This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCCN.2017.2702163, IEEE Transactions on Cognitive Communications and Networking

Location Privacy Preservation in Database-driven Wireless Cognitive Networks through Encrypted Probabilistic Data Structures

Mohamed Grissa, Attila A. Yavuz, and Bechir Hamdaoui
Oregon State University, grissam,attila.yavuz,hamdaoui@oregonstate.edu

***Abstract*—In this paper, we propose new location privacy preserving schemes for database-driven cognitive radio networks (CRNs) that protect secondary users' (SUs) location privacy while allowing them to learn spectrum availability in their vicinity. Our schemes harness probabilistic set membership data structures to exploit the structured nature of spectrum databases (DBs) and SUs' queries. This enables us to create a compact representation of DB that could be queried by SUs without having to share their location with DB, thus guaranteeing their location privacy. Our proposed schemes offer different cost-performance characteristics. Our first scheme relies on a simple yet powerful two-party protocol that achieves unconditional security with a plausible communication overhead by making DB send a compacted version of its content to SU which needs only to query this data structure to learn spectrum availability. Our second scheme achieves significantly lower communication and computation overhead for SUs, but requires an additional architectural entity which receives the compacted version of the database and fetches the spectrum availability information in lieu of SUs to alleviate the overhead on the latter. We show that our schemes are secure, and also demonstrate that they offer significant advantages over existing alternatives for various performance and/or security metrics.**

***Index Terms*—Database-driven spectrum availability, location privacy preservation, cognitive radio networks, set membership data structures.**

I. Introduction

Cognitive radio networks (CRNs) have emerged as a key technology for addressing the problem of spectrum utilization inefficiency [2]–[8]. CRNs allow unlicensed users, also referred to as *secondary users (SUs)*, to access licensed frequency bands opportunistically, so long as doing so does not harm licensed users, also referred to as *primary users (PUs)*. In order to enable SUs to identify vacant frequency bands, also called white spaces, the federal communications commission (FCC) has adopted two main approaches: *spectrum sensing-based approach* and *geo-location database-driven approach*.

In the sensing-based approach [9], SUs themselves sense the licensed channels to decide whether a channel is available prior to using it so as to avoid harming PUs. In the database-driven approach, SUs rely on a geo-location database (DB) to obtain channel availability information. For this, SUs are required to be equipped with GPS devices so as to be able to query DB on a regular basis using their exact locations. Upon receipt of a query, DB returns to SU the list of available channels in its vicinity, as well as the transmission parameters that are to be used by SU. This database-driven approach has advantages over the sensing-based approach. First, it pushes the responsibility and complexity of complying with spectrum policies to DB. Second, it eases the adoption of policy changes by limiting updates to just a handful number of databases, as opposed to updating large numbers of devices [10].

Companies, like Google and Microsoft, are selected by FCC to administrate these geo-location databases, following the guidelines provided by *PAWS (Protocol to Access White-Space)* [10]. *PAWS* protocol defines guidelines and operational requirements for both the spectrum database and the SUs querying it. These requirements include: SUs need to be equipped with geo-location capabilities, SUs must query DB with their specific location to check channel availability before starting their transmissions, DB must register SUs and manage their access to the spectrum, DB must respond to SUs' queries with the list of available channels in their vicinity along with the appropriate transmission parameters.

Despite their effectiveness in improving spectrum utilization efficiency, database-driven CRNs suffer from serious security and privacy threats. The disclosure of location privacy of SUs has been one of such threats to SUs when it comes to obtaining spectrum availability from DBs. This is simply because SUs have to share their locations with DB to learn about spectrum availability. The fine-grained location, when combined with publicly available information, can lead to even greater private information leakage. For example, it can be used to infer private information like shopping patterns, preferences, behavior and beliefs, etc. [11]. Being aware of such potential privacy threats, SUs may refuse to rely on DB for spectrum availability information. Therefore, there is a critical need for location-privacy preserving schemes for database-driven spectrum access.

A. Our Contribution

In this paper, we propose two location privacy-preserving schemes for database-driven CRNs with different performance and architectural benefits. The first scheme, *location privacy in database-driven CRNs* ($LPDB$), provides optimal location privacy to SUs within DB's coverage area by leveraging *set membership data structures* (used to test whether an element is a member of a set) to construct a compact version of DB. The second scheme, *$LPDB$ with two servers* ($LPDBQS$), minimizes the overhead at SU's side at the cost of deploying an additional entity in the network. The cost-performance tradeoff gives more options to system designers

This manuscript is an extension of [1], published in: Computer Networks and Information Security (WSCNIS), 2015 World Symposium on.

to decide which topology and which approach is more suitable to their specific requirements.

Both approaches exploit two important facts: (i) Spectrum databases are highly structured [10]; and (ii) SUs queries contain always the same device-specific characteristics (e.g., device type, antenna hight, frequency range, etc.) [10]. The highly structured property of the database refers to the fact that DB's structure is usually agreed upon by the FCC and the database administrators, like Google, Microsoft, etc, and that the queries and messages exchanged by DB and SUs have a specific format in terms of what data they include. This well-defined information is available to both database administrators and SUs which allows them to have an idea on what kind of data the other party will include in its query/response, and also to compact both DB's content and the queries using probabilistic data structures. In fact, and as recommended by the *PAWS* standard, the database should always reply to SUs with a set of predetermined information. This allows DB to compact its content to include only this information, which significantly reduces queried data sizes, and enables SUs to emulate DB's response when querying the probabilistic data structure as we show next.

A desirable property of our schemes is their simplicity that is expected to facilitate their applicability in real-life applications. Our proposed schemes offer various cost-performance trade-offs that can meet the requirements of different applications. We study these tradeoffs and show that high privacy and better performance for SUs' 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. In addition, we study the impact of system parameters on the performances of our proposed schemes, and compare them against those obtained via existing approaches.

Compared to our preliminary work [1], this paper provides: (i) A new scheme, $LPDBQS$, with multiple deployment scenarios, that improves the overhead on SUs' side by relying on an extra architectural entity; (ii) An improvement to our previously proposed scheme, $LPDB$, by incorporating spectrum sensing to reduce the impact of the false positive rate of the used *set membership data structure* on spectrum availability information's accuracy; (iii) A detailed security analysis of the proposed schemes; and (iv) More detailed performance analysis with more evaluation metrics.

The remainder of this paper is organized as follows: We discuss related work in Section II. We present our system and threat models along with our security assumptions in Section III. Section IV provides a brief overview of the *set membership data structure* that we use in this paper. In Section V-A, we present our first scheme $LPDB$. We describe our second scheme $LPDBQS$ in Section V-B. We evaluate and analyze the performance of the proposed schemes in Section VII, and conclude our work in Section VIII.

II. Related Work

Despite its importance, the location privacy issue in CRNs only recently gained interest from the research community [12]. Some works focused on addressing this issue in the context of collaborative spectrum sensing [13]–[17] while others focused on addressing it in the context of dynamic spectrum auction [18]. However, these works are not within the scope of this paper as we focus on the location privacy issue in database-driven CRNs.

Protecting SUs' location privacy in database-driven CRNs is a very challenging task, since SUs are required to provide their physical locations to DB in order for them to be able to learn about spectrum opportunities in their vicinities. Recently developed techniques mostly adopt either the *k-anonymity* [19], *Private Information Retrieval (PIR)* [20], or *differential privacy* [21] concepts. However, direct adaptation of such concepts yield either insecure or extremely costly results. For instance, *k-anonymity* guarantees that SU's location is indistinguishable among a set of k points, which could be achieved through the use of dummy locations by generating $k-1$ properly selected dummy points, and performing k queries to DB using both the real and dummy locations. For example, Zhang et al. [22] rely on this concept to make each SU query DB by sending a square cloak region that includes its actual location. Their approach makes a tradeoff between providing high location privacy and maximizing some utility, which makes it suffer from the fact that achieving a high location privacy level results in a decrease in spectrum utility.

PIR, on the other hand, allows a client to obtain information from a database while preventing the database from learning which data is being retrieved. Several approaches have used this approach. For instance, Gao et al. [23] propose a PIR-based approach, termed $PriSpectrum$, that relies on the PIR scheme of Trostle et al. [24] to defend against a newly identified attack that exploits spectrum utilization pattern to localize SUs. Troja et al. [25], [26] propose two other PIR-based approaches that try to minimize the number of PIR queries by either allowing SUs to share their availability information with other SUs [25] or by exploiting trajectory information to make SUs retrieve information for their current and future positions in the same query [26]. Despite their merit in providing location privacy to SUs these PIR-based approaches incur high overhead especially in terms of computation.

Using *differential privacy*, Zhang et al. [27] rely on the *ϵ-geo-indistinguishability* mechanism [28] to make SUs obfuscate their location. However, such a mechanism introduces noise to SU's location which may impact the accuracy of the spectrum availability information retrieved.

There have also been other privacy-enhancing technologies (PETs) that are not specific to CRNs but are designed to enable private queries over a database in general. However, many of these PETs are designed for applications that do not fit in the context of CRNs. For instance, oblivious random access memory (ORAM) [29] aims to enable a user to outsource its encrypted data to a database and to offer him/her the possibility to access this data while hiding the access patterns from the database [12]. Searchable symmetric encryption [30] is another PET that is largely deployed to privately outsource one's data to another party while maintaining the ability to selectively search over it [12]. These PETs are designed for protecting queries and searches over data that is outsourced to a database, which is completely different from the CRN scenario where the queried data belong to the database itself.

III. System Model and Security Assumptions

A. Database-driven CRN Model

We first consider a CRN that consists of a set of SUs and a geo-location database (DB). SUs are assumed to be enabled with GPS and spectrum sensing capabilities, and to have access to DB to obtain spectrum availability information within its operation area. To learn about spectrum availability, a SU queries DB by including its location and its device characteristics. DB responds with a list of available channels at the specified location and a set of parameters for transmission over those channels. SU then selects and uses one of the returned channels. While using the channel, SU needs to recheck its availability on a daily basis or whenever it changes its location by 100 meters as mandated by *PAWS* [10].

We then investigate incorporating a third entity to the network along with DB and SUs. This entity, referred to as *query server* (QS), has a dedicated high throughput link with DB. QS is used to guarantee computational location privacy while reducing the computational and communication overhead especially on SUs' side.

B. Security Model and Assumptions

DB and QS are assumed to be honest but curious. That is, DB and QS follow the protocol honestly but may try to infer information on the input of other parties beyond what the output of the protocol reveals. Specifically, our objective is to prevent these two entities from learning SUs' location. Therefore, our security assumptions are as follows:

Security Assumption 1. *DB and QS do not modify the integrity of their input. That is, (i) DB does not maliciously change SU's query's content; (ii) QS does not modify the input that it receives from DB or SU.*

Security Assumption 2. *DB and QS do not collude with each other to infer the location of SUs from their queries.*

We further assume that the communication between different entities is secured by a cryptographic protocol like TLS [31] as suggested by *PAWS* [10]. This eliminates the risk of an adversary trying to eavesdrop the communication.

IV. Set Membership Data Structures

Our proposed privacy-preserving schemes utilize *set membership data structures* to exploit the highly structured property of DB. There are several data structures that are designed for set membership tests, e.g. *bloom filter* [32], *cuckoo filter* [33], etc. However, in this paper, we opt for *cuckoo filter* as the building block of our schemes. We use *cuckoo filter* to construct a compact representation of the spectrum geo-location database as explained in Sections V-A & V-B. What motivates our choice is that *cuckoo filter* offers the highest space efficiency among its current well known alternatives, such as *bloom filters*. Besides, it has been proven to be more efficient than these alternatives especially for large sets. Finally, the *cuckoo filter* enjoys fast $Lookup$ and $Insert$ operations that are beneficial to our schemes.

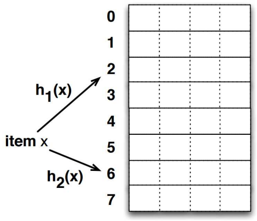


Figure 1: Cuckoo Filter: 2 hashes per item, 8 buckets each containing 4 entries

A cuckoo filter [33] uses *cuckoo hashing* [34] and is designed to serve applications that need to store a large number of items while targeting low false positive rates and requiring storage space smaller than that required by bloom filters. A false positive occurs when the membership test returns that an item exists in the *cuckoo filter* (i.e., belongs to the set) while it actually does not. A false negative, on the other hand, occurs when the membership test returns that an item does not exist while it actually exists. In *cuckoo filters*, false positives are possible, but false negatives are not, and the target false positive rate, denoted throughout this paper by ϵ, can be controlled but has a direct impact on the filter's size. Figure 1 shows an example of a *cuckoo filter* that uses two hashes per item and contains 8 buckets each with 4 entries.

A cuckoo filter has mainly two functions: An $Insert$ function that stores items in the filter, and a $Lookup$ function that checks whether an item exists in the filter. In the $Insert$ operation, cuckoo filter stores a fingerprint f of each item x, as opposed to storing the item itself. The space cost, in bits, of storing one item in the cuckoo filter using the $Insert$ function depends on the target false positive rate ϵ and is given by $(log_2(1/\epsilon)+log_2(2\beta))/\alpha$ where α is the load factor of the filter which defines its maximum capacity, and β is the number of entries/slots per bucket. Once the maximum feasible, α, is reached, insertions are likely to fail, and hence, the filter must expand in order to store more items [33]. The $Lookup$ operation is performed by first computing a fingerprint of the desired item and two indexes, representing the potential locations (or buckets) of this item in the filter, and then checking whether these two locations contain the item.

V. Proposed Schemes

In this section, we describe our proposed schemes. The first scheme, $LPDB$, is simple as it involves only two parties, SUs and DB, and provides unconditional location privacy to SUs within the coverage area of DB. The second scheme, $LPDBQS$, offers computational privacy with a significantly reduced overhead on SUs' side compared to $LPDB$, but at the cost of introducing an extra architectural entity.

Since we are unable to access the actual spectrum database, we relied on two sources to have an estimate of this structure: First, we have relied on the recommendation of the PAWS standard [10], which defines the interaction between SUs and DB and what information they should exchange. Second, we used graphical web interfaces provided to the public by white

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCCN.2017.2702163, IEEE Transactions on Cognitive Communications and Networking

space database operators like Google [35], Microsoft [36], iconectiv [37], etc. These web interfaces comply with PAWS recommendation and allow an interested user to specify a location of interest and learn spectrum availability in that location to emulate the interaction between a SU and DB in real world. While the purpose of these interfaces was initially to provide a working platform as a showcase for FCC to acquire approval for operating spectrum database, we believe it has enough information to enable us to estimate the structure of the database and SUs' queries.

As required by PAWS, SUs must be registered with DB to be able to query it for spectrum availability. Registered SU starts by sending an initialization query to DB which replies by informing the SU of specific parameterized-rule values. These parameters include time periods beyond which the SU must update its available-spectrum data, and maximum location change before needing to query DB again. Afterwards, SU queries DB with an available spectrum query which contains its geolocation, device identifier, capabilities (to limit DB's response to only compatible channels) and antenna characteristics (e.g. antenna height and type). DB then replies with the set of available channels in the SU's location along with permissible power levels for each channel.

Based on these interactions between SU and DB, which we learned from *PAWS* and the database web interfaces, we estimate the structure of DB to be as illustrated in Table I. Each row corresponds to a different combination of location pairs ($locX$,$locY$) and channel chn. One location may contain several available channels at the same time. Note that even if the real structure deviates from the one illustrated in Table I (e.g. more/different attributes, more tables, etc), our schemes can be adapted to the new structure of both the queries and the database by designing or using a different probabilistic data structure(s). Also, even in this case, the *PAWS* standard requires that DB always replies to spectrum availability queries with a set of predetermined values that have to be in the database no matter what structure it has. In that case, DB only needs to insert these values in the cuckoo filter and this could be done independently from the database structure.

TABLE I: Simplified example of DB's structure

	$locX$	$locY$	ts	chn	avl	par^1	$\cdots$	par^n
row_1	$locX_1$	$locY_1$	t	chn_1	0	par_1^1	$\cdots$	par_1^n
row_2	$locX_1$	$locY_1$	t	chn_2	1	par_2^1	$\cdots$	par_2^n
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
row_i	$locX_2$	$locY_2$	t	chn_1	1	par_i^1	$\cdots$	par_i^n
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
row_r	$locX_r$	$locY_r$	t	chn_1	0	par_r^1	$\cdots$	par_r^n

$avl = 1$ means chn is available and $avl = 0$ means chn is not available.

A. *LPDB*

In this section, we describe our basic scheme, which is referred to as *location privacy in database-driven CRNs* ($LPDB$). The novelty of $LPDB$ lies in the use of set membership data structures to construct a compact (space efficient) representation of DB that can be sent to querying SUs to inform them about spectrum availability.

In our scheme, instead of sending its location, a SU sends only its characteristics (e.g., its device type, its antenna type, etc.), as specified by PAWS [10], to DB, which then uses them to retrieve the corresponding entries in all possible locations. DB then puts these entries in a *cuckoo filter* and sends it to SU. Upon receiving this filter, SU constructs a query that includes its characteristic information, its location, and one of the possible channels with its associated parameters. SU then looks up this query in the received *cuckoo filter* to see whether that channel is available in its current location.

Parameters that are inserted in the response of DB may include the location, time stamps, the available channels, and the transmission power to be considered when using those channels. SU's characteristics and DB parameters could be agreed upon beforehand between DB and SUs to make sure that SU queries the *cuckoo filter* with the right parameters.

The proposed $LPDB$ scheme is illustrated in Algorithm 1, and briefly described as follows: First, each SU starts by constructing $query$ to be sent to DB by including a set of characteristics, which are specific to the querying device, along with a time stamp ts. DB then retrieves the entries that correspond to $query$ and constructs a *cuckoo filter* CF (which could be done offline). Since DB contains availability status for each channel in each location, the number of entries satisfying $query$ will still be huge and one way to further reduce it is to retrieve only the information about available channels and ignore the other ones. Afterwards, DB concatenates the data in each row to construct x_j as illustrated in Step 6, inserts it to CF and sends CF to SU.

SU constructs a string y by concatenating its location coordinates with a combination of one channel and its possible transmission parameters and tries to find whether y exists in CF by using the $Lookup$ operation of CF. SU keeps changing the channel and the associated parameters until it finds the string y in CF or until SU tries all possible channels. Note that, depending on the false positive rate ϵ of CF, even if the $Lookup$ operation returns $True$ it does not necessarily mean that the specified channel is available. Setting ϵ to be very small makes the probability of having such a scenario very small, thus reduces the risk of using a busy channel, but this cannot be done without increasing the size of CF. To further reduce the risk of falling into this case, we have also included an additional sensing step to confirm the query's result and give more accurate information about the status of the channel of interest. If SU finds y in CF, then it needs to sense the specific channel found in y to confirm its availability. SU can conclude that this channel is free and thus can use it only if the sensing result coincides with CF's response.

If, after trying all possible combinations, SU does not find y in CF, this means that no channel is available in the specified location as *cuckoo filters* do not incur any false negatives.

When the size of DB is not large, then $LPDB$ works well (as will be shown Section VII) by providing unconditional privacy with reasonably small amounts of overhead. However, a scalability issue may arise when the location resolution is very small (resolution used in DB could be as small as 50 meters) and/or the area covered by DB is large (e.g. at the scale of a country). In this case, the number of locations, and

Algorithm 1 $LPDB$ Algorithm

```
1: $SU$ queries $DB$ with $query \leftarrow f(char, ts)$;
2: $DB$ retrieves $resp$ containing $r$ entries satisfying $query$;
3: $DB$ constructs $CF$;
4: for $j = 1, \ldots, r$ do
5:     if $avl_j = 1$ then
6:         $x_j \leftarrow (locX_j \| locY_j \| chn \| ts \| \ldots)$;
7:         $DB$ inserts $x_j$ into $CF$: $CF.Insert(x_j)$;
8: $DB$ sends $CF$ to $SU$;
9: $SU$ initializes $decision \leftarrow$ Channel is busy
10: for all possible combinations of $par$ do
11:     $SU$ computes $y \leftarrow (locX \| locY \| chn_i \| ts \| \ldots \| par^n)$;
12:     if $CF.Lookup(y)$ then
13:         $SU$ senses $chn$;
14:         if $Sensing(chn) \leftarrow$ available then
15:             $decision \leftarrow chn$ is available; break;
    return $decision$
```

thus the number of entries in DB, can be large, and then even after relying on the *cuckoo filter*, the size of the data to be transmitted may still be impractical (e.g. in the order of gigabytes). This depends on the desired resolution and DB's covered area. Next, we present a discussion about a possible way to deal with this scalability issue in the case of a very large DB.

Performance-privacy tradeoff: As discussed before, $LPDB$ may suffer from a scalability issue when the size of DB's coverage area is very large. We can address this issue through the following observation. When the covered area is large and/or the location resolution is small, allowing DB to learn one of SU's coordinates can drastically reduce the number of entries that DB retrieves. This leads to considerably reduce the size of CF to be transmitted, thus making the approach scalable. Interestingly, in the case of large areas, revealing one of SU's coordinates does not make it any easier for DB to infer SU's location. To illustrate this, let's for example assume that DB covers the entire surface of the United States, as shown in Figure 2. Allowing DB to learn one coordinate (e.g the latitude) means that it can only learn that SU is located somewhere on the blue line that spans the latitude of the whole country. But since DB does not know the longitude of SU, then knowing the latitude only does not offer any practical information about SU's location.

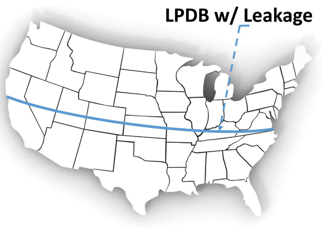


Figure 2: Location Leakage

This, as shown in Section VII-A, drastically reduces the size of CF transmitted by DB at the cost of loosing the unconditional location privacy of SUs. However, when the coverage area of DB is large, even revealing one of the coordinates still achieves high location privacy of SUs. Indeed, since databases (like those managed by Microsoft and Google) may cover an entire nation of the size of the United States, the revealed information is not sufficient to localize SU, yet, this reduces our scheme's overhead substantially. The example of the United States in Figure 2 shows that our scheme can offer high privacy even when one of the coordinates is revealed. Throughout, we refer to this variant of our scheme as $LPDB$ with leakage.

It is worth reiterating that when the covered area is not too large, then the size of the *cuckoo filter* is practical and there is no need to reveal one of SU's coordinates. In this case, our scheme, $LPDB$, provides unconditional privacy without incurring much overhead. The system regulator can decide about which approach to follow depending on the system constraints and the size DB's covered area.

B. $LPDBQS$

In this section, we propose a new scheme, $LPDBQS$, which offers better performance at SUs' side than that of $LPDB$. This comes at the cost of deploying an additional entity, referred to as *query server* (QS), and having a computational security as opposed to unconditional. QS is introduced to handle SUs' queries instead of DB itself, which prevents DB from learning information related to SUs' location information. QS learns nothing but secure messages sent by SUs to check the availability of a specific channel.

- *Intuition*: We introduce QS to avoid sending CF, which might be large, to SU. Instead, CF, that contains $HMAC$ secure entries inserted by DB using a secret key provided by SU, is sent to QS through a high throughput link pre-established with DB. SU just needs to query, using $HMAC$ messages, QS which looks for its queries in CF. Using $HMAC$, SU can hide the content of the query string, which includes its location information, among others, from QS which ignores the key used to construct the hashed query and the CF. This not only prevents QS from learning the query's content but also the entry that matches it in the filter. As most of the computation and communication overhead are incurred by both DB and QS, this scheme is the most efficient in terms of overhead incurred by SUs. $LPDBQS$ is summarized in Algorithm 2 and described in the following.

First, SU starts by sending a secret key k, pre-established beforehand, along with its device characteristics to DB. DB then retrieves only the entries that have available channels and that also comply with the device characteristics of the querying SU. Afterwards, DB constructs a *cuckoo filter* CF_k and inserts into it the entries retrieved in the previous step as shown in Steps 4-7. $Insert_{HMAC_k}$, in Step 7, is a modified version of the $Insert$ procedure, where the fingerprint is replaced by an $HMAC_k$ function. DB uses $HMAC_k$ with the secret key k, provided by SU, to construct hashed entries and insert them to CF_k. DB then sends CF_k to QS via the high speed link that connects DB to QS.

SU constructs a string y by concatenating its location coordinates with a combination of one channel and its possible

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCCN.2017.2702163, IEEE Transactions on Cognitive Communications and Networking

Algorithm 2 *LPDBQS* Algorithm

```
1:  $SU$ queries $DB$ with $query \leftarrow f(k, char, ts)$;
2:  $DB$ retrieves $resp$ containing $r$ entries satisfying $char$;
3:  $DB$ constructs $CF_k$;
4:  for $j = 1, \ldots, r$ do
5:      if $avl_j = 1$ then
6:          $x_j \leftarrow (locX_j \| locY_j \| ts \| \ldots \| row_j(c))$;
7:          $CF_k.Insert_{HMAC_k}(x_j)$;
8:  $DB$ sends $CF_k$ to $QS$ over a high throughput link;
9:  $SU$ initializes $decision \leftarrow$ Channel is busy
10: for all possible combinations of $par$ do
11:     $SU$ computes $y \leftarrow (locX \| locY \| chn \| ts \| \ldots \| par^n)$;
12:     $SU$ computes $y_k \leftarrow HMAC_k(y)$ and sends it to $QS$;
13:     $QS$ looks up for $y_k$ in $CF_k$ using $Lookup$;
14:     if $CF_k.Lookup(y_k)$ then
15:         $SU$ senses $chn$;
16:         if $Sensing(chn) \leftarrow$ available then
17:             $decision \leftarrow chn$ is available; break;
        return $decision$
```

transmission parameters. Subsequently, SU hashes y using an $HMAC$ with the secret key k and sends the new value y_k to QS to find out whether CF_k of QS contains y_k. If the query's combination is found in CF_k, then SU needs to take one further step: It senses the channel that was included in the query. If the result of the sensing complies with the outcome of the $Lookup$ operation in CF_k, then SU can conclude that this channel is available and, thus, it can use it for its future transmissions. In this case, SU can stop querying QS. The sensing operation is added to confirm the outcome of querying the *cuckoo filter* and overcome the risk of falling into the case of a false positive result that would eventually make SU interfere with primary transmissions. In case the sensing result is different from the outcome of the $Lookup$ operation, then SU keeps changing the channel and the associated parameters until QS finds y_k in CF_k or until SU tries all possible channels and combinations.

This scheme considerably reduces the overhead perceived by SUs, as much of the computation is performed offline by DB, and SUs do not need to download the *cuckoo filters* which are only sent to QS over a high throughput link.

If DB knows the possible device characteristics of the querying SUs, this can help to further reduce the incurred overhead. Indeed, DB can pre-compute several *cuckoo filters* for each possible combination of potential device parameters offline by relying on a set of secret keys $\mathcal{K} = \{k_1, \ldots, k_z\}$ that it generated beforehand. For each combination of parameters, DB constructs multiple CF_k with different keys from $\mathcal{K}$ to make sure that each SU uses a different filter. SUs are not required to generate their own keys as in the previous variant. Whenever a SU queries DB for spectrum opportunities, DB shares a secret key k with it and sends the corresponding CF_k to QS. SU uses k to construct its hashed strings and query CF_k of QS just like in Algorithm 2.

Leveraging a Secure Hardware: As long as DB and QS do not collude, as stated in Security Assumption 2, neither of them can infer the coordinates of SUs from the keyed one-way function output. To mitigate the non-collusion requirement between FC and QS, $LPDBQS$ could be implemented in a slightly different way by relying on a secure hardware (e.g., a secure co-processor or a trusted platform module) that can perform cryptographic operations without exposing its embedded private key. This hardware can be deployed inside DB itself and play the role of QS. Such a high-end secure hardware is physically shielded from penetration [38], and any tampering from the adversary, potentially DB, triggers the automatic erasure of sensitive memory areas containing critical secrets [39]. When a secure hardware meets the FIPS 140-2 level 4 [40] physical security requirements, it becomes infeasible for FC to tamper with the operations executed by this hardware. DB sends the *cuckoo filters* to this hardware, and SUs have to query this hardware to learn about spectrum availabilities.

VI. Security Analysis

In this section, we analyze the security of our proposed schemes $LPDB$ and $LPDBQS$.

Theorem 1. *Under Security Assumptions 1 and 2, $LPDB$ does not leak any information on SUs' location.*

Proof: We construct a history list $\mathcal{H}$ of each entity's knowledge about SUs' information during the execution of $LPDB$.

$\boldsymbol{SU}$. A SU cannot learn anything about other SUs information nor the filters $\{CF_{i,t}\}_{i=1,t=t_0}^{n-1,t_f}$ that they receive from DB as the communication between each SU and DB is secured, i.e. $\mathcal{H}_{SU} = \emptyset$. Note that, even if, a SU would learn the filters of other SUs, i.e. $\mathcal{H}_{SU} = \{CF_{i,t}\}_{i=1,t=t_0}^{n-1,t_f}$, $\mathcal{H}_{SU}$ includes no information about SUs' location.

$\boldsymbol{DB}$. In Step 1 of Algorithm 1, DB learns $\mathcal{H}_{DB} = \{char\}_{i=1}^{n}$ which contains the characteristics of the querying SUs. $\mathcal{H}_{DB}$ may include information like frequency ranges in which SU can operate, antenna characteristics, etc. This information is not related to the querying SUs' location. This shows that the knowledge that DB gains during the execution of $LPDB$ does not allow it to infer SUs' location when they try to learn about spectrum opportunities. $LPDB$ offers an unconditional privacy in the sense that DB's knowledge about SUs' location, during the execution of $LPDB$, does not increase compared to its initial knowledge, which is necessarily the coverage area of DB. □

Theorem 2. *Under Security Assumptions 1 and 2 $LPDBQS$ does not leak any information about SUs' location beyond $\kappa - HMAC$ secure values.*

Proof: We construct a history list of each entity's knowledge during the execution of $LPDBQS$.

$\boldsymbol{SU}$. As the communication between different entities is secured, SUs cannot learn any information about the communicated information of other entities, i.e. $\mathcal{H}_{SU} = \emptyset$.

$\boldsymbol{DB}$. In Line 1 of Algorithm 2, DB learns $\mathcal{H}_{DB} = \{k_{i,t}, char_i, ts_t\}_{i=0,t=t_0}^{n,t_f}$. Obviously, SUs' secret keys $\{k_{i,t}\}_{i=0,t=t_0}^{n,t_f}$ and timestamp values $\{ts_t\}_{t=t_0}^{t_f}$ cannot leak any information about SUs' location since these values

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCCN.2017.2702163, IEEE Transactions on Cognitive Communications and Networking

are not correlated to their physical location. Similarly, their characteristics $\{char_i\}_{i=1}^{n}$ contain information about SUs' devices capabilities, like their possible transmit powers, antennas height, etc, which cannot be used to localize them. This proves that DB's knowledge about SUs' location during the execution of $LPDBQS$ does not differ from its initial knowledge; i.e. that SUs are within DB's covered area.

$\boldsymbol{QS}$. As indicated in Lines 8 & 12 of Algorithm 2, the only information that QS can learn during the execution of $LPDBQS$, is $\mathcal{H}_{QS} = \{y_{k_{i,t}}, CF_{k_{i,t}}\}_{i=1,t=t_0}^{n,t_f}$. $\{y_{k_{i,t}}\}_{i=1,t=t_0}^{n,t_f}$ are as secure as $HMAC$. The elements of $\{CF_{k_{i,t}}\}_{i=1,t=t_0}^{n,t_f}$ are computed using a pseudo random function (as an $HMAC$ is also a pseudo random function) with SUs' secret keys $\{k_{i,t}\}_{i=1,t=t_0}^{n,t_f}$, where $\{k_{i,t}\}_{i=1,t=t_0}^{n,t_f} \overset{\$}{\leftarrow} \{0,1\}^{\kappa}$ and κ is the security level. $\{y_{k_{i,t}}\}_{i=1,t=t_0}^{n,t_f}$ are independent from each other. The same applies to $\{CF_{k_{i,t}}\}_{i=1,t=t_0}^{n,t_f}$. Each query from $\{y_{k_{i,t}}\}_{i=1,t=t_0}^{n,t_f}$ has a corresponding $HMAC$ key, which means that even for the same SU querying the same information, there will be randomly independent and uniformly distributed outputs generated by DB and SUs. Since only SUs and DB know the keys $\{k_{i,t}\}_{i=1,t=t_0}^{n,t_f}$ and that these keys are updated for every query made by SUs, QS cannot learn any information about SUs' location as long as it does not collude with DB as stated in Security Assumption 2. Correlating queries $\{y_{k_{i,t}}\}_{i=1,t=t_0}^{n,t_f}$ to SUs' physical location is equivalent to breaking the underlying $HMAC$ or PRF, which is of probability $1/2^{\kappa}$.

We can conclude that $LPDBQS$ is as secure as the underlying $HMAC$. □

VII. Evaluation and Analysis

In this section, we evaluate the performance of our proposed schemes. We consider that DB's covered area is modeled as a $\sqrt{m} \times \sqrt{m}$ grid that contains m cells each represented by one location pair ($locX$,$locY$) in DB. We use the efficient *cuckoo filter* implementation provided in [41] for our performance analysis with a very small false positive rate $\epsilon = 10^{-8}$ and a load factor $\alpha = 0.95$. In addition, since personal/portable $TVBD$ devices of SUs can only transmit on available channels in the frequency bands $512 - 608$ MHz (TV channels $21-36$) and 614-698 MHz (TV channels $38-51$), this means that users can only access 31 white-space TV band channels in a dynamic spectrum access manner [42]. Therefore, in our evaluation we set the number of TV channels $s = 31$.

Since in practice, at a given time, only a percentage of DB's entries contains available channels, we have ran an experiment to learn what a realistic value of this percentage might be. We denote this percentage (averaged over time and space) as ϱ. We have used the Microsoft online white spaces database application [36] to identify and measure ϱ by monitoring 8 different US locations (Portland, San Faransico, Houston, Miami, Seattle, Boston, New York and Salt Lake City) for few days with an interval between successive measurements of 3 hours. Our measurements show that ϱ is about 6.8%.

Not only does this experiment allow us to evaluate the communication overhead, but also the computational overhead, especially from the database side since both overheads are linear functions of the percentage ϱ as we show in Table II.

There are several factors that influence the performance of both $LPDB$ and $LPDBQS$. One of these factors is the percentage ϱ which has a significant influence on the performance of our schemes as we show in Table II and Figure 6. Also, the number of cells in the grid covered by DB has a direct impact on the size of DB, and thus on the communication and computational overheads of DB as highlighted in Table II and Figures 3 and 5a. In fact, as the number of cells increases, the size of DB increases and so does the computational complexity of constructing the cuckoo filter. In addition, the false positive rate, ϵ, has an impact on the cost of storing one record in the cuckoo filter and subsequently on the communication overhead as we illustrate and discuss in Figure 4 and Table II. Finally, the fraction of positive queries, f_p, can impact the lookup performance as we show and discuss in Figure 7a. We discuss these factors in more details in the next section.

Next, we also compare our schemes with respect to existing approaches in terms of (i) communication and computational overhead, and (ii) location privacy. Since the schemes in [22], [27] try to achieve a different goal, which is the mutual location privacy between SUs and PUs, we do not include them in our overhead analysis. Note that, since the PIR protocol used in [25] has not been specified, we use the protocol proposed by Trostle et al. [24] used in *PriSpectrum* [23] in our performance comparison.

A. Communication and Computation Overhead

1) Communication Overhead: We provide analytical expressions of the communication overhead of these schemes in Table II. For $LPDB$, we provide two expressions of the overhead with respect to two scenarios: (i) when SUs do not reveal one of their coordinates, (ii) when one of the coordinates is revealed by SUs. In both scenarios the data transmitted consist basically of $query$, sent by SU, and the response of DB to it. The size of the response generated by DB depends on the number of entries in DB that satisfy $query$ and on the space needed to store each of these entities in CF. The number of entries for $LPDB$ is given by $\varrho \cdot s \cdot m$ and reduces to $\varrho \cdot s \cdot \sqrt{m}$ when one of the coordinates is revealed by SU. $s \cdot m$ and $s \cdot \sqrt{m}$ provide the number of entries in DB that satisfy the query of SU for both scenarios. ϱ gives the percentage of those entries with available channels. $LPDBQS$ incurs a slightly higher communication overhead than $LPDB$ from a system point of view, as SU needs to additionally send a maximum of $s \cdot \sigma_{HMAC}$ to QS. However, most of this overhead is incurred between DB and QS as SUs do not have to download CFs from DB anymore. For illustration purpose, we plot in Figure 3 the system communication overhead of the different schemes using the expressions established in Table II.

As shown in Figure 3, and as expected, $LPDB$ is clearly more expensive than the other schemes in terms of communication overhead even when ϱ, determined experimentally, is equal to 6.8%. However, revealing one of the coordinates brings a huge gain and makes our scheme even better than

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCCN.2017.2702163, IEEE Transactions on Cognitive Communications and Networking

TABLE II: Communication and computation overhead of proposed and existent schemes

Scheme	Communication	Computation		
		DB	***SU***	***QP***
$\boldsymbol{LPDB}$	$\sigma_{query} + \varrho \cdot s \cdot m \cdot (log_2(1/\epsilon) + log_2(2\beta))/\alpha$	$\varrho \cdot s \cdot m \cdot insert$	$s \cdot (Hash + lookup)$	-
$\boldsymbol{LPDB\ w/\ leakage}$	$\sigma_{query} + \varrho \cdot s \cdot \sqrt{m} \cdot (log_2(1/\epsilon) + log_2(2\beta))/\alpha$	$\varrho \cdot s \cdot \sqrt{m} \cdot insert$	$s \cdot (Hash + lookup)$	-
$PriSpectrum$ [23]	$(2\sqrt{m} + 3)\lceil \log p \rceil$	$\mathcal{O}(m) \cdot Mulp$	$4\sqrt{m} \cdot Mulp$	-
$\boldsymbol{LPDBQS}$	$\sigma_{query} + \varrho \cdot s \cdot m \cdot (log_2(1/\epsilon) + log_2(2\beta))/\alpha + s \cdot \sigma_{HMAC}$	$\varrho \cdot s \cdot m \cdot insert$	$s \cdot HMAC$	$s \cdot lookup$
Troja et al [26]	$(2 + d) \cdot log_2\ N$	$\mathcal{O}(m) \cdot Mulp$	$4\sqrt{m \cdot v} \cdot Mulp$	-
Troja et al [25]	$n_g \cdot b \cdot \log_2 q + (2\sqrt{m} + 3)\lceil \log p \rceil$	$\mathcal{O}(m) \cdot Mulp$	$n_g \cdot b \cdot (2Expp + Mulp) + 4\sqrt{m} \cdot Mulp$	-

Variables: $insert$ and $lookup$ denote the cost of one $Insert$ and $Lookup$ operations in the Cuckoo Filter. β is the number of entries in a bucket of the cuckoo filter. p is a large prime used in the blinding factor of $PriSpectrum$, q is a large prime used in [25], b denotes the number of bits that an SU shares with other SUs in [25], n_g is the number of SUs within a same group in [25], v is the size of a block in DB [26], and d is the umber of DB segments in [26]. $Mulp$ and $Expp$ denote a modular multiplication and a modular exponentiation operations over modulus p. σ_u denotes the amount of data exchanged during a process u, where $u \in \{query, HMAC\}$.

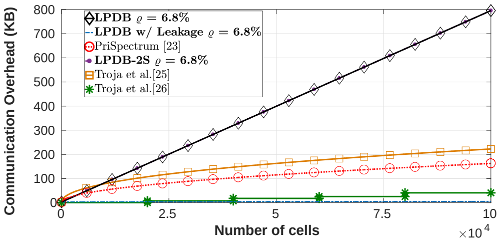


Figure 3: Communication Overhead

existing approaches, yet without compromising the location privacy. $LPDBQS$ has almost the same communication overhead as $LPDB$ but with the difference that most of this overhead is incurred between DB and QS.

We study also the impact of varying the target false positive rate, ϵ, on the cost of inserting one record in the CF in bits as illustrated in Figure 4. This has a direct impact on the size of the filter and thus the communication overhead of our schemes. We do this for multiple values of β, which is the number of slots per bucket in the cuckoo filter. As shown in Figure 4, targeting a smaller value of ϵ costs more bits to store an item in the filter and subsequently increases the communication overhead. Increasing the value of β will require more bits per item to achieve the same target ϵ as illustrated in the Figure. However, cuckoo filter still achieves significantly better than other probabilistic data structures like space-optimized bloom filter as shown in the Figure, which again justifies our choice of the cuckoo filter technique.

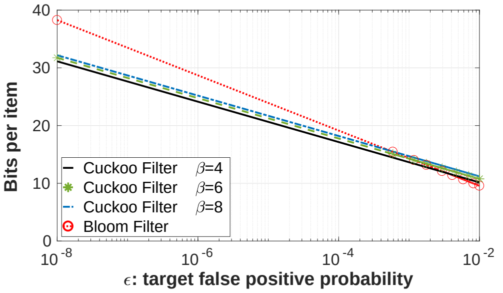


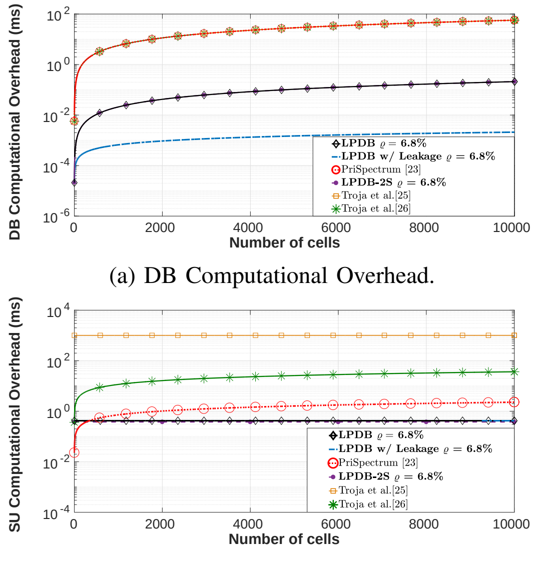
Figure 4: False positive rate vs. space cost per element

2) Computational Overhead: We also investigate the efficiency of our proposed schemes in terms of their computational overhead. We evaluate the computation required at each entity separately, and we provide the corresponding analytical expression of the overhead as shown in Table II. Again we provide two estimated costs for both scenarios of $LPDB$. The computation of DB is given in terms of the number of insertions it has to perform into CF. This depends on the number of DB entries that comply with $query$ considering only the available channels. This number is equal to $\varrho \cdot s \cdot m$ in $LPDB$ and reduces to $\varrho \cdot s \cdot \sqrt{m}$ in $LPDB\ with\ leakage$. For the computational cost at the SU's side, $LPDB$'s overhead depends solely on the number of possible channels, s, and the cost of one $Hash$ and one $Lookup$ operations, as shown in Table II. One of the reasons that motivated our use of the *cuckoo filter*, as we mentioned earlier, is that it is characterized by an extremely fast $Lookup$ operation. This allows SUs to check whether a specific combination, y, exists in the filter, i.e. whether channel is available, very efficiently. $LPDB$'s overhead at SU's side does not depend on the size of DB since any lookup query to CF always reads a fixed number of buckets (at most two) [33], which makes our scheme more scalable than existing approaches in terms of computation when the size of DB increases. In $LPDBQS$, DB performs the same computation as in $LPDB$. The $Lookup$ operations on CF are now outsourced to QS instead of SUs and QS needs to perform a maximum of $s \cdot lookup$ for every querying SU, which is very fast to perform as we mentioned earlier. Every SU needs to only construct $HMAC$-strings $\{y_{k_t}\}_{t=t_0}^{t_f}$ which could be done extremely quickly and could even be precomputed. Note that the PIR-based approaches have similar cost on DB's side, since in any PIR scheme, the server is destined to have $\mathcal{O}(m)$ computation [24].

For illustration purpose, we plot in Figure 5 the computational overhead incurred by each SU and DB, in the different schemes using the expressions established in Table II.

Our schemes are much more efficient than existing approaches at both DB and SU sides as shown in Figures 5a & 5b. The gap keeps increasing considerably as the number of cells (i.e., the size of DB) increases. This is due to the fact that these approaches' cost is dominated by an increasing number of modular multiplications which are very expensive compared to the $Insert$ and $Lookup$ operations of the *cuckoo filter* in our schemes.

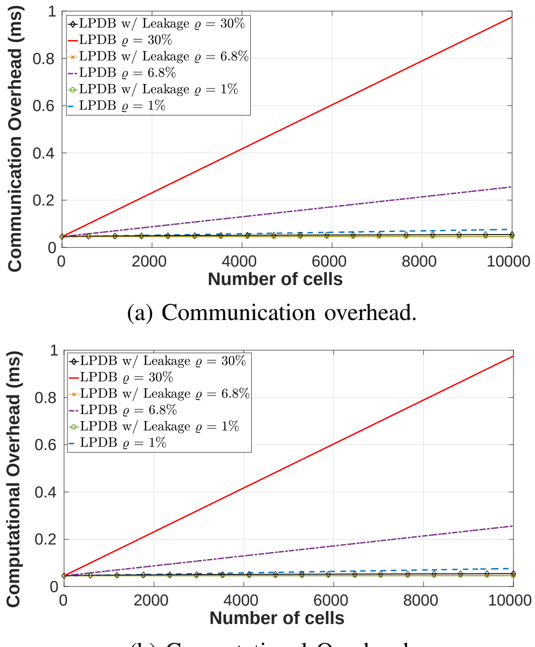
We also evaluate the impact of other parameters on the overhead perceived by both SUs and DB as shown in Figure 7. First, in Figure 7a, we illustrate the variation of the



(a) DB Computational Overhead.

(b) SU Computational Overhead.

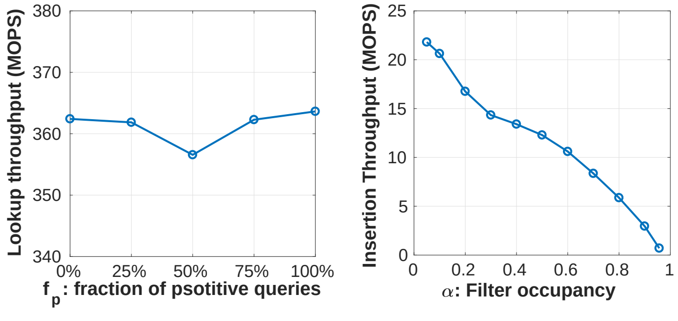
Figure 5: Computation Comparison



(a) Communication overhead.

(b) Computational Overhead.

Figure 6: Impact of varying ϱ.



(a)

(b)

Figure 7: 7a. Lookup performance when a filter achieves its capacity. 7b. Insertion throughput for different load factors α.

throughput of the lookup operations in million operations per second (MOPS) in a cuckoo filter of size $112MB$ as a function of the fraction of positive queries f_p, i.e. queries for items that actually exist in the filter. This clearly shows the efficiency of the lookup operations that SU or QS has to perform to check availability information within CF. CF always fetches two buckets and thus achieves about the same performance when the queries are 100% positive or 100% negative and drops when $f_p = 50\%$ for which the CPU's branch prediction is least accurate [33].

We also assess the insertion throughput that DB experiences to construct the CF as a function of the load factor α as shown in Figure 7b. As opposed to the lookup throughput shown in Figure 7a, CF has a decreasing insert throughput when it is more filled (though their overall construction speed is still high). This is mainly due to the fact that CF may have to move a sequence of existing fingerprints recursively before successfully inserting a new item, and this process becomes more expensive when the load factor grows higher [33].

3) Impact of varying the percentage ϱ of entries with available channels: We also study the impact of ϱ on the overhead incurred by our schemes. For this, we plot in Figure 6 the communication and the system computational overheads for different values of ϱ. We plot only $LPDB$ and $LPDB$ *with leakage* as $LPDBQS$ has almost the same overhead as $LPDB$. As shown in Figure 6, both overheads behave similarly in the way that decreasing ϱ when one of the coordinates is revealed doesn't impact much our scheme. $LPDB$ *w/ Leakage* has the smallest overhead compared to the case where no leakage is allowed. On the other hand, decreasing this parameter drastically reduces the overhead of $LPDB$ and even makes it comparable to $LPDB$ *w/ Leakage* in terms of communication and computation. This means that in the case where only 1% or less of DB entries have available channels, there is no need to reveal one of the coordinates to reduce the overhead.

B. Location privacy

We compare our schemes to existing approaches in terms of location privacy level by presenting the security problems on which they rely as illustrated in Table III. We also precise the localization probability of SUs under these schemes. The best probability that could be achieved is $1/m$, i.e. SUs are within DB coverage area. If one of the schemes is broken then this probability increases considerably.

$LPDB$ offers unconditional security, as SUs do not share any information that could reveal their location. $LPDB$ could be seen as a variant of PIR in which the server sends a whole copy of the database to the user and this is the only way to achieve information theoretic privacy (i.e. cannot be broken even with computationally unbounded adversary) in a single-server setting. Even if one of the coordinates is intentionally revealed by a SU, its location is still indistinguishable from $\sqrt{m} - 1$ remaining possible locations.

The approaches in [23], [25], [26] rely on computational PIR protocols to preserve SUs' location privacy. The security

TABLE III: Location privacy

Scheme	Security level	Localization probability
$\boldsymbol{LPDB}$	Unconditionally secure	$1/m$
$\boldsymbol{LPDB\ w/\ leakage}$	Uncond. within 1 coordinate	$\sqrt{1/m}$
$PriSpectrum$ [23]	Computational PIR	$1/m$
Troja et al [26]	Computational PIR	$1/m$
Troja et al [25]	Computational PIR	$1/m$
$\boldsymbol{LPDBQS}$	$\kappa - HMAC$	$1/m$
Zhang et al. [22]	*k-anonymity*	$1/k$
Zhang et al. [27]	Geo-Indistinguishability	$1/r$

Variables: r is the radius of the ϵ-geo-indistinguishability mechanism in [27].

of Computational PIR protocols' is established against a computationally bounded adversary based on well-known cryptographic problems that are hard to solve (e.g. discrete logarithm or factorization [43]). This means that these approaches offer lower security level than $LPDB$.

The approach proposed by Zhang et al. [22] relies on the concept of *k-anonymity*, which offers very low privacy level as the probability of identifying the location of a querying SU is equal to $1/k$. Also, an approach cannot be proved to satisfy *k-anonymity* unless assumptions are made about the attacker's auxiliary information. For instance, dummy locations are only useful if they look equally likely to be the real location from the adversary's point of view. Any auxiliary information that allows the attacker to rule out any of those locations would immediately violate the definition.

As we have shown in Section VI, $LPDBQS$ is as secure as its underlying $HMAC$ which is breakable only with probability of $1/2^{\kappa}$, where κ is the security level. For the same security level, $HMAC$ incurs much less communication overhead than that of the computational PIR protocols in [23], [25], [26].

Zhang et al. [27] propose an approach whose privacy depends on the ϵ-geo-indistinguishability [28] mechanism, which is derived from the *differential privacy* concept. In this mechanism, a SU sends a randomly chosen point z close to its location, but that still allows it to get a useful service. An informal, definition of this mechanism as given in [28] is as follows: A mechanism satisfies ϵ-geo-indistinguishability if and only if for any radius $r > 0$, the user enjoys ℓ-privacy within a radius r, where $\ell = \epsilon r$ and ϵ is the privacy level per unit of distance. A user is said to enjoy ℓ-privacy within r if, by observing z, the adversary's ability to find the user's location among all points within r, does not increase by more than a factor depending on ℓ compared to the case when z is unknown [28]. The smaller ℓ the stronger the privacy the user enjoys. SU can specify its privacy level requirement by providing the radius r it is concerned about, and the privacy level that it wishes for this specific radius. Relying on this mechanism in the context of CRN, is problematic because, first, it introduces some noise to SU's location which may cause erroneous spectrum availability information and, subsequently, interference with primary transmissions. Second, to avoid facing the previous issue, SU may need to pick the radius that can still give it accurate information which means necessarily that $r << \sqrt{m}$. Hence, even though the adversary will be unable to pinpoint the exact location of the SU, it will be able though to learn that it is within the radius r from the shared location z.

In summary, as can be seen in Table III and as explained above, $LPDB$ offers the highest location privacy level as it achieves information-theoretic security. $LPDBQS$ can offer similar security guarantees as computational PIR-based approaches but with significantly better computational and communication overhead thanks to the use of $HMAC$.

VIII. Conclusion

In this paper, we have proposed two location privacy preserving schemes, called $LPDB$ and $LPDBQS$, that aim to preserve the location privacy of SUs in database-driven CRNs. They both use *set membership data structures* to transmit a compact representation of the geo-location database to either SU or QS, so that SU can query it to check whether a specific channel is available in its vicinity. These schemes require different architectural and performance tradeoffs.

Acknowledgment

This work was supported in part by the US National Science Foundation under NSF award CNS-1162296.

References

[1] M. Grissa, A. A. Yavuz, and B. Hamdaoui, "Cuckoo filter-based location-privacy preservation in database-driven cognitive radio networks," in *WSCNIS*. IEEE, 2015, pp. 1–7.

[2] "Spectrum policy task force report," Federal Communications Commission, Tech. Rep. ET Docket No.02-135, 2002.

[3] B. Khalfi, M. B. Ghorbel, B. Hamdaoui, and M. Guizani, "Optimal power allocation for smart-grid powered point-to-point cognitive radio system," in *ComComAp, 2014 IEEE*, pp. 316–320.

[4] H. Zhu, C. Fang, Y. Liu, C. Chen, M. Li, and X. S. Shen, "You can jam but you cannot hide: Defending against jamming attacks for geo-location database driven spectrum sharing," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 10, pp. 2723–2737, 2016.

[5] M. Guizani, B. Khalfi, M. B. Ghorbel, and B. Hamdaoui, "Large-scale cognitive cellular systems: resource management overview," *IEEE Communications Magazine*, vol. 53, no. 5, pp. 44–51, 2015.

[6] N. Adem and B. Hamdaoui, "The impact of stochastic resource availability on cognitive network performance: modeling and analysis," *Wireless Communications and Mobile Computing*, 2015.

[7] B. Khalfi, M. B. Ghorbel, B. Hamdaoui, and M. Guizani, "Distributed fair spectrum assignment for large-scale wireless dsa networks," in *International Conference on Cognitive Radio Oriented Wireless Networks*. Springer, 2015, pp. 631–642.

[8] N. Adem and B. Hamdaoui, "Delay performance modeling and analysis in clustered cognitive radio networks," in *Global Communications Conference (GLOBECOM), 2014 IEEE*. IEEE, 2014, pp. 193–198.

[9] W. Wang and Q. Zhang, *Location Privacy Preservation in Cognitive Radio Networks*. Springer, 2014.

[10] L. Zhu, V. Chen, J. Malyar, S. Das, and P. McCann, "Protocol to access white-space (paws) databases," 2015.

[11] S. B. Wicker, "The loss of location privacy in the cellular age," *Communications of the ACM*, vol. 55, no. 8, pp. 60–68, 2012.

[12] M. Grissa, B. Hamdaoui, and A. A. Yavuz, "Location privacy in cognitive radio networks: A survey," *IEEE Communications Surveys & Tutorials*, 2017.

[13] 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*. IEEE, 2012, pp. 729–737.

[14] M. Grissa, A. A. Yavuz, and B. Hamdaoui, "Lpos: Location privacy for optimal sensing in cognitive radio networks," in *Global Communications Conference (GLOBECOM), 2015 IEEE*. IEEE, 2015.

[15] W. Wang and Q. Zhang, "Privacy-preserving collaborative spectrum sensing with multipleservice providers," *Wireless Communications, IEEE Transactions on*, 2015.

[16] M. Grissa, A. 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*. IEEE, 2016, to be published.

[17] ——, "Preserving the location privacy of secondary users in cooperative spectrum sensing," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 2, pp. 418–431, 2017.

[18] S. Liu, H. Zhu, R. Du, C. Chen, and X. Guan, "Location privacy preserving dynamic spectrum auction in cognitive radio network," in *Distributed Computing Systems (ICDCS), 2013 IEEE 33rd International Conference on*. IEEE, 2013, pp. 256–265.

[19] M. Gruteser and D. Grunwald, "Anonymous usage of location-based services through spatial and temporal cloaking," in *Proceedings of the 1st international conference on Mobile systems, applications and services*. ACM, 2003, pp. 31–42.

[20] B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan, "Private information retrieval," *JACM*, vol. 45, no. 6, 1998.

[21] C. Dwork, "Differential privacy: A survey of results," in *International Conference on Theory and Applications of Models of Computation*. Springer, 2008, pp. 1–19.

[22] 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*.

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

[24] J. Trostle and A. Parrish, "Efficient computationally private information retrieval from anonymity or trapdoor groups," in *International Conference on Information Security*. Springer, 2010, pp. 114–128.

[25] E. Troja and S. Bakiras, "Leveraging p2p interactions for efficient location privacy in database-driven dynamic spectrum access," in *Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 2014.

[26] ——, "Efficient location privacy for moving clients in database-driven dynamic spectrum access," in *2015 24th International Conference on Computer Communication and Networks (ICCCN)*. IEEE, 2015.

[27] Z. Zhang, H. Zhang, S. He, and P. Cheng, "Achieving bilateral utility maximization and location privacy preservation in database-driven cognitive radio networks," in *Mobile Ad Hoc and Sensor Systems (MASS), 2015 IEEE 12th International Conference on*. IEEE, 2015.

[28] M. E. Andrés, N. E. Bordenabe, K. Chatzikokolakis, and C. Palamidessi, "Geo-indistinguishability: Differential privacy for location-based systems," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. ACM, 2013, pp. 901–914.

[29] O. Goldreich and R. Ostrovsky, "Software protection and simulation on oblivious rams," *JACM*, vol. 43, no. 3, pp. 431–473, 1996.

[30] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: improved definitions and efficient constructions," *Journal of Computer Security*, vol. 19, no. 5, pp. 895–934, 2011.

[31] T. Dierks and C. Allen, "The tls protocol version 1.0," 1999.

[32] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[33] B. Fan, D. G. Andersen, M. Kaminsky, and M. D. Mitzenmacher, "Cuckoo filter: Practically better than bloom," in *Proc. of the 10th ACM Int'l Conference on emerging Networking Experiments and Technologies*, 2014, pp. 75–88.

[34] R. Pagh and F. F. Rodler, "Cuckoo hashing," *Journal of Algorithms*, vol. 51, no. 2, pp. 122–144, 2004.

[35] "Google white spaces database," https://www.google.com/get/spectrumdatabase/channel.

[36] "Microsoft white spaces database," http://whitespaces-demo.cloudapp.net.

[37] "iconectiv white spaces database," https://spectrum.iconectiv.com/main/home/contour_vis.shtml.

[38] B. Yee and J. D. Tygar, "Secure coprocessors in electronic commerce applications." in *USENIX Workshop on Electronic Commerce*, 1995.

[39] F. PUB, "Security requirements for cryptographic modules," Ph.D. dissertation, National Institute of Standards and Technology, 1999.

[40] N. F. PUB, "140-2: Security requirements for cryptographic modules," *Information Technology Laboratory, National Institute of Standards and Technology*, 2001.

[41] "Cuckoo filter implementation," https://github.com/efficient/cuckoofilter.

[42] F. C. Commission, "Electronic code of federal regulations title 47, chapter 1, subchapter a: Part 15-television band devices," 2015.

[43] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone, *Handbook of applied cryptography*. CRC press, 1996.

Mohamed Grissa (S'14) received the Diploma of Engineering (with highest distinction) in telecommunication engineering from Ecole Superieure des Communications de Tunis, Tunis, Tunisia, in 2011, and the M.S. degree in electrical and computer engineering (ECE) from Oregon State University, Corvallis, OR, USA, in 2015. He is currently working toward the Ph.D. degree at the School of Electrical Engineering and Computer Science (EECS), Oregon State University, Corvallis, OR, USA.

Before pursuing the Ph.D. degree, he worked as a Value Added Services Engineer at Orange France Telecom Group from 2012 to 2013. His research interests include privacy and security in wireless networks, cognitive radio networks, IoT and eHealth systems.

Attila A. Yavuz (S'05–M'10) received a BS degree in Computer Engineering from Yildiz Technical University (2004) and a MS degree in Computer Science from Bogazici University (2006), both in Istanbul, Turkey. He received his PhD degree in Computer Science from North Carolina State University in August 2011. Between December 2011 and July 2014, he was a member of the security and privacy research group at the Robert Bosch Research and Technology Center North America. Since August 2014, he has been an Assistant Professor in the School of Electrical Engineering and Computer Science, Oregon State University, Corvallis, USA. He is also an adjunct faculty at the University of Pittsburgh's School of Information Sciences since January 2013.

Attila A. Yavuz is interested in design, analysis and application of cryptographic tools and protocols to enhance the security of computer networks and systems. His current research focuses on the following topics: Privacy enhancing technologies (e.g., dynamic symmetric and public key based searchable encryption), security in cloud computing, authentication and integrity mechanisms for resource-constrained devices and large-distributed systems, efficient cryptographic protocols for wireless sensor networks.

Bechir Hamdaoui (S'02–M'05–SM'12) is presently an Associate Professor in the School of EECS at Oregon State University. He received the Diploma of Graduate Engineer (1997) from the National School of Engineers at Tunis, Tunisia. He also received M.S. degrees in both ECE (2002) and CS (2004), and the Ph.D. degree in ECE (2005) all from the University of Wisconsin-Madison. His research interest spans various areas in the fields of computer networking, wireless communications, and mobile computing, with a current focus on distributed optimization, parallel computing, cognitive networks, cloud computing, and Internet of Things. He has won several awards, including the 2016 EECS Outstanding Research Award and the 2009 NSF CAREER Award. He is presently an Associate Editor for IEEE Transactions on Wireless Communications (2013-present). He also served as an Associate Editor for IEEE Transactions on Vehicular Technology (2009-2014), Wireless Communications and Mobile Computing Journal (2009-2016), and for Journal of Computer Systems, Networks, and Communications (2007-2009). He is currently serving as the chair for the 2017 IEEE INFOCOM Demo/Posters program. He has also served as the chair for the 2011 ACM MOBICOM's SRC program, and as the program chair/co-chair of several IEEE symposia and workshops, including GC 16, ICC 2014, IWCMC 2009-2017, CTS 2012, and PERCOM 2009. He also served on technical program committees of many IEEE/ACM conferences, including INFOCOM, ICC, and GLOBECOM. He has been selected as a Distinguished Lecturer for the IEEE Communication Society for 2016 and 2017. He is a Senior Member of IEEE, IEEE Computer Society, IEEE Communications Society, and IEEE Vehicular Technology Society.