Preserving Secondary Users' Privacy in Cognitive Radio Networks

Zhengrui Qin, Shanhe Yi, Qun Li
Dept of Computer Science, College of William & Mary
{zhengrui,syi,liqun}@cs.wm.edu

Dmitry Zamkov
University of Illinois at Urbana-Champaign
zamkov2@illinois.edu

***Abstract*—Cognitive radio plays an important role in improving spectrum utilization in wireless services. In the cognitive radio paradigm, secondary users (SUs) are allowed to utilize licensed spectrum opportunistically without interfering with primary users (PUs). To motivate PU to share licensed spectrum with SU, it is reasonable for SU to pay PU a fee whenever the former is utilizing the latter's licensed spectrum. SU's detailed usage information, such as when and how long the licensed spectrum is utilized, is needed for PU to calculate payment. Providing usage information to PU, however, may compromise SU's privacy. To solve this dilemma, we are the first to propose a novel privacy-preserving mechanism for cognitive radio transactions through commitment scheme and zero-knowledge proof. This mechanism, on one hand, only allows PU to know the total payment to SU for a billing period, plus a little portion of SU's usage information. On the other hand, it guarantees PU that the payment is correctly calculated. We have implemented our mechanism and evaluated its performance.**

I. Introduction

Cognitive radio has emerged as a fundamental approach in improving the utilization of spectrum resource through dynamic spectrum sharing. As a key technology in wireless communication, cognitive radio enables secondary users (SUs) to opportunistically share the licensed spectrum with primary users (PUs). To make the spectrum sharing attractive, it is essential to design incentive mechanisms for both PU and SU. It is reasonable that the owner of the licensed spectrum, i.e., PU, charges SU whenever the latter is utilizing licensed spectrum. Recently, several approaches, such as game theory and auction theory, have been applied to this topic, maximizing either PU's interests or the social welfare. However, SU's privacy in cognitive radio transactions has been neglected or left untouched.

In a typical cognitive radio transaction, SU's payment to PU depends on SU's detailed usage information, such as when and how long the licensed spectrum has been utilized. PU needs this usage information to calculate or verify the payment, but detailed usage information is sensitive to SU and the disclosure of this information may compromise SU's privacy. Therefore, protecting PU's interests and preserving SU's privacy simultaneously becomes very challenging. To the best of our knowledge, none of the existing work has addressed this problem in cognitive radio transactions.

In this paper, we propose a novel privacy-preserving mechanism for cognitive radio transactions. This mechanism not only preserves SU's privacy but also protects PU's interests. In this mechanism, PU only knows a little portion of SU's sensitive information during a billing period with the result that SU's privacy is preserved. At the same time, PU knows the total payment for a billing period and is guaranteed that the payment is correctly calculated and PU's interests are protected. This mechanism employs commitment schemes and zero-knowledge proof. At the end of a billing period, SU commits all detailed usage information and the payment for each utilization instance, and provides a zero-knowledge proof for each utilization instance that the payment is correctly calculated. These commitments and proofs, along with the total fee, are sent to PU. Due to the *hiding* property of the commitment scheme, the commitments do not reveal any information about the detailed usage information. Due to the *binding* property of the commitment scheme, SU cannot deny the values that are used to create the commitments. Furthermore, by verifying the zero-knowledge proofs provided by SU, PU is able to verify that the payment for each utilization instance is correct. To prevent SU from committing fraud, such as choosing not to submit all utilization instances or submitting false utilization instances, we introduce a random-checking monitor that can provide some ground-truth information on the spectrum utilization status. PU can opportunistically query the monitor to ask for a few pieces of ground-truth information, and use this information to challenge SU. Once SU is challenged, it must provide proofs to match the random-checking information.

Our main contributions in this paper are three-fold. First, we have pointed out an important security problem in cognitive radio literature, preserving SU's privacy, which has been neglected for a long time. Second, we are the first to address this problem and have proposed a privacy-preserving mechanism to protect PU's interests and preserve SU's privacy at the same time. Third, we have implemented our mechanism and evaluated its performance.

The rest of paper is organized as follows. Section II is the related work. Section III describes our privacy-preserving model, including threat model, trust model, and system architecture. In Section IV, we review some preliminaries and construct our privacy-preserving protocol in detail. In Section V, we present the security analysis. In Section VI, we propose the monitoring scheme and examine its effectiveness. We implement and

978-1-4799-3360-0/14/$31.00

evaluate our proposed protocol in Section VII, and conclude our paper in Section VIII.

II. Related Work

Cognitive radio is the key approach that allows SUs to opportunistically access the licensed spectrum owned by PUs through spectrum sensing [1]–[6]. Here we will review some work on cognitive radio transactions, and discuss some privacy and security issues in cognitive radio market.

A. Cognitive Radio Transaction

In a cognitive radio network with incentive mechanism, PU wants to maximize its revenue, and SU wants to maximize its utility, which is usually the gain from the wireless service minus the payment. Therefore, cognitive radio pricing is an indispensable component for incentive cognitive radio networks, and has been addressed by many researchers, such as through auction models [7]–[9]. In this paper, our focus is on SU's privacy, which is neglected in auction models, rather than pricing itself. We assume that PU announces a pricing policy to all SUs, similar to the uniform pricing model in [7]. Then every SU who utilizes the licensed spectrum pays PU according to the pricing policy. Our interest is to hide SU' detailed usage information from PU and thus preserve SU' privacy.

B. Privacy and Security

Usage privacy usually relates to when, where and how a consumer, such as a person or an object, is consuming a non-free resource. The owner of the resource may need the detailed usage information to calculate a payment to charge the customer, while the detailed information may breach the customer's privacy. This dilemma exists in many areas, such as smarter metering and vehicular electronic tolling. Privacy-preserving solutions have been proposed in smart metering [10], vehicular tolling [11], [12], and other topics [13]–[15], and can potentially be applied to some attacks in smart grid and social networks [16]–[18].

Cognitive radio marketing faces the same dilemma. Usually SU is charged based on its spectrum usage profile, such as when and how long it utilizes a certain channel, while the usage profile inevitably breaches its privacy. Though much work has been done on the incentive mechanism for cognitive radio, to the best of our knowledge, SU's privacy related to fine-grained usage profile has never been addressed.

III. Privacy-Preserving Model

We consider a typical cognitive radio network, which consists of a primary user (PU) and multiple secondary users (SUs). The PU has some non-utilized spectrum to sell to SUs. In a cognitive radio transaction, PU announces the pricing policy for the non-utilized spectrum, typically a price function depending on frequency, bandwidth, time, etc. SU utilizes some spectrum resources and pays PU according to the pricing policy at the end of a billing period, such as a day or a month.

A. Adversary and Threat Model

In this paper, we assume secure communication. Messages among all entities are signed by the senders and can be verified by the recipients. Our focus in this paper is on PU's interests and SU's privacy. We consider the following threats from the prospective of both PU and SU.

PU is concerned about the following threats:

- **T1.** SU does not report all its utilization data;
- **T2.** SU reports false utilization data, such as the time when it starts to use a spectrum resources and the time duration of one utilization instance;
- **T3.** SU uses incorrect price to calculate the subfee for a utilization instance and thus reports false subfee;
- **T4.** SU reports false total fee for a billing period.

SU is concerned the following threat:

- **T5.** PU learns a large amount of the detailed usage information to breach SU's privacy.

B. Design Goals

There are two goals for our privacy-preserving mechanism. The first one is to preserve SU's privacy. The second one is to protect PU's interests.

SU should never send its detailed utilization information to PU in plain text. The detailed utilization information, such as when (time) and how (frequency, bandwidth) SU utilizes the spectrum, compromises SU's privacy. Instead of the utilization information in plain text, SU commits to the utilization information and sends the commitments to PU. As we explain later, the commitments do not disclose any information about the committed values, SU's privacy is thus preserved.

At the same time, PU's interests must be protected. PU must be ensured that SU does not commit fraud. Even though PU only obtains the commitments of SU's utilization information, it must have a way to verify the payment through the commitments. We resort to a homomorphic commitment scheme, which can hide SU's utilization information and help PU verify the payment.

C. Trust Model

To achieve the design goals, trustworthy information about SU's utilization is required. Otherwise, there is no way for PU to determine whether SU commits fraud. Hence, we introduce a trusted random-checking monitor, which wakes up to check the utilization status of the spectrum in a way that SU cannot predict. The monitor is trusted, but it cannot scan all the spectrum channels all the time due to limited monitoring resources or cost concern. For instance, the monitor may not be able to cover all spectrum channels at the same time due to its functionality constraints, or the cost concern may refrain the monitor from monitoring all channels all the time even it is capable to do so.

We assume that SU does not know the pattern how the monitor scans all channels. Since the monitor records some information on the status of spectrum utilization, SU who does

not report or reports false information has the risk to be caught. Once a fraud is caught, we can impose a high penalty to deter SU from lying. Therefore, the monitor will enforce SU to report true spectrum utilization, i.e., when and how long SU user has used a certain channel.

D. System Architecture

The system model comprises three components, *Primary User* (PU), *Secondary User* (SU), and *Monitor* (M), as shown in Fig. 1. PU announces pricing policy to SU, which is in the form of segments, with each segment representing the unit price for a certain channel in a period of time. SU calculates a subfee for each utilization instance based on the pricing policy if he utilizes a certain channel for a period of time. At the end of a billing period, SU sums up all the subfees to obtain a total fee and send it to PU, together with the commitments of values in each utilization instance. PU only obtains the information of the total payment from SU, but it does not know when and how long SU has utilized a channel, nor does it know which channel SU has utilized. In this way, SU's privacy is preserved. To check whether SU commits fraud, PU can ask for some observations from the monitor M, and require SU to reveal the corresponding commitments. If the observations and commitments are matched, then SU is considered honest; otherwise, a high penalty will be imposed. We limited the number of random-checking observations that a primary user can ask for during a billing period such that only a limited amount of SU's private information is leaked. To guarantee the integrity of messages, all three entities implement a signature scheme such that all messages cannot be forged.

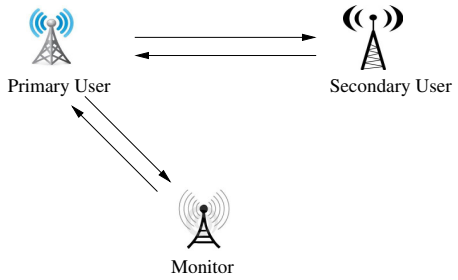

Fig. 1. System architecture for privacy-preserving pricing in cognitive radio network.

Remark: In the above architecture, we introduce a new entity, the monitor M, into the system to randomly check whether SU is utilizing a channel. Alternatively, we may let SU carry out the monitoring functionality [19]. When an SU transmits a message over a licensed channel, it concatenates the message and the current time, and hash the concatenated message. Then the SU signs the hashed value. Other SUs nearby can capture the signed message in the corresponding channel. When PU asks an SU u_i for utilization instances of another SU u_j, user u_i can provide user u_j's signature together with the corresponding time to PU. With the signature, user u_j cannot deny its utilization of the licensed spectrum. To motivate SU to participate in monitoring, PU can offer some rewards to SUs who provide monitoring information.

IV. Protocol Construction

In this section, we will first review some preliminaries that are extensively used in our privacy-preserving protocol. Then we will show our protocol in detail.

A. Preliminaries

1) Signature Scheme: A signature scheme consists of three fundamental components, key generation (**Keygen**), signature generation (**Sign**), and verification (**Verify**). **Keygen** generates a key pair (sk, pk), where sk and pk are the secret key and the public key, respectively. **Sign**(sk, m) outputs a signature m_s on message m. **Verify**(pk, m_s, m) outputs *accept* if m_s is a valid signature of message m; otherwise, it outputs *reject*. A signature must be unforgeable such that an adversary should not be able to obtain the signature m_s on message m unless he knows the secret key sk or has previously obtained a signature on m.

2) Commitment Scheme: A commitment scheme consists of three fundamental phases, commitment setup phase (**ComSetup**), commit phase (**Commit**), and open phase (**Open**). **ComSetup** generates the parameters of the commitment scheme, par. **Commit**(par, x) outputs a commitment c_x to x and auxiliary information $open_x$. **Open**(par, c_x, x, $open_x$) outputs *accept* if c_x is the commitment to x and $open_x$ is the corresponding auxiliary information. A commitment scheme has two important properties, $hiding$ and $binding$. The hiding property guarantees that c_x does not reveal any information about x, and the binding property guarantees that c_x cannot be opened to values other than x.

A commitment scheme is said additively homomorphic if, given two commitments c_{x_1} and c_{x_2} with openings $open_{x_1}$ and $open_{x_2}$ respectively, **Open**(par, $c_{x_1} \cdot c_{x_2}$, $x_1 + x_2$, $open_{x_1} + open_{x_2}$) outputs *accept*.

3) Zero-knowledge Proof: A zero-knowledge proof of knowledge is a method for a prover to prove to a verifier that a statement is true, without revealing anything other than the veracity of the statement. A zero-knowledge proof of knowledge has three important property, *completeness*, *soundness*, and *zero-knowledge*. Completeness guarantees that an honest verifier will be convinced by an honest prover if the statement is true. Soundness guarantees that no malicious prover can convince a verifier that a statement is true if it is actually false, except with some negligible probability. *Zero-knowledge* guarantees that the verifier learns nothing other than the truth of the statement.

For instance, a prover has a value x and its commitment $c_x = g^x h^{open_x}$, where g and h are parameters of commitment scheme. He sends only c_x, g, and h to a verifier. By means of zero-knowledge proof, the prover can prove to the verifier that he knows the committed value x and its corresponding open message $open_x$, without revealing x and $open_x$. During a typical proof, the verifier inquires the prover with some random challenge, and the prover makes a response to the challenge. Then the verifier verifies some equalities. This interactive proof can be turned into non-interactive zero-knowledge proof via Fiat-Shamir heuristic [20].

B. Construction Sketch

As in Fig. 1, there are in total three entities in the system: primary user (PU), secondary users (SUs), and a random-checking monitor (M). Our privacy-preserving protocol has three stages, i.e., *initialization*, *transaction*, and *random-checkinging*.

1) Initialization: At the beginning, each entity generates a key pair for signature scheme, i.e., (sk_{PU}, pk_{PU}) for PU, (sk_{SU}, pk_{SU}) for SU, (sk_M, pk_M) for the monitor M. Each entity distributes its own public key, and stores the public keys from other entities. With these key pairs, each entity can verify the origin and integrity of each message received. SU also generates the parameter par for its commitment scheme, and share par with PU.

When PU has some licensed spectrum f (such as a channel) to sell, he announces tuples in the form of $< T_{b_i}, T_{e_i}, f >$, each of which denotes that a certain SU may use spectrum f in the time interval $[T_{b_i}, T_{e_i}]$. PU also establishes a function $F : (T_{b_i}, T_{e_i}, f) \rightarrow \Phi$ that maps every tuple $< T_{b_i}, T_{e_i}, f >$ to a price $p_i \in \Phi$, where p_i is the unit price ($\$/second$) for spectrum f in the time interval $[T_{b_i}, T_{e_i}]$*. PU signs the pricing policy Φ to get Φ_s, where $\Phi_s = \textbf{Sign}(sk_{PU}, \Phi)$, and sends (Φ_s, Φ) to SU.

When SU receives Φ_s, it runs $\textbf{Verify}(pk_{PU}, \Phi_s, \Phi)$. If the verification outputs *accept*, SU stores the pricing policy Φ; otherwise, SU notifies PU that the message is tampered.

2) Transaction: When SU utilizes a segment of the spectrum with frequency f, it records the starting time t_i and the time duration $\triangle t_i$, denoted as a tuple $< t_i, \triangle t_i, f >$, which is called a utilization instance. First, it determines which pricing policy to use. Suppose $T_{b_j} \leq t_i \leq T_{e_j}$, then the unit price is p_j. Then it calculates the payment $fee_i = p_j \triangle t_i$†. Next, it computes a payment tuple $u_i = (c_{t_i}, c_{\triangle t_i}, c_f, c_{fee_i}, c_{p_j}, \pi_i)$, where c_x is a homomorphic commitment of x with auxiliary information $open_x$ and parameter par, $x \in \{t_i, \triangle t_i, f, fee_i, p_j\}$, π_i is the non-interactive zero-knowledge proof that (1) SU knows all openings of all commitments; (2) fee_i committed in c_{fee_i} is the product of p_j and $\triangle t_i$ committed in c_{p_j} and $c_{\triangle t_i}$, respectively; (3) SU possesses a C-L signature [21] on p_j computed by PU that the price p_j belongs to Φ and it is the right one to calculate the subfee fee_i.

Suppose SU has utilized n tuples of spectrum resources in a billing period. At the end of the billing period, SU adds up all the subfees of n tuples to obtain a total fee $fee_{total} = \sum_{i=1}^{n} fee_i$, and adds up all the openings to obtain an opening for the total fee, $open_{fee_{total}} = \sum_{i=1}^{n} open_{fee_i}$. Then SU composes a payment message m that consists of $(fee_{total}, open_{fee_{total}}, id)$ and all the payment tuples $u_i = (c_f, c_{t_i}, c_{\triangle t_i}, c_{fee_i}, c_{p_j}, \pi_i)$, i.e., $m = (fee_{total}, open_{fee_{total}}, id, u_{i=1}^{n})$. SU signs message m to obtain its signature m_s, where $m_s = \textbf{Sign}(sk_{SU}, m)$ and sends (m, m_s) to PU.

When receiving (m, m_s, id), PU first runs $\textbf{Verify}(pk_{SU}, m_s, m)$ to verify SU's signature on the message m. If the verification outputs *accept*, PU then verifies the proof π_i in each payment tuple u_i, $\forall i \in [1, n]$. If all the verifications outputs *accept*, PU times up commitments c_{fee_i} of all payment tuples to obtain a commitment $c_{fee_{total}} = \prod_{i=1}^{n} c_{fee_i}$. At the end, PU checks whether $open_{fee_{total}}$ is a valid opening by running $\textbf{Open}(par, c_{fee_{total}}, fee_{total}, open_{fee_{total}})$. If it outputs *accept*, PU will query the monitor M as described below.

3) Random-checking: The monitor, M, scans to check whether a SU is utilizing the spectrum with frequency f (such as a channel). For each observation, it generates a signed statement ϕ that an SU with its identity id is utilizing the spectrum with frequency f at time t_c, i.e., $\phi = (t_c, f, id)$. To check whether a certain SU submits all payment tuples and the payment for each tuple is correctly calculated, PU sends a request message $r = (request, id)$, together with its signature $r_s = \textbf{Sign}(sk_{PU}, r)$, to M to ask for one statement ϕ for SU with identity id. When receiving the request, M verifies PU's signature by running $\textbf{Verify}(pk_{PU}, r_s, r)$. If the verification is accepted, M sends one piece of statement ϕ with signature, i.e., $\phi_s = \textbf{Sign}(sk_M, \phi)$, to PU. Then PU relays ϕ_s to SU. SU first verifies whether ϕ_s has a valid signature from M. If the signature is valid, it searches among its payment tuples and finds the one with the time domain $[t_i, t_i + \triangle t_i]$ such that $t_i < t_c < t_i + \triangle t_i$. Once the tuple is found, SU will send all the openings for all the commitments in that payment tuple in a signed message, $o_s = \textbf{Sign}(sk_{SU}, open_{t_i}, open_{\triangle t_i}, open_f, open_{fee_i})$. Then PU can verify whether the SU has submitted the payment tuple corresponding to ϕ and whether the SU computes the price fee_i correctly. PU can request more than one statements at a time. However, the number of statements must be limited. If SU commits fraud, a penalty will be imposed.

The whole protocol is shown in Fig. 2.

V. Security Analysis

In this section, we will analyze whether our protocol in Section IV can thwart all the threats mentioned in Section III-A.

A. Privacy and Interest

Claim 1: Our protocol can defend against threats **T1**, **T2**, **T3**, and **T4**, thus the primary user's interests are protected.

The random-checking monitor and the imposed penalty defend against both **T1** and **T2**. If a secondary user purposely chooses not to report some of its utilization data, it has the risk to be caught by the monitor, and the imposed penalty once being caught deters a secondary user from committing **T1**. Similarly, a secondary user has the risk to be caught by the monitor if it commits **T2** by reporting false utilization data. Regarding **T3**, the proof that a secondary user possesses a C-L signature on the pricing policy makes sure that the

*Here we assume a linear pricing policy. Our scheme can be adopted to other complicated pricing policy, such as cumulative policy in [10].

†When $\triangle t_i$ spans two pricing policies, we break this segment into two segments at the boundary of two pricing polices.

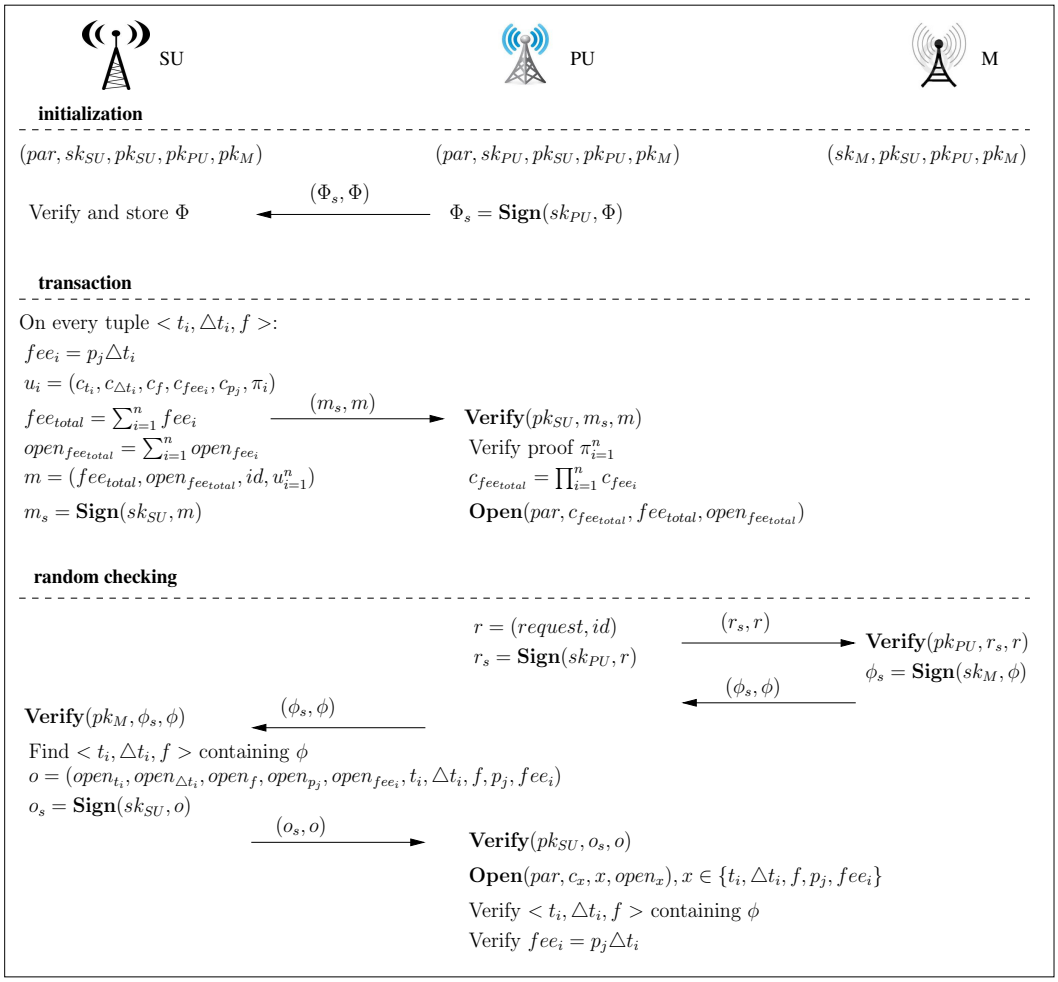

Fig. 2. Structure of the protocol. SU=Secondary User, PU=Primary User, M=Monitor.

correct unit price is used to calculate the subfee for a utilization instance. Furthermore, a secondary user cannot commit fraud **T4** once all subfees are correct. The homomorphic property of the commitments scheme can guarantee that the total fee is the sum of all the subfees. □

Claim 2: A secondary user's privacy is preserved except that information in the responses of queries is disclosed.

At the end of a billing period, the secondary user only sends the total fee and its id in plain text to the primary users. All other information is sent as commitments and zero-knowledge proofs. As we know, the hiding property of a commitment and the zero-knowledge property of a zero-knowledge proof guarantee that all information in commitments and zero-knowledge proofs is not revealed to the primary user. The only information disclosed is the one in the responses of queries when the secondary user responds to the queries made by the primary user. □

B. Discussion

In our protocol, besides the information in the responses of queries, the primary user can also learn the number of utilization instances. For each utilization instance, the secondary user calculates a subfee and sends the commitment of the subfee to the primary user. By counting the number of subfee commitments, the primary user learns the number of utilization instances. Though this information does not reveal any detailed information in each utilization instance, it does disclose how often a secondary user utilizes the licensed spectrum. The primary user can also infer whether a secondary user is more active than another one. To hide the number of utilization instances, the secondary user can submit some dummy utilization instances, such as some tuples $< t_i, \triangle t_i, f >$ with $\triangle t_i = 0$. In this way, the primary user cannot learn the exact number but a upper bound of utilization instances. To even hide the upper bound, i.e., to make the upper bound meaningless, each secondary user can submit a large amount of dummy instances. Too many dummy instances, however, introduce heavy overhead, especially for less active secondary users.

Since the responses of queries disclose secondary users' information, we must limit the number of queries a primary user can make on an individual secondary user. And we assume that the information disclosed in the limited number of queries is tolerable. If even this limited information is not tolerable, we can hide the information disclosed in the responses of queries by changing a little bit the system architecture. In the architecture shown in Fig.2, the monitor does not communicate with the secondary user. We can add a communication link

from the monitor to the secondary user. When the primary user queries the monitor, the monitor randomly chooses a statement ϕ, for instance, that the secondary user is utilizing a channel at time t_u. Instead of sending t_u to the primary user as in our protocol, the monitor only sends the commitment of t_u to the primary user. At the same time, the monitor sends the opening message of the commitment of t_u to the secondary user. Then the secondary user proves to the primary user by interval check and zero-knowledge proof that (1) it knows the opening of commitment of t_u and (2) t_u is a point of time in one of its utilization instances submitted. Even there is no link from the monitor to the secondary user, we can utilize public key cryptography to let the primary user relay t_u to the secondary user; the monitor encrypts t_u with the secondary user's public key, and only the secondary user can decrypt it since no other entity has its secret key.

VI. Random Checking

As described above, random checking plays an important role in our protocol. In this section, we will first propose rules for the monitor to efficiently capture secondary users' utilization instances. Then we will analyze the effectiveness of random checking. Specifically, we show that the monitor can capture enough random checking instances even when the probability to capture each individual instance is low, and the captured random checking instances can considerably catch a secondary user who commits fraud.

A. Monitoring Scheme

Suppose a primary user has n channels of licensed spectrum to share with l secondary users. And there is one monitor that can only monitor one channel at a time. Our problem is that what strategy the monitor should apply such that the monitor can capture as many utilization instances among all secondary users and all channels.

1) Problem Statement: Suppose that the traffic on all channels is the same. We assume that on each channel both the duration of busy time and the duration of idle time follow exponential distributions with parameter λ_B and λ_I, respectively. The average busy and idle times are denoted by $T_B = 1/\lambda_B$ and $T_I = 1/\lambda_I$, respectively. We also assume that when a monitor jumps from one channel to another, there is a switch cost; that is, it wastes a period of time T_c. The objective is to capture as many utilization instances as possible in a given period of time T_0 among all secondary users. As the monitor does not know how secondary users choose channels and when they utilize the chosen channels, we assume that secondary users randomly choose channels and utilization instances can be randomly distributed on each channel.

2) Monitoring Rules: It is hard to give an optimal solution to the above problem deterministically. Here we propose some heuristic rules to control the monitor. Remember that a random check instance is a point of time that a certain secondary user is utilizing a channel. The monitor can choose to either stay on the current channel or switch to another channel. Our intuition is to let the monitor make a better choice between stay and switch. Suppose at a time the monitor is not scanning any channel, and now it jumps to a channel to start scanning. Then the monitor have to decide what to do after jumping to a channel. There are in total two scenarios: (i) the monitor jumps to a channel currently busy; (ii) the monitor jumps to a channel currently idle.

• When a monitor jumps to a channel which is currently busy, it will stay until it is able to figure out which secondary user is utilizing that channel. After the monitor has determined the secondary user's identity, i.e., captured a random-check instance, it will determine whether to stay on the current channel or switch to another channel. If it chooses to stay, the probability that it encounters a busy channel during T_c, p_{stay_b}, is:

$$\begin{aligned} p_{stay_b} &= \int_0^{T_c}\int_0^{y} \lambda_B\lambda_I e^{-\lambda_B x} e^{-\lambda_I(y-x)} dxdy \\ &= \begin{cases} 1 - \frac{1}{\lambda_B-\lambda_I}(\lambda_B e^{-\lambda_I T_c} - \lambda_I e^{-\lambda_B T_c}) & \text{if } \lambda_B \neq \lambda_I \\ 1 - \lambda_B T_c e^{-\lambda_B T_c} - e^{-\lambda_B T_c} & \text{if } \lambda_B = \lambda_I \end{cases} \end{aligned} \tag{1}$$

In the case of switch, the monitor wastes T_c of time for monitoring. At the time point of T_c, the probability that the channel is busy, p_{switch}, is:

$$p_{switch} = \frac{T_B}{T_B + T_I} \tag{2}$$

We can obtain the critic switch cost, T_c^*, by setting $p_{stay_b} = p_{switch}$. When $T_c > T_c^*$, we have $p_{stay_b} > p_{switch}$; the monitor should continue to stay on the channel that it is currently monitoring. When $T_c < T_c^*$, we have $p_{stay_b} < p_{switch}$; the monitor should switch to another channel which has been monitored less frequently. When $T_c = T_c^*$, we have $p_{stay_b} = p_{switch}$; the monitor can choose to either stay on the current channel or switch to another channel.

• When the monitor jumps to an idle channel, we can obtain another critic switch cost $T_c^{*'}$ similarly. In the case of stay, during the time period of T_c, the probability that the monitor encounters a busy channel, p_{stay_i}, is:

$$p_{stay_i} = \int_0^{T_c} \lambda_I e^{-\lambda_I x} dx = 1 - e^{-\lambda_I T_c} \tag{3}$$

In the case of switch, at the time point of T_c, the probability that the channel is busy, p_{switch}, is same as Eq(2). By setting $p_{stay_i} = p_{switch}$, we get $T_c^{*'} = T_I \ln \frac{T_B}{T_B+T_I}$. When $T_c > T_c^{*'}$, we have $p_{stay_i} > p_{switch}$; the monitor should continue to stay on the idle channel that it is monitoring. When $T_c < T_c^{*'}$, we have $p_{stay_i} < p_{switch}$; the monitor should switch to another channel which has been monitored less frequently. When $T_c = T_c^{*'}$, we have $p_{stay_i} = p_{switch}$; the monitor can choose to either stay on the current channel or switch to another channel.

We summarizes the rules as follows:

1) When a monitor jumps to a channel that is currently busy, it stays until it identifies the secondary user that is utilizing the channel. Then the monitor chooses to stay or switch according the values of T_c and T_c^* as described above;

2) When a monitor jumps to a channel that is currently idle, it chooses to stay or switch according T_c and $T_c^{*'}$ as described above.

B. Random-check Effectiveness

Since the monitor cannot cover all the channels all the time, it only captures a portion of the total utilization instance. Note that in Section V, we assume that the random-check mechanism refrains a secondary user from committing fraud. Here we analyze the effectiveness of the random-check mechanism. We will show that a secondary user has very high probability to be caught if it commits fraud even the monitor only captures a limited number of random-check instances.

Let p be the probability that the monitor captures a secondary user in one second interval when the latter is utilizing a channel. To simplify the analysis, we assume that the probability is the same for all channels in the licensed spectrum. Let n be the total time (in seconds) that a secondary user has utilized the licensed spectrum[‡]. Let X be the number of captures in n seconds. Note that a utilization instance may last far more than one second, so it is possible that a utilization instance is captured by the monitor more than once. Obviously, X follows the binomial distribution with parameter n and p, i.e., $X \sim B(n, p)$. The probability mass function is:

$$Pr(X = k) = \binom{n}{k} p^k (1-p)^{n-k} \tag{4}$$

The expected number of captures is $E(X) = np$. The probability that a secondary user is captured at least k times in n second utilization is:

$$Pr(X >= k) = 1 - \sum_{i=0}^{k-1} \binom{n}{i} p^i (1-p)^{n-i} \tag{5}$$

Fig.3 shows the relation between p and n when $Pr(X >= 5) = 0.95$. We can see that n decreases almost exponentially as p increases. Even when p is as small as 0.01, it just takes 300 seconds for the monitor to capture a secondary user at least 5 times. Therefore, it dose not take a long time before the monitor obtains considerable amount of captures even the probability to capture a secondary user in one second is small.

Now we will show the probability with which a secondary user can be caught if it commits fraud. Suppose a secondary user has utilized the licensed spectrum for n seconds, and it only submits $n - m$ seconds of data. We also assume the primary user queries the monitor for k instances of random-check. The probability that the secondary user is caught committing fraud is:

$$Pr[caught] = 1 - \binom{n-k}{n-m-k} / \binom{n}{n-m} \tag{6}$$

Fig.4 shows the relation between m and $Pr[caught]$ when $n = 200$ and $k = 5$. We can see that the more data a secondary user chooses not to submit, the higher chance he will be caught. For instance, if a secondary user purposely chooses not to submit 20 second data (10% of total utilization data), it has about 41% of chance to be caught; if it chooses not submit 50 second data (25% of total utilization data), it has about 77% of chance to be caught. To refrain the secondary user from committing fraud, we can set a penalty higher than the gain such that a rational secondary user would not choose to commit fraud.

[‡]The secondary user could use any channel in the licensed spectrum.

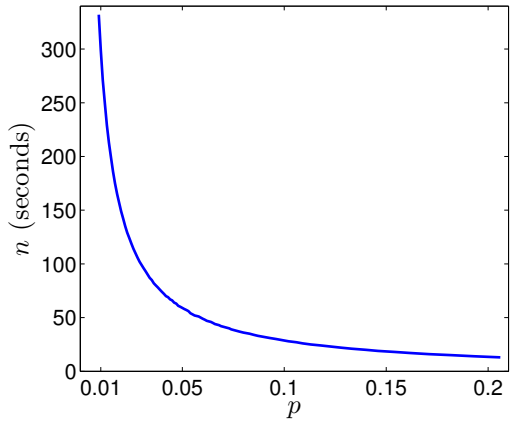

Fig. 3. The relation between p and n when $Pr(X >= 5) = 0.95$.

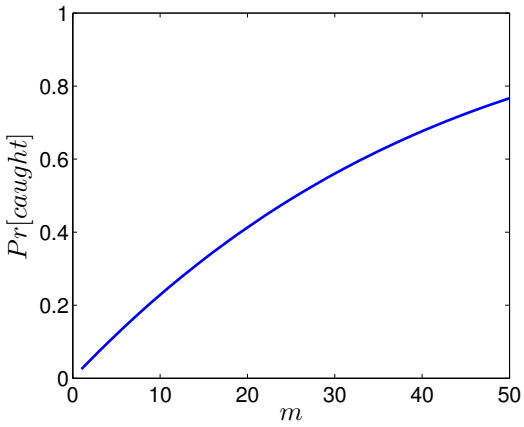

Fig. 4. The relation between m and $Pr[caught]$ when $n = 200$ and $k = 5$.

VII. Implementation and Evaluation

In this section, we will first present in detail our implementation of commitment scheme, signature scheme, and zero-knowledge proof, which are intensively used in our protocol. Then we will evaluate the running time of our protocol.

A. Commitment Scheme

In the implementation of commitment scheme, we use Pedersen commitments [22]. Pedersen commitments are theoretically hiding and binding, relying on a group G of prime order p with generators g and h. Under this scheme, a commitment C_x to message $x \in Z_p$ has the form of $C_x = g^x h^{o_x}$, where o_x is an *opening* nonce chosen uniformly at random in Z_p. In our implementation, we use a group G based on bilinear pairing

over elliptic curves [23]. Here we say that the commitment scheme is done in ECC (Elliptic Curve Cryptography) domain.

B. Signature Scheme

All messages are signed by the sender and verified by the recipient in our protocol. For messages between the monitor and the primary user, and messages from the secondary user to the primary user, any unforgeable signature scheme can be used. In our implementation, we choose ECDSA signature scheme [24] in ECC domain. As to the signature scheme for the message from the primary user to the secondary user, especially the pricing policy, we utilize ECC version of C-L signature based on elliptic curve groups [21]. C-L signature scheme [21] can not only compute a signature on a commitment message but also provide zero-knowledge proof of knowledge of a signature on a committed value. These properties are leveraged to ensure the primary user that the price used by the secondary user during billing process is correct.

C. Non-Interactive Zero-Knowledge Proof

Recall that in the privacy-preserving protocol,the primary user sends the pricing policy in plain text together with pricing policy signed by CL-signature scheme to the secondary user. The secondary user then calculates the payment for each utilization instance ($< t_i, \triangle t_i, f >$) and sends the commitment of the payment back to the primary user. The secondary user needs to use non-interactive zero-knowledge proof to prove that: (1) t_i is in an interval $[T_{b_j}, T_{e_j}]$ such that $T_{b_j} \leq t_i \leq T_{e_j}$; (2) the payment fee_i is the product of p_j and $\triangle t_i$; and (3) p_j is indeed the price associated with $< T_{b_j}, T_{e_j}, f >$. Therefore, the basic building blocks are a non-interactive zero-knowledge proof that a committed value lies in an interval, a non-interactive zero-knowledge proof that a committed value is the product of two committed value, and a non-interactive zero-knowledge proof that a committed value has a CL-signature.

1) Interval Check: Suppose we need to prove that a committed value x lies in an interval $[a, b]$, i.e., $a \leq x \leq b$. It is equivalent to prove that $x - a \geq 0$ and $b - x \geq 0$. To prove that an integer $v >= 0$, we employ the non-interactive zero-knowledge proof by Groth [25]. It is well-known from number theory that $4v + 1$ can be written as a sum of three squares if integer v is non-negative.

In the proof in [25], integer commitment scheme is constructed with special RSA modulus (here we say RSA domain), in which the order of generators is unknown. However, all the commitments in our protocol are constructed in ECC domain, where the order of generators is known. We need first to convert a commitment in ECC domain into a commitment in RSA domain. We then use non-interactive zero-knowledge proof to prove that a commitment in ECC domain and a commitment in RSA domain commit to the same value. Next we prove that the committed value in RSA domain is non-negative with the non-interactive zero-knowledge proof mentioned above.

2) Proof of Multiplication: Giving commitments of p_j, $\triangle t_i$, and fee_i, i.e., $C_{p_j} = g^{p_j} h^{o_{p_j}}$, $C_{\triangle t_i} = g^{\triangle t_i} h^{o_{\triangle t_i}}$, and $C_{fee_i} = g^{fee_i} h^{o_{fee_i}}$, we need to prove that $fee_i = p_j \triangle t_i$. We adopt the notation introduced by Camenisch and Stadler [26]. The proof is as follows:

$$
\begin{aligned}
NIPK\{(&fee_i, o_{fee_i}, \triangle t_i, o_{\triangle t_i}, p_j, o_{p_j}) : \\
&(C_{fee_i}, o_{fee_i}) = Commit(par, fee_i) \wedge \\
&(C_{p_j}, o_{p_j}) = Commit(par, p_j) \wedge \\
&(C_{\triangle t_i}, o_{\triangle t_i}) = Commit(par, \triangle t_i) \wedge \\
&fee_i = p_j \triangle t_i \\
\}&
\end{aligned}
$$

In the above notation, $NIPK$ stands for *Non-Interactive Proof of Knowledge*. The variables in the parenthesis, i.e., $fee_i, o_{fee_i}, \triangle t_i, o_{\triangle t_i}, p_j, o_{p_j}$, denote quantities whose knowledge is being proven, while all other values are known to the verifier.

3) Possession of a CL-signature: In the above proof of multiplication, it lacks of another piece of proof that p_j is the correct unit price that should be used. This is done by the proof of possession of a CL-signature on p_j. That is, the secondary user proves to the primary user that the former possesses a CL-signature computed by the latter on $< T_{b_j}, T_{e_j}, f, p_j >$ that states that p_j is the unit price to be paid for (T_{b_j}, T_{e_j}, f). We implement the proof of knowledge of a CL-signature on a commitment in [21].

D. Running time

We have implemented all the functionality required to the privacy-preserving protocol in C. We use both pbc library [27] and GMP library [28]. Our platform is Intel(R) Xeon(R) CPU at 1.02GHz on OPENSUSE operating system. We can measure the running time for each individual block, such as ECC commitment, integer commitment, CL-signature, proof of possession of CL-signature, proof of multiplication, and interval checking. All blocks, except interval check, are done in ECC domain; the key size is 160 bits (192 bit key for ECDSA). The interval check involves both ECC and RSA modulus; for ECC part, the key size is still 160 bits, and for RSA part, the key size is 2048. These choices of key sizes are believed to achieve high security. The running time for all fundamental blocks is shown in Table I.

TABLE I
THE RUNNING TIME OF INDIVIDUAL BLOCKS.

	Key size	time (ms)
Sign a policy (ECC)	160	16
Verify a policy (ECC)	160	18
Generate commitment (ECC)	160	4.5
Verify commitment (ECC)	160	2.4
Prove interval check (ECC/RSA)	160/2048	30
Verify interval check (ECC/RSA)	160/2048	20
Prove possession of C-L signature (ECC)	160	20
Verify possession of C-L signature (ECC)	160	11
Prove multiplication (ECC)	160	4.9
Verify multiplication (ECC)	160	4.7
ECDSA (ECC)	192	1.2

We also measure the running time for pricing policy generation, proving a bill, and verifying a bill, which is shown in

Table II.

TABLE II
THE RUNNING TIME OF PROOF AND VERIFICATION.

	Key size	time per reading (ms)
policy generation	160	35
prove bill	160	100
verify bill	160	65
random checking	192	1.2

We generate some synthetic data to simulate a real scenario. We assume that there is a secondary user that utilizes licensed spectrum in a day. This secondary user can use several channels while it can only use one at a time. We suppose that the time duration of utilization instances follows an exponential distribution, and the time duration between two utilization instances also follows an exponential distribution. We then generate different traces for a secondary user. For the pricing policy, we introduce three linear policies, each of which spans 8 hours. Table III shows the total time required for both the primary user and the secondary user for different traces for a billing period of one day.

TABLE III
THE RUNNING TIME OF ONE-DAY BILLING.

# of utilization instances	PU (second)	SU (second)
12	0.84	1.3
17	1.1	1.8
24	1.6	2.5
37	2.4	3.8
48	3.2	5.0

VIII. CONCLUSION

In this paper, we propose a privacy-preserving protocol for cognitive radio transactions. We utilize commitment scheme and zero-knowledge proof to preserve secondary user's privacy and use a random-checking monitor to protect the primary user's interests. Furthermore, we formulate the monitoring problem and analytically propose efficient rules to control the random-checking monitor such that the monitor can capture as many utilization instances as possible. Finally, we implement our protocol and evaluate the performance. To the best of our knowledge, it is the first work to address the privacy of secondary users in cognitive radio transactions.

ACKNOWLEDGMENT

The authors would like to thank all the reviewers for their helpful comments. This project was supported in part by US National Science Foundation grants CNS-1117412, CNS-1320453, and CAREER Award CNS-0747108.

REFERENCES

[1] J. Mitola III and G. Maguire Jr, "Cognitive radio: making software radios more personal," *IEEE Personal Communications*, vol. 6, pp. 13–18, 1999.

[2] Z. Qin, Q. Li, and G. Hsieh, "Defending against cooperative attacks in cooperative spectrum sensing," *IEEE Trans on Wireless Communications*, vol. 12, pp. 2680–2687, 2013.

[3] S. Haykin, "Cognitive radio: brain-empowered wireless communications," *IEEE journal on selected areas in communications*, vol. 23, pp. 201–220, 2005.

[4] H. Li, X. Cheng, K. Li, X. Xing, and T. Jing, "Utility-based cooperative spectrum sensing scheduling in cognitive radio networks," in *IEEE Infocom Mini-Conference*, April 14-19 2013, pp. 165–169.

[5] H. Li, X. Cheng, K. Li, C. Hu, N. Zhang, and W. Xue, "Robust collaborative spectrum sensing schemes for cognitive radio networks," *IEEE Trans on Parallel and Distributed Systems*, 2013.

[6] Q. Yan, M. Li, T. Jiang, W. Lou, and Y. T. Hou, "Vulnerability and protection for distributed consensus-based spectrum sensing in cognitive radio networks," in *IEEE Infocom*, 2012, pp. 900–908.

[7] S. Gandhi, C. Buragohain, L. Cao, H. Zheng, and S. Suri, "A general framework for wireless spectrum auctions," in *IEEE DySPAN*, 2007, pp. 22–33.

[8] D. Niyato and E. Hossain, "Market-equilibrium, competitive, and cooperative pricing for spectrum sharing in cognitive radio networks: Analysis and comparison," *IEEE Trans on Wireless Communications*, vol. 7, pp. 4273–4283, 2008.

[9] X. Wang, Z. Li, P. Xu, Y. Xu, X. Gao, and H.-H. Chen, "Spectrum sharing in cognitive radio networksan auction-based approach," *IEEE Trans on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 40, pp. 587–596, 2010.

[10] A. Rial and G. Danezis, "Privacy-preserving smart metering," in *Proceedings of the 10th annual ACM workshop on Privacy in the electronic society*, 2011, pp. 49–60.

[11] R. Popa, H. Balakrishnan, and A. Blumberg, "VPriv: protecting privacy in location-based vehicular services," in *Proceedings of the 18th conference on USENIX security symposium*, 2009, pp. 335–350.

[12] J. Balasch, A. Rial, C. Troncoso, B. Preneel, I. Verbauwhede, C. Geuens, K. Leuven, and S. ICRI, "PrETP: Privacy-preserving electronic toll pricing," in *Usenix Security*, vol. 10, 2010.

[13] J. Teng, B. Zhang, X. Bai, Z. Yang, and D. Xuan, "Incentive-driven and privacy-preserving message dissemination in large scale mobile networks," *IEEE Trans on Parallel and Distributed Systems*, 2013.

[14] W. Wei, F. Xu, and Q. Li, "MobiShare: Flexible privacy-preserving location sharing in mobile online social networks," in *IEEE Infocom*, 2012, pp. 2616–2620.

[15] B. Zhang, J. Teng, X. Bai, Z. Yang, and D. Xuan, "P3-coupon: A probabilistic system for prompt and privacy-preserving electronic coupon distribution," in *IEEE International Conference on Pervasive Computing and Communications*, 2011, pp. 93–101.

[16] Z. Qin, Q. Li, and M. Chuah, "Unidentifiable attacks in electric power systems," in *Proceedings of IEEE/ACM International Conference on Cyber-Physical Systems*, 2012, pp. 193–202.

[17] W. Wei, F. Xu, C. C. Tan, and Q. Li, "Sybildefender: Defend against sybil attacks in large social networks," in *IEEE Infocom*, 2012, pp. 1951–1959.

[18] Z. Qin, Q. Li, and M. Chuah, "Defending against unidentifiable attacks in electric power grids," *IEEE Trans on Parallel and Distributed Systems*, pp. 1961–1971, 2013.

[19] Q. Yan, M. Li, F. Chen, T. Jiang, W. Lou, Y. T. Hou, and C.-T. Lu, "Non-parametric passive traffic monitoring in cognitive radio networks," in *IEEE Infocom*, 2013, pp. 1240–1248.

[20] A. Fiat and A. Shamir, "How to prove yourself: practical solutions to identification and signature problems," in *Advances in Cryptology-CRYPTO*, 1987, pp. 186–194.

[21] J. Camenisch and A. Lysyanskaya, "Signature schemes and anonymous credentials from bilinear maps," in *Advances in Cryptology–CRYPTO*, 2004, pp. 56–72.

[22] T. P. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," in *Advances in CryptologyCRYPTO*, 1992, pp. 129–140.

[23] J. H. Silverman, *The arithmetic of elliptic curves*. Springer, 2009.

[24] D. Johnson, A. Menezes, and S. Vanstone, "The elliptic curve digital signature algorithm (ECDSA)," *International Journal of Information Security*, vol. 1, pp. 36–63, 2001.

[25] J. Groth, "Non-interactive zero-knowledge arguments for voting," in *Applied Cryptography and Network Security*, 2005, pp. 467–482.

[26] J. Camenisch and M. Stadler, *Proof systems for general statements about discrete logarithms*. Citeseer, 1997.

[27] B. Lynn, "PBC library," *Online: http://crypto. stanford. edu/pbc*, 2006.

[28] T. Granlund *et al.*, "GMP, the GNU multiple precision arithmetic library," *Online: http://gmplib.org/*, 1991.