Detection of PUE Attacks in Cognitive Radio Networks Based on Signal Activity Pattern

ChunSheng Xin, *Senior Member, IEEE* and Min Song, *Senior Member, IEEE*

Abstract—Promising to significantly improve spectrum utilization, *cognitive radio networks* (CRNs) have attracted a great attention in the literature. Nevertheless, a new security threat known as the *primary user emulation* (PUE) attack raises a great challenge to CRNs. The PUE attack is unique to CRNs and can cause severe *denial of service* (DoS) to CRNs. In this paper, we propose a novel PUE detection system, termed *Signal activity Pattern Acquisition and Reconstruction System*. Different from current solutions of PUE detection, the proposed system does not need any a priori knowledge of primary users (PUs), and has no limitation on the type of PUs that are applicable. It acquires the activity pattern of a signal through spectrum sensing, such as the ON and OFF periods of the signal. Then it reconstructs the observed signal activity pattern through a reconstruction model. By examining the reconstruction error, the proposed system can smartly distinguish a signal activity pattern of a PU from a signal activity pattern of an attacker. Numerical results show that the proposed system has excellent performance in detecting PUE attacks.

Index Terms—Cognitive radio network, primary user emulation attack, primary user emulation detection

1 Introduction

Many studies have indicated that a significant amount of licensed spectrum is considerably under-utilized in both temporal and spatial domains. Such *spectrum holes* or *white spaces* offer a great opportunity for wireless communication. As such *cognitive radio networks* (CRNs) have been proposed to exploit this opportunity. In CRNs, unlicensed or *secondary users* (SUs) dynamically search for idle licensed spectrum bands or *channels* through spectrum sensing, and access them for communications. On the other hand, to ensure the access privilege of licensed or *primary users* (PUs), the SUs are required not to generate harmful interference to PUs. In other words, once a PU signal is detected on a channel, SUs have to give up channel access. However, such a requirement raises a security threat to CRNs, known as the *primary user emulation* (PUE) attack. Specifically, an attacker, which may be a malicious user or a selfish SU, can transmit a PU signal using its cognitive radio. The benign SUs, which cannot distinguish that the PU signal is from a PU or an attacker, have to evacuate from the channel. The PUE attack causes *denial of service* (DoS) to the CRN and can result in severe performance degradation, due to service disruption at the MAC and higher layers. Hence, the PUE attack greatly limits the spectrum access opportunity of SUs.

- *C. Xin is with the Department of Electrical and Computer Engineering, Old Dominion University, Norfolk, VA 23529 USA. E-mail: cxin@odu.edu.*
- *M. Song is with the Department of Electrical Engineering and Computer Science, University of Toledo, Toledo, OH 43606 USA. E-mail: min.song@utoledo.edu.*

Manuscript received 26 Dec. 2012; revised 3 Aug. 2013; accepted 22 Aug. 2013. Date of publication 12 Sep. 2013; date of current version 15 May 2014. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier 10.1109/TMC.2013.121

While CRNs are also vulnerable to other security attacks that are common to wireless networks, the PUE attack is unique to CRNs and is a fundamental threat to CRNs. As such, it has attracted many research efforts. The existing studies on PUE attacks can be classified into two categories, detection or defense. The strategy for defense based schemes [1] is similar to the one for anti-jamming. The SU does not care if a busy channel is due to a signal from a PU or an attacker. It simply senses the channels, and picks an idle channel for communications by some game theoretic strategies that maximize the chance to 'escape' from the attackers. Nevertheless, this approach is not effective for PUE attacks launched by selfish SUs, which have the internal knowledge of an CRN, and also not effective for Sybil PUE attacks where one malicious user launches many attacks on different channels simultaneously [2].

The detection based schemes aim to verify if the transmitter of a PU signal is a PU or an attacker [3]–[5]. There are primarily two approaches, *location verification* or *hardware fingerprint verification*. The location verification approach [3], [5] assumes that the PUs are TV towers with locations known to SUs. The main goal is to develop algorithms to estimate the location of a PU signal transmitter, e.g., through a wireless sensor network [5] or a technique called *location belief propagation* [3], and then look up the database of TV tower locations, to find whether the transmitter is an attacker (if the estimated location is not in the database). The second approach for PUE detection is to use the radio hardware fingerprint of the transmitter. The authors in [4] presented an interesting work that extracts the transmitter fingerprint from a signal, namely the phase shift difference, carrier frequency deviation from the ideal signal, etc., which are unique for a given transmitter. To detect PUE attacks, a received signal is processed by two modules. The first module extracts the fingerprint from the signal,

and the second module extracts the ID of the PU from the signal, e.g., the MAC address. The paper assumed that the SU has some way to check if a PU ID is valid, e.g., through an ID database of PUs, and hence an attacker has to use a previously overheard PU ID. Hence, if there is another signal (previously received) that has the same PU ID, but a different fingerprint, then a PUE attack is alarmed. There are also PUE detection studies that target a particular type of PUs. For instance, the authors in [6] proposed an interesting method that exploits the correlation between RF signals and acoustic information to detect the PUE attacker emulating a wireless microphone.

While the existing studies on PUE detection are promising, the problem is not completely solved. The current solutions need certain critical a priori knowledge of PUs. Specifically, the location verification approach assumes that the locations of PUs are known to the SU, and the fingerprint approach assumes that the IDs of PUs are known to the SU (as otherwise an attacker can simply use an arbitrary ID different from any previously overheard PU ID, so that the detection algorithm would not find two fingerprints with the same ID). Furthermore, the current solutions can be applied to only certain type of PUs. The location verification approach requires that PUs be static, such as TV towers. The fingerprint approach requires that the PU ID be included in the signal and can be extracted by SUs, which is often difficult, e.g., for analog PUs.

In this paper, we propose a novel PUE detection system, termed *Signal activity Pattern Acquisition and Reconstruction System* (SPARS). In the ensuing discussion, if not otherwise noted, an attacker refers to a PUE attacker, a signal refers to a PU signal, and a transmitter refers to a PU signal transmitter, which may be a PU or an attacker. We define a *signal activity pattern* (SAP) of a transmitter as a series of *ON* and/or *OFF* periods of the transmitter along the time. An ON period refers to the duration of a busy period that the transmitter is transmitting and the SUs must be refrained from communications. An OFF period refers to the duration of an idle period between two adjacent ON periods. Different from current solutions on PUE detection, SPARS does not have limitation on the type of PUs, i.e., SPARS can be applicable to all types of PUs. Furthermore, SPARS does not need any a priori knowledge of PUs. It acquires the SAP of a transmitter through spectrum sensing, and compares it with SAPs of PUs through a *SAP reconstruction model*. If the observed SAP is not 'like' the SAPs of PUs, which is measured by the reconstruction error, then the transmitter is an attacker.

Our motivation is that while an attacker can cheat on the signal itself, it cannot cheat on its objective, i.e., causing DoS to the CRN. An attacker can transmit a PU signal, but its SAP is expected to be different from the ones of PUs. This is because the objective of the attacker is to occupy the channel to cause DoS to the CRN. Therefore, the attacker aims to significantly decrease the channel availability to the CRN, e.g., by increasing the ON periods and/or decreasing the OFF periods. Thus the attacker creates a different SAP from PUs. On the other hand, if an attacker also cheats on its SAP, i.e., manipulates its spectrum occupation to be similar to the one of PUs, we argue that such a 'mild' PUE attack is tolerable by the CRN, and hence defeats the DoS objective of the attacker. This is because a CRN usually selects the operation channels with low spectrum occupation by PUs. An attacker with a similarly low spectrum occupation is not a serious threat to the CRN, as the CRN has been designed with the mild disruption (from PUs) in mind, and hence is tolerable to a PUE attack that causes a mild disruption. Therefore, by targeting the objective of the attacker, SPARS is effective to detect PUE attack.

We use a Bayesian method and sparse modeling to develop a good SAP reconstruction model to compare a candidate SAP with the SAPs of PUs. The sparse modeling has been widely used in the literature to solve various problems in science and engineering fields [7]–[14], due to its good performance of data/signal reconstruction. In [7], the authors studied the dictionary structure for sparse representation of multi-dimension data and presented an efficient algorithm to learn the dictionary. The author in [8] showed that for most sparse approximation problems, the solution can be obtained efficiently. The authors in [9] used sparse modeling for face recognition. The problem of face recognition is formulated as a classification problem among multiple classes of objects and the objects are constructed with sparse representation. In [10], the authors presented a sparse coding model to obtain high-dimensional sparse representations of sensory data. The authors in [11] applied sparse modeling for unusual event detection in videos, through online reconstruction of an event using a dynamically learned dictionary. In [12], the authors used sparse modeling for speech recognition. The authors in [13] studied sparse modeling in computer vision and image processing, and presented the neurobiological implication of sparse modeling. The authors in [14] presented an efficient algorithm to solve the sparse coding or the sparse optimization problem.

Before we get into the details of SPARS, we summarize our main contributions as follows.

- We have designed a PUE detection system, SPARS, which does not need a priori PU knowledge such as the locations or IDs of PUs, and has no limitation on the type of applicable PUs such as static PUs or PUs with extractable IDs.
- We have developed a SAP reconstruction model through a Bayesian method, to train SPARS and reconstruct an observed SAP. We have incorporated the sparsity and other principles in the model design, to achieve excellent performance for SAP reconstruction.
- We directly target the objective of the attacker which the attacker cannot hide, and utilize a technique called *tolerance interval* to test the normality of the reconstruction error and accordingly detect the PUE attack.

The remainder of the paper is organized as follows. Section 2 describes the network model. Section 3 describes the architecture and the main idea. The SAP reconstruction model is presented in Section 4. Sections 5 and 6 describe the system training and the PUE detection. Section 7 develops the Chernoff bounds for the performance of SPARS. Section 8 introduces a statistics based PUE detection scheme, which is studied comparatively with SPARS.

Section 9 presents performance evaluation and Section 10 concludes the paper.

2 Network Model and Assumptions

There are two types of attackers: 1) a selfish SU that wants to use a channel exclusively and thus launches a PUE attack to drive benign SUs out of the channel; and 2) a malicious user that simply wants to cause DoS to the CRN. While there is a slight difference, we do not differentiate them for the purpose of detection, since their common objective is to cause DoS.

We adopt a fingerprint technique such as the one in [4] to track the transmitter of a signal, and hence SPARS is applicable to both static and mobile PUs. Specifically, each SU independently carries out spectrum sensing on every channel. For each received signal, an SU extracts its fingerprint, and records the ON and OFF periods of the signal, which are stored in a database together with the fingerprint of the signal. The fingerprint essentially serves as the transmitter ID. Hence with a little abuse of the language, we say that each SU has recorded the ON and OFF periods of every PU signal transmitter (identified by the fingerprint). The spectrum sensing can be carried out either continuously or periodically in the time domain. In the former case, an ON or OFF period is a continuous random variable. In the latter case, let T_s denote the duration of the spectrum sensing cycle. An ON or OFF period is then a discrete random variable, represented as the number of time unit T_s. SPARS can be run periodically or on-demand as necessary. When SPARS is invoked to detect if a transmitter is an attacker, the last k ON or OFF periods of this transmitter (identified by the fingerprint) are used to compose the SAP to be fed to SPARS.

In most studies of PUE detection [3]–[5], only one PU transmitter is assumed to be transmitting at any time. While this assumption is usually true, we introduce a technique that can handle the scenario when multiple transmitters send PU signals simultaneously, within the sensing range of an SU. The *directional spectrum sensing* technique in [15] can be used to distinguish the transmitters that are simultaneously transmitting signals. This technique allows an SU to select a given spatial direction and listen to the signal only at that direction. Here we assume that the transmitters at the same direction of an SU are not transmitting simultaneously, as otherwise they would likely be interfering each other. Combining the directional spectrum sensing and the fingerprint technique, we can address both mobile PUs and simultaneously transmitting PUs.

At last, in this paper, we assume that there is only one class of PUs that have similar SAP features, i.e., with similar distributions for the ON/OFF periods. Nevertheless, the proposed PUE detection approach can be extended to address the scenario with multiple classes of PUs that have different SAP features, which will be studied in our future work.

3 Architecture of SPARS

The main idea of SPARS is to use a set of n vectors $\boldsymbol{B}_1, \ldots, \boldsymbol{B}_n \in \mathbb{R}^k$, which are called *bases* in this paper, to reconstruct a SAP. Specifically, let a column vector $\boldsymbol{Y} \in \mathbb{R}^k$ denote a SAP. Our objective is to reconstruct $\boldsymbol{Y}$ using bases $\boldsymbol{B}_1, \ldots, \boldsymbol{B}_n$ as

$$\boldsymbol{Y} = \sum_{i=1}^{n} \boldsymbol{B}_i W_i, \tag{1}$$

where W_i $(1 \le i \le n)$ is the weight associated with base $\boldsymbol{B}_i$ to compose $\boldsymbol{Y}$.

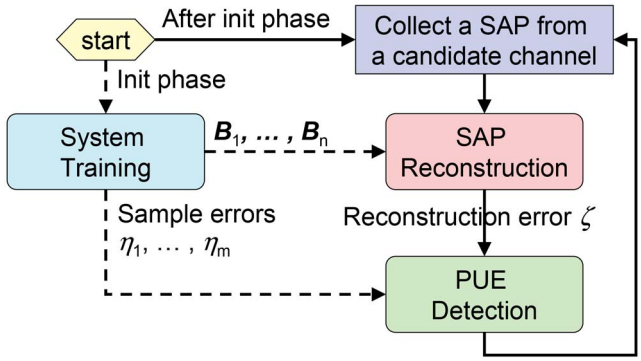

Fig. 1. SPARS architecture.

The bases $\boldsymbol{B}_1, \ldots, \boldsymbol{B}_n$ are carefully learned through a training process to capture the essential features of the SAPs of PUs. For the reconstruction model in (1) to be a good model, it is typically not a determined system, but an overdetermined or underdetermined system. Hence the reconstruction of a SAP results in a *reconstruction error*, which is utilized by SPARS to detect PUE attack. This is because the SAPs of PUs can be reconstructed by (1) very well, i.e., with a small reconstruction error, as the bases have captured the essential features of such SAPs. On the other hand, as discussed in Section 1, the SAPs of attackers would have different features from the SAPs of PUs, since the attackers aim to cause severe DoS to the CRN. Therefore, if we use the bases to reconstruct a SAP of an attacker, it would turn out that the reconstruction cannot be performed well, i.e., we would have a large reconstruction error. Thus, from the reconstruction error, we can tell if the transmitter of SAP $\boldsymbol{Y}$ is an attacker or a PU.

Fig. 1 illustrates the architecture of SPARS. It consists of three modules: *system training*, *SAP reconstruction*, and *PUE detection*. In the initial CRN setup phase, an SU passively performs spectrum sensing to collect a set of SAPs from PUs for the purpose of training SPARS. This set of SAPs is called the *training data set*. The system training module learns the bases $\boldsymbol{B}_1, \ldots, \boldsymbol{B}_n$ from the training data set. After learning the bases, the system training module also computes the reconstruction errors for the training data set, $\eta_1, \ldots, \eta_m$, called the *sample errors* in the figure. This module can be re-run periodically to update the training data set and the bases.

After SPARS is trained, then an SU can use it for PUE detection. Suppose the SU wants to find if there is an attacker in a candidate channel. It first collects a SAP from this channel. Then the SU uses the SAP reconstruction module of SPARS to reconstruct this SAP using the learned bases $\boldsymbol{B}_1, \ldots, \boldsymbol{B}_n$, and compute the reconstruction error ζ for this SAP. Next, the PUE detection module is used to check if ζ falls in a *tolerance interval* of the sample errors $\eta_1, \ldots, \eta_m$, which have been obtained in the initial system

training. If it does not fall in the tolerance interval, then this SAP is treated from an attacker, and the transmitter of this SAP is alarmed as an attacker.

Both the SAP reconstruction and the system training rely on a good *SAP reconstruction model*. The system training needs the model to select the best bases $\boldsymbol{B}_1, \ldots, \boldsymbol{B}_n$ for the purpose of SAP reconstruction in the future. On the other hand, the SAP reconstruction module needs the model to select best weights $W_1, \ldots, W_n$ to minimize the reconstruction error. Following the work in [7]–[9], [13], we develop an SAP reconstruction model through a Bayesian method in the next section.

4 SAP Reconstruction Model

A determined model in (1) has poor performance for reconstructing arbitrary input data (SAP in this paper), because it targets only the training data during system training. In other words, a determined model captures all features of the training data, some of which are actually not desirable when reconstructing the future input data, as they represent deviations of training data from a typical input data. Therefore, it is desirable to design an overdetermined or underdetermined model in practice. In this paper, we adopt an underdetermined model, which is robust to noise and other interference of the data. An underdetermined model also has other desirable features such as greater flexibility to reconstruct the input data. With an underdetermined model, we do not get an exact representation of $\boldsymbol{Y}$ by $\boldsymbol{B}_1, \ldots, \boldsymbol{B}_n$ and $W_1, \ldots, W_n$. Instead, we get an approximate representation of $\boldsymbol{Y}$ with an error term $\boldsymbol{E} = [E_1, \ldots, E_k]^{\mathrm{T}}$. In other words, as an underdetermined model, (1) becomes

$$\boldsymbol{Y} = \boldsymbol{E} + \sum_{i=1}^{n} \boldsymbol{B}_i W_i, \tag{2}$$

where the number of bases n is larger than the number of elements k in $\boldsymbol{Y}$.

Furthermore, to prevent overfitting, a sparse model that uses only a small number of bases to reconstruct the input data is preferred over a complex model that uses a majority of bases to reconstruct the input data. This is because although an overfitted complex model is usually better to represent the training data (with a smaller error), it is vulnerable to small fluctuations of future input data, and hence has poor reconstruction performance [7]. In contrast, a sparse model captures only the essential features of the data and is not affected by small fluctuations of the input, which are typically caused by features of minor importance. Therefore, a *sparsity requirement* is imposed to the model so that only a small number of bases are used to represent $\boldsymbol{Y}$. Sparse modeling has been widely used in the literature to solve a wide range of problems, due to its good performance of data/signal reconstruction and preventing overfitting, see [7]–[11].

For the ease of description, sometimes we use a matrix $\boldsymbol{B} = [\boldsymbol{B}_1, \ldots, \boldsymbol{B}_n]$ to denote the bases and a column vector $\boldsymbol{W} = [W_1, \ldots, W_n]^{\mathrm{T}}$ to denote the weight. To avoid confusion, we briefly describe the style of notations in this paper. We use a bold symbol to denote a matrix, e.g., $\boldsymbol{B}$, or a column vector, e.g., $\boldsymbol{W}$. For a matrix $\boldsymbol{B}$, we use $\boldsymbol{B}_j$ to represent the jth column vector, and B_{ij} to represent the element on row i and column j of $\boldsymbol{B}$. For a vector $\boldsymbol{W}$, we use W_j to denote the jth element. We list the notations for this section in Table 1. With this notation style, we rewrite (2) to

TABLE 1
Notations for Section 4

$\boldsymbol{Y}, Y_i$	An arbitrary SAP and its element
$\boldsymbol{B}, \boldsymbol{B}_i$	Bases matrix, and its column vector
$\boldsymbol{W}, W_i$	Weights to reconstruct $\boldsymbol{Y}$
$\boldsymbol{E}, E_i$	Error of reconstructing $\boldsymbol{Y}$ and its element Y_i
σ	Variance of error E_i, an element of $\boldsymbol{E}$
$\boldsymbol{B}^*$	Optimal bases
k	Number of elements in $\boldsymbol{Y}$
n	Number of bases in $\boldsymbol{B}$
$p(\boldsymbol{Y})$	Probability distribution of $\boldsymbol{Y}$
$p(\boldsymbol{Y} \mid \boldsymbol{B})$	Probability distribution of $\boldsymbol{Y}$ represented by bases $\boldsymbol{B}$
$p(\boldsymbol{W})$	Prior probability distribution of $\boldsymbol{W}$
$p(W_i)$	Prior probability distribution of W_i
$p(\boldsymbol{Y} \mid \boldsymbol{W}, \boldsymbol{B})$	Probability distribution that SAP $\boldsymbol{Y}$ occurs for a given $\boldsymbol{W}$ and bases $\boldsymbol{B}$
$\|\boldsymbol{Y}\|_2$	ℓ^2 norm of vector $\boldsymbol{Y}$
$\|\boldsymbol{B}\|_F$	Frobenius norm of matrix $\boldsymbol{B}$
c	Constraint for the Frobenius norm of $\boldsymbol{B}$
λ	A constant combining the parameters of the probability distributions of W_i and E_i

$$\boldsymbol{Y} = \boldsymbol{E} + \boldsymbol{B}\boldsymbol{W}. \tag{3}$$

The objective of the model in (3) is to reconstruct an arbitrary SAP $\boldsymbol{Y}$, using a fixed set of bases $\boldsymbol{B}$ and a variable weight $\boldsymbol{W}$. SAP $\boldsymbol{Y}$ can be viewed as a random variable. Similarly, $\boldsymbol{W}$ is also a random variable. The bases $\boldsymbol{B}$ are currently unknown and our objective is to derive $\boldsymbol{B}$ for the model, to minimize the reconstruction error. Let $p(\boldsymbol{Y})$ denote the probability distribution of $\boldsymbol{Y}$. In order to represent $\boldsymbol{Y}$ using (3), the probability distribution of $\boldsymbol{B}\boldsymbol{W}$ needs to match $p(\boldsymbol{Y})$ well, so that the error term $\boldsymbol{E}$ is minimized. The term $\boldsymbol{B}\boldsymbol{W}$ is the representation of $\boldsymbol{Y}$ by bases $\boldsymbol{B}$, and hence its probability distribution is denoted as $p(\boldsymbol{Y} \mid \boldsymbol{B})$. A good matching of $p(\boldsymbol{Y})$ and $p(\boldsymbol{Y} \mid \boldsymbol{B})$ would result in a set of bases $\boldsymbol{B}$ that capture the essential features of $\boldsymbol{Y}$. Given the prior probability distribution of weight $\boldsymbol{W}$, denoted as $p(\boldsymbol{W})$, and the probability distribution that a SAP $\boldsymbol{Y}$ occurs for a given weight $\boldsymbol{W}$ (and bases $\boldsymbol{B}$), denoted as $p(\boldsymbol{Y} \mid \boldsymbol{W}, \boldsymbol{B})$, the probability distribution $p(\boldsymbol{Y} \mid \boldsymbol{B})$ can be obtained as follows.

$$p(\boldsymbol{Y} \mid \boldsymbol{B}) = \int p(\boldsymbol{W}) p(\boldsymbol{Y} \mid \boldsymbol{W}, \boldsymbol{B}) d\boldsymbol{W}. \tag{4}$$

In a Bayesian context, the probability distribution of $\boldsymbol{Y}$ is our belief of the uncertainty of $\boldsymbol{Y}$. Given weight $\boldsymbol{W}$ and bases $\boldsymbol{B}$, the uncertainty of $\boldsymbol{Y}$ is reduced to the uncertainty of the error $\boldsymbol{E}$. As a general practice, the element E_i of the error term $\boldsymbol{E}$ can be assumed to follow a Gaussian distribution with 0 mean and variance σ^2. Therefore, $p(\boldsymbol{Y} \mid \boldsymbol{W}, \boldsymbol{B})$, the

probability distribution of SAP $\boldsymbol{Y}$ given weight $\boldsymbol{W}$ and bases $\boldsymbol{B}$, is as follows

$$p(\boldsymbol{Y} \mid \boldsymbol{W}, \boldsymbol{B}) = \frac{1}{(2\pi)^{\frac{k}{2}}\sigma^k} e^{-\frac{\|\boldsymbol{Y}-\boldsymbol{BW}\|_2^2}{2\sigma^2}}, \tag{5}$$

where k is the number of elements of vector $\boldsymbol{Y}$, and $\|\bullet\|_2$ is the ℓ^2 norm, with $\|\boldsymbol{Y}\|_2 = \sqrt{Y_1^2 + \cdots + Y_k^2}$.

When representing SAP $\boldsymbol{Y}$ in (3), it is desirable to make the weighted terms $\boldsymbol{B}_1 W_1, \ldots, \boldsymbol{B}_n W_n$ statistically independent. Furthermore, as discussed earlier, the model should have a sparse structure, i.e., only a small number of $\boldsymbol{B}_1, \ldots, \boldsymbol{B}_n$ are needed to reconstruct $\boldsymbol{Y}$, or equivalently, only a small number of elements of $\boldsymbol{W}$ are non-zeros. These two requirements are critical to build a good reconstruction model, and can be imposed through applying a suitable prior probability distribution of weight $\boldsymbol{W}$, $p(\boldsymbol{W})$, to the model. The requirement of statistical independence can be imposed by selecting a prior probability distribution of $\boldsymbol{W}$ such that it is a product of the probability distributions of the elements $W_1, \ldots, W_n$. That is, $p(\boldsymbol{W})$ is given as

$$p(\boldsymbol{W}) = \prod_{i=1}^{n} p(W_i), \tag{6}$$

where $p(W_i)$ denotes the probability distribution of W_i, the ith element of $\boldsymbol{W}$.

The requirement of sparse representation of $\boldsymbol{Y}$ can be achieved by assigning each element of $\boldsymbol{W}$ a prior probability distribution that is peaked around 0. Then the random variable W_i ($1 \le i \le n$) has a large probability to be 0 (or close to 0), and hence the average number of non-zero elements among $W_1, \ldots, W_n$ is small, which meets the sparsity requirement. In this paper, we select the Laplace distribution with 0 mean for W_i. Hence $p(W_i)$ is given as

$$p(W_i) = \frac{1}{2b} e^{-\frac{|W_i|}{b}}, \tag{7}$$

where b is the shape parameter, and $|W_i|$ is the absolute value of W_i. With the Laplace distribution, $p(W_i)$ is peaked around 0. The parameter b is to control the concentration of W_i around the mean 0. The smaller the b, the more concentrated the W_i is, and hence the more sparse the model is.

Combining (4)–(7), we obtain $p(\boldsymbol{Y} \mid \boldsymbol{B})$, the probability distribution of $\boldsymbol{Y}$ represented as the bases. To measure how close this probability distribution is to the real probability distribution of $\boldsymbol{Y}$, $p(\boldsymbol{Y})$, we can use a statistical divergence function. We adopt the Kullback–Leibler divergence function to measure the difference of $p(\boldsymbol{Y})$ from $p(\boldsymbol{Y} \mid \boldsymbol{B})$, denoted as D, which is given as

$$\begin{aligned} D &= \int p(\boldsymbol{Y}) \log \frac{p(\boldsymbol{Y})}{p(\boldsymbol{Y} \mid \boldsymbol{B})} d\boldsymbol{Y} \\ &= \int p(\boldsymbol{Y}) \left(\log p(\boldsymbol{Y}) - \log p(\boldsymbol{Y} \mid \boldsymbol{B}) \right) d\boldsymbol{Y}. \end{aligned} \tag{8}$$

A smaller value of D means a smaller difference of $p(\boldsymbol{Y})$ from $p(\boldsymbol{Y} \mid \boldsymbol{B})$. If $D = 0$, then $p(\boldsymbol{Y}) = p(\boldsymbol{Y} \mid \boldsymbol{B})$.

To accurately reconstruct $\boldsymbol{Y}$ using the bases $\boldsymbol{B}$ by (3), we need to select $\boldsymbol{B}$ to minimize D. Since $p(\boldsymbol{Y})$ is not affected by the selection of $\boldsymbol{B}$, the objective is to maximize $p(\boldsymbol{Y} \mid \boldsymbol{B})$ in (8). From (4), the optimal $\boldsymbol{B}$, denoted as $\boldsymbol{B}^*$, is obtained as

$$\begin{aligned} \boldsymbol{B}^* &= \arg\max_{\boldsymbol{B}} p(\boldsymbol{Y} \mid \boldsymbol{B}) \\ &= \arg\max_{\boldsymbol{B}} \int p(\boldsymbol{W}) p(\boldsymbol{Y} \mid \boldsymbol{W}, \boldsymbol{B}) d\boldsymbol{W}. \end{aligned} \tag{9}$$

Since we have selected the Laplace distribution for W_i, then $p(\boldsymbol{W})$ is peaked around a point in the $\mathbb{R}^n$ space. On the other hand, $p(\boldsymbol{Y} \mid \boldsymbol{W}, \boldsymbol{B})$ is a Gaussian distribution, which is also peaked around a certain point in $\mathbb{R}^k$ space. Therefore, $p(\boldsymbol{W})p(\boldsymbol{Y} \mid \boldsymbol{W}, \boldsymbol{B})$ is expected to peak around a point in the $\boldsymbol{W}$ space. With this property, we can approximately maximize the integral $\int p(\boldsymbol{W})p(\boldsymbol{Y} \mid \boldsymbol{W}, \boldsymbol{B})d\boldsymbol{W}$ by maximizing the product $p(\boldsymbol{W})p(\boldsymbol{Y} \mid \boldsymbol{W}, \boldsymbol{B})$. Therefore, (9) is approximately equivalent to the following optimization.

$$\boldsymbol{B}^* \approx \arg\max_{\boldsymbol{B}} \left(\max_{\boldsymbol{W}} p(\boldsymbol{W}) p(\boldsymbol{Y} \mid \boldsymbol{W}, \boldsymbol{B}) \right). \tag{10}$$

For the optimization in (10), we need to add a constraint for the norm of $\boldsymbol{B}$, to prevent $\boldsymbol{W} = [0, \ldots, 0]_n$. This is because if there exist a set of bases $\boldsymbol{C}$ and a weight $\boldsymbol{U} = [U_1, \ldots, U_n]^{\mathrm{T}}$ as the solution for problem (10), then there always exist another weight $\boldsymbol{U}' = \frac{\boldsymbol{U}}{s}$ and bases $\boldsymbol{C}' = s\boldsymbol{C}$, where $s \gg \max\{|U_1|, \ldots, |U_n|\}$, as the solution since $\boldsymbol{C}'\boldsymbol{U}' = \boldsymbol{C}\boldsymbol{U}$. However, the latter solution has a weight $\boldsymbol{U}' = [0, \ldots, 0]_n$, which would make the SAP reconstruction model useless. Therefore, to prevent this situation, we need to add a constraint c for the Frobenius norm of $\boldsymbol{B}$ as follows

$$\|\boldsymbol{B}\|_F^2 = \sum_{i=1}^{n} \|\boldsymbol{B}_i\|_2^2 = \sum_{i=1}^{n} \sum_{j=1}^{k} B_{ji}^2 \le c. \tag{11}$$

To maximize $p(\boldsymbol{W})p(\boldsymbol{Y} \mid \boldsymbol{W}, \boldsymbol{B})$ in (10), we note that

$$\begin{aligned} & p(\boldsymbol{W})p(\boldsymbol{Y} \mid \boldsymbol{W}, \boldsymbol{B}) \\ &= \left[\frac{1}{(2\pi)^{\frac{k}{2}}\sigma^k} e^{-\frac{\|\boldsymbol{Y}-\boldsymbol{BW}\|_2^2}{2\sigma^2}} \right] \left[\prod_{i=1}^{n} \frac{1}{2b} e^{-\frac{|W_i|}{b}} \right] \\ &= \varrho e^{-\frac{\|\boldsymbol{Y}-\boldsymbol{BW}\|_2^2}{2\sigma^2} - \sum_{i=1}^{n} \frac{|W_i|}{b}} \\ &= \varrho e^{-\frac{1}{2\sigma^2}\left(\|\boldsymbol{Y}-\boldsymbol{BW}\|_2^2 + \frac{2\sigma^2}{b} \sum_{i=1}^{n} |W_i| \right)} \\ &= \varrho e^{-\frac{1}{2\sigma^2}\left(\|\boldsymbol{Y}-\boldsymbol{BW}\|_2^2 + \lambda \sum_{i=1}^{n} |W_i| \right)}, \end{aligned} \tag{12}$$

where

$$\varrho = \frac{1}{(2\pi)^{\frac{k}{2}}\sigma^k (2b)^n}, \text{ and } \lambda = \frac{2\sigma^2}{b}.$$

From (12), we see that to maximize $p(\boldsymbol{W})p(\boldsymbol{Y} \mid \boldsymbol{W}, \boldsymbol{B})$, it is equivalent to minimize the exponent $\|\boldsymbol{Y} - \boldsymbol{BW}\|_2^2 + \lambda \sum_{i=1}^{n} |W_i|$ since ϱ is a constant. Therefore, optimizing the bases in (10) is transformed to

$$\boldsymbol{B}^* = \arg\min_{\boldsymbol{B}} \left(\min_{\boldsymbol{W}} \|\boldsymbol{Y} - \boldsymbol{BW}\|_2^2 + \lambda \sum_{i=1}^{n} |W_i| \right),$$

subject to constraint (11). Finally, the reconstruction of SAP $\boldsymbol{Y}$ in (3) is transformed into solving the following optimization problem.

$$\min_{\boldsymbol{B},\boldsymbol{W}} \|\boldsymbol{Y}-\boldsymbol{B}\boldsymbol{W}\|_2^2+\lambda\sum_{i=1}^{n}|W_i| \tag{13}$$

$$\text{subject to } \|\boldsymbol{B}\|_F^2 \le c. \tag{14}$$

In the literature, an optimization problem with the form in (13), i.e., a least square term and an ℓ^1 norm regularization term, is called a sparse coding problem [9], [13], [14], as the ℓ^1 norm term $\lambda\|\boldsymbol{W}\|_1=\lambda\sum_{i=1}^{n}|W_i|$ has an effect to enforce a sparse structure of $\boldsymbol{W}$, i.e., a majority of elements of $\boldsymbol{W}$ are zeros. Problem (13) can be used to obtain an optimal set of bases $\boldsymbol{B}^*$ as well as reconstruct a SAP given the bases. To obtain $\boldsymbol{B}^*$, we will need to feed a set of training SAPs. We discuss this problem in the next section. To reconstruct a SAP $\boldsymbol{Y}$ given a set of bases $\boldsymbol{B}^*$, we remove the constraint (14) and let $\boldsymbol{B}=\boldsymbol{B}^*$ in (13) to minimize over $\boldsymbol{W}$ only. We discuss it in Section 6.

5 SPARS Training

To train SPARS to obtain an optimal set of bases $\boldsymbol{B}^*$, we need a set of training data, i.e., a set of SAPs. We can collect the training data in the initial setup phase of the CRN. At the beginning of CRN setup, each SU passively performs spectrum sensing on each channel to collect the SAP information (ON and OFF periods) of every PU signal transmitter. In this initial sensing phase, the transmitters of PU signal are expected to be (genuine) PUs. This is because in this phase, the selfish SUs would not launch PUE attack, since data transmission is not started yet, and they do not get any benefit for occupying a channel. Furthermore, a malicious attacker would not be aware of a CRN being set up in the field, since the SUs are not transmitting. (We assume that there is no internal malicious SUs inside the CRN in the initial setup phase.) Hence a malicious attacker would not launch a PUE attack either, since it thinks that there is no CRN yet, and launching a PUE attack simply wastes its energy. In the case that a malicious attacker does not care if a CRN is being set up and sends signals on some arbitrarily selected channels anyhow, a CRN would usually avoid such channels and selects the channels that are lightly utilized as the operation channels. Hence, the PUE attack in the initial CRN setup phase is relieved by the selection of operation channels. Therefore, we assume that during the initial sensing phase of a CRN setup, there is no PUE attack, so that the training data set consists of SAPs of PUs. After the CRN is set up, the training data set can be continuously updated by randomly incorporating the SAPs which are recognized as from PUs by SPARS. The bases can be updated accordingly to accommodate the dynamic changes of SAP features.

We list the notations for this and next sections in Table 2. Let m denote the number of SAPs in the training data set, and let $\boldsymbol{X}_1,\ldots,\boldsymbol{X}_m$ denote these SAPs, where $\boldsymbol{X}_i\in\mathbb{R}^k$. Let the column vector $\boldsymbol{S}_i\in\mathbb{R}^n$ $(1\le i\le m)$ denote a weight to be used to reconstruct training data $\boldsymbol{X}_i$. We let matrix $\boldsymbol{X}=[\boldsymbol{X}_1,\ldots,\boldsymbol{X}_m]$ denote the training data set, and let $\boldsymbol{S}=[\boldsymbol{S}_1,\ldots,\boldsymbol{S}_m]$ denote the weight matrix. Given the training data set $\boldsymbol{X}$, the learning of bases is a process to find $\boldsymbol{B}$ and $\boldsymbol{S}$ such that

$$\boldsymbol{X}\approx\boldsymbol{B}\boldsymbol{S}. \tag{15}$$

TABLE 2
Notations for Sections 5 and 6

$\boldsymbol{X},\boldsymbol{X}_i,X_{ji}$	Training data set (matrix, column vectors, and elements)
$\boldsymbol{B},\boldsymbol{B}_i,B_{ji}$	Bases
$\boldsymbol{S},\boldsymbol{S}_i,S_{ji}$	Weights to reconstruct $\boldsymbol{X}$ using $\boldsymbol{B}$
$\|\boldsymbol{S}_i\|_1$	ℓ^1 norm of vector $\boldsymbol{S}_i$
$\boldsymbol{B}^*,\boldsymbol{B}_i^*,B_{ji}^*$	Optimal bases obtained after system training
m	Number of SAPs in the training data set $\boldsymbol{X}$
k	Number of ON or OFF periods in each SAP
n	Number of bases in $\boldsymbol{B}$
η_i	SSE of reconstructing $\boldsymbol{X}_i$
$\boldsymbol{Y},Y_i$	An observed SAP and its elements
$\boldsymbol{W},W_i$	Weights to reconstruct $\boldsymbol{Y}$ using $\boldsymbol{B}^*$
ζ	SSE of reconstructing $\boldsymbol{Y}$
$\tilde{\gamma}$	Tolerance interval upper limit
$\tilde{\mu},\tilde{\sigma}^2$	SSE actual mean and variance
$\hat{\mu},\hat{\sigma}^2$	SSE sample mean and variance
α,β	Parameters for the (α,β) tolerance interval

Based on the analysis in the preceding section, Problem (15) can be transformed into the following problem[1].

$$\min_{\boldsymbol{B},\boldsymbol{S}} \|\boldsymbol{X}-\boldsymbol{B}\boldsymbol{S}\|_F^2+\lambda\sum_{i=1}^{m}\|\boldsymbol{S}_i\|_1 \tag{16}$$

$$\text{subject to } \|\boldsymbol{B}\|_F^2 \le c, \tag{17}$$

where $\|\bullet\|_F$ is the Frobenius norm of a matrix, and $\|\boldsymbol{S}_i\|_1=\sum_{j=1}^{n}S_{ji}$ is the ℓ^1 norm of vector $\boldsymbol{S}_i$.

Problem (16) is an optimization problem over $\boldsymbol{B}$ and $\boldsymbol{S}$. It is a convex optimization problem if one of the variables ($\boldsymbol{B}$ or $\boldsymbol{S}$) is fixed. Therefore, (16) can be solved by iteratively optimizing the objective over $\boldsymbol{B}$ or $\boldsymbol{S}$ while fixing the other. There are efficient algorithms to solve the sparse coding problem. In this paper, we use the feature-sign and Lagrange dual algorithms in [14] to solve (16). After it is solved, we obtain a set of optimal bases, denoted as $\boldsymbol{B}^*$.

Next, we discuss the error when using the learned bases $\boldsymbol{B}^*$ to reconstruct the training data $\boldsymbol{X}_i$. Let $\tilde{E}_{vi}$ denote the error to reconstruct the vth element of $\boldsymbol{X}_i$. $\tilde{E}_{vi}$ is given as

$$\tilde{E}_{vi}=\left(X_{vi}-\sum\nolimits_{j=1}^{n}B_{vj}^*S_{ji}\right). \tag{18}$$

Let η_i denote the *sum of square of errors* (SSE) to reconstruct $\boldsymbol{X}_i$. Then η_i is given as

$$\eta_i=\left\|\boldsymbol{X}_i-\boldsymbol{B}^*\boldsymbol{S}_i\right\|_2^2=\sum_{v=1}^{k}\tilde{E}_{vi}^2. \tag{19}$$

The reconstruction error $\tilde{E}_{vi}$ can be seen as a random variable. Therefore, if k is reasonably large, by the central limit theorem, η_i is an approximate Gaussian random variable with some unknown mean $\tilde{\mu}$ and variance $\tilde{\sigma}^2$. Algorithm 1 describes the system training, with the output as the obtained optimal bases $\boldsymbol{B}^*$ and the SSE η_i $(1\le i\le m)$.

1. Note that $\boldsymbol{B}$ does not necessarily satisfy the RIP condition. Nevertheless, [8] shows that for most matrices, the sparse solution can be found.

Algorithm 1 SPARS training

Input: Training data set $\boldsymbol{X}$, number of bases n, parameters λ, c

Output: Optimal bases $\boldsymbol{B}^*$, SSEs $\eta_1, \ldots, \eta_m$

1. Solve problem (16) with constraint (17) to get $\boldsymbol{B}^*$ and $\boldsymbol{S}$ using the feature-sign and Lagrange dual algorithms in [14].
2. **for** $1 \le i \le m$ **do**
3. Compute SSE η_i by (18) and (19).
4. **end for**

6 SAP Reconstruction and PUE Detection

Given the bases $\boldsymbol{B}^*$ obtained from the system training in last section, we can use them to reconstruct an observed SAP $\boldsymbol{Y} = [Y_1, \ldots, Y_v, \ldots, Y_k]^{\mathrm{T}}$ using the model in (3). It is transformed into the following problem, which now becomes a least square problem with an ℓ^1 norm regularization to enforce a sparse structure.

$$\min_{\boldsymbol{W}} \left\| \boldsymbol{Y} - \boldsymbol{B}^* \boldsymbol{W} \right\|_2^2 + \lambda \left\| \boldsymbol{W} \right\|_1. \tag{20}$$

Let $E_v = \left(Y_v - \sum_{j=1}^{n} B^*_{vj} W_j \right)$ denote the error to reconstruct the vth element of $\boldsymbol{Y}$. Let ζ denote the SSE to reconstruct $\boldsymbol{Y}$ using the bases $\boldsymbol{B}^*$. Then we have

$$\zeta = \left\| \boldsymbol{Y} - \boldsymbol{B}^* \boldsymbol{W} \right\|_2^2 = \sum_{v=1}^{k} E_v^2. \tag{21}$$

As discussed in last section, ζ is an approximate Gaussian random variable with some unknown mean $\tilde{\mu}$ and variance $\tilde{\sigma}^2$.

We use the SSE ζ to measure the normality of SAP $\boldsymbol{Y}$. Specifically, let γ denote the α quantile of the Gaussian distribution with mean $\tilde{\mu}$ and variance $\tilde{\sigma}^2$, i.e., $\gamma = \tilde{\mu} + \tilde{\sigma} z_\alpha$, where z_α is the α quantile of the standard Gaussian distribution, i.e., $\Pr(Z \le z_\alpha) = \alpha$ where Z is a standard Gaussian random variable. Then if ζ is larger than γ, the observed SAP $\boldsymbol{Y}$ is treated as an abnormal SAP, i.e., a PUE attack. Otherwise, $\boldsymbol{Y}$ is seen as a normal SAP. Unfortunately, the SSE mean $\tilde{\mu}$ and variance $\tilde{\sigma}^2$ are unknown, and hence we cannot obtain γ.

We introduce a technique called *tolerance interval* [16, Ch2] to test normality of SAP $\boldsymbol{Y}$. The tolerance interval is similar to the quantile, but has a benefit that it can be computed from the sample mean and variance, without needing the actual mean and variance. We can obtain the SSE sample mean and variance from $\eta_1, \ldots, \eta_m$, which have been obtained in last section for reconstructing the training data set $\boldsymbol{X}$, since each training data $\boldsymbol{X}_i$ is a random SAP sample.

Based on $\eta_1, \ldots, \eta_m$, we can compute the SSE sample mean and variance as

$$\hat{\mu} = \frac{1}{m} \sum_{i=1}^{m} \eta_i, \qquad \hat{\sigma}^2 = \frac{1}{m-1} \sum_{i=1}^{m} \left(\eta_i - \hat{\mu} \right)^2. \tag{22}$$

We expect that an abnormal SAP has a large reconstruction error. Hence we use a one-sided tolerance interval in the form of $[-\infty, \tilde{\gamma}]$. The upper tolerance limit $\tilde{\gamma}$ is given as

$$\tilde{\gamma} = \hat{\mu} + \theta \hat{\sigma}, \tag{23}$$

where θ is a parameter that we discuss next. (Similarly, the lower tolerance limit is given as $\tilde{\gamma}' = \hat{\mu} - \theta \hat{\sigma}$.)

An (α, β) one-sided tolerance interval $[-\infty, \hat{\mu} + \theta\hat{\sigma}]$ means that a proportion β of the entire SSE population (i.e., $100\beta\%$ population) falls in the interval $[-\infty, \hat{\mu} + \theta\hat{\sigma}]$ with $100\alpha\%$ confidence. In other words, we have

$$\Pr\left\{ \Pr(\zeta \le \hat{\mu} + \theta\hat{\sigma} \mid \hat{\mu}, \hat{\sigma}) \ge \beta \right\} = \alpha. \tag{24}$$

Given $\alpha, \beta, \hat{\mu}, \hat{\sigma}$, (24) can be solved to obtain θ as follows [16, Ch2].

$$\theta = \frac{1}{\sqrt{m}} t_{m-1,\alpha}(z_\beta \sqrt{m}),$$

where m is the sample size when computing $\hat{\mu}$ and $\hat{\sigma}$, z_β is the β quantile of the standard Gaussian distribution, and $t_{m-1,\alpha}(z_\beta\sqrt{m})$ denotes the α quantile of the noncentral Student's t distribution with $m-1$ degree of freedom and the noncentrality parameter $z_\beta\sqrt{m}$. While $t_{m-1,\alpha}(z_\beta\sqrt{m})$ may be computed from the inverse of the cumulative probability distribution function of the noncentral Student's t distribution, a convenient approximation for computing θ is given as [17]

$$\theta \approx \frac{z_\beta + \sqrt{z_\beta^2 - \tau\omega}}{\tau}, \qquad \tau = 1 - \frac{z_\alpha^2}{2(m-1)}, \quad \omega = z_\beta^2 - \frac{z_\alpha^2}{m}, \tag{25}$$

where z_β and z_α are the β and α quantiles of the standard Gaussian distribution, respectively. The formula in (25) is easier for computation since the quantile for the standard Gaussian distribution is widely available.

Given the tolerance interval $[-\infty, \tilde{\gamma}]$, we test SAP $\boldsymbol{Y}$ by examining the SSE ζ in (21). If ζ falls in the tolerance interval, i.e., $\zeta \le \tilde{\gamma}$, then $\boldsymbol{Y}$ is a normal SAP. Otherwise, $\boldsymbol{Y}$ is an abnormal SAP and the corresponding signal transmitter is alarmed as an attacker. Algorithm 2 describes how an SU carries out the PUE detection using SPARS that output an alarm signal for a SAP $\boldsymbol{Y}$ collected from a candidate channel, to indicate that the corresponding transmitter is an attacker or a PU.

After identifying an attacker, there may be several options for subsequent actions, such as reporting to the corresponding authority, or reporting to the nearby nodes of the attacker to penalize the latter, e.g., not forwarding the traffic of the attacker (effective for selfish SU attacker) or even ignoring the PU signal. We do not get into details of such actions, as this is out of the scope of this paper.

7 Bound Analysis of SPARS

We develop the Chernoff bound for the SPARS system. In the literature, the Chernoff bound is typically given for only Bernoulli random variables that have values of 0 or 1. However, the Chernoff bound can be derived for a general distribution from the Markov inequality. If X is a nonnegative random variable and $a > 0$ is a constant, then the Markov inequality is

$$\Pr(X > a) \le \frac{\mathbb{E}(X)}{a}. \tag{26}$$

Algorithm 2 PUE detection by SPARS at an SU

Input: Parameters $\lambda, c, \alpha, \beta, m, n$

1. **if** CRN is in the initial setup phase **then**
2. Passively carry out spectrum sensing to collect m SAPs, $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_m$, on candidate channels, which are used as the training data set $\boldsymbol{X}$.
3. Apply Algorithm 1 on $\boldsymbol{X}$ with parameters λ, c, and n to obtain the bases $\boldsymbol{B}^* = [\boldsymbol{B}_1^*, \ldots, \boldsymbol{B}_n^*]$ and SSEs $\eta_1, \ldots, \eta_m$.
4. **end if**
5. **loop**
6. For a SAP $\boldsymbol{Y}$ collected from a candidate channel, solve problem (20) to get $\boldsymbol{W}$.
7. Compute SSE ζ by (21).
8. Compute $\hat{\mu}$ and $\hat{\sigma}$ by (22) using η_i from Algorithm 1.
9. Compute θ by (25).
10. Compute $\tilde{\gamma}$ by (23).
11. **if** $\zeta > \tilde{\gamma}$ **then**
12. SAP $\boldsymbol{Y}$ is from an attacker. *Alarm* = YES.
13. **else**
14. SAP $\boldsymbol{Y}$ is from a PU. *Alarm* = NO.
15. **end if**
16. **end loop**

If Y is an arbitrary random variable that can also have negative values, the Markov inequality can be still applied to obtain the tail probability by letting $X = e^{sY}$ where $s > 0$ is some constant. Since e^{sY} is a nonnegative random variable, we can apply the Markov inequality on it. Furthermore, since e^{sY} is an increasing function, we have

$$\Pr(Y > a) = \Pr(e^{sY} > e^{sa}) \leq \frac{\mathbb{E}(e^{sY})}{e^{sa}} = e^{g(s)-sa}, \qquad (27)$$

where $g(s) = \log \mathbb{E}(e^{sY})$ is the cumulant generating function of the random variable e^{sY}. (27) is called the Chernoff bound when s is selected such that $g(s) - sa$ is minimized and $s > 0$.

Next we derive the Chernoff bound for the SSE ζ in (21). Recalling from the preceding section that ζ follows the Gaussian distribution with mean $\tilde{\mu}$ and variance $\tilde{\sigma}^2$. The cumulant generating function of the Gaussian distribution is

$$g(s) = \frac{\tilde{\sigma}^2}{2}s^2 + s\tilde{\mu}.$$

To minimize $g(s) - sa$, we let

$$\frac{d(g(s) - sa)}{ds} = \tilde{\sigma}^2 s + \tilde{\mu} - a = 0.$$

$$\Longrightarrow s = \frac{a - \tilde{\mu}}{\tilde{\sigma}^2} \text{ for } a > \tilde{\mu}.$$

Hence

$$\min_s(g(s) - sa) = -\frac{(a - \tilde{\mu})^2}{2\tilde{\sigma}^2}.$$

Thus, from (27), the Chernoff bound for ζ is

$$\Pr(\zeta > a) \leq e^{-\frac{(a-\tilde{\mu})^2}{2\tilde{\sigma}^2}} \text{ for } a > \tilde{\mu}. \qquad (28)$$

In practice, the mean $\tilde{\mu}$ and the variance $\tilde{\sigma}^2$ are unknown. In this case, we can use the SSE sample mean and variance, $\hat{\mu}$ and $\hat{\sigma}^2$, to replace $\tilde{\mu}$ and $\tilde{\sigma}^2$ in (28) to get the approximate Chernoff bound.

8 PUE Detection by Statistics

In this section, we present another PUE detection approach, termed *PUE detection by statistics* (PDS), which also utilizes the signal activity pattern of PU signals. PDS examines two most important statistics, the mean and variance, of the signal ON/OFF periods in a SAP to see if they are abnormal. It works as follows. In the initial setup phase of a CRN, similar to SPARS, each SU passively carries out spectrum sensing to collect SAPs, denoted as $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_m$ and put them into a training data set $\boldsymbol{X}$. Next, it compute the mean and variance of the ON/OFF periods in each SAP $\boldsymbol{X}_i$, denoted as μ_i and v_i. Therefore, from the training data set, PDS obtains a vector of means $[\mu_1, \ldots, \mu_m]$ and a vector of variances $[v_1, \ldots, v_m]$. For the ease of description, let $M(\boldsymbol{Y})$ or $M(Y_1, \ldots, Y_k)$ represent the function to compute the mean of a vector $\boldsymbol{Y} = [Y_1, \ldots, Y_k]$, and let $V(\boldsymbol{Y})$ or $V(Y_1, \ldots, Y_k)$ represent the function to compute the variance of vector $\boldsymbol{Y}$. They are given as

$$M(\boldsymbol{Y}) = M(Y_1, \ldots, Y_k) = \frac{1}{k}\sum_{i=1}^{k} Y_i,$$

$$V(\boldsymbol{Y}) = V(Y_1, \ldots, Y_k) = \frac{1}{k-1}\sum_{i=1}^{k} (Y_i - M(\boldsymbol{Y}))^2.$$

By the central limit theorem, the mean and variance of the ON/OFF periods in a SAP $\boldsymbol{Y}$, denoted as $\mu_{\boldsymbol{Y}}$ and $\sigma_{\boldsymbol{Y}}^2$, approximately follow the Gaussian distribution, as long as the number of ON/OFF periods in SAP $\boldsymbol{Y}$ is reasonably large. Therefore, we can use (23) and (25) to compute the tolerance limits for $\mu_{\boldsymbol{Y}}$ and $\sigma_{\boldsymbol{Y}}^2$, respectively, based on the sample mean and variance of vectors $[\mu_1, \ldots, \mu_m]$ and $[v_1, \ldots, v_m]$, respectively. With the tolerance limits, we can test if SAP $\boldsymbol{Y}$ is normal (from a genuine PU) or abnormal (from an attacker) by checking the mean and variance of the ON/OFF periods against the tolerance limits. Algorithm 3 formally describes PDS.

9 Performance Evaluation

In this section, we evaluate the performance of SPARS through simulations. We use $n = 128$ bases. The constraints λ and c are both assumed as 1. The SAP size k is assumed as 30. We assume 30 PU signal transmitters randomly spread on 20 channels, with minimum one and maximum 3 transmitters on each channel. To focus on evaluation of the PUE detection, we do not consider cooperative spectrum sensing and assume that each SU independently performs spectrum sensing to acquire SAPs. Hence without loss of generality, we study the PUE detection of one SU only. We use two distributions, exponential and Pareto distributions, to simulate the ON and OFF periods of each transmitter. As discussed earlier, we assume that in the initial sensing phase of the CRN setup, each SU passively performs spectrum sensing and acquires the ON and OFF periods from each transmitter into a training data set. (The training data set may also be collected offline and injected into each SU before deployment.) We use a training data set with 200 training SAPs.

Algorithm 3 PUE detection by statistics (PDS) at an SU

1. **if** CRN is in the initial setup phase **then**
2. Passively carry out spectrum sensing to collect SAPs, denoted as $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_m$.
3. **for** $1 \leq i \leq m$ **do**
4. Calculate the mean $\mu_i = M(\boldsymbol{X}_i)$ and variance $v_i = V(\boldsymbol{X}_i)$ for SAP $\boldsymbol{X}_i$.
5. **end for**
6. Compute the sample mean and variance of vector $[\mu_1, \ldots, \mu_m]$ as $\hat{\mu}_{\text{mean}} = M(\mu_1, \ldots, \mu_m)$ and $\hat{\sigma}^2_{\text{mean}} = V(\mu_1, \ldots, \mu_m)$
7. Compute the tolerance limit of the ON periods mean $\tilde{\gamma}_{\text{mean}}$ using $\hat{\mu}_{\text{mean}}$ and $\hat{\sigma}_{\text{mean}}$ by (23) and (25). (For the OFF periods mean, compute the lower tolerance limit $\tilde{\gamma}'_{\text{mean}}$.)
8. Compute the sample mean and variance of vector $[v_1, \ldots, v_m]$ as $\hat{\mu}_{\text{var}} = M(v_1, \ldots, v_m)$ and $\hat{\sigma}^2_{\text{var}} = V(v_1, \ldots, v_m)$
9. Compute the tolerance limit of the variance $\tilde{\gamma}_{\text{var}}$ using $\hat{\mu}_{\text{var}}$ and $\hat{\sigma}_{\text{var}}$ by (23) and (25)
10. **end if**
11. **loop**
12. For a SAP $\boldsymbol{Y}$ collected from a candidate channel, compute the mean $\mu_{\boldsymbol{Y}} = M(\boldsymbol{Y})$ and variance $\sigma^2_{\boldsymbol{Y}} = V(\boldsymbol{Y})$.
13. **if** $\mu_{\boldsymbol{Y}} \leq \tilde{\gamma}_{\text{mean}}$ and $\sigma^2_{\boldsymbol{Y}} \leq \tilde{\gamma}_{\text{var}}$ **then**
14. {or $\mu_{\boldsymbol{Y}} \geq \tilde{\gamma}'_{\text{mean}}$ for OFF periods}
15. SAP $\boldsymbol{Y}$ is from a PU. *Alarm* = NO.
16. **else**
17. SAP $\boldsymbol{Y}$ is from an attacker. *Alarm* = YES.
18. **end if**
19. **end loop**

We run the simulation with 30 batches. Each batch includes 30 experiments with a different seed. In our experiments, the initial SPARS training typically takes a few minutes on a desktop PC with a Intel Core i5 2.67 GHz processor. The reconstruction of a SAP after the initial phase is in the order of milliseconds. To simulate varying scenarios, in each experiment, the number of attackers is generated as a random number between 0 and 30, instead of using a fixed number of attackers. The ON periods are simulated with the exponential distribution, with the mean ON period for PUs as 1 time unit, and the mean ON period for attackers varying from 1.5 to 2 time units. A larger mean ON period indicates that the channel availability for SUs is smaller. The OFF periods are simulated with the Pareto distribution using the shape parameter 1.9, and the location parameter 4 for PUs. The location parameter for attackers varies from 2 to 3, and the shape parameter is the same. This results in a mean OFF period of attackers being 50% to 75% of the mean OFF period of PUs. A smaller mean OFF period indicates that the channel availability for SUs is smaller.

Being an approach that targets general scenarios of the PUE detection without limiting the PU type and/or requiring a priori knowledge of PUs, SPARS takes a very different methodology from existing PUE detection schemes. This makes it difficult to quantitatively compare the performance of SPARS with the ones of other PUE detection schemes. For instance, the location verification approach [3], [5] assumes that the PU location is known in advance, and the performance depends on how accurately the spectrum sensing can estimate the distance to the transmitter. In contrast, SPARS does not need such accurate localization. Hence, in this paper, we do not quantitatively compare SPARS with the existing PUE detection schemes.

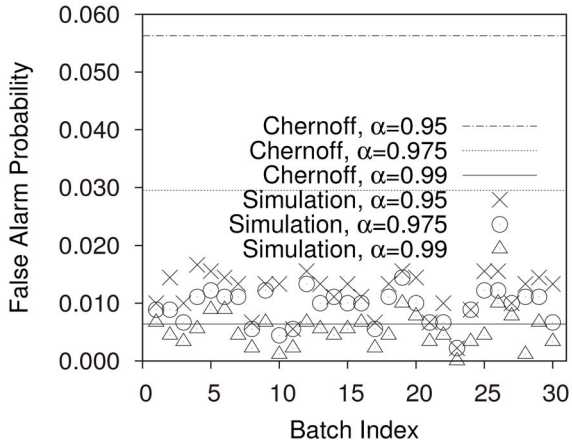

Fig. 2. False alarm probability for detection of SAPs consisting of ON periods.

Next, we first examine the performance of SPARS for detecting straight attackers that do not forge their SAPs, and then evaluate SPARS comparatively with PDS for detecting *smart attackers* that forge their SAPs.

9.1 Detection of Straight Attacker

Fig. 2 plots the false alarm probability of the PUE detection by SPARS, i.e., the probability that a PU is alarmed as an attacker by SPARS, when the ON periods are used as the SAP for detection. The X-axis indicates the batch ID of the experiments. The false alarm probability is averaged over the 30 experiments in each batch. We use three tolerance limits for PUE detection, with $\beta = 0.999$ and $\alpha = 0.95$, 0.975, and 0.99, respectively. We can see that the false alarm probability meets the expectation of the tolerance limits. That is, the false alarm probability is expected to be not larger than $1 - \alpha$. We also plot the Chernoff bounds for the false alarm probability with the three tolerance limits as the parameters. We can see that the Chernoff bounds are quite tight. Nevertheless, one may note that the Chernoff bounds here are approximations as we have used the sample mean and standard deviation instead of the real mean and standard deviation of the SSE ζ, which are unknown. Hence, such tight Chernoff bounds may be due to this substitution. Fig. 3 illustrates the false alarm probability when the OFF periods are used as the SAP for PUE detection. The false alarm probabilities are slightly higher than for ON periods, but still approximately meet the expectation of the tolerance limits.

Next we examine the miss-detection probability, i.e., the probability that an attacker is not detected. Fig. 4 illustrates the miss-detection probability when the ON periods are used in the SAP for PUE detection. The miss-detection probability is averaged on all experiments of all batches. We also use three tolerance limits, with $\alpha = 0.95$ and $\beta = 0.99$, 0.995, and

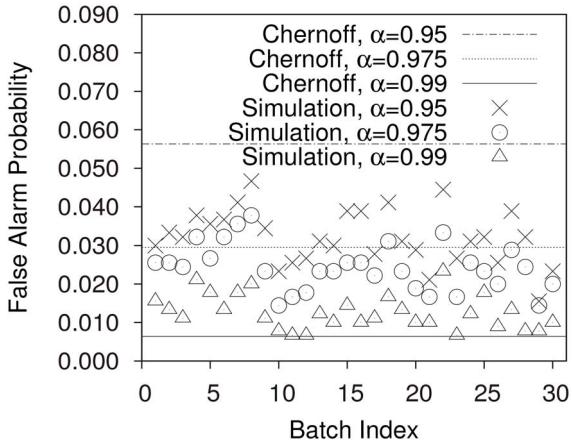

Fig. 3. False alarm probability for detection of SAPs consisting of OFF periods.

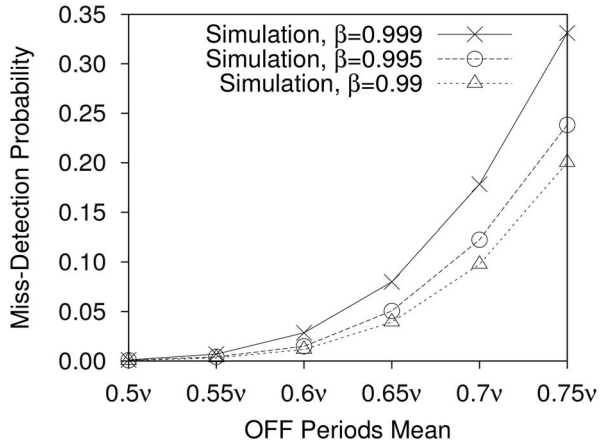

Fig. 5. Miss-detection probability for detection of SAPs consisting of OFF periods.

0.999, respectively. In the figure, μ denotes the mean ON period of PUs. The X-axis indicates the mean ON period of attackers, e.g., 1.5μ indicates 1.5 times of μ. We see that if the mean ON period of attackers is 70% larger than the one of PUs, most attackers are caught by SPARS. If the mean ON period of attackers increases to twice as the one for PUs, i.e., 2μ, then almost all attackers are caught by SPARS. Note that the typical spectrum occupation of PUs is low, e.g., 10%. Hence even doubling the spectrum occupation by an attacker does not really result in a high spectrum occupation, while on the other hand, the attacker is in the risk of almost surely being caught by SPARS. Therefore, SPARS is effective to detect PUE attack. Among the three tolerance limits, a smaller β indicates that the tolerance limit is smaller or tighter; hence an attacker is more likely to be caught.

Fig. 5 illustrates the miss-detection probability when the OFF periods are used as the SAP for PUE detection. The ν denotes the location parameter of the Pareto distribution to generate the OFF periods of PUs, which is proportional to the mean. Given the same shape parameter, 0.5ν indicates that the mean OFF period of attackers is about half of the mean OFF period of PUs. When the mean OFF period of attackers decreases by 25% (0.75ν), about 33% attackers are not caught by SPARS. However, if the mean OFF period of attackers is about half of the mean OFF period of PUs, almost all the attackers are caught, with the miss-detection probability close to 0.

Next, we examine the *receiver operating characteristic* (ROC) curve of SPARS, which is a plot of the true positive rate, i.e., 1− miss-detection probability, versus the false positive rate, i.e., the false alarm probability. Fig. 6 plots the ROC curves of SPARS for detecting SAPs of ON periods, when the ON periods of attackers' SAPs are generated with 1.5μ, 1.8μ, and 2μ, respectively, as the mean. We use the $(0.95, 0.999)$ tolerance limit for PUE detection in SPARS. The ROC curves indicate that SPARS is effective to detect SAPs of attackers, with high true positive rates versus low false positive rates. In particular, when the mean ON period of attackers is 1.8 times of the mean ON period of PUs or larger, the true positive rate is close to 1 versus a false positive rate close to 0. Fig. 7 illustrates the ROC curves of SPARS for detecting SAPs of OFF periods, when the OFF periods of attackers' SAPs are generated with 0.5ν, 0.65ν, and 0.75ν, respectively, as the location parameter of the Pareto distribution. The observations are similar to those in Fig. 6, and confirm that SPARS is effective to detect SAPs of attackers. In particular, when the mean OFF period of attackers is 0.5 times of the mean OFF period of PUs or smaller, the true positive rate is almost 1 versus a low false positive rate.

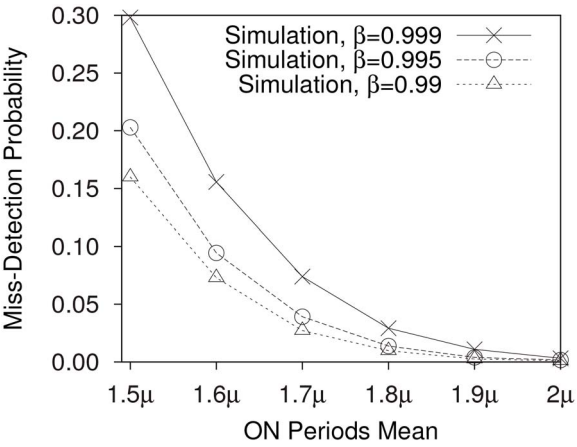

Fig. 4. Miss-detection probability for detection of SAPs consisting of ON periods.

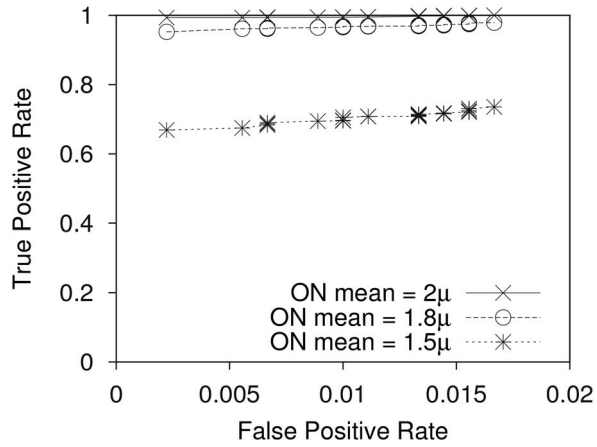

Fig. 6. ROC curves of SPARS for detecting SAPs consisting of ON periods.

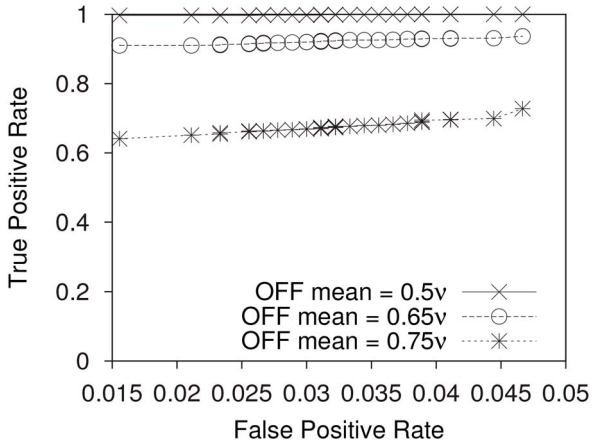

Fig. 7. ROC curves of SPARS for detecting SAPs consisting of OFF periods.

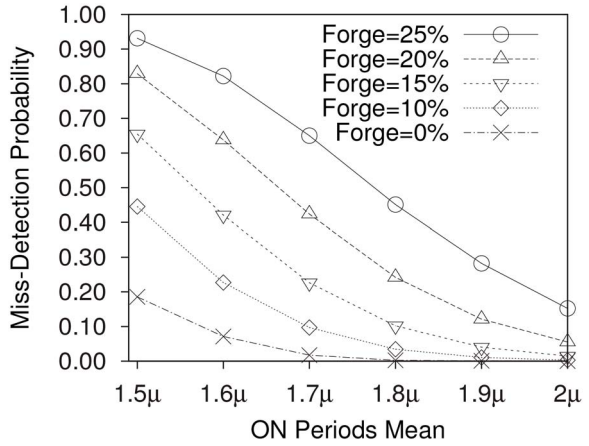

Fig. 9. Miss-detection probability of PDS for detecting smart attackers.

Putting it all together, for attackers to cause DoS to the CRN, they want to increase ON periods and/or decrease OFF periods. With SPARS, if an attacker increases the average ON period by more than 100% or decreases the average OFF period to about half, which results in only a mild increase of spectrum occupation, it will be almost surely caught by SPARS. On the other hand, if an attacker wants to avoid the detection of SPARS, it has to have almost the same average ON and OFF periods as PUs. However, as discussed in Section 1, such an attacker would have low spectrum occupation, which defeats the objective of the PUE attack and would be tolerable by the CRN. Therefore, SPARS is effective to detect and deter PUE attacks.

9.2 Detection of Smart Attackers

In this section, we consider smart attackers that forge their SAPs, and evaluate SPARS comparatively with PDS. Both schemes can detect PUE attacks in general scenarios. For straight attackers, PDS also has a very good performance comparable to the one of SPARS. Nevertheless, as to be seen, the drawback of PDS is on detecting smart attackers.

A smart attacker can manipulate its ON/OFF periods so that the mean ON/OFF period is more close to the one of PUs. Specifically, a smart attacker randomly generates a small fraction of very short ON periods, e.g., using a mean close to 0, to intentionally decrease the mean ON period in a SAP. Similarly, it can randomly generate a small fraction of very long OFF periods, to intentionally increase the mean OFF period in a SAP.

Figs. 8 and 9 plot the miss-detection probabilities of SPARS and PDS, respectively, for detecting SAPs of ON periods. The 'Forge' in the figures indicates the forging fraction of the ON periods in a SAP. For instance, 'Forge = 0.2' indicates that 20% ON periods in a SAP are forged. We use the $(0.95, 0.999)$ tolerance limit for the PUE detection for both SPARS and PDS. PDS needs two tolerance limits, one for testing the mean and the other for testing the variance. Both tolerance limits use the same α and β values. From Fig. 8, we can see that SPARS is robust to smart attackers as the miss-detection probability does not increase much even with up to 25% forged ON periods. In contrast, the miss-detection probability of PDS in Fig. 9 deteriorates significantly when the forging fraction of the ON periods increases. Comparing Fig. 8 and Fig. 9, we can see that SPARS is much more robust than PDS for detecting smart attackers. For instance, with 25% forged ON periods, the miss-detection probability of PDS is much higher than the one of SPARS. In other words, smart attackers can easily defeat PDS by forging a fraction of the ON/OFF periods in its SAPs, while they will be still caught by SPARS. The results of using

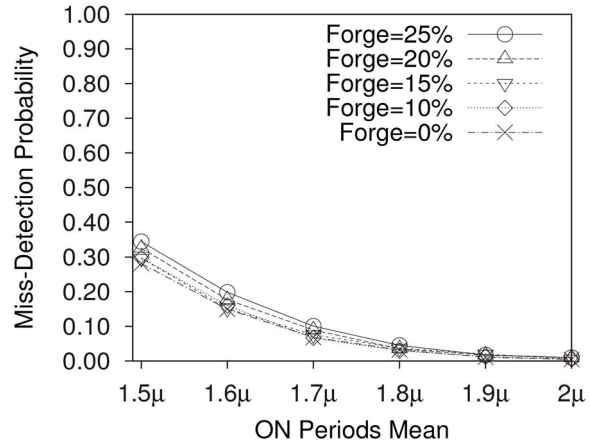

Fig. 8. Miss-detection probability of SPARS for detecting smart attackers.

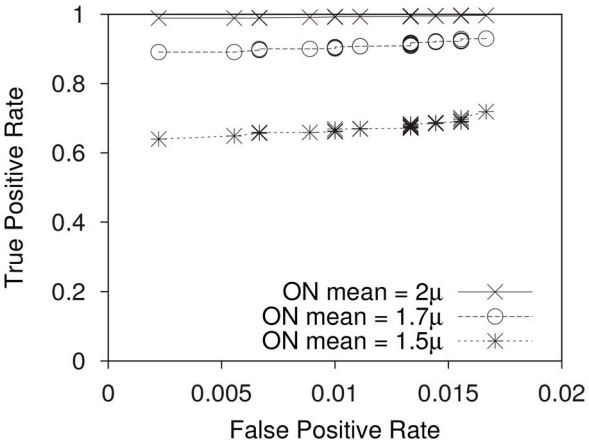

Fig. 10. ROC curves of SPARS for detecting smart attackers.

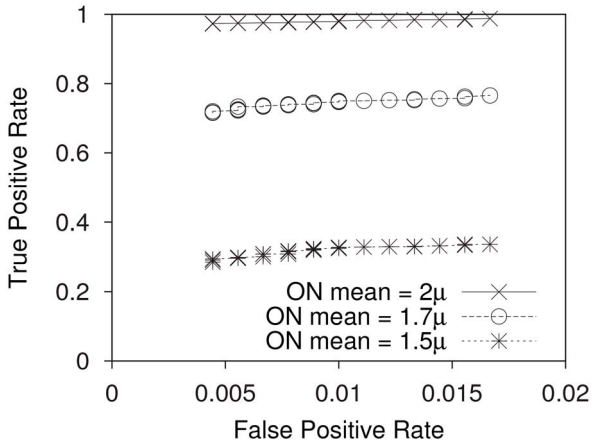

Fig. 11. ROC curves of PDS for detecting smart attackers.

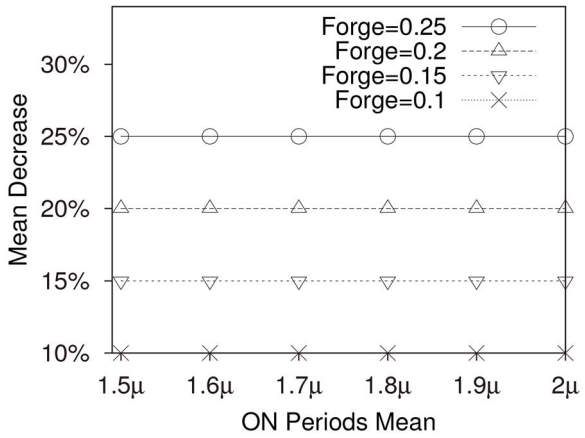

Fig. 13. Decrease of the ON periods mean caused by forged SAPs.

OFF periods for SAPs have similar observations and are omitted.

Figs. 10 and 11 plot the ROC curves of SPARS and PDS, respectively, for detecting smart attackers, with the forging fraction being 20%. Comparing Fig. 10 and Fig. 11, we can see that SPARS is much more effective for detecting smart attackers, as the true positive rate of SPARS is relatively much higher versus the false positive rate. For instance, in the case that the mean ON periods of attackers is 1.7μ, the true positive rate of PDS versus the 0.01 false positive rate is around 0.75, while the one of SPARS is approximately 0.9.

One may wonder if forging the SAPs by smart attackers would increase the variance or standard deviation of the ON/OFF periods of the SAPs, such that PDS can use a tighter tolerance limit for variance to detect smart attackers. Fig. 12 illustrates the percentage increase of the standard deviation of the ON periods in the SAPs of smart attackers, with different forging fractions. We can see that the increase of the standard deviation of the ON periods is rather small; hence tightening the tolerance limit for variance in PDS does not work well. For instance, even with 25% ON periods being forged, the increase of the standard deviation is only about 3%. Capturing such an increase requires a tolerance limit that results in a very high false alarm probability. At last, we plot the percentage decrease of the ON periods mean in the SAPs of smart attackers in Fig. 13. The decrease of the ON periods mean is approximately the forging fraction, which is expected as the smart attacker uses a very small mean to generate the forged ON periods.

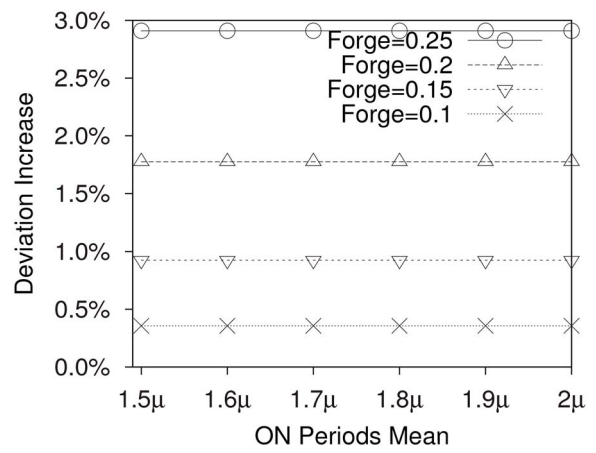

Fig. 12. Increase of the standard deviation of ON periods caused by forged SAPs.

In summary, PDS is a fast PUE detection approach and performs well for straight PUE attacks, but not robust to smart PUE attacks. On the other hand, SPARS is more robust to smart PUE attacks than PDS, but is a more complex system. Fortunately, although it is complex, SPARS is efficient to detect PUE attacks. After the initial training, the PUE detection by SPARS takes only a few milliseconds in our experiments. The initial training time takes several minutes in our experiments, which is, nevertheless, compensated by the good performance of SPARS.

10 Conclusion and Future Work

We have presented a PUE detection system termed SPARS, which acquires the signal activity pattern (SAP) of PU signal transmitters. It uses a SAP reconstruction model to reconstruct an observed SAP and finds if the SAP belongs to an attacker based on the reconstruction error. Different from current solutions on the PUE detection, SPARS does not need a priori knowledge of PUs, and has no limitation on the type of applicable PUs. The performance evaluation indicates that SPARS is robust and effective to detect both straight and smart PUE attackers, even though the smart attackers may forge the SAPs.

In our future work, we will extend SPARS for the PUE detection when there are multiple classes of PUs, and different classes of PUs have different signal activity patterns. Specifically, we will extend SPARS to classify an observed SAP to see if it belongs to a certain class of PUs. If yes, then this SAP is from a PU. Otherwise, it is from an attacker. To achieve this objective, we will need to examine the structure of the weights in the reconstruction of a SAP, in addition to the reconstruction error.

Acknowledgments

The research of ChunSheng Xin is supported in part by US NSF under grants CNS-1418012 and ECCS-1418013. The research of Min Song is supported in part by US NSF CAREER Award CNS-1248092 and NSF IPA Independent Research and Development (IR/D) Program. Any opinion, finding, and conclusions or recommendations expressed in this material;

are those of the author and do not necessarily reflect the views of the US National Science Foundation.

REFERENCES

[1] H. Li and Z. Han, "Dogfight in spectrum: Combating primary user emulation attacks in cognitive radio systems, Part II: Unknown channel statistics," *IEEE Trans. Wireless Commun.*, vol. 10, no. 1, pp. 274–283, Jan. 2011.

[2] Y. Tan, K. Hong, S. Sengupta, and K. Subbalakshmi, "Using Sybil identities for primary user emulation and byzantine attacks in dsa networks," in *Proc. IEEE GLOBECOM*, Houston, TX, USA, 2011.

[3] Z. Yuan, D. Niyato, H. Li, and Z. Han, "Defense against primary user emulation attacks using belief propagation of location information in cognitive radio networks," in *Proc. IEEE WCNC*, Cancun, Mexico, 2011.

[4] N. Nguyen, R. Zheng, and Z. Han, "On identifying primary user emulation attacks in cognitive radio systems using nonparametric Bayesian classification," *IEEE Trans. Signal Process.*, vol. 60, no. 3, pp. 1432–1445, Mar. 2012.

[5] R. Chen, J.-M. Park, and J. Reed, "Defense against primary user emulation attacks in cognitive radio networks," *IEEE J. Sel. Areas Commun.*, vol. 26, no. 1, pp. 25–37, Jan. 2008.

[6] S. Chen, K. Zeng, and P. Mohapatra, "Hearing is believing: Detecting mobile primary user emulation attack in white space," in *Proc. IEEE INFOCOM*, 2011.

[7] R. Rubinstein, M. Zibulevsky, and M. Elad, "Double sparsity: Learning sparse dictionaries for sparse signal approximation," *IEEE Trans. Signal Process.*, vol. 58, no. 3, pp. 1553–1564, Mar. 2010.

[8] D. L. Donoho, "For most large underdetermined systems of equations, the minimal l1-norm near-solution approximates the sparsest near-solution," *Wiley Commun. Pure Appl. Math.*, vol. 59, no. 7, pp. 907–934, 2006.

[9] J. Wright, A. Yang, A. Ganesh, S. Sastry, and Y. Ma, "Robust face recognition via sparse representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 2, pp. 210–227, Feb. 2009.

[10] J. Yang, K. Yu, and T. Huang, "Efficient highly over-complete sparse coding using a mixture model," in *Proc. 11th ECCV*, Heraklion, Greece, 2010, pp. 113–126.

[11] B. Zhao, L. Fei-Fei, and E. Xing, "Online detection of unusual events in videos via dynamic sparse coding," in *Proc. IEEE Conf. CVPR*, Providence, RI, USA, 2011, pp. 3313–3320.

[12] O. Vinyals and L. Deng, "Are sparse representations rich enough for acoustic modeling?" in *Proc. 13th Annu. Conf. International Speech Communication Association*, Portland, OR, USA, Sept. 2012.

[13] B. A. Olshausen and D. J. Fieldt, "Sparse coding with an over-complete basis set: A strategy employed by v1?" *Vision Res.*, vol. 37, no. 23, pp. 3311–3325, Dec. 1997.

[14] H. Lee, A. Battle, R. Raina, and A. Y. Ng, "Efficient sparse coding algorithms," in *Proc. NIPS*, 2006.

[15] A. Madanayake, C. Wijenayake, N. Tran, S. Hum, L. Bruton, and T. Cooklev, "Directional spectrum sensing using tunable multi-d space-time discrete filters," in *Proc. IEEE Workshop CORAL*, San Francisco, CA, USA, Jun. 2012.

[16] K. Krishnamoorthy and T. Mathew, *Statistical Tolerance Regions: Theory, Applications, and Computation*. Hoboken, NJ, USA: Wiley, 2009.

[17] M. G. Natrella, *Experimental Statistics, NBS Handbook 91*. Washington, DC, USA: U.S. Department of Commerce, 1963.

ChunSheng Xin received the Ph.D. degree in computer science and engineering from the State University of New York at Buffalo, Buffalo, NY, USA, in 2002. He is an Associate Professor with the Department of Electrical and Computer Engineering, Old Dominion University, Norfolk, VA, USA. His current research interests include cybersecurity, cognitive radio networks, wireless communications and networking, cyber-physical systems, and performance evaluation and modeling. His research is supported by multiple NSF grants, and results in numerous papers in leading journals and conferences, as well as several book chapters and one patent. He has served as an Associate Editor of international journals, and an External Consultant on cybersecurity for industry. He is a senior member of IEEE.

Min Song received the Ph.D. degree in computer science from the University of Toledo, Toledo, OH, USA, in 2001. He is a Professor with the Electrical Engineering and Computer Science Department at the University of Toledo. He is currently serving the US National Science Foundation as a Program Director of CNS/CISE. He is the recipient of NSF CAREER Award. His professional career is comprised of a total of 25 years of work experience in academia, government, and industry. He was the Founding Director of a Networking System Division in an IT company, and launched an international journal and served as the Editor-in-Chief. He has acted as an Editor or Guest Editor of 13 international journals, and served as a General chair, Technical Program Committee Chair, and Session Chair for numerous international conferences. He is a senior member of IEEE.