# CR-Honeynet: A Cognitive Radio Learning and Decoy Based Sustenance Mechanism to Avoid Intelligent Jammer

Suman Bhunia, Edward Miles, Shamik Sengupta, and Felisa Vázquez-Abad

***Abstract*—Cognitive Radio Network (CRN) enables secondary users to borrow unused spectrum from the proprietary users in a dynamic and opportunistic manner. However, dynamic and open access nature of available spectrum brings a severe sustenance challenge amongst CRNs which makes them vulnerable to various spectrum etiquette attacks. Jamming-based denial-of-service (DoS) attack poses serious threats to legitimate communications and packet delivery. A rational attacker targets specific transmission characteristics to find the highest impacting connection of CRN and causes maximum disruption. With the help of software-defined radios, we have shown that an attacker can intelligently target a particular communication. In this paper, inspired by the honeypot concept in cybercrime, we propose a honeynet based defense mechanism, which aims to deter the attacker from jamming legitimate communications. The honeynet passively learns the attacker's strategy from the history of attacks and actively adapts preemptive decoy mechanisms to prevent attacks on legitimate communications. Simulation results show that the with the help of honeynet mechanism, CRN successfully avoids jamming attacks and thereby improves system performance regarding packet delivery ratio. We further built a prototype using off-the-shelf software defined radio that proves the effectiveness of the proposed mechanism.**

***Index Terms*—Cognitive Radio, Jamming, Honeynet, Stochastic Learning, USRP, GNURadio**

## I. Introduction

Dynamic Spectrum Access (DSA) based Cognitive Radio (CR) [2] aims to provide a solution for spectrum scarcity by allowing Secondary Users (SUs) to use unused licensed spectrum on a non-interfering basis. In contrast to conventional wireless technologies, a CRN can reconfigure itself by controlling its operating frequency, channel bandwidth, modulation techniques, transmission power, etc. [2]. Because of the authorized primary user (PU) priority, SUs must periodically sense the channel of communication for the presence of the PU. If the current channel is blocked by the presence of PU of that channel, SU must switch to another free channel.

The "open" philosophy of the cognitive radio paradigm makes CRN susceptible to jamming-based denial-of-service (DoS) attacks by smart malicious users [2]–[4]. An attacker can scan through channels, identify legitimate SU communications and then transmit a jamming signal on the same channel or fragment of the channel causing disruptive interference to the SU, which in effect can completely block the legitimate SU's transmission [2], [3]. However, note that, from an intelligent and rational attacker's perspective, jamming a communication randomly will not yield optimal result; rather an attacker can be most disruptive if it targets the communication that impacts the CRN most severely upon interruption [5]–[9]. The attacker succeeds in determining highest impacting communication [8], [10]. Many transmission characteristics can be used to find out the target transmission, as for example, highest transmission power, highest data rate, modulation scheme, packet inter-arrival time, frequency shift, quality of route with end-to-end acknowledgments, etc [11]–[16]. Again, an attacker may also use combination of transmission characteristics instead of a single characteristics to find out the highest impacting communication. To defend against such attackers, a CRN must learn about the strategy (the targeted transmission characteristic(s)) that the attacker uses to figure out the highest impacting communication. The attacker's strategy of finding highest impacting communication can be used as a trap by the defending CRN to detract the attacker from attacking legitimate communications.

Honeypot is a well-known tool in cybersecurity where a resource, called honeynode is dedicated with lowest possible security to collect extensive data from cyber attacks [17]. The attackers or cyber bots get a false impression of hacking a real system by intruding the honeynode. Analyzing this data allows the defender to extract information or pattern of the attacks and even diagnose or detect new threats or vulnerabilities in the system that are yet to be discovered. In this paper we propose *CR-Honeynet*, a honeypot based defense mechanism where the CRN passively learns the *strategy of the attacker* using stochastic learning, and then place an active decoy namely *honeynode* to entice the attacker for jamming the honeynode transmission. Thus, the attacker gets a false impression of attacking the highest impacting communication whereas legitimate SU communications avoid attacks and reduce attack impact on the CRN. One or multiple SUs act as honeynode in each transmission period. The SU acting as honeynode refrains from transmitting its own data packets and transmits garbage data with specific transmission characteristics. Such transmission characteristics lure the attacker to jam Honeynode's transmission. The transmission characteristics that the attacker aims is learned from the history of attacks. As an example, if an attacker targets highest transmission power then the honeynode transmits with highest possible power

This research was supported by NSF CAREER grant CNS #1346600. A preliminary version of this paper has been presented at IEEE MILCOM 2014 [1].

S Bhunia is with University of California, Davis. E Miles is with IntelliSource. S Sengupta is with University of Nevada, Reno. F Vázquez-Abad is with City University of New York. (e-mail: sbhunia@ucdavis.edu, elmiles@nevada.unr.edu, ssengupta@unr.edu, and felisav@hunter.cuny.edu)

while all other SUs keep their transmission power lower than Honeynode's power.

The evolving nature of the attacker, as well as dynamic and stochastic nature of the wireless medium, pose several challenges to the learning mechanism. Suspicious of being trapped, an attacker may intentionally change its strategy of finding highest impacting communication. Also, due to erroneous and stochastic nature of wireless medium, an attacker may err in sensing CRN's highest impacting connection. Such error may result in an attack on different SU communication instead of the communication with desired/targeted characteristics. Such circumstances must be taken into account for effective luring. In this paper, we use statistical monitoring threshold to decide whether the changes in recent attack pattern is due to an error in attacker's sensing or whether the attacker has changed its attacking strategy. Our proposed stochastic learning mechanism correctly detects attacker's strategy with a probability of $0.958$ within 15 iterations and identifies a change in attacker's strategy dynamically with 95% confidence interval within five iterations. The simulation results show that CR-Honeynet learns attacker's strategy correctly and adapt to attacker's strategy change dynamically which in effect enhances CRN's performance regarding packet delivery ratio. We further developed a state-of-the-art testbed using off-the-shelf software defined radios. An attacker is developed using GNURadio that can sense the ongoing communications and target a particular channel to jam. The CR-Honeynet is built as a network of five nodes. The setup shows the effectiveness of the defense mechanism proposed. In summary, the main contributions of this paper are:

- A novel jamming defense mechanism that uses the intelligence of the attacker for its benefit.
- Stochastic model for learning the attacker's strategy in a noisy and transient wireless environment.
- Regime change detection model to detect when an attacker changes its strategy after being suspicious of a honeynode trap.
- Prototype developed using off-the-shelf hardware to prove the feasibility of the proposed mechanism.

The rest of the paper proceeds as follows: in Section II, we discuss the motivation for our work and background studies. Section III presents our proposed model. In Section IV we describe our simulator and then analyzes CR-Honeynet's performance. The prototype system and the experiments are described in Section V. Finally Section VI concludes the paper.

## II. Background studies

### A. Jamming attack

In traditional wireless networks, the user of a particular channel has proprietary access to that channel and thus has the right to penalize any trespassers. The threat of penalty can discourage potential attackers. However, if a CRN is accessing a channel, the SUs are only borrowing the channel, and they have no grounds from which they can fend off attackers. Thus, SUs are left vulnerable to malicious jamming attacks [2]. Jamming can be broadly categorized into two types [18], [19]. In *physical layer jamming*, the attacker jams the channel of communication by sending strong noise or jamming signals. The *data-link / MAC layer jamming* targets several vulnerabilities present in the MAC layer protocol [6], [20]. Jamming essentially means disrupting communication of legitimate users.

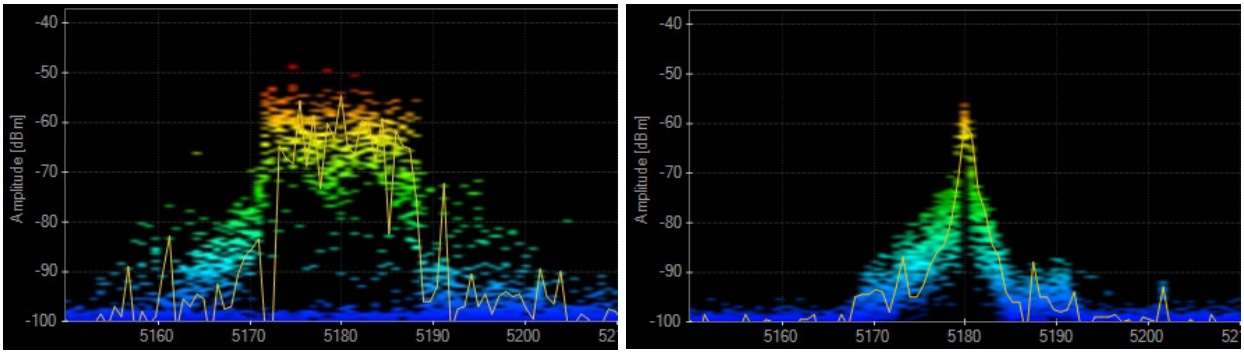
To illustrate the effect of jamming, we ran an experiment in our lab. Two computers were configured to communicate over a WLAN (IEEE 802.11-a) channel 36 (centered at 5.18 GHz). When communicating at full throttle, it achieved end-to-end throughput of 11 Mbps. We observed the Power Spectral Density (PSD) over the channel using the Wi-spy spectrum analyzer [21]. The PSD for healthy communication is shown in Figure 1a. The plot shows that the transmission is using a 20 MHz channel as well as some energy leakage to the neighboring channels. Then we started transmitting a very narrow band jamming signal of 2MHz from a GNU Radio [22] enabled USRP board [23]. In the presence of the jamming signal, the genuine transmission was blocked entirely as can be viewed in Figure 1b where only the jamming signal is visible. The attacker is exploiting the vulnerability of IEEE 802.11 MAC that enforces a node to sense the channel before transmission. When the legitimate transmitter senses that there is some energy on the channel, it refrains from transmission. In effect, the attacker successfully jams the channel with minimal cost. Irrespective of the jamming technique, a target node suffers a significant amount of data or packet loss and sometimes completely loses the channel. CRN being a next-generation intelligent network should incorporate a mechanism to mitigate, avoid or prevent these attacks.

### B. Detection of jamming attacks

Due to the noise in the wireless medium, detection of jamming is crucial in combating with an attacker. An excellent survey of different detection mechanisms for jamming based DoS attack is presented in [8]. It is difficult to detect jamming based on a single system parameter correctly. Several system parameters such as received-signal-strength, packet-send-ratio, packet-delivery-ratio, carrier-sensing-time, etc. are used to model jamming detection system. Consistency check among system parameters is used for more efficient detection. Authors of [24] have classified spectrum usage anomaly detection algorithms. Through different fusion algorithms, anomalies in spectrum usage can be detected successfully with higher efficiency. A cross-layer detection mechanism of anomalous spectrum attack was proposed in [25], [26] where the network maps the jammed geographical region using spectrum sensing reports sent from each SU that are equipped with localization module.

### C. Defense against jamming attacks

Already proposed defense mechanisms against jamming-based DoS attack can be broadly categorized into *spatial filtering*, *spatial retreat*, *mapping jammed region*, *spread spectrum*, *strategic power allocation game*, *channel hopping*, etc. In *spatial filtering* approach, mobile nodes use adaptive beamforming to filter out the signal coming from a jammer [27], [28]. To defend mobile jammer, defenders use periodic




(a) Normal communication (b) Jamming Signal

Fig. 1: PSD for data communication and jamming signal

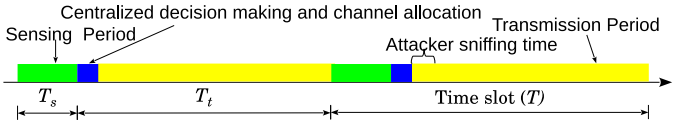



Fig. 2: Time slots in cognitive cycle

measurements of the direction of arrival for jamming signal; and then predict the movement of the jammer in between two observation intervals to create the optimized beamform to keep jammers in null regions [29], [30]. In *Spatial Retreat* [31] mobile nodes relocate themselves physically to avoid jamming. The constraint of this approach is that the nodes are required to be highly mobile which is not realistic for static nodes. In *Mapping Jammed Region* [32] approach, the intensely populated multi-hop CRN avoids routing through the links that have been affected by jamming. This mechanism fails if there is only one path and that link is attacked. In *Spread Spectrum* [33] technique, low bandwidth data stream uses higher bandwidth channel to pass the information irrespective of jamming.

In *strategic power allocation* technique [34]–[36], the interaction between SUs and attacker is modeled as a game where both players choose optimal power allocation strategies that maximize their incentive. These game model how the players should behave where they see incomplete information such as channel usage, utilities where the players are limited by their power budget, transmission bandwidth, etc. Several other game models are proposed [9], [37]–[40] where the jammer's action is treated as random decision process and the defender tries to play with an optimal strategy where it performs different plans with different probabilities. The underlying assumption of these games is that the action space of both the players is known to each other.

In *Channel hopping* technique, the node which is under attack migrates its channel of communication upon detection of jamming [19]. Authors of [41] proposed proactive frequency hopping where the nodes change its channel of operation irrespective of attacks to avoid jamming. The authors considered a fixed number of channels of the attacker that is known to an SU, which in reality is difficult to achieve. In [42], the authors proposed a random jump pattern called Tri-CH that achieves high-security level. The random jump pattern is a permutation of the available channels. It guarantees that the transmitter and receiver will be able to communicate with a specific number of jumps or within a bounded time.

Majority of the previous works have assumed that the attacker is naive and does not evolve. Thus, none of these works have focused on learning the strategy of attacker where the attacker is also dynamic and changes its policy of choosing the target communication characteristics. In our previous work [43], [44], we introduced the concept of honeynet in CRN and presented the benefits of dedicating one SU as honeynode in a multichannel CRN provided the honeynode is successful of enticing the attacker. We presented a stochastic model for honeynode selection which proved that in the case of uniform traffic, selecting an SU with lowest queue size is optimal concerning overall system performance. Extending on previous work, in this paper we present CR-Honeynet, where the CRN learns the strategy of the attacker and then dedicates one SU as honeynode. Honeynode acts as the optimal target for the attacker so that the attacker gets the false satisfaction of attacking highest impacting SU communication, thus reducing attack impact on other legitimate communications.

## III. Proposed Model

In this section we describe the system model, its assumptions, challenges and the proposed defense mechanism.

### A. Assumptions

The general system assumptions are:

i) **Network:** The network consists of one central controller and several SUs. The central controller oversight the channel used by SUs. For an example, consider LTE-U [45] network. Here the base station communicates with the cellular phones on licensed spectrum, however, for higher data rate, the nodes can communicate through unlicensed bands.

ii) **Time slots:** To protect PU transmissions, SUs are required to perform periodic spectrum sensing and evacuate promptly upon the return of the PU [2]. A time slot consists of a sensing period followed by a transmission period as can be seen in Figure 2. SUs scan the wireless environment for free channels in the *sensing period* ($T_s$). SUs send the spectrum usage report to the central

controller for decision. During transmission period ($T_t$), an SU transmits packets through its channel dedicated by a centralized controller.

iii) **Jammer:** Jammer is itself an SU with the same power of a regular SU [3].

iv) **Attack:** We assume that an SU cannot switch its channel during the transmission period as it is unaware of the condition of the other channels and can change only on the next transmission period. Upon being attacked, all data packets transmitted by the SU are lost [3].

v) **Common control channel:** The network uses an out-of-band common control channel for control message communication between the central controller and the SUs. This channel is proprietary and the attacker can not attack due to the heavy penalty associated with attacking a licensed spectrum [46].

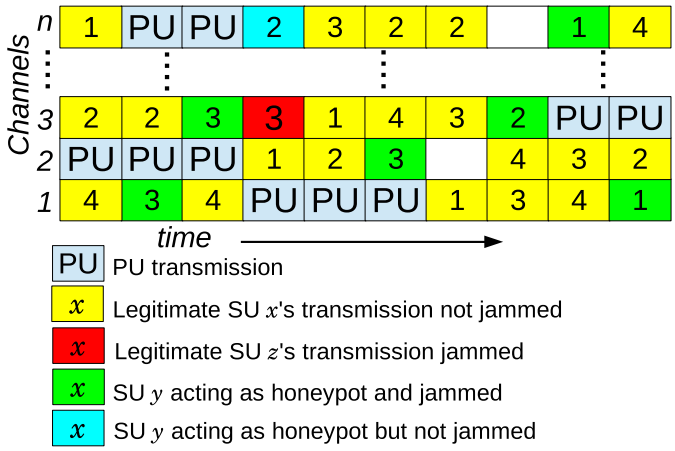



Fig. 3: A snapshot of CR-Honeynet channel allocation

### B. Model for attackers

In this paper we consider three types of attacking strategies, as follows:

I: **Static attacker:** Attacker targeting a particular channel.

II: **Adaptive attacker** Targets specific SU transmission characteristic(s).

III: **Random attacker** Randomly targets a channel with an active SU transmission on every time slot.

Attacking strategy of type I and III causes less harm on a CRN as it does not search for the best communication that causes the highest impact in CRN. However, an intelligent and rational attacker of type II can choose any transmission characteristics to determine the best communication for the attack that causes the highest impact on the CRN. From the CRN's point of view, it is difficult to generate such characteristic space. Such targeted characteristic space of an attacker can be learned by two methods: *manually by domain experts* or through *automatic learning* from data obtained for a long time. For the first step, we are dealing with the first method and wish to extend our model to perform the second option and learn an attacker's possible strategical viewpoints by automatic learning. We present a generalized model considering the $d$ possible transmission characteristics or a combination of transmission characteristics.

### C. CR-Honeynet defense mechanism

In CR-Honeynet, the central controller assigns the role of honeynode to an SU at the beginning of each transmission period. Figure 3 illustrates this channel allocation based on time domain (Sensing period not shown). Due to the error of the attacker or strategy change of the attacker, some attacks are trapped by honeynode transmission, and others disrupt legitimate SU communications. We define a parameter, *attractiveness of honeynode* ($\xi$) as the probability that the honeynode transmission is attacked, conditional on observing a jamming attack.

When acting as a honeynode, an SU doesn't transmit its packets; instead, it queues all its incoming packets and sends garbage data packets. Honeynode allocation results in more delay as well as packet drop due to finite buffer sizes for the chosen SU, both of which are undesirable. If the attractiveness of the honeynet ($\xi$) is low, then the CRN will suffer the delay caused by honeynode allocation as well as the packet drop with a probability of $(1-\xi)$ due to the attacks on legitimate SUs other than the one chosen as honeynode. The threshold, lowest attractiveness of the honeynet ($\xi^*$) is the value where the net gain is zero; below $\xi^*$ the CRN is better off facing the loss from the attacks than dedicating one SU intending to lure the attacker.

In accordance to an attacker's strategies, we define the following honeynode strategies:

i: If the attack strategy is believed to be of type-I then the vulnerable channel will be assigned to the honeynode.

ii: If the attack strategy is believed to be of type-II then the actual target property should be learned and used as a lure for the honeynode.

iii: If the attack strategy is believed to be of type-III then we use a *special honeynode strategy* that delays all but the honeynode's transmissions in order to reduce the number of vulnerable channels to 1.

If there are $\mathcal{C}$ available channels, then an attacker must sense for activity on each of them. Let's assume switching a channel incurs a delay of $\kappa$ ($\kappa = 7.6ms$ has been measured for Atheros WiFi [41]). The attacker needs at most $\mathcal{C}\kappa$ time units to scan all available channels for activity. Under the special strategy, we must, therefore, delay all SUs at least $\mathcal{C}\kappa$ units of time beyond the sensing period, during which only the honeynode will transmit. When using the special strategy there is an added loss or cost of luring as all the other SUs are delayed in their transmissions, albeit much less than the delay caused to the chosen honeynode. So this strategy should be avoided by CR-Honeynet if possible. In contrast, type-III is the only strategy that increases the attractiveness of honeynet ($\xi$) to 1.

### D. Stochastic model

We assume that at time slot $n$ the attacker's strategy $S_n$ follows a random switching process, with consecutive switching times $T_k \in \mathbb{N}$. The model need not be a *Hidden Markov Model*, but we assume that the holding times $h_k = T_{k+1} - T_k$ are long

enough for learning. We will specify the exact assumptions later on.

The *base model* for an attacker with a type-II strategy is stated now. Because of measurement errors, the attacker may not always be successful in identifying the correct communication to attack. Let $p_1$ denote the probability of attacking the communication with the target characteristics. We will assume that the number of available channels is larger than $d$, and use $d$ of the SUs as learning probes, each with a different target property. Counting only the time slots when one of the probes is attacked, the total number of attacks to each of the probes within $n$ such time slots is modeled as a multinomial random variable with probabilities:

$$p_1(\theta) = \frac{\theta}{\theta + d - 1}; \quad p_i(\theta) = \frac{1}{\theta + d - 1}, \text{ for } i \in \{2, 3, \ldots, d\}, \tag{1}$$

where $\theta > 1$.

The above model corresponds to the situation where probe $k = 1$ is targeted and hit with probability $p_1 < 1$. Under error measurement, any other probe will be attacked with equal probability $p_i, i \neq 1$. The number $\theta = p_1/p_i$ provides the ratio between $p_1$ and the rest. For the base model, using the fact that all other probabilities are equal, $\theta = (d-1)p_1/(1-p_1)$.

Define the function:

$$\phi(\theta, n) = \sum_{y \in \mathcal{P}(n)} \frac{n!}{y_1!\, y_2!\, , \ldots y_d!} p_1(\theta)^{y_1} \left(\frac{p_1(\theta)}{\theta}\right)^{\sum_{i=2}^{d} y_i} \tag{2}$$

where the summation is over the set of all possible observations of a sample of size $n$ of the multinomial with parameters (eq. 1) where the first component dominates the others, that is:

$$\mathcal{P}(n) = \left\{ y \in \mathbb{N}^d : \sum_{k=1}^{d} y_i = n, \text{ and } y_1 \geq y_i; i \geq 2 \right\}. \tag{3}$$

It is straightforward to show that this is the exact probability of correct selection in a sample of size $n$ from the base model when the maximum likelihood estimator is used. Specifically, let $Y_i(n)$ count the number of attacks to probe $i$ under the base model, so that: $(Y_1(n), Y_2(n), \ldots Y_3(n)) \sim M(p(\theta), n)$, then the MLE for the parameter $p_i$ is simply $\hat{p}_i(n) = Y_i(n)/n$ and $\phi(\theta, n) = \mathbb{P}(Y_1(n) = \max(Y_1(n), \ldots, Y_d(n))$.

Let $\alpha \in (0, 1)$ be a confidence level for statistical significance. Then under the base model we can calculate the sample size required to ensure a probability of correct selection of at least $1 - \alpha$:

$$N^*(\theta, d) = \min\left(n : \ \phi(\theta, n) \geq 1 - \alpha\right). \tag{4}$$

Bechhofer *et al.*[47] have tabulated the function $\phi(\theta, n)$ for $d = 2, 3, 4$ using various values of $\theta$ and $n$. For example, if $d = 4$, then a sample size of $n = 25$ ensures a correct selection with level $\alpha = 0.200579$ when $\theta = 2$, and with level $\alpha = 0.038559$ when $\theta = 3$.

Suppose that honeynet correctly identifies a lure, but $p_1 < \xi^*$. Clearly, the best it can do here is to use its (correct) guess for the honeynode, but this will provide at most a probability $p_1$ that the honeynode will be attacked. Because $p_1$ is below the threshold, it will not be worth using honeynode in this case and we use special honeynet strategy similar to type-III. Thus, such values of $\xi^*$ provide a threshold value $\theta^* = \frac{(d-1)\,\xi^*}{(1-\xi^*)}$ below which it is not worth using honeynode.

**Definition:** We call a *naive* attacker one of type II where the probability of error in measurement is lower than $1 - \xi^*$, and we assume that $\mathbb{P}(h_k < N^*(\theta^*, d)) = 0$.

The above definition says that this type of attack is fairly accurate (usually $p_1 \gtrapprox .85$) and also that the strategy is kept long enough to learn the target probe. Specifically, because $\theta \geq \theta^*$ for a naive attacker, then $N^*(\theta, d) \leq N^*(\theta^*, d)$ if we use the MLE to identify the target with $\arg\max(Y_i(n))$ for $n \approx N^*(\theta, d)$.

### *E. Learning attacker's strategy*

When the learning mechanism starts, given a confidence level $\alpha$, the number $n = N^*(\theta^*, d)$ is calculated as a first estimate for an adequate sample size to detect type II attackers. When $d < \mathcal{C}$ it is possible that error in measurements results in false attacks to communications that have not been allocated any lures. Thus, we will focus only on time slots when attacks happen to lures. According to our model, this "sampled" process corresponds to the base model for attackers of type-II. Given $n$, define $\tau(n)$ as the total number of time slots required to see $n$ attacks to the lures.

During the learning phase, the $d$ different lures for type-II attacks are assigned to $d$ different communications among the available ones with uniform probability and no honeynode is yet allocated. Let $Y_i(0) = 0; i = 1, \ldots, d$ and define for each $i = 1, \ldots, d$ and the counting processes:

$$Y_i(k) = Y_i(k-1) + \mathbf{1}_{\{i\text{-th lure is attacked at time } k\}} \tag{5}$$

for $k = 1, 2 \ldots$, where the notation $\mathbf{1}_{\{A\}}$ stands for the indicator function of event $A$ (or Dirac delta). In parallel, define $C(1) = c$, if $c$ is the first channel to suffer an attack, and let

$$C(k+1) = C(k)\mathbf{1}_{\{\text{channel } c \text{ is attacked at time } k+1\}}. \tag{6}$$

Because we have allocated the lures randomly among SU communications, it follows that

$$\mathbb{P}(C(k) = 1 \,|\, \text{type II or III}) \leq \max\left\{ \left(\frac{1}{d}\right)^k, \left(\frac{1}{\mathcal{C}}\right)^k \right\} \tag{7}$$

Define $n_0$ as the smallest power that makes this probability smaller than our given confidence level $\alpha$, that is, when $d < \mathcal{C}$

$$n_0 = \lceil \log(1/\alpha) - \log(d) \rceil. \tag{8}$$

The number of tests to check for type I is thus typically very small. For example, if $\alpha = 0.001, d = 2$ then $n_0 = 7$, for $\alpha = 0.005$ and $d = 6$, $n_0 = 4$.

If $C(n_0) = c$, we declare having learned that the attack is of type I and we identify $c$ as the target channel. From this point onwards, we place the honeynet in this channel and keep monitoring. Because attacks of type I are not subject to error in identifying the channel, as soon as $C(k) = 0$ we declare a regime change and re-set the learning phase.

Otherwise, if $C(n_0) = 0$ then we keep assigning lures to channels for as many time slots are required to observe $n = N^*(\theta^*, d)$ attacks to lures. Gelfand *et al.*[48] provides a comparison between various estimators and confidence intervals for $\hat{p}_1$. In particular, his findings support the fact that under attacks of type II the approximate confidence interval based on the CLT is adequate, even for small to moderate sample sizes. Following this approximation, if

$$\hat{p}_1 - 1.96\sqrt{\frac{\hat{p}_1(1-\hat{p}_1)}{n}} \geq \xi^* \tag{9}$$

then we declare having learned that the strategy is of type II and we identify the lure. From this time onwards, we use the honeynode with that lure and start the monitoring phase. Notice that by construction, naive attacks are ensured to be correctly identified with probability at least $1 - \alpha$.

If (eq. 9) does not hold, then we do not have significant evidence that our candidate lure will be sufficiently effective. From this point onwards (whether the attacker is of type II but with large measurement errors, or of type III) we use the special honeynode allocation by delaying all other SUs. It is important to note that while the regular honeynode entails a delay for the chosen SU, the special strategy delays all of the rest of the SUs, albeit by a much smaller amount of time.

### F. Regime change detection: monitoring phase

Once the learning period is over, the corresponding honeynet strategy is used and honeynet keeps monitoring the attack counts, keeping track of running window averages. This is the monitoring phase where the honeynet is sensing for a possible change in attacker's strategy, as follows.

If the honeynet is under the assumption of a type I attack, then it keeps track of $C(k), k \geq n_0$ until the first time slot where $C(k) = 0$. Then it restarts the learning phase.

If the honeynet is under the assumption of a type II naive attacker then it uses sliding window averages to test for regime changes. During the monitoring phase the honeynet uses the detected lure for the honeynode allocation. Honeynet's first monitoring test uses a standard control chart for frequencies, and the second proposed method uses a regression for the slope of the frequency of attacks. Let $\hat{p}_1(k); k \geq n$ be as in (eq. 11) re-calculated with increasing observations beyond the initial horizon $n$ and call

$$L(k) = \hat{p}_1(k) - 3\sqrt{\frac{\hat{p}_1(k)(1-\hat{p}_1(k))}{w}}. \tag{10}$$

Given a window of size $w$ time slots, let $\tilde{\xi}_w(k)$ be the estimate of $p_1$ (and also of $\xi$) for time slot $k > n$ using the observations $(Y_{(d)}(k-w), \ldots, Y_{(d)}(k))$. As soon as $\tilde{\xi}_w(k) < L(k)$, the honeynet declares a change of regime and restarts the learning phase (resetting all counters).

The regression test works very similarly. (To complete, regression with the window and test for $H_0 : \beta < 0$, where $\beta$ is the slope or method of residuals).

Finally, if the honeynet is operating under the assumption of a type III attack or a type II attack for small $\xi$, then honeynode's current strategy is the special honeynet strategy that delays all but the honeynode. The honeynet keeps a new counter $H(k) = H(k-1) \times \mathbf{1}_{\{\text{honeynode is attacked}\}}$, initialized at the value 1. As soon as the attack goes to another channel ($H(k) = 0$), the honeynet declares that the attacker is not aiming at random, but it must be targeting now either a specific channel or a specific property of the transmissions. Then the honeynet restarts the learning phase.

## IV. Simulation and results

### A. Simulator

We coded a *tick based simulator* [49] using *Python* for simulating the CR-Honeynet. In the simulation, we have considered 20 SUs and 1 attacker which can effectively attack one SU communication. The CR-Honeynet dedicates 1 SU as honeynode in each slot. The attacker follows algorithm 1 and the honeynet follows algorithm 2. All SUs generate packets in accordance with *Poisson* process and queue them while in sensing period or when that SU is acting as a honeynode. During transmission period, SUs that are not acting as honeynode transmit packets from the queue. Packet transmission time ($S_n$) follows uniform distribution of 0.1 - 1.7 ms. A sensing Period ($T_s$) of 50 ms and a transmission Period ($T_t$) of 950 ms has been considered for the cognitive cycle. We consider attacker has target transmission characteristics ($d$) space as 4. From the CRN's point of view, attractiveness threshold ($\xi^*$) is considered as 0.6. Type-I learning horizon ($n_0$) and Type II learning horizon ($N^*(\theta^*, d)$) are calculated as 5 and 15 respectively. We run the simulation for 5,000,000 ms *simulation time* with 100,000 ms as *warm-up time* [1].

**Algorithm 1:** Algorithm for attacker

```
if strategy = attack particular channel then
    scan channel c ∈ C in the initial stage of T_t
    if SU is active on c then
        attack on channel c
else if strategy = attack transmission characteristics x then
    Scan all c_i ∈ C at initial stage of T_t
    attack the channel which have highest x
else if strategy = attack randomly then
    Scan all c_i ∈ C at initial stage of T_t
    attack randomly a channel c where SU is active
```

### B. Learning attacker's strategy

We plot $\phi(\theta^*, n)$ (eq. 2) with respect to learning period ($N^*$) in Figure 4. We can clearly see that with an increase in $N$, confidence level also increases. We have defined earlier, confidence level for statistical significance ($\alpha = 1 - \phi$). From (eq. 4) we can get the optimal $N^*$. For our simulation we have considered $\xi^* = 0.6$. We see that $N^* = 15$ ensures correct learning with level $\alpha = 0.015$ for $d = 4$. From the

[1]To obtain reliable steady state results for system staring with empty queue, a simulator run for *warm-up period* [49] without recording data. Once the warm period is over simulator starts gathering data

**Algorithm 2:** Algorithm for CR-Honeynet

Calculate $n_0, N^*$ based upon $d$ and $\xi^*$
Reset all counters such as $y, n$ etc.
Run initial learning phase for $n_0$ slots
**if** *all attack happens on channel* $c \in \mathcal{C}$ **then**
    Put honeynet on $c$ in every slot until attack observed on other channel.
    Go to step 2
**else**
    Continue counting for $n = N^*(\theta^*, d)$ slots
    **if** $\hat{p}_1 - 1.96\sqrt{\hat{p}_1(1-\hat{p}_1)/n} \geq \xi^*$ **then**
        $lure = \text{argmax}(y)$
        put honeynode with $lure$ on every slot until $\xi_w(k) < L(k)$
        Go to step 2
    **else**
        Use special honeynet strategy until honeynode is not attacked. Go to step 2.

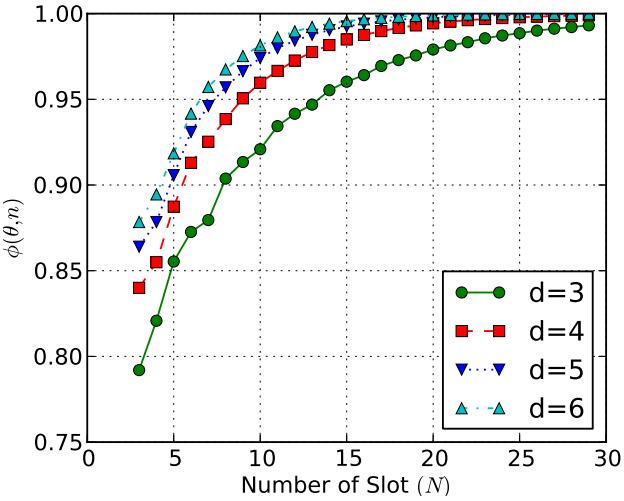



Fig. 4: Confidence level of Learning with pilot simulation

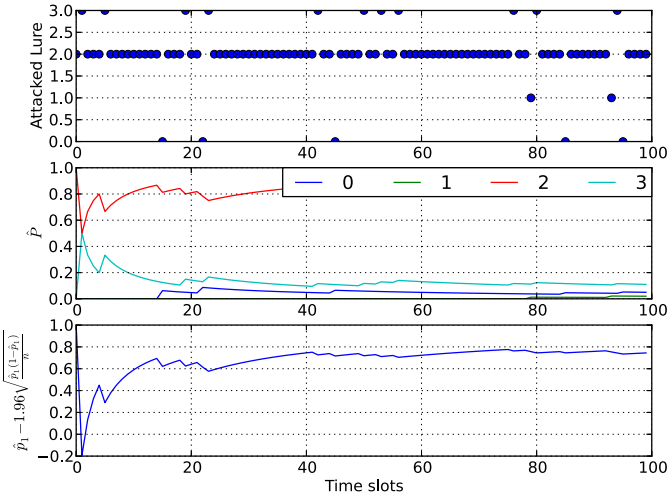



Fig. 5: Depiction of Learning while $d = 4$

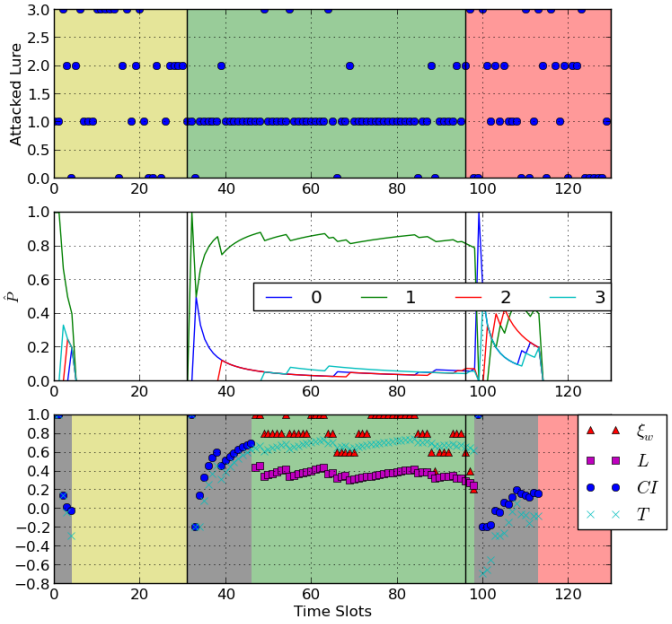



Fig. 6: Honeynet Learning phase corresponding to attacker's strategy change from I to II and then to III

figure we can conclude that for a certain desired confidence of learning $(\phi(\theta^*, n))$, an increase in the number of lured characteristics $(d)$, results in a decrease in required slots for learning $(N^*)$. In another way, the more transmission characteristics or combination of transmission characteristics an attacker can target, CR-honeynet takes lesser time to learn with the same confidence.

Figure 5 provides an illustration of a learning phase. In this scenario, the attacker with type II strategy is aiming for lure 2 (the lure is characteristics of transmission) to attack. Lure 2 is actually attacked with a probability 0.8. The actual attacks on the various lures are shown on the upper subplot. The middle subplot depicts $\hat{P}$ for the different lures calculated using (eq. 11). The third subplot provides $\hat{p}_1 - 1.96\sqrt{\frac{\hat{p}_1(1-\hat{p}_1)}{n}}$ which can be taken as a measure of learning. With $d = 4$ and $\xi^* = 0.6$, the probability of correct selection after $N^* = 15$ samples is 0.985.

### C. Dynamic evolution with change in attacker's strategy

Figure 6 and 7 provide the results for different experiments, each of which corresponds to a different sequence of attack processes $\{S_n; n = 1, 2, \ldots\}$. The upper subplots provide the attacker's strategy. *Yellow, green* and *red* colors indicate type-I, type-II and type-III attacking strategies respectively. Blue dots indicates the attacker's aimed transmission characteristics to find highest impacting communication. Here we have used 4 types of lure, i.e. transmission characteristics $(d = 4)$.

Middle subplots give CR-Honeynet's observation of $\hat{p}$ (eq. 11) for different lures. It uses the MLE estimators for the two highest probabilities:

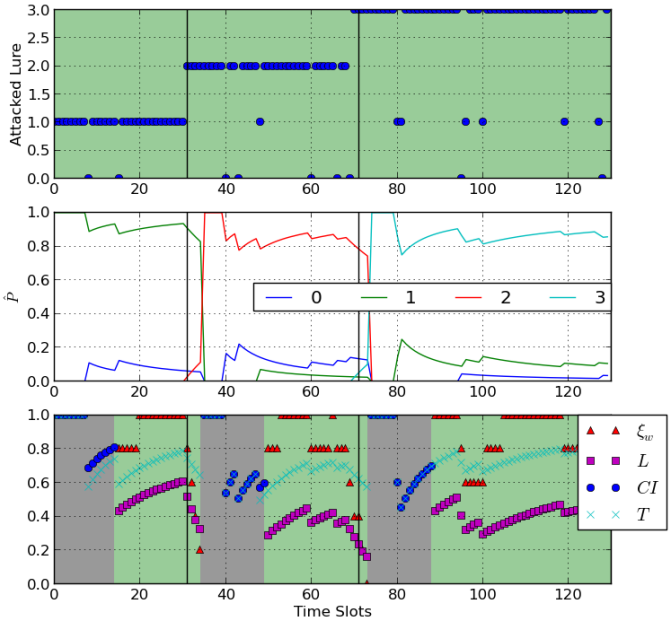



Fig. 7: Honeynet Learning phase corresponding to attacker of type II and attacker is changing its target lure

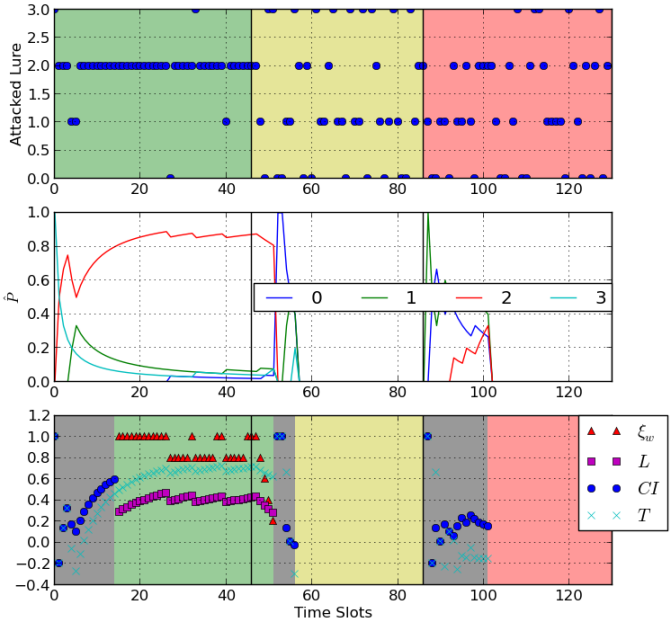



Fig. 8: Honeynet Learning phase corresponding to attacker's strategy change from II to I and then to III

$$\hat{p}_1 = \frac{Y_{(d)}(\tau(n))}{n}; \quad \hat{p}_2 = \frac{Y_{(d-1)}(\tau(n))}{n}, \tag{11}$$

where the notation $(x_{(1)}, \ldots, x_{(d)})$ is the usual notation for the ordered statistics.

In lower subplots of these 3 figures, we present phases of Honeynet. Background colors *Grey, yellow, green* and *red* indicate the learning phase, type-I, type-II and type-III defense strategies respectively. Then we plot the estimation of $CI = \hat{p}_1 - 1.96\sqrt{\frac{\hat{p}_1(1-\hat{p}_1)}{n}}$ in the learning phase. We can see that with increase in slots, $CI$ is increasing. When the learning phase is over and honeynet decides which strategy to take, it change its phase. When it detects a regime change, it enters to the learning phase again. When it is in type-II honeynet strategy, the honeynet monitors $L(k)$ and $\tilde{\xi}_w$. Honeynet enters learning phase when $\tilde{\xi}_w \leq L(k)$. An approximate test of level 0.05 which decides whether the attack is of type II or III is to test if $T > 0$, for the statistics:

$$T = (\hat{p}_1 - \hat{p}_2) - 1.96\sqrt{\frac{\hat{p}_1(1-\hat{p}_1) + \hat{p}_2(1-\hat{p}_2) - \hat{p}_1\,\hat{p}_2}{n}}. \tag{12}$$

If $T \leq 0$ then we infer, the attacks are "sufficiently random" between at least two contenders.

Figure 6 shows how the honeynet learns the change of strategy of attacker dynamically. We see that, for type-I attack, honeynet learns in 5 iterations. To distinguish between type II and III, honeynet takes 15 slots. When the attacker deviates from type I, honeynet learns it on the next iteration. However, when the attacker is in type II and changes its strategy, honeynet takes 2 iterations to detect the change in strategy of attack.

Figure 7 depicts a scenario where the attack strategy is of type II. It changes its targeted SU transmission characteristics

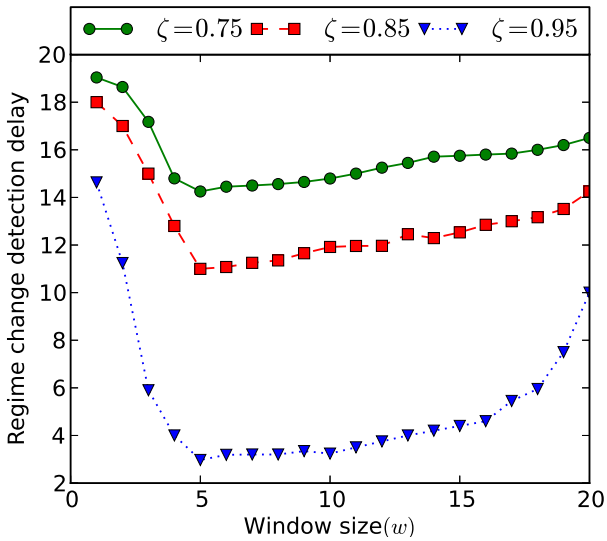



Fig. 9: Regime change detection delay for type II attacker

dynamically. For the first phase, attacker aims characteristics 1 and then 2 and then 3. We can see that for imperfect scanning, the attack may actually happen on a different lure. Honeynet identifies the correct strategy and particular type of attack in 15 iterations. We can see that for this particular simulation, honeynet detect attack strategy change after 3 iterations in the first case and after 2 iterations in the second one.

Figure 8 depicts a simulation scenario where attacker changes its type from II to I and then to III. Here, we can see that honeynet takes 5 iterations to learn that attacker has changed its strategy from type II to type I. From all 3 plots we can see that both $CI$ and $T$ gives indication of learning efficiency.

### D. Optimal window to detect regime change

When honeynode is placed with the wrong lure, legitimate communications are disrupted. We see honeynet detects the regime change very quickly, which decrease the loss. Mainly, the loss is during the learning phase when CR-Honeynet does not deploy honeynode. To see how long it takes for the honeynet to detect the regime change while in type II lure strategy, we present a comparison to select optimal window size in fig. 9. An attacker of type-II strategy is attacking a particular lure with probability $\zeta$. We simulated for 3 different values of $\zeta$. For every value of $\zeta$ and $w$, the simulation is run for 100,000 slots to ensure accurate results. In this simulation, the attacker is changing its targeted transmission characteristics randomly with mean interval of 100 steps. We can clearly see that, window size $w = 5$ provide optimal result i.e. it can detect regime change very quickly and efficiently.

### E. Overall system performance

We now code an *Event Driven Simulator* to compare the system performance between using honeynet and not using honeynet for an infinite buffer CRN. For simplicity, we have considered 20 SUs and kept $\xi = 0.8$. We vary average packet inter-arrival time $(\lambda)$ to examine system performance with varying load. We observe that for all values of $\lambda$, with CR-Honeynet the average packet dropping probability is 0.01, while without honeynet results packet dropping probability of 0.05 . Figure 10a provides the comparison of average queuing delay for a SU. From the figure, we can conclude that, using honeynet for lower $\lambda$ is highly beneficial as packet drop is minimized. Better packet delivery ratio is achieved at the cost of higher packet delay. In our future work we shall try to get an estimation of $\xi^*$ that can regulate CRN to use honeynode or not, depending $\lambda$ and traffic type (elastic, non-elastic, real-time etc.)

### F. Effect of SU sensing error

In CR-Honeynet architecture, multiple SUs transmit over multiple channels simultaneously during the transmission cycle. After the transmission cycle is over, the defender networks determine which channel is jammed based on the number of dropped packets. If a channel observes packet drop during the transmission cycle while the signal strength on that channel being high, that channel is flagged to be jammed. Thus all the jamming effects are successfully determined. Packet drop can be the result of two incidents: 1) that channel is intentionally jammed by an attacker or 2) PU of that channel started transmitting during the transmission cycle. Also if an SU erroneously detects a channel to be free while the PU is present, all the packets will be dropped as well. If jamming is detected on multiple channels, the defender needs to decide which one is due to the attacker and what kind of bait it attacked. If we recognize jamming on the presumed decoy bait that the CR-Honeynet deployed, the defender assumes it as a success. However, if the attack did not happen on the bait channel, the defender counts it as the attack happening on another SU other than the bait. For a defender network with fewer SUs, the defender is confused most of the time and restarts the learning phase quickly. For a network with more SUs, there is a lower number of such confusions, and as a result, the effect of sensing error reduces.

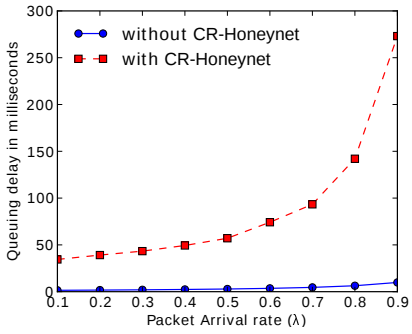
We simulated the scenario of PU sensing error with varying number of SUs. Figure 10b provides the obtained throughput regarding the number of packets successfully transmitted. We can see if the SUs can sense PU presence without any error, the throughput is very high. Note that, with an increase in the number of SUs, the throughput increases but it does not increase linearly as there will always be packet drop due to sudden PU arrival during the transmission phase. In addition to this, SU throughput is also limited by the frequent sensing phase as well as an SU's service as a honeynode. When SUs incur error in sensing the PU presence, the SU's transmission will be hampered for the entire transmission cycle. So, the effective throughput is reduced. We can see that for a lower number of SUs, they are massively affected by this error. However, when the number of SUs increases, the effect of the PU sensing error reduces. Thus for a defender network with a higher number of SUs, it is always beneficial to use CR-Honeynet.

### G. Comparison with state-of-the art protocols

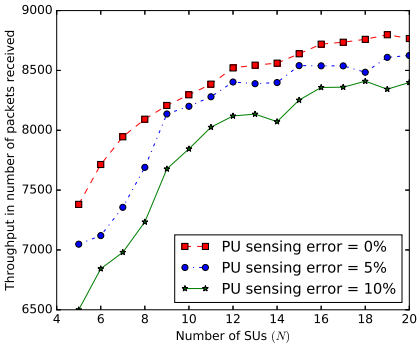
In this section, we compare the performance of CR-Honeynet with the state-of-the-art channel hopping based jamming defense techniques namely Tri-CH [42]. The simulation parameters are kept same as described in the earlier section. We have used only one adaptive jammer. Figure 10c provides the actual throughput obtained by a CRN. In case of Tri-CH, the SUs have to keep hopping channels until it is finally able to synchronize with its receiver. On the contrary, CR-Honeynet does not hop channels during a transmission period. It is true that some packets are lost since an SU is jammed but it is unable to switch the channel. Overall, the CR-Honeynet obtains higher throughput than that of TRi-CH.

## V. System development

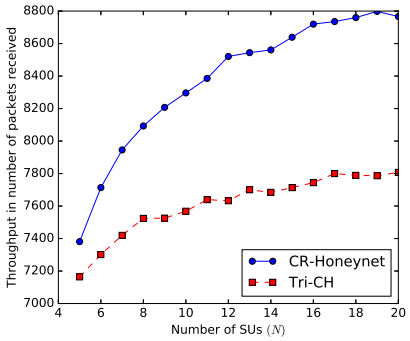
In this section, we describe the system architecture of the prototype and present the experimental evaluation. Figure 11 illustrates of our testbed, and for reader's convenience, a video demonstration depicting the prototype functionality is presented in [50]. We designed the testbed with multiple SUs, a central controller (CC) which would get data from all the SUs, and an intelligent attacker. The prototype is built using USRP radios [23], [51], each connected with one laptop. The spectrum usage is visualized using a spectrum analyzer. This setup uses one transmitter for each SU to communicate with the CC and one of the SUs act as the honeynode in each slot. The CC has a jamming detector, and the SU has a controller that would allow it to change transmission characteristics at the beginning of a time slot. Two different attackers are designed as well. The intelligent attacker listens to ongoing SU transmissions in the network and jam one channel based on a target characteristic. The other is a much more simple which is manually configured to attack specific channels we choose. Next, we will discuss how each component works,




(a) Overall queuing delay for $\xi = 0.8$




(b) Overall throughput($\xi = 0.9$)




(c) Overall throughput($\xi = 0.9$)

Fig. 10: Simulation results for $\mathcal{N} = 20$

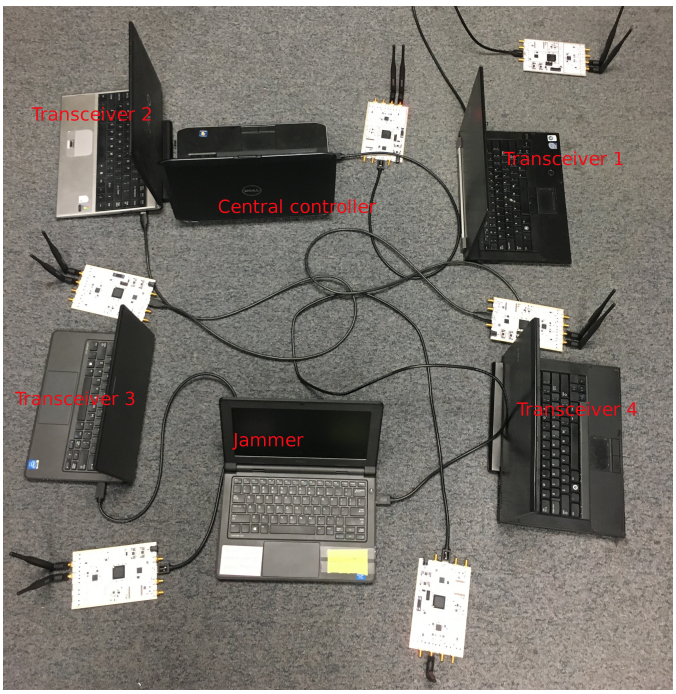



Fig. 11: Prototype setup for 4 SU with a central controller.

what it does, how it does it, and finally discuss the evaluation of the prototype.

### A. Central controller (CC) design in GNURadio

The CC in this system is designed first to work with two channels, and then it is updated to work with four. In this setup, one of the SUs is designated as the honeynode. CC monitors the incoming data and the signal strengths for each channel separately. If it detects an anomaly for a channel (such as data is not decoded while signal strength is above a threshold), it raises a jamming flag. If CC raises a flag for SU other than the honeynode, then it triggers channel switching between the jammed SU and the honeynode so that the SU will now be back on a unjammed channel. Through a backup common control channel, the message is relayed to the SU so that they can use proper frequencies. If the swapped SU still cannot communicate for a significant time, then the CC will assume the SU is no longer transmitting and swap it with the honeynode again and leave that SU alone until it starts receiving data from it. Swapping back and forth like this also makes the system a little more interference resistant if interference between channels is not allowing data to get through on a channel. Now we shall describe the design of CC in GNURadio.

TABLE I: Channel parameters Parameters

| System parameters | Variable names | Default values |
|---|---|---|
| Center Frequency | freq | 2.44GHz |
| Number of Channels | num_c | 4 |
| Channel Bandwidth | c_width | 200KHz |
| Band Pass Filter Low Cutoff | low_cutoff | -70KHz |
| Band Pass Filter High Cutoff | high_cutoff | 70KHz |
| Sample Rate | samp_rate | 800KHz |
| Guard Band | guard_band | 30KHz |

*1) GNURadio blocks for CC:* For the initial two SU setup, the blocks used are a USRP Source, a Message Strobe, a Jamming Detection/Defense, and two each of the following: Frequency Xlating FIR Filters, GFSK Demods, Packet Decoders, File Sinks, Byte Sensors, and UDP Sinks. For the updated four SU, the blocks used are a USRP Source, a Message Strobe, a Jamming Detection/Defense and four of each of the following: Frequency Xlating FIR Filters, GFSK Demods, Packet Decoders, File Sinks, Byte Sensors, and UDP Sinks.

*2) CC design parameters:* There are seven parameters being used in the CC design as can be seen in Table I. The center frequency designates which frequency for the USRP source to listen to. The number of channels is used in multiple calculations including to determine the overall sampling rate of the CC. The channel bandwidth is set to designate how wide each channel is and is used to calculate the overall sampling rate as well as the center frequency of each channel. The low and high bandpass filter cutoffs are used by the Frequency Xlating FIR Filters to provide guard bands for the channels. The Sample Rate is the overall sampling rate of the entire flowgraph and is set by the number of channels multiplied by the channel width. The guard band is used to help determine the high and low bandpass filter cutoffs.

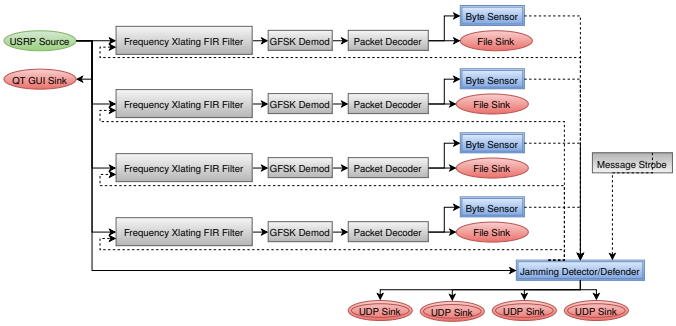



Fig. 12: GNURadio Flowgraph of the central controller

*3) GNURadio flowgraph for CC design:* Figure 12 depicts a flowgraph of the CC designed in GNURadio for static scenario. Note that when we introduce dynamic change in transmission characteristics, we need to code everything in Python as the static flowgraph cannot change dynamic parameter changes for USRPs. The data is received from the USRP Source as a wideband transmission that is then filtered and shifted by the Frequency Xlating FIR Filters for each channel in the network. This shift and filter isolate a single SU's data to pass through GFSK demodulation. Once demodulation is complete, the data gets passed to a Packet Decoder. At this point, there is no more processing to be done to the data so it gets sent to the Byte Sensor and the File Sink. When the Byte Sensor receives bytes it uses the GNURadio message passing system to let the Jamming Detect/Defense block know how many bytes it received with a timestamp. As long as the Byte sensors for the SUs are receiving bytes, the Jamming Detector/Defense block doesn't do anything. If it stops receiving bytes for a certain amount of time, raises a jamming flag. The UDP Sinks are used as a control channel between the CC and SUs.

*4) Custom Blocks:* The two custom blocks that we built for the CC are the Byte Sensing block and the Jamming Detection/Defense block. The Byte Sensing block is simple in design. It has one input which accepted bytes from the packet decoder and one message output that connected to the Jamming Detection/Defense. Since the *work*[2], the block will only work when the Packet Decoder successfully decodes bytes of data. When it does receive bytes, the block uses GNURadio's message passing system to let the Defender block know what time that channel received bytes and which SU this sensor is connected to.

We also had to build the Jamming Detector/Defender. This block takes GNURadio message input probes as input from the Byte sensor blocks for each SUs. It also has one 64-bit complex input taken directly from the USRP source. This block outputs a message output for each SUs Frequency Xlating Fir Filter and a 32-bit floating point output for each SU's UDP sink. The block contains many different variables to store data about each SU including the time of the last known byte, whether the SU is jammed, which frequency each SU is on, and more. The block loops through each SU in its list

[2]GNURadio has a function, *work* which is called when data is received at the input of a block.

TABLE II: SU Parameters

| SU parameter | Variable name | Default value |
|---|---|---|
| Center Frequency | freq | 2.44GH |
| Sample Rate | samp_rate | 800KH |
| Guard Band | guard_band | 30KH |
| Gain | gain | Varies by SU |
| Payload Size | payload | 1024 |

first checking to see if the SU is the honeynode. If it is not the honeynode then it checks how long any bytes have been received from that SU. If it has not received any byte for a threshold time the defender assumes the SU is jammed and swaps its frequency with that of the honeynode in an attempt to unjam the SU. Then if that SU can not communicate for a longer period, the CC assumes it is off and swaps it with the honeynode again. For the defender to swap frequencies, it creates a message with the new frequency and send it to the Frequency Xlating Fir Filter corresponding to that SU and also sends the new frequency through the floating point output of the corresponding SU so it also knows to change. This message passing through the common control channel ensures proper rendezvous between the SU and the CC.

*5) CC Limitations:* The most profound limitation in the CC is the interference that is sometimes created by the SUs as they move between channels. This interference can sometimes cause an SU to stay jammed even after it swaps. A temporary solution has been found in the form of controlling the gain of the SU and by having the frequency of the SU swap with the honeynode again if it persists. This can sometimes put the SU back into a favorable position where it can get its data through.

### *B. Secondary User (SU) design in GNURadio*

The SU for this system is designed to get bytecode of a file, encode, modulate, and filter the data, and then send the data through the USRP. It is also built to be able to change its channel of communication dynamically by the transmission control block. CC sends control messages to the transmission control blocks of the SUs through UDP to change the channel of communications.

*1) GNURadio blocks for SU design:* Each SU includes a UDP Source, File Source, transmitter Controller, Packet Encoder, Band Pass Filter, Multiply Constant, QT GUI Sink, USRP Sink, and QT GUI Range.

*2) Parameters for SU design:* Table II lists the parameters used by the SUs. The center frequency is different for each SU and is the frequency that the USRP is transmitting on. The sample rate is the bandwidth of the channel. The guard band determines the high and low cutoffs for the bandpass filter. The gain is used to set the gain of the USRP which can be controlled by the user at runtime for testing purposes. The payload size is used in the UDP source to determine the size of the UDP packets being sent by the CC.

*3) GNURadio flowgraph for an SU:* Figure 13 shows the flowgraph of an SU. The file source block loads the bytecode of the data to be transmitted which is then passed to the packet encoder where it is encoded for GFSK modulation. The baseband signal is passed through a bandpass filter to get

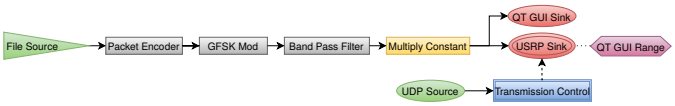



Fig. 13: GNURadio flowgraph of the SU design

rid of any undesired frequencies. Next the data passes through a multiply block which is used to help fine tune the signal. Then it is sent to the USRP for radio transmission. Each SU also contains a UDP source block which is connected to the UDP sink blocks located within the CC through the common control channel. This data is sent to a transmission control block we built to change to a new frequency determined by the CC.

*4) Custom GNURadio blocks for an SU:* The transmission Control block is designed to take data from a UDP Source. This data is the new frequency for the SU to switch to if the CC detects jamming. When it receives this data, it uses the GNURadio message passing system to send a message to the USRP to set the new frequency.

*5) SU limitations:* The limitation is changing frequencies depends on the connection between the UDP-source SU and corresponding UDP-sink blocks in the CC retaining their connection. Also when the SU is changing frequency, sometimes its data will go into a different SUs file sink because it didn't switch fast enough.

### C. Intelligent attacker design in GNURadio

We design an intelligent attacker with the capability of sensing the wireless channels and then attacking a channel based on a target characteristic. The two most prominent characteristics we tested against are high transmission power and longest transmission time. In mode11 the attacker would sense and attack the channel with the highest transmission power. In mode-2, the attacker jams the channel that had been transmitting the longest during the sensing time.

*1) GNURadio blocks:* To build the intelligent attacker we had to use a USRP Source, Stream to Vector, FFT, Vector To Stream, Jammer, and USRP Sink.

*2) Parameters for attacker:* The parameters for intelligent attacker design are listed in Table III. The center frequency sets the frequency that the USRP Source is listening to, but it does not fix the frequency of the USRP sink. This is because the USRP Source will always listen to 2.44e9GHz whereas the USRP Sink's frequency will change each time the attacker targets a new channel. The number of channels is set so that the jammer knows how many channels it needs to account for in the FFT data. The Sampling rate is the overall sampling rate of the network. The guard band is used to set the high and low cutoffs for the bandpass filter. This is needed because we are testing the system with the attacker only being able to attack a single channel at a time.

*3) GNURadio flowgraph for intelligent attacker:* The flowgraph for this attacker can be seen in figure 14. This attacker takes data from a USRP Source that is set to the same frequency as the CC's network. This data is output as a stream which is then turned to a vector of size 1024 by the stream to

TABLE III: Intelligent attacker design parameters

| Parameter | Variable Name | Default Value |
|---|---|---|
| Center Frequency | freq | 2.44GH |
| Number of Channels | num_c | 4 |
| Channel Bandwidth | c_width | 200KH |
| Sample Rate | samp_rate | 800KH |
| Guard Band | guard_band | 30KH |


Fig. 14: Flowgraph of an intelligent attacker.

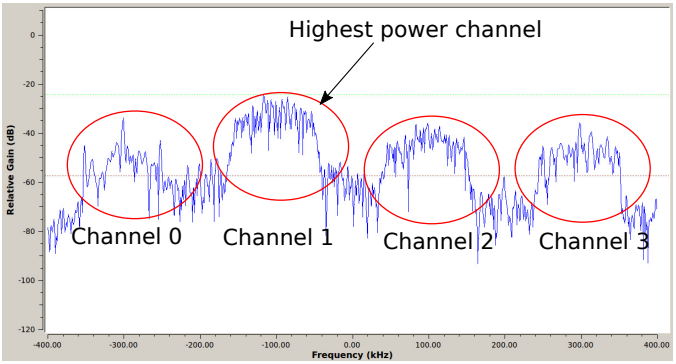



Fig. 15: FFT plot of the signal observed by an attacker. If intelligent attacker targets highest power channel then the SU transmitting on channel 1 sould be attacked.

vector block. This newly created vector is passed to an FFT block which performs the transformation and then passes the data to the Jammer/Sensor. After analyzing the data for a set amount of time, the Jammer will then beginning outputting a jamming signal on a specific channel based on the attack mode of the jammer. This jamming signal is passed through a bandpass filter and then sent to the USRP Sink for RF transmission.

*4) Custom GNURadio blocks for attacker:* Only the jammer/sensor block is created for the intelligent attacker. This block has one input that accepts 64-bit complex vectors of size 1024. It also has an output of a stream of 64-bit complex numbers and a message output that is used to change the frequency of the USRP. The block starts by taking in a vector which is the FFT data of the network if it is in the sensing mode. Then it splits the data based on the number of channels present in the network. Each segment represents the data on a specific channel. This the block loops through the segments and checks the average of that segment against a threshold. If the average is greater than that threshold then the block will update the information it has for that specific channel. Namely the strength of the transmission and how long it has been on. If the block is in jamming mode then it will use the data it gathered from sensing and choose the channel which matches it's search criteria. Then it outputs the jamming signal on target channel. The block will jam that channel for a set amount of time before it goes back into sensing mode and it repeats the process. In mode-1, the intelligent attacker will target the SU with the highest transmission power and can be seen in Figure 15. In mode-2 the intelligent attacker will target the SU that has the longest transmission time as can be seen

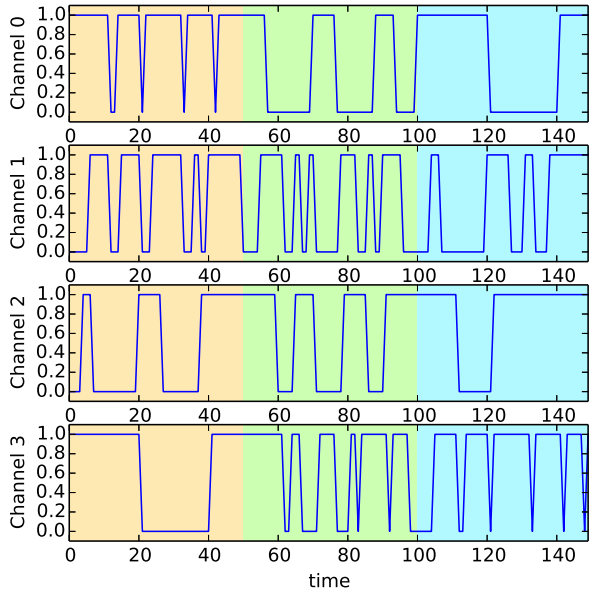



Fig. 16: Actual plot of transmission time on different channels. If intelligent attacker targets channel with maximum transmission, in the first time slot channel 0 will be targeted, in the second time slot channel 3 will be targeted, and in the third time slot channel 2 will be targeted.

in Figure 16.

*5) Attacker's limitations:* The only limitation of this attacker is noise in the environment can give a false positive that the channel is on. Although this isn't much of an issue because the noise only registers for a very short amount of time and the power is usually really low. This means it usually won't change much regarding the Jammer's sensing time.

### D. Manual Attacker Design

The manual attacker is designed mostly as a testing and debugging tool. It enabled us to test different scenarios while initially designing the system and it also retained its usefulness after the system is built to test specific jamming patterns.

*1) GNURadio blocks for attacker design:* The manual attacker had by far the simplest flowgraph only using a Signal Source, Band Pass Filter, USRP Sink, QT GUI Sink, and two QT GUI Ranges.

*2) Parameters for an attacker:* There are only three parameters used by the manual attacker as can be seen in Table IV. The center frequency is the frequency that the USRP is transmitting on and can be changed at runtime by the user to transmit and jam specific frequencies. The sample rate is the bandwidth of the channel. The guard band is used to determine the high and low-frequency cutoffs for the bandpass filter. The gain is used to set the transmission power of the USRP and can also be changed at runtime to change the strength of the jamming.

*3) GNURadio flowgraph of manual attacker:* The flowgraph is depicted in Figure 17. It starts with a signal source

TABLE IV: Manual Attacker Parameters

| Parameter | Variable Name | Default Value |
|---|---|---|
| Center Frequency | freq | 2.4397GH |
| Sample Rate | samp_rate | 800KH |
| Guard Band | guard_band | 30KH |
| Gain | gain | Varies by SU |

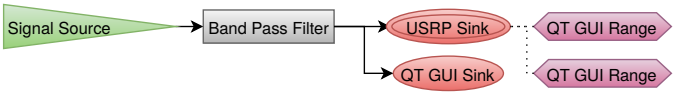



Fig. 17: Flowgraph of the manual attacker that can be changed at runtime to attack a specific channel

that outputs a cosine wave. That signal is then sent to a bandpass filter to remove any undesired frequencies in the signal. After filtering the signal, it is sent to the USRP for transmission. While running the user can change the gain and center frequency of the USRP to jam specific channels are different strengths.

*4) Limitations of manual attacker:* The only limitation of the manual attacker is that it must be manually moved to attack different channels. On its own, it contains no sensing or detection capability. This attacker represents one that just chooses a channel to attack and then jams it.

### E. Experiments with prototype

*1) Experiments with two SUs:* Experimenting with this system is with just two SUs and one CC. Both SUs have a file to transmit, but one of them is designated the honeynode. In this setup, when the CC detects jamming on a non-honeynode SU, it would swap the frequencies without a problem and the data would continue to come through. We test this with both the manual attacker as well as the intelligent attacker and in both cases when the CC detected jamming the channels would swap. The only issue encountered with this is the cross-channel interference between the two SUs. Although most of the time it would not be much of an issue, sometimes it would cause no data to come through the non-honeynode SU at all making the CC think it is no longer transmitting. This stage is just for testing the initial design, and since it is working, we moved on to the next part.

*2) Experiments with four SUs:* Once we got the system working for two SUs, we moved on to four SUs. In this case three of them would be actively transmitting, and the fourth would be the honeynode. Interference is a much bigger issue in this experiment because there are now two extra SUs to deal with. To help minimize interference, we increased the size of the guard bands and updated the defender to swap the jammed SU with the honeynode. This extra swap would help get some extra bytes from that SU while each one tried to stabilize in its new frequency with its new neighbors. With the issue of interference partially improved, we could begin testing the system with the attackers.

When testing with the manual attacker, the defender would place the honeynode where the attacker is and only move if some interference made another channel seem like it is being jammed. As the manual attacker is moved to another

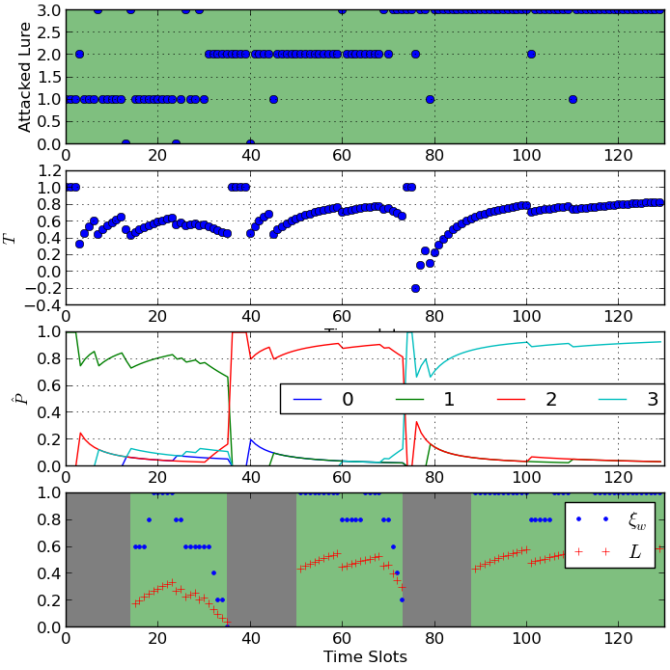



Fig. 18: Experiment result with 4 lures characteristics

frequency, the honeynode would follow it. After numerous tests with the manual attacker, we then move on to testing with the intelligent attacker. When the intelligent attacker is introduced to the system, it would target either the SU with the highest transmission power or the longest transmission time. When the defender detected the presence of jamming on one of those channels, it would move the honeynode to the jammed frequency until it detected another being jammed.

### *F. Experiment results*

In the testbed, we have deployed an intelligent attacker (i.e. type-II). We have set four sets of characteristics based on transmission power, packet arrival rate, packet length and packet inter-arrival gap. Now, the attacker chooses target characteristics and scan through the channel to detect the target channel. It also changes its strategy dynamically. Figure 7 depicts the experiment results. The attacker chose the lure 1 and kept attaching to that characteristics. The defender places a honeynode at the interval 15. The attacker changes its strategy at interval 35 and again in 70. The honeynet's learning period is colored gray. The green color means the defender is using an active decoy. The results can be compared with the results of the simulation described earlier and clearly support the effectiveness of the CR-Honeynet.

## VI. Conclusions and Future Works

In this paper, we propose CR-Honeynet, a CRN sustenance mechanism, which exploits the fact that an intelligent and rational attacker aims for specific transmission characteristics to gain the highest impact out of jamming. The stochastic learning model presented in the paper shows that the honeynet can confidently learn the attacker's strategy and dynamically evolve with attacker's strategy change. The mechanism efficiently lures the attacker towards attacking the active decoy trap and thus bypassing attacks on legitimate SU communications. The state-of-the-art testbed developed using off-the-shelf software defined radios prove the effectiveness of the mechanism. Currently, the mechanism has a drawback of not placing active decoy while it is passively learning attacker's strategy. In the future, we shall investigate more to improve the learning mechanism where the honeynet would be able to predict the attacker's strategy change and can place an active decoy to mitigate an attack.

## References


[1] S. Bhunia, S. Sengupta, and F. Vazquez-Abad, "CR-Honeynet: A Learning & Decoy Based Sustenance Mechanism against Jamming Attack in CRN," in *IEEE MILCOM*, pp. 1173–1180, 2014.

[2] A. Fragkiadakis, E. Tragos, and I. Askoxylakis, "A survey on security threats and detection techniques in cognitive radio networks," *IEEE Communications Surveys Tutorials*, vol. 15, no. 1, pp. 428–445, 2013.

[3] S. Bhattacharjee, S. Sengupta, and M. Chatterjee, "Vulnerabilities in cognitive radio networks: A survey," *Computer Communications*, vol. 36, no. 13, pp. 1387–1398, 2013.

[4] T. C. Clancy and N. Goergen, "Security in cognitive radio networks: Threats and mitigation," in *IEEE CrownCom*, 2008.

[5] T. X. Brown and A. Sethi, "Potential cognitive radio denial-of-service vulnerabilities and protection countermeasures: A multi-dimensional analysis and assessment," *Mobile Networks and Applications*, vol. 13, no. 5, pp. 516–532, 2008.

[6] D. Thuente and M. Acharya, "Intelligent jamming in wireless networks with applications to 802.11 b and other networks," in *MILCOM*, 2006.

[7] S. Anand, S. Sengupta, K. Hong, K. Subbalakshmi, R. Chandramouli, and H. Cam, "Exploiting channel fragmentation and aggregation/ bonding to create security vulnerabilities," *IEEE Transactions on Vehicular Technology*, 2014.

[8] K. Pelechrinis, M. Iliofotou, and S. V. Krishnamurthy, "Denial of service attacks in wireless networks: The case of jammers," *Communications Surveys & Tutorials, IEEE*, vol. 13, no. 2, pp. 245–257, 2011.

[9] B. Wang, Y. Wu, K. R. Liu, and T. C. Clancy, "An anti-jamming stochastic game for cognitive radio networks," *Selected Areas in Communications, IEEE Journal on*, vol. 29, no. 4, pp. 877–889, 2011.

[10] C. Karlof and D. Wagner, "Secure routing in wireless sensor networks: Attacks and countermeasures," *Ad hoc networks*, vol. 1, no. 2, pp. 293–315, 2003.

[11] "GR-Inspector." https://github.com/gnuradio/gr-inspector.

[12] C. G. Wheeler and D. R. Reising, "Assessment of the Impact of CFO on RF-DNA Fingerprint Classification Performance," in *IEEE ICNC*, 2017.

[13] T. J. Bihl, K. W. Bauer, and M. A. Temple, "Feature selection for rf fingerprinting with multiple discriminant analysis and using zigbee device emissions," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 8, 2016.

[14] R. Klein, M. A. Temple, M. J. Mendenhall, and D. R. Reising, "Sensitivity analysis of burst detection and rf fingerprinting classification performance," in *IEEE ICC*, 2009.

[15] D. R. Reising, M. A. Temple, and J. A. Jackson, "Authorized and rogue device discrimination using dimensionally reduced rf-dna fingerprints," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 6, pp. 1180–1192, 2015.

[16] P. K. Harmer, D. R. Reising, and M. A. Temple, "Classifier selection for physical layer security augmentation in cognitive radio networks," in *IEEE ICC*, 2013.

[17] Z. Zhan, M. Xu, and S. Xu, "Characterizing honeypot-captured cyber attacks: Statistical framework and case study," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 11, pp. 1775–1789, 2013.

[18] S. Misra, S. K. Dhurandher, A. Rayankula, and D. Agrawal, "Using honeynodes for defense against jamming attacks in wireless infrastructure-based networks," *Computers & electrical engineering*, vol. 36, no. 2, pp. 367–382, 2010.

[19] W. Xu, T. Wood, W. Trappe, and Y. Zhang, "Channel surfing and spatial retreats: defenses against wireless denial of service," in *Proceedings of the 3rd ACM workshop on Wireless security*, pp. 80–89, ACM, 2004.

[20] G. Noubir and G. Lin, "Low-power DoS attacks in data wireless LANs and countermeasures," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 7, no. 3, pp. 29–30, 2003.

[21] "Wi-Spy Spectrum analyzer." http://www.metageek.net/products/wi-spy/.

[22] "GNU Radio." http://gnuradio.org/redmine/projects/gnuradio/wiki.

[23] "Universal Software Radio Peripheral (USRP) Kit." https://www.ettus.com/product/details/UN200-KIT.

[24] V. Chatzigiannakis, G. Androulidakis, K. Pelechrinis, S. Papavassiliou, and V. Maglaris, "Data fusion algorithms for network anomaly detection: classification and evaluation," in *ICNS*, pp. 50–50, IEEE, 2007.

[25] C. Sorrells, L. Qian, and H. Li, "Quickest detection of denial-of-service attacks in cognitive wireless networks," in *Homeland Security (HST), 2012 IEEE Conference on Technologies for*, pp. 580–584, IEEE, 2012.

[26] M. Spuhler, D. Giustiniano, V. Lenders, M. Wilhelm, and J. B. Schmitt, "Detection of reactive jamming in DSSS-based wireless communications," *IEEE Transactions on Wireless Communications*, vol. 13, no. 3, 2014.

[27] S. Bhunia, V. Behzadan, P. A. Regis, and S. Sengupta, "Adaptive Beam Nulling in Multihop Ad Hoc Networks Against a Jammer in Motion," *Computer Networks*, vol. 109, pp. 50–66, 2016.

[28] S. Bhunia, V. Behzadan, P. A. Regis, and S. Sengupta, "Performance of Adaptive Beam Nulling in Multihop Ad-Hoc Networks under Jamming," in *IEEE CSS*, 2015.

[29] S. Bhunia and S. Sengupta, "Distributed Adaptive Beam Nulling to Mitigate Jamming in 3D UAV Mesh Networks," in *IEEE ICNC*, 2017.

[30] S. Bhunia, P. A. Regis, and S. Sengupta, "Distributed adaptive beam nulling to survive against jamming in 3d uav mesh networks," *Computer Networks*, 2018.

[31] W. Xu, K. Ma, W. Trappe, and Y. Zhang, "Jamming sensor networks: attack and defense strategies," *IEEE Network*, no. 3, pp. 41–47, 2006.

[32] C. Sorrells, P. Potier, L. Qian, and X. Li, "Anomalous spectrum usage attack detection in cognitive radio wireless networks," in *IEEE HST*, 2011.

[33] C. Popper, M. Strasser, and S. Capkun, "Anti-jamming broadcast communication using uncoordinated spread spectrum techniques," *Selected Areas in Communications, IEEE Journal on*, vol. 28, no. 5, pp. 703–715, 2010.

[34] R. El-Bardan, V. S. S. S. Nadendla, S. Brahma, and P. K. Varshney, "On ARQ-based wireless communication systems in the presence of a strategic jammer," in *IEEE GlobalSIP*, 2014.

[35] R. El-Bardan, S. Brahma, and P. K. Varshney, "Strategic power allocation with incomplete information in the presence of a jammer," *IEEE Transactions on Communications*, vol. 64, no. 8, pp. 3467–3479, 2016.

[36] R. El-Bardan, V. Sharma, and P. K. Varshney, "Learning equilibria for power allocation games in cognitive radio networks with a jammer," in *IEEE GlobalSIP*, 2016.

[37] S. Singh and A. Trivedi, "Anti-jamming in cognitive radio networks using reinforcement learning algorithms," in *Wireless and Optical Communications Networks (WOCN)*, 2012.

[38] S. Mneimneh, S. Bhunia, F. Vázquez-Abad, and S. Sengupta, "A game-theoretic and stochastic survivability mechanism against induced attacks in cognitive radio networks," *Pervasive and Mobile Computing*, 2017.

[39] C. Chen, M. Song, C. Xin, and J. Backens, "A game-theoretical anti-jamming scheme for cognitive radio networks," *IEEE Network*, vol. 27, no. 3, pp. 22–27, 2013.

[40] Y. Gwon, S. Dastangoo, C. Fossa, and H. Kung, "Fast Online Learning of Antijamming and Jamming Strategies," in *IEEE GLOBECOM*, 2015.

[41] V. Navda, A. Bohra, S. Ganguly, and D. Rubenstein, "Using channel hopping to increase 802.11 resilience to jamming attacks," in *IEEE INFOCOM*, 2007.

[42] G.-Y. Chang, S.-Y. Wang, and Y.-X. Liu, "A jamming-resistant channel hopping scheme for cognitive radio networks," *IEEE Transactions on Wireless Communications*, 2017.

[43] S. Bhunia, S. Sengupta, and F. Vázquez-Abad, "Performance analysis of CR-honeynet to prevent jamming attack through stochastic modeling," *Pervasive and Mobile Computing*, vol. 21, pp. 133–149, 2015.

[44] S. Bhunia, X. Su, S. Sengupta, and F. Vázquez-Abad, "Stochastic Model for Cognitive Radio Networks under Jamming Attacks and Honeypot-Based Prevention," in *ICDCN*, Springer Berlin Heidelberg, 2014.

[45] C. Cano and D. J. Leith, "Coexistence of wifi and lte in unlicensed bands: A proportional fair allocation scheme," in *Communication Workshop (ICCW), 2015 IEEE International Conference on*, pp. 2288–2293, IEEE, 2015.

[46] FCC, "Jammer Enforcement." https://www.fcc.gov/general/jammer-enforcement.

[47] R. E. Bechhofer, S. Elmaghraby, and N. Morse, "A single-sample multiple-decision procedure for selecting the multinomial event which has the highest probability," *The Annals of Mathematical Statistics*, pp. 102–119, 1959.

[48] A. Gelfand, J. Glaz, L. Kuo, and T.-M. Lee, "Inference for the maximum cell probability under multinomial sampling," *Naval Research Logistics (NRL)*, vol. 39, no. 1, pp. 97–114, 1992.

[49] S. Ross, *Simulation*. Elsevier Science, 2012.

[50] "Video of CR-Honeynet prototype." http://sbhunia.me/research/honeynet/.

[51] S. Bhunia and S. Sengupta, "Implementation of interface agility for duplex dynamic spectrum access radio using usrp," in *IEEE MILCOM*, 2017.