**Definition 3. Tree-based Group Elliptic Curve Diffie-Hellman (TGECDH) [23]:** is a dynamic and contributory group key establishment protocol that permits multiple users to collaboratively establish and update a group key \(K\).

**Definition 4. Group Key independence:** given a subset of previous keys, an attacker cannot know any other group key.

**Definition 5. Elliptic Curve Discrete Logarithm Problem (ECDLP)**: given an elliptic curve \(E\) over \(GF(q)\) and points \((P, Z) \in E\), find an integer \(x\), if any exists, s.t. \(Z = xP\).

**Definition 6. Digital Signature:** A digital signature scheme \(SGN\) is used to validate the authenticity and integrity of a message \(m\). It contains three components defined as follows:

- **Key generation algorithm (Kg):** returns a private/public key pair given a security parameter \(\kappa\), \((SK_{DS}, PK_{DS}) \leftarrow SGN.Kg(1^\kappa)\).
- **Signing algorithm (Sign):** takes as input a message \(m\) and the secret key \(SK_{DS}\) and returns a signature \(\sigma\), \(\sigma \leftarrow SGN.Sign(SK_{DS}, m)\).
- **Verification algorithm (Ver):** takes as input the public key \(PK_{DS}\), \(m\) and \(\sigma\). It returns 1 if valid and 0 if invalid, \(\{0,1\} \leftarrow SGN.Ver(PK_{DS}, m, \sigma)\).

Note that communications are made over a secure (authenticated) channel maintained with a symmetric key (e.g., via SSL/TLS) to ensure confidentiality and authentication. For the sake of brevity, we will only write encryptions but not the authentication tags (e.g., Message Authentication Codes [24]) for the rest of the paper.

In the following we present the two schemes that we propose in this paper. For convenience and before getting into the details of the proposed approaches, we have summarized the different notations that we use in the remaining parts of this paper in Table I.

### IV. LP-2PSS

We now present our first proposed scheme, which is a voting-based approach designed for the 2-party cooperative spectrum sensing network, consisting of one \(FC\) and a set of \(SUs\). Throughout, we refer to this scheme by LP-2PSS (location privacy for 2-party spectrum sensing architecture). LP-2PSS achieves the aforementioned security objectives via an innovative integration of the \(OPE, TGECDH\) and YM protocols. Voting-based spectrum sensing offers several advantages over its aggregation-based counterparts as discussed in Section III, but requires comparing FC’s threshold \(\tau\) and SUs’ RSSs, thereby forcing at least one of the parties to expose its information to the other. One solution is to use a secure comparison protocol, such as YM, between \(FC\) and each \(SU\), which permits \(FC\) to learn the total number of \(SUs\) above/below \(\tau\) but nothing else. However, secure comparison protocols involve several costly public key crypto operations (e.g., modular exponentiation), and therefore \(O(n)\) invocations of such a protocol per sensing period, thus incurring prohibitive computational and communication overhead.

- **Intuition:** The key observation that led us to overcome this challenge is the following: If we enable \(FC\) to learn the relative order of RSS values but nothing else, then the number of YM invocations can be reduced drastically. That is, the knowledge of relative order permits \(FC\) to execute YM protocol at worst-case \(O(\log(n) / \tau)\) by utilizing a binary-search type approach, as opposed to running YM with each user in total \(O(n)\) overhead. This is where \(OPE\) comes into play. The crux of our idea is to make users \(OPE\) encrypt their RSS values under a group key \(K\), which is derived via TGECDH at the beginning of the protocol. This allows \(FC\) to learn the relative order of encrypted RSS values but nothing else (and users do not learn each others’ RSS values, as they are sent to \(FC\) over a pairwise secure channel). \(FC\) then uses this knowledge to run YM protocol by utilizing a binary-search strategy, which enables it to identify the total number of users above/below \(\tau\) and then compares it to \(\lambda\). As \(FC\) may try to maliciously modify the value of \(\tau\) as stated in Security Assumption 1, this makes it easier for it to infer the RSS values of \(SUs\), thus their location. We rely on digital signatures to overcome this limitation. A digital signature is used by \(SUs\) to verify the integrity of the information that was
sent by FC during the execution of YM protocol and signed by the service operator as we explain in more details next. This strategy makes LP-2PSS achieve SU’s location privacy with efficient spectrum sensing, fault-tolerance and network dynamism simultaneously.

Before we describe our protocol in more details, we first highlight how we improve the YM protocol proposed in [25] as shown next.

A. Our Improved YM.ECElG Scheme

To achieve high efficiency, we improve the YM protocol in [25], in which only the initiator of the protocol learns the outcome, and call this new improved scheme YM.ECElG. YM.ECElG, described next, is used by our proposed LP-2PSS to perform secure comparisons. Our secure comparison scheme improves YM protocol proposed in [25] in two aspects: (i) We adapt it to work with additive homomorphic encryption (specifically ECElG) to enable compact comparison operations in Elliptic Curves (EC) domain. (ii) The final stage of YM.ECElG requires solving ECDLP (Definition 5), which is only possible with small plaintext domains, and this is the case for our 8-bit encoded RSS values required by IEEE 802.22 standard [26]. However, despite small plaintext domain, solving ECDLP with brute-force is still costly. We improve this step by adapting Pollard-Lambda method [27] to solve the ECDLP for the reverse map, which offers decryption efficiency and compactness. The Pollard-Lambda method is designed to solve the ECDLP for points that are known to lie in a small interval, which is the case for RSS values [27]. Below, we outline our optimized YM.ECElG.

- **Notation:** Let $\gamma = |\text{RSS}| = |\tau|$ denote the size in bits of the RSS value of a SU and $\tau$ of FC to be privately compared. Also, let $n$ denote the average number of SU’s per sensing period, $q$ be a large prime number, $\mathcal{E}$ an elliptic curve over a finite field $GF(q)$, $Z$ a point on the curve with prime order $m$. $(sk, pk)$ is a private/public key pair of Elliptic Curve ElGamal (ECElG) encryption [28], generated under $(\mathcal{E}, q, Z, m)$. Let $\pi = (\gamma, \mathcal{E}, q, Z, m, (sk, pk))$ be YM.ECElG parameters generated by FC which is the initiator of the protocol. YM.ECElG returns $b \leftarrow ECElGama(\tau, \text{RSS}, \pi)$, where $b = 0$ if $\tau < \text{RSS}$ and $b = 1$ otherwise. Only FC learns $b$ but (FC, SU) learn nothing else. For simplicity during the description of YM.ECElG, we denote $\tau$ as $x$ and RSS as $y$.

YM.ECElG, as in YM, is based on the fact that $x$ is greater than $y$ if $S_x^1$ and $S_y^0$ have a common element where $S_x^1$ and $S_y^0$ are the 1-encoding of $x$ and the 0-encoding of $y$ respectively. The 0-encoding of a binary string $s = s_\gamma \ldots s_1 \in \{0, 1\}^\gamma$ is given by $S_y^0 = \{s_\gamma \ldots s_{\gamma-1} \ldots s_1 | s_i = 0.1 \leq i \leq \gamma\}$ and the 1-encoding of $s$ is given by $S_x^1 = \{s_\gamma \ldots s_{\gamma-1} \ldots s_1 | s_i = 1.1 \leq i \leq \gamma\}$. For example, if we have a string $s = 1011011$, then $S_y^0 = \{101, 10111\}$ and $S_x^1 = \{1101, 10111, 10011\}$. If we want to compare two values $x = 46 = 101110$ and $y = 45 = 101101$, we need first to construct $S_x^1 = \{1, 101, 10111, 10011\}$ and $S_y^0 = \{1101, 10111\}$. Since $S_x^1 \cap S_y^0 \neq \emptyset$, then $x > y$.

FC with a private input $z = x_\gamma \ldots x_1$ generates $\pi$ for encryption and decryption $(E, D)$ then prepares a $2 \times \gamma$-table $T[i, j], i \in \{0, 1\}, 1 \leq j \leq \gamma$ such that $T[x_i, i] = E(1)$ and $T[x_i, i] = E(r_i)$ for a random $r_i$ in the subgroup $G_q$ and finally sends $T$ to SU. SU with private input $y = y_\gamma y_{\gamma-1} \ldots y_1$ computes $c_t$ for each $t = t_\gamma t_{\gamma-1} \ldots t_1 \in S_y^0$ as follows

$$c_t = T[t_\gamma] \oplus T[t_{\gamma-1}] \ldots \oplus T[t_1, i]$$

with $\oplus$ denotes Elliptic Curve point addition operations ($\oplus$ replaces $\times$ in the original YM scheme). SU then prepares $l = \gamma - |S_y^0|$ random encryptions $z_i = (a_i, b_i) \in G_q^2$, $1 \leq j \leq l$ and permutes $c_t$'s and $z_i$'s to obtain $\hat{c}_1, \ldots, \hat{c}_\gamma$ which are sent back to FC that decrypts $D(\hat{c}_i) = m_i, 1 \leq i \leq \gamma$ via Pollard-Lambda algorithm [27] and decides $x > y$ iff some $m_i = 0$ ($m_i = 1$ in the original YM). The different steps of this protocol are summarized in Figure 1.

![Fig. 1: YM.ECElG protocol](image-url)

B. LP-2PSS Description

Next we describe our proposed scheme LP-2PSS whose main steps are outlined in Algorithm 1.

- **Initialization:** The service operator sets up the value of energy threshold $\tau$. FC sets up ECElG crypto parameters, voting threshold and users reputation weights values. Initially, all the users are considered credible so the weight vector $w$ is constituted of ones. FC, then, constructs the table $T$ used in YM protocol as described in Section IV-A with $\tau$ as input using the FC’s ECElG public key pk. Notice here that since the same $\tau$ is always used during different sensing periods, the table $T$ can be precomputed during the Initialization phase. This considerably reduces this protocol’s computational overhead. Then the service operator that manages the network signs $T$ using a digital signature scheme with secret key $SK_{DS}$. This digital signature is used to make sure that FC does not maliciously modify the value of $\tau$ to learn RSS values of users and thus infer their locations. The service operator then shares the public key $PK_{DS}$ with SUs to use it for verifying the integrity of $T$ and thus of $\tau$. SUs establish a group key $K$ via TGECDH, with which they OPE encrypt their RSS values during the private sensing. FC also establishes a secure channel $chn_i$ with each user $SU_i$.

- **Private Sensing:** Each $SU_i$ OPE encrypts its RSS value with group key $K$ and sends ciphertext $c_t$ to FC over $chn_i$. FC then sorts ciphertexts as $c_{min} \leq \ldots \leq c_{max}$ (as all RSS’s are OPE encrypted under the same $K$) without learning corresponding RSS values, and the secure channel $chn_i$ protects the communication of $SU_i$ from other users as well as from outside attackers. FC then initiates YM.ECElG first with the
Algorithm 1 LP-2PSS Algorithm

Initialization: Executed only once.
1: Service operator sets \( \tau \).
2: FC generates \( \pi \), sets \( \lambda \) and \( \omega \leftarrow 1 \).
3: FC pre-computes \( T \) using \( pk \).
4: Service operator computes \( \sigma \leftarrow SGN.Sign(SK_{DS}, T) \).
5: Service operator shares \( PK_{DS} \) with SUs.
6: \( \mathcal{G} = \{ SU_{i} \}_{i=1}^{n} \) establish \( K \) via TGECDH protocol.
7: FC establishes \( c_{\text{chn}} \) with each \( SU_{i} \) for \( i = 1, \ldots, n \).

Private Sensing: Executed every sensing period \( t_{w} \)
8: \( SU_{i} \) computes \( c_{\text{chn}_{i}} \leftarrow OPE.\mathcal{E}_{K}(RSS_{i}) \) for \( i = 1, \ldots, n \).
9: \( SU_{i} \) sends \( c_{\text{chn}_{i}} \) to FC over \( c_{\text{chn}} \) for \( i = 1, \ldots, n \).
10: FC sorts encrypted RSS values as \( c_{\text{min}} \leq \ldots \leq c_{\text{max}} \).
11: FC runs \( b_{\text{id}_{\text{max}}} \leftarrow YM.ECEIG(RSS_{\text{max}}, \tau, \pi) \) with \( SU_{\text{id}_{\text{max}}} \) having \( c_{\text{max}} \).
12: \( SU_{\text{id}_{\text{max}}} \) verifies \( T \) using \( \sigma \).
13: if \( SGN.Ver(PK_{DS}, T, \sigma) = 0 \) then
14: \( SU_{\text{id}_{\text{min}}} \) leaves the sensing
15: Go to Step 17.
16: if \( b_{\text{id}_{\text{max}}} = 0 \) then \( \text{dec} \leftarrow \text{Channel free} \), \( \{ b_{i} \}_{i=1}^{n} \leftarrow 0 \).
17: else FC runs \( b_{\text{id}_{\text{min}}} \leftarrow YM.ECEIG(RSS_{\text{min}}, \tau, \pi) \) with \( SU_{\text{id}_{\text{max}}} \) having \( c_{\text{min}} \).
18: \( SU_{\text{id}_{\text{min}}} \) verifies \( T \) using \( \sigma \).
19: if \( SGN.Ver(PK_{DS}, T, \sigma) = 0 \) then
20: \( SU_{\text{id}_{\text{min}}} \) leaves the sensing
21: Go to Step 27.
22: if \( b_{\text{id}_{\text{min}}} = 1 \) then \( \text{dec} \leftarrow \text{Channel busy} \), \( \{ b_{i} \}_{i=1}^{n} \leftarrow 1 \).
23: else repeat
24: 
25: FC computes \( I \leftarrow \text{BinarySearch}(\mathcal{G}) \)
26: FC runs \( b_{I} \leftarrow YM.ECEIG(RSS_{I}, \tau, \pi) \) with \( SU_{I} \) having \( c_{I} \).
27: \( SU_{I} \) verifies \( T \) using \( \sigma \).
28: if \( SGN.Ver(PK_{DS}, T, \sigma) = 0 \) then
29: \( SU_{I} \) leaves the sensing
30: until \( RSS_{I} = \tau \) \( \leq RSS_{I-1} \)
31: FC assigns \( b_{i} \leftarrow 0 \) for \( i = 1, \ldots, I-1 \) and \( b_{j} \leftarrow 1 \) for \( j = I, \ldots, n \).
32: FC computes \( v \leftarrow \sum_{i=1}^{n} w_{i} \times b_{i} \)
33: if \( v \geq \lambda \) then \( \text{dec} \leftarrow \text{Channel busy} \)
34: else \( \text{dec} \leftarrow \text{Channel free} \)
35: FC updates \( \{ \varphi_{i} \}_{i=1}^{n} \) and \( \{ w_{i} \}_{i=1}^{n} \) as in Eqs. (3) & (4)
36: return \( \text{dec} \)

Update after \( \mathcal{G} \) Membership Changes or Breakdown:
36: if \( SU(s) \) join/leave \( \mathcal{G} \) or breakdown in \( t_{w} \) then
37: New group \( \mathcal{G}' \) form new \( K' \) using TGECDH.
38: FC updates \( \lambda \) and \( \pi \) as \( \lambda' \) and \( \pi' \), respectively, if required.
39: Execute the private sensing with \( (K', \lambda', \pi') \).

\( SU_{\text{id}_{\text{max}}} \) that has the highest RSS value \( RSS_{\text{max}} \). If it is smaller than energy sensing threshold \( \tau \), then the channel is free. Otherwise, FC initiates YM.ECEIG with the user that has \( RSS_{\text{min}} \). If it is greater than \( \tau \), then the channel is busy. Otherwise, to make the final decision based on the optimal sensing threshold \( \lambda \), FC runs YM.ECEIG according to the binary-search strategy which guarantees the decision at the worst \( O((log(n)) \) invocations. Note that before participating in YM.ECEIG, each SU first verifies the integrity of \( T \) using the digital signature \( \sigma \) that was provided by the service operator as indicated in Steps 12, 18 & 27. A SU that detects a change in the value of \( T \) refuses to participate in the sensing to prevent FC from learning any sensitive information regarding its location. In that case the system stops and the malicious intents of FC are detected.

In Steps 16, 22 & 31 of Algorithm 1, FC constructs the vector of local decisions of SUs after running the private comparisons between \( \tau \) and RSS values. Based on the decision vector \( b \) and the weights vector \( w \) that was computed previously, FC computes \( v \) in Step 32 using Equation 2 to finally make the final decision \( \text{dec} \) using voting threshold \( \lambda \). FC then computes the credibility score and the weights that will be given to all users in the next sensing period. If \( SU_{i} \) has a decision \( b_{i} \neq \text{dec} \), its assigned weight decreases. But if a SU makes the same decision as FC, it is assigned the highest weight. The main steps of the private sensing phase are summarized in Figure 2.

---

Fig. 2: LP-2PSS's Private Sensing phase

- **Update after \( \mathcal{G} \) Membership Changes or Breakdown:** At the beginning of \( t_{w} \), if membership status of \( \mathcal{G} \) changes, a new group key is formed via TGECDH, and then FC updates \( \lambda \). If some SUs breakdown and fail to sense or send their measurements, \( \lambda \) also must be updated. In new sensing period, Algorithm 1 is executed with new parameters and group key.

**Choice of digital signature**

Choosing the right digital signature scheme depends on the network and users constraints. In the following we briefly...
our Security Objectives 1. Note that LP-3PSS and SU the final decision. FC
compact scheme among all alternatives but with an additional
compactness among the alternatives but with an additional
compactness in terms of signature and key size and/or computational efficiency.

Another scheme could be ECDSA [30] which is an elliptic curve analogue of the DSA [31] digital signature scheme. It provides more compact signatures than its counterparts thanks to the use of Elliptic Curve crypto. It has a moderate speed, though, in terms of verification and encryption compared to RSA. It is more suitable for situations where the communication overhead is the main concern.

One-time signatures, e.g. HORSE [32] and its variants [33], [34], are digital signatures that are based on one-way functions without a trapdoor which makes them much faster than commonly used digital signatures, like RSA. The main drawbacks of this kind of digital signatures are their large size and the complexity of their ”one-timed-ness” which requires a new call to the key generation algorithm for each use. In our context, we should not worry about the latter since we sign T only once so we don’t have to regenerate the keys. In that case, one-time signatures may be the best option when computation speed at SU’s is the main concern.

NTRU [35] signature could also be applied here. It provides a tradeoff between signature size and computational efficiency. Indeed it has a moderate signature size that is larger than the one of ECDSA but it is faster than both ECDSA and RSA in key generation, signing and verification.

V. LP-3PSS

We now present an alternative scheme that we call LP-3PSS (location privacy for 3-party spectrum sensing architecture), which offers higher privacy and significantly better performance than that of LP-2PSS, but at the cost of deploying an additional entity in the network, referred to as Gateway (GW) (thus ”3P” refers to the 3 parties: SU’s, FC, and GW).

GW enables a higher privacy by preventing FC from even learning the order of encrypted RSS values of SU’s (as in LP-2PSS). GW also learns nothing but secure comparison outcome of a RSS values andτ as in YM but only using OPE. Thus, no entity learns any information on RSS or τ beyond a pairwise secure comparison, which is the minimum information required for a voting-based decision.

• Intuition: The main idea behind LP-3PSS is simple yet very powerful: We enable GW to privately compare n distinct OPE encryptions of τ and RSS values, which were computed under n pairwise keys established between FC and SU’s. These OPE encrypted pairs permit GW to learn the comparison outcomes without deducing any other information. GW then sends these comparison results to FC to make the final decision. FC learns no information on RSS values and SU’s cannot obtain the value of τ, which complies with our Security Objectives 1. Note that LP-3PSS relies only on symmetric cryptography to guarantee the location privacy of SU’s. Hence, it is the most computationally efficient and compact scheme among all alternatives but with an additional

entity in the system. LP-3PSS is described in Algorithm 2 and outlined below.

Algorithm 2 LP-3PSS Algorithm

Initialization: Executed only once.
1: Service operator sets τ.
2: FC sets λ and w ← 1.
3: FC establishes k_{FC,i} with SU_i, i = 1, ..., n.
4: GW establishes k_{GW,i} with SU_i, i = 1, ..., n.
5: FC establishes k_{FC,GW} with GW.
6: FC computes τ_i ← \mathcal{E}_{FC,GW}(OPE.E_{k_{FC,i}}(τ)), i = 1, ..., n and sends \{τ_i\}_{i=1}^n to GW.

Private Sensing: Executed every sensing period t_w
7: SU_i computes τ_i ← \mathcal{E}_{k_{GW,i}}(OPE.E_{k_{FC,i}}(RSS_i)), i = 1, ..., n and sends SU_i to GW.
8: GW obtains OPE.E_{k_{FC,i}}(τ) ← D_{k_{FC,GW}}(τ_i) and OPE.E_{k_{FC,i}}(RSS_i) ← D_{k_{GW,i}}(τ_i), i = 1, ..., n.
9: for i = 1, ..., n do
10: if OPE.E_{k_{FC,i}}(RSS_i) < OPE.E_{k_{FC,i}}(τ) then b_i ← 0
11: else b_i ← 1
12: GW computes τ ← \mathcal{E}_{k_{FC,GW}}(\{b_i\}_{i=1}^n) and sends τ to FC.
13: FC decrypts τ and computes v ← \sum_{i=1}^n w_i × b_i
14: if v ≥ \lambda then dec ← Channel busy
15: else dec ← Channel free
16: FC updates \{ϕ_1\}_{i=1}^n and \{ω_i\}_{i=1}^n as in Eqs. (3) & (4)
return dec

Update after \mathcal{G} Membership Changes or Breakdown:
17: if SU_j joins CRN then
18: SU_j establishes k_{FC,j} with FC and k_{GW,j} with GW.
19: if SU’s join/leave/breakdown then
20: FC updates λ as λ'.
21: Execute the private sensing with λ'.

• Initialization: Service operator and FC set up spectrum sensing and crypto parameters. Let (E, D) be IND-CPA secure [24] block cipher (e.g. AES) encryption/decryption operations. FC establishes a secret key with each SU and GW. GW establishes a secret key with each SU. FC encrypts τ with OPE using k_{FC,i}, i = 1, ..., n. FC then encrypts OPE ciphertexts with E using k_{FC,GW} and sends these \{τ_i\}_{i=1}^n to GW, i = 1, ..., n. Since these encryptions are done offline at the beginning of the protocol, they do not impact the online private sensing phase. FC may also pre-compute a few extra encrypted values in the case of new users joining the sensing.

• Private Sensing: Each SU_i encrypts RSS_i with OPE using k_{FC,i}, which was used by FC to OPE encrypt τ value. SU_i then encrypts this ciphertext with E using key k_{GW,j}, and sends the final ciphertext τ_i to GW. GW decrypts 2n ciphertexts \{τ_i\} with \mathcal{D} using k_{FC,GW} and k_{GW,j}, which yields OPE encrypted values. GW then compares each OPE encryption of RSS with its corresponding OPE encryption of τ. Since both were encrypted with the same key, GW can compare them and conclude which one is greater as in Step 10. GW stores the outcome of each comparison
in a binary vector $b$, encrypts and sends it to $FC$. Finally, $FC$ compares the summation of votes $v$ to the optimal voting threshold $\lambda$ to make the final decision about spectrum availability and updates the reputation scores of the users.

- **Update after $G$ Membership Changes or Breakdown**: Each new user joining the sensing just establishes a pairwise secret key with $FC$ and $GW$. This has no impact on existing users. If some users leave the network, $FC$ and $GW$ remove their secret keys, which also has no impact on existing users. In both cases, and also in the case of a breakdown or failure, $\lambda$ must be updated accordingly.

![Fig. 3: LP-3PSS protocol](image)

**Remark 1.** A malicious $FC$ in LP-3PSS following Security Assumption 1 may want to maliciously modify the value of $\tau$. But since $GW$ is the one that performs the comparison between RSS values and $\tau$, changing $\tau$ maliciously has almost no benefit to $FC$ as it does not have access to individual comparison outcomes. This makes LP-3PSS robust against this malicious $FC$.

It is worth iterating that the $GW$ only needs to perform simple comparison operations between the RSS values of the $SUs$ and the energy sensing threshold $\tau$ of the $FC$ as we explained earlier. Thus, such an entity does not interfere with the spectrum sensing process in the CRN. Moreover, it does not need to be provided with large computational resources as these comparisons are very simple and fast to perform. It could be a standalone entity, one of the $SUs$ that is dedicated to perform the tasks of the $GW$ or even a secure hardware that is deployed inside the $FC$ itself as we discuss next. This gives multiple options to system designers. If FCC’s regulation allows introducing an additional entity to the CRN, then $GW$ could be deployed without any concern. If not, system designers could consider introducing a secure hardware within $FC$ or dedicating one of the $SUs$ to perform the tasks of $GW$.

**LP-3PSS with Secure Hardware**

LP-3PSS could also be implemented in a slightly different way by relying on a secure hardware deployed within the $FC$ itself instead of using a dedicated gateway. All the computation that is performed by $GW$ could be relayed to this hardware. This secure hardware, which is referred to as secure co-processor ($SCPU$) or as trusted platform module (TPM) in the literature, is physically shielded from penetration, and the I/O interface to the module is the only way to access the internal state of the module [36]. An $SCPU$ that meets the FIPS 140-2 level 4 [37] physical security requirements guarantees that $FC$ cannot tamper with its computation. Any attempt to tamper with this $SCPU$ from $FC$ that results somehow in penetrating the shield, leads to the automatic erasure of sensitive memory areas containing critical secrets.

The $SCPU$ may provide several benefits to the network. First, there is no need anymore of adding a new standalone entity managed by a third party to the network as was the case with $GW$. Also, despite its high cost, having an $SCPU$ deployed within $FC$ itself may reduce the communication latency that is incurred by having a gateway that needs to communicate with $FC$ and with every user in the network.

In terms of performance, it was proven in [38] that at a large scale the computation inside an $SCPU$ is orders of magnitude cheaper than equivalent cryptography that is performed on an unsecured server hardware, despite the overall greater acquisition cost of secure hardware.

All of this makes using an $SCPU$ a good alternative to using a dedicated gateway in the network thanks to its performance and the security guarantees that it provides.

**VI. SECURITY ANALYSIS**

We first describe the underlying security primitives, on which our schemes rely, and then precisely quantify the information leakage of our schemes, which we prove to achieve our Security Objectives 1. At the end of this section, we discuss the security of the modified versions of our schemes.

**Fact 1.** An $OPE$ is indistinguishable under ordered chosen-plaintext attack (IND-OCPA) [21] if it has no leakage, except the order of ciphertexts (e.g. [39], [40]).

**Fact 2.** YM.ECElG is secure by Definition 2 if ECElG cryptosystem [28], whose security relies on the ECDLP (Definition 5), is secure.

**Fact 3.** TGECdH is secure with key independence by Definition 4 if ECDLP is intractable by Definition 5.

Let $E$ and $OPE.E$ be IND-CPA secure [24] and IND-OCPA secure symmetric ciphers, respectively. $(\{RSS_i\}_{i=1}^{n,\ell}, \tau)$ are RSS values and $\tau$ of each $SU_{i}$, and $FC$ for sensing periods $j = 1, \ldots, \ell$ in a group $G$. $(L_1, L_2, L_3)$ are history lists, which include all values learned by entities $SU_i$, $FC$ and $GW$, respectively, during the execution of the protocol for all sensing periods and membership status of $G$. Vector $V$ is a list of IND-CPA secure values transmitted over secure (authenticated) channels. $V$ may be publicly observed by all entities including external attacker $A$. Hence, $V$ is a part of all lists $(L_1, L_2, L_3)$. Values (jointly) generated by an entity such as cryptographic keys or variables stored only by the entity itself (e.g., $\lambda$, $\pi$) are not included in history lists for the sake of brevity. Moreover, information exchanged during the execution of YM.ECElG protocol are not included in history lists, since they do not leak any information by Fact 2.
Theorem 1. Under Security Assumptions 1, LP-2PSS leaks no information on \( \left( \{ \text{RSS}_i \}^n_{i=1}, \{ \text{TGECDH} \}^n_{i=1}, \{ \text{SCPU} \}^n_{i=1} \right) \) beyond IND-CPA secure \( \{ \tilde{V}^j \}^\ell_{j=1}, \{ \text{IND-SCPU} \}^\ell_{j=1} \) secure order of tuple \( \{ \tilde{K} \} = \{ \text{OPE}, \text{E}_{\text{FC}} \} \{ \tilde{V}^j \}^\ell_{j=1}, \text{OPE}, \text{E}_{\text{FC}} \} \{ \text{RSS}^j \}^\ell_{j=1} \), and \( \{ b^j \}^\ell_{j=1} \rightarrow \text{FC} \).

Proof: \( \tilde{V}^j = \{ \text{ch}^j \}^\ell_{j=1} \) at Step 6 of Algorithm 1. History lists are as follows for each sensing period \( j = 1, \ldots, \ell \):

\[
\mathcal{L}_1 = \tilde{V}^j, \quad \mathcal{L}_2 = (\{ b^j \}^n_{i=1}, \tilde{V}^j, \tilde{Z}^j),
\]

where \( \{ b^j \}^n_{i=1} \) are the outcomes of YM.ECEIG protocol (Steps 11, 17 & 26 of Algorithm 1). By Fact 2, YM.ECEIG protocol leaks no information beyond \( \{ b^j \}^n_{i=1} \) to FC and no information to anyone else. Variables in \( \{ \mathcal{L}_1, \mathcal{L}_2 \} \) are IND-CPA and IND-OCPA secure, and therefore leak no information beyond the order of tuples in \( \tilde{Z}^j \) to FC by Fact 1.

Any membership status update on \( \mathcal{G} \) requires an execution of TGECDH protocol, which generates a new group key \( \tilde{K}^j \). By Fact 3, TGECDH guarantees key independence property (Definition 4), and therefore \( \tilde{K}^j \) is only available to new members and is independent from previous keys. Hence, history lists \( \{ \mathcal{L}_1, \mathcal{L}_2 \} \) are computed identically as described above for the new membership status of \( \mathcal{G} \) but with \( \tilde{K}^j \), which are IND-CPA secure and IND-OCPA secure.

Using a digital signature gives SU’s the possibility to learn the intentions of FC and detect whether it is trying to locate them. Since no SU wants its location to be revealed, SUs will simply refuse to participate in the sensing upon detection of malicious activity of FC by verifying the signed messages. The only way that FC can learn the location of a SU in this case is when this SU continues to participate in the sensing even after detecting the malicious intents of FC.

Corollary 1. Theorem 1 and Theorem 2 guarantee that in our schemes, RSS values and \( \tau \) are IND-OCPA secure for all sensing periods and membership changes. Hence, our schemes achieve Objectives 1.

A. Discussion about SCPU-based LP-3PSS’s security

The security of SCPU-based LP-3PSS could be reduced to that of the SCPU that is used. Since no direct communication exists between FC and SUs, the only way for FC to learn RSS values of SUs is by compromising the SCPU. Having the secret keys that were used to OPE encrypt SUs’ RSS values, a successful attempt to break into this secure hardware by FC will allow it to decrypt the RSS values and learn SUs’ locations. However, as mentioned earlier, a SCPU that complies with the physical security requirements of FIPS 140-2 level 4 [37] should guarantee that such a breach does not happen. And thus, FC should not be able to retrieve the data in the SCPU even though the latter is deployed inside the malicious FC itself.

B. Discussion about collusion between different entities

We also investigate how our schemes perform under collusion. We discuss different collusion scenarios for each proposed scheme separately.

For LP-2PSS, if multiple SUs collude to learn another SU’s location information, their collusion can only allow them to learn IND-CPA secure values \( \tilde{V} \) which contain the OPE-encrypted RSS values transmitted over the authenticated secure channel between the target SU and FC. This means that collusion among SUs does not allow them to learn RSS measurements of other SUs and, thus, nor their location. The second scenario is when FC colludes with some SUs to encrypt their RSS measurements. Only in this case would FC be able to learn SUs’ locations. Therefore, LP-2PSS is robust against collusion among compromised SUs, but assumes that FC cannot collude with SUs.

Similar reasoning applies to LP-3PSS. Collusion among SUs does not allow them to infer other SUs’ locations. And if SUs collude with GW, they can only manage to learn the encrypted RSS measurements of the SUs but nothing more, as each SU OPE encrypts its RSS measurement with its own secret key. Also, collusion between FC and some SUs cannot reveal the RSS measurements of other SUs as the latter send their OPE encrypted RSS values through their authenticated channels established individually with the GW. This prevents colluding SUs and FC from accessing the private information of other SUs and subsequently localizing them. Thus, for LP-3PSS, only collusion between GW and FC could reveal RSS measurements of all SUs as FC has the secret keys that were used by SUs to OPE encrypt their RSS values before sending them to GW. However, this specific collusion scenario could be dealt with, for example, by deploying a secure hardware within FC to play the role of GW. The inherent nature of such a hardware prevents FC from accessing it and colluding with it. We provided an explanation to this in Section V. This proofs that LP-3PSS is...
TABLE II: Computational overhead comparison

<table>
<thead>
<tr>
<th>Scheme</th>
<th>FC</th>
<th>Computation</th>
</tr>
</thead>
<tbody>
<tr>
<td>LP-2PSS</td>
<td>[\gamma/2 \cdot (2 + \log n) \cdot (PMulQ + PAddQ + √\delta \cdot Pol)]</td>
<td>[(4γ - 6) \cdot PAddQ + OPE + μ \cdot (2 \log n + 2) \cdot PMulQ]</td>
</tr>
<tr>
<td>LPOS</td>
<td>[1/2 \cdot (2 + \log n) \cdot γ \cdot [p] \cdot Mulp]</td>
<td>[(2γ \cdot [p] + 2γ) \cdot Mulp + OPE + 2\mu \log n \cdot PMulQ]</td>
</tr>
<tr>
<td>ECEG</td>
<td>[PMulQ + PAddQ + √\delta \cdot Pol]</td>
<td>[2PMulQ + PAddQ]</td>
</tr>
<tr>
<td>PPSS</td>
<td>[H + (n + 2) \cdot Mulp + (2^{-(n+2)} \cdot n + 2) \cdot Expp]</td>
<td>[H + 2Expp + Mulp]</td>
</tr>
<tr>
<td>LP-3PSS</td>
<td>[D + \beta(t) \cdot (E + OPE_E)]</td>
<td>[OPE_E + E]</td>
</tr>
<tr>
<td>PDAFT</td>
<td>[2Expp + 1\cdot Mulp]</td>
<td>[2Expp + 1\cdot Mulp]</td>
</tr>
</tbody>
</table>

(i) Variables: \(κ\) security parameter, \(N\): modulus in Paillier, \(p\): modulus of El Gamal, \(H\): cryptographic hash operation, \(K\): secret group key of OPE, \(Expp\) and \(Mulp\) denote a modular exponentiation and a modular multiplication over modulus \(u\) respectively, where \(u \in \{N, N^2, p\}\). \(InvN^2\): modular inversion over \(N^2\), \(PMulQ\): point multiplication of order \(Q\), \(PAddQ\); point addition of order \(Q\), \(γ\): number of servers needed for decryption in PDAFT. (ii) Parameters size: For a security parameter \(κ\) = 80, suggested parameter sizes by NIST 2012 are given by: \(|N| = 1024\), \(|p| = 1024\), \(|Q| = 192\) as indicated in [41]. (iii) YM.ECEG-Mambo: The communication cost for one comparison is \(4γ \cdot |Q|\). The total computational cost of the scheme for one comparison is \(γ \cdot (PMulQ + 5PAddQ + √3 \cdot Pol) - 6PAddQ\). (iv) ECEG: The decryption of the aggregated message in ECEG is done by solving the constrained ECDLP problem on small plaintext space similarly to [5] via Pollard’s Lambda algorithm, which requires \(O(√\delta \cdot Pol)\) computation and \(O(log(\sqrt{δ}))\) storage [42], where \(δ = a - b\) if \(RSS \in [a, b]\) and \(Pol\) is the number of point operations in Pollard Lambda algorithm which varies depending on algorithm implementation used. For \(SU\)'s overhead, the left columns shows the cost for a normal \(SU\) in ECEG and the right column shows the cost of the \(SU\) that plays the role of a gateway in ECEG. (v) \(TGECDH\): It permits the alteration of group membership (i.e., join/leave), on average \(O(log(\sqrt{δ}))\) communication and computation (i.e., ECC scalar multiplication) [43]. (vi) OPE: We rely on OPE scheme proposed by Boldyreva [21] for our evaluation because of its popularity and public implementation but our schemes can use any secure OPE scheme (e.g., [21], [39], [40]) as a building block. (vi) \(E\): We rely on AES [44] as our \((E, D)\) for our cost analysis.

VII. PERFORMANCE EVALUATION

We now evaluate our proposed schemes, LP-2PSS and LP-3PSS, by comparing LP-2PSS to its predecessor LPOS [17], ECEG and PPSS as these schemes are all designed for the sensing architecture without a gateway, and comparing LP-3PSS to PDAFT as both are designed for the sensing architecture with a gateway.

A. Existing Approaches: PPSS, ECEG, and PDAFT

PPSS [5] uses secret sharing and the Privacy Preserving Aggregation (PPA) process proposed in [45] to hide the content of specific sensing reports and uses dummy report injections to cope with the DLP attack.

In ECEG, \(SU\)'s encrypt their RSSSs with \(FC\)'s ECEG public key. One of the nodes aggregates these ciphertexts including its own and then sends the aggregated result to \(FC\). The \(FC\) then decrypts the aggregated result with its ECEG private key and makes the final decision.

PDAFT [16] combines Paillier cryptosystem [46] with Shamir’s secret sharing [47], where a set of smart meters sense the consumption of different households, encrypt their reports using Paillier, then send them to a gateway. The gateway multiplies these reports and forwards the result to the control center, which selects a number of servers (among all servers) to cooperate in order to decrypt the aggregated result. PDAFT requires a dedicated gateway, just like LP-3PSS, to collect the encrypted data, and a minimum number of working servers in the control center to decrypt the aggregated result.

B. Performance Analysis and Comparison

We focus on computational overhead and computational overhead. We consider the overhead incurred during the sensing operations but not that related to system initialization (e.g., key establishment), where most of the computation and communication is done offline. We model the membership change events in the network as a random process \(R\) that takes on 0 and 1, and whose average is \(μ\). \(R = 0\) means that no change occurred in the network and \(R = 1\) means that some \(SU\)'s left/joined the sensing task. Let \(β(t)\) be a function that models the average number of \(SU\)'s that join the sensing at the current sensing period \(t\).

We precisely that our performance analysis is not based on a simulation but rather on measuring the computational and communication overhead involved in the cryptographic operations that we deployed, like YM.ECEG protocol and OPE. This gives us an idea about how our schemes perform compared to existent approaches in terms of incurred overhead. The execution times of the different primitives and protocols were measured on a laptop running Ubuntu 14.10 with 8GB of RAM and a core M 1.3 GHz Intel processor, with cryptographic libraries MIRACL [48], Crypto++ [49] and Louismullie’s Ruby implementation of OPE [50]. C++ implementations that we developed for the optimized ECEG and the YM.ECEG schemes will be provided for public use.

Computational Overhead: Table II provides an analytical computational overhead comparison including the details of variables, parameters and the overhead of building blocks.

In LP-2PSS, \(FC\) requires only a logarithmic number of YM.ECEG executions. An \(SU\) requires a small constant number of Point additions \(PAddQ\), one OPE encryption and group key update, which is necessary only \(μ\) percent of the time when there is a change in the network (with only a logarithmic overhead in the number of \(SU\)'s). The signature verification operation, that new \(SU\)'s have to perform upon joining the sensing, is extremely fast in most of the digital signature schemes compared to the system overall computational overhead that we study in this section. This makes the delay introduced by the digital signature negligible compared to the overall computational overhead incurred by LP-2PSS regardless of the used digital signature scheme. Thus, we don’t consider this delay in our evaluation. This makes LP-2PSS much more efficient than ECEG and PPSS, especially for a relatively large number of \(SU\)'s.

In LP-3PSS, \(FC\) requires only a small constant number not only robust against collusion among \(SU\)'s themselves, but also against collusion between \(FC\) and compromised \(SU\)'s.
of \((D, E, OPE)\) operations. An \(SU\) requires one \(OPE\) and \(E\) encryptions of its \(RSS\). Finally, \(GW\) requires one \(D\) operation per \(SU\) and one \(E\) of vector \(b\). All computations in \(LP-3PSS\) rely on only symmetric cryptography, which makes it the most computationally efficient scheme among all alternatives as discussed below.

For illustration purpose, we plot in Figure 4 the system end-to-end computational overhead of the different schemes. Figure 4(a) shows that \(LP-2PSS\) incurs an overhead that is comparable to that incurred by \(ECEG\), but much lower than that incurred by \(PPSS\). Figure 4(a) shows also that \(LP-2PSS\) performs slightly better than its predecessor \(LPOS\).

Figure 4(b) shows that \(LP-3PSS\) is several order of magnitudes faster than \(PDAFT\) for any number of \(SUs\).

![Figure 4: Computation Overhead, \(\beta = 5, \mu = 20\% \& \kappa = 80\)](image)

Notice that the key generation and signing operations are done only once at the beginning of the protocol as \(\tau\), and thus \(T\), should be static over time unless a dramatic change in the system environment occurs which leads to the re-execution of the \(LP-2PSS\)’s initialization phase. That is why these operations are not counted for the operational overhead of our scheme.

We also study the impact of the security parameter, \(\kappa\), which controls the encryption key length, by varying it in accordance with \(NIST\)’s recommendations [41]. Note that this assesses the suitability of a scheme for a long term deployment in a stable networking infrastructure. Figure 5, evaluating the schemes under three values of \(\kappa\), shows that our schemes are the least impacted by increasing security parameters. It also shows that \(LP-3PSS\) is significantly more efficient than \(PDAFT\) in terms of computation overhead for all entities. Note that our schemes achieve a delay, which is well below the 2-second computation delay required by \(IEEE 802.22\) standard for TV white space management [26]. This standard requires that the system handles dynamism in the network and that \(RSS\) values lie within the interval \([-104, 23.5]dB\) and are encoded under 8 bits. Figures 5(a) & 5(c) show the gain in computational performance of \(LP-2PSS\) over \(LPOS\) especially for high security levels and from the \(SUs\) side.

### Communication Overhead:

Table III provides the analytical communication overhead comparison. \(LP-2PSS\) requires \(\log(n)\) message exchanges for \(YM\) protocol, \(n\) \(OPE\) ciphertexts and \(\log(n)\) messages for group key update (only needed \(\mu\) percent of the time when there is a membership change). If some \(SUs\) join the \(CRN\), \(LP-2PSS\) requires sharing the digital signature \(\sigma\) of message \(T\) and the public key \(PK_{DS}\) used to construct this signature with \(\beta\) new \(SUs\). \(LP-3PSS\) requires \((n+1)\ \epsilon\) ciphertexts and single \(\zeta\), which are significantly smaller than the values transmitted by \(PDAFT\).

We further compare our schemes with their counterparts in terms of communication overhead in Figure 6. Figure 6(a) illustrates the communication overhead induced by \(LP-2PSS\) using different digital signature schemes (\(HORS, ECDSA\) and \(NTRU\)) compared to the original scheme, \(LPOS\), and also to existent approaches \(PPSS\) and \(ECEG\). This Figure shows that \(LP-2PSS\) is more efficient than \(PPSS\) and \(ECEG\) due to the use of elliptic curve cryptography with smaller key sizes. Using \(ECDSA\) or \(NTRU\) seems to be the best option in terms of communication overhead as expected. For a large number of \(SUs\), using a digital signature scheme with large signature size like \(HORS\) does not prevent \(LP-2PSS-HORS\) from performing better than existent approaches especially for a large number of \(SUs\). Figure 6(b) shows that \(LP-3PSS\) has the smallest communication overhead when compared with \(PDAFT\), since it relies on symmetric cryptography only. \(PPSS\) and \(PDAFT\) have a very high communication overhead due to the use of expensive public key encryptions (e.g., Pailler [46]).

We also study and show in Figure 7 the impact of the security parameter, \(\kappa\), on the communication overhead. Note that the performance gap between our schemes and their counterparts drastically grows when \(\kappa\) is increased, showing the suitability of our schemes for long term deployment. Our schemes possess this desirable feature, thanks to their innovative use of compact cryptographic primitives. Figures 6(a) & 7(a) show again how efficient \(LP-2PSS\) is compared to the original \(LPOS\) in terms of communication overhead.

Overall, our performance analysis indicates that \(LP-3PSS\) is more efficient than \(LP-2PSS\), and significantly more efficient than all other counterpart schemes in terms of computation and communication overhead, even for increased values of the security parameters, but with the cost of including an additional entity. Moreover, Figures 5 \& 7 show that our schemes are impacted much less by increased security parameters when compared to existing alternatives, and therefore are ideal for long term deployment. Note that our performance analysis lacks the evaluation of the \(CPUS\)-based version of \(LP-3PSS\) due to the fact that this hardware is very expensive.

### VIII. Conclusion

We developed two efficient schemes for cooperative spectrum sensing that protect the location privacy of \(SUs\) with a low cryptographic overhead while guaranteeing an efficient spectrum sensing. Our schemes are secure and robust against \(SUs\)’ dynamism, failures, and maliciousness. Our performance analysis indicates that our schemes outperform existing alternatives in various metrics.

### References

Communication Overhead (KB) for varying $n = 1200$, $\beta = 5$ & $\mu = 20\%$

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Communication</th>
</tr>
</thead>
<tbody>
<tr>
<td>LP-2PSS</td>
<td>$2\gamma \cdot</td>
</tr>
<tr>
<td>ECEG</td>
<td>$4</td>
</tr>
<tr>
<td>PSS</td>
<td>$</td>
</tr>
<tr>
<td>LP-3PSS</td>
<td>$(n + 1) \cdot e_{E}$</td>
</tr>
<tr>
<td>PDAFT</td>
<td>$</td>
</tr>
</tbody>
</table>

$e_{OPE} = 128$ bits: maximum ciphertext size obtained under $OPE$ encryption, $e_E$: size of ciphertext under $E$. $|\sigma|$ and $|PK_{DS}|$ are respectively the size of the digital signature and the public key of the digital signature scheme DS.

Fig. 5: Computational Overhead Variation with Respect to Security Parameter $\kappa$ for $n = 1200$, $\beta = 5$ & $\mu = 20\%$

![Graph of Communication Overhead Variation](image)

Fig. 6: Communication Overhead, $\beta = 5$, $\mu = 20\%$ & $\kappa = 80$

![Graph showing Communication Overhead](image)

Fig. 7: Communication Overhead with varying $\kappa = (80, 112, 128)$ for $n = 1200$, $\beta = 5$ & $\mu = 20\%$.

![Graph showing Communication Overhead with varying $\kappa$](image)


