Wireless Communications under Broadband Reactive Jamming Attacks

Song Fang, Yao Liu, and Peng Ning, *Senior Member, IEEE*

Abstract—A reactive jammer jams wireless channels only when target devices are transmitting; Compared to constant jamming, reactive jamming is harder to track and compensate against [2], [38]. Frequency hopping spread spectrum (FHSS) and direct sequence spread spectrum (DSSS) have been widely used as countermeasures against jamming attacks. However, both will fail if the jammer jams all frequency channels or has high transmit power. In this paper, we propose an anti-jamming communication system that allows communication in the presence of a broadband and high power reactive jammer. The proposed system transmits messages by harnessing the reaction time of a reactive jammer. It does not assume a reactive jammer with limited spectrum coverage and transmit power, and thus can be used in scenarios where traditional approaches fail. We develop a prototype of the proposed system using GNURadio. Our experimental evaluation shows that when a powerful reactive jammer is present, the prototype still keeps communication, whereas other schemes such as 802.11 DSSS fail completely.

Index Terms—Wireless communication, jamming attacks, reactive jammer, broadband

1 Introduction

Jamming attacks are well-known threats to wireless communication. A jammer uses a radio frequency (RF) device to transmit wireless signals. Due to the shared nature of wireless medium, signals of the jammer and the sender collide at the receiver, and the signal reception process is disrupted. Anti-jamming techniques have been extensively studied and proposed in the literature over the past decades (e.g., [7], [11], [16], [23], [26], [28], [41]). Frequency hopping spread spectrum (FHSS) (e.g., [11], [33]) and direct sequence spread spectrum (DSSS) (e.g., [19], [28]) are dominantly used for the anti-jamming purpose.

In FHSS, the sender and the receiver switch their communication channels periodically to avoid being jammed. In DSSS, the sender multiplies the original message with a pseudo-random sequence to obtain the spreading gain. If the jammer's power is not strong enough to overwhelm the DSSS signals with the spreading gain, the receiver can use the same pseudo-random sequence to recover the message.

Although FHSS and DSSS techniques were developed more than 30 years ago, until now these techniques and their variants are all limited by a common assumption that the jammer can only jam part of the communication channels or has limited transmit power. Unfortunately, if the jammer is broadband (i.e., it can jam all channels simultaneously) or has a high transmit power to overcome the spreading gain, these methods fail to maintain the anti-jamming communication. Hence, it seems that a broadband and high-power jammer is perfect and invincible. However, when such a jammer adopts a reactive jamming strategy, a closer examination on the jammer's behavior reveals its "Achilles Heel".

Reactive jamming attacks are among the most effective jamming attacks [2]. Compared to constant jamming, reactive jamming is not only cost effective for the jammer, but also hard to track and remove due to its intermittent jamming behaviors [2]. To be reactive, a reactive jammer "stays quiet when the channel is idle, but starts transmitting a radio signal as soon as it senses activity on the channel" [41]. Channel sensing causes a short delay. For example, energy detection is the most popular channel sensing approach with very small sensing time [13]. When implemented in a fully parallel pipelined FPGA for fast speed [6], energy detection requires more than $1\,\mathrm{ms}$ to detect the existence of target signals for a 0.6 detection probability and -110 dBm signal strength. In addition, upon detecting the target signal, the jammer needs to switch its status from quiet to transmitting. The switching process further takes time. Therefore, before the jammer actually jams, the sender has already transmitted $\Delta t R$ bits, where Δt is the reaction time of the jammer and R is the transmission rate of the sender.

This observation provides insights into designing countermeasures to deal with the broadband and high-power reactive jammers. It is easy for people to conceive that the receiver may collect information bits from the unjammed parts of received packets and try to assemble these bits together to obtain a meaningful message. However, signifi cant technical challenges exist to prevent this intuition from being transformed into a real-world realization. For example, transmission errors like lost or duplicate bits may happen when there exist jamming attacks or a retransmission mechanism is employed. A small number of lost/duplicate bits can make many bits mis-aligned, causing the failures in reconstructing the original message.

- *S. Fang and Y. Liu are with the Department of Computer Science and Engineering, University of South Florida, Tampa, FL 33620. E-mail: songf@mail.usf.edu, yliu@cse.usf.edu.*
- *P. Ning is with the Department of Computer Science, North Carolina State University, Raleigh, NC 27695-8206. E-mail: pning@ncsu.edu.*

Manuscript received 3 Apr. 2014; revised 31 Oct. 2014; accepted 28 Jan. 2015. Date of publication 2 Feb. 2015; date of current version 18 May 2016. 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 no. 10.1109/TDSC.2015.2399304

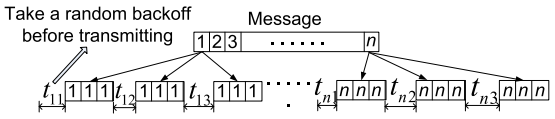


Fig. 1. Sender design.

In this paper, we aim to create techniques that can solve the major challenges in utilizing the unjammed bits survived in the reaction time of a reactive jammer to establish the anti-jamming communication. Based on the proposed techniques, we implemented a real-world prototype anti-jamming system, which can collect unjammed bits and assemble them into an original message under the broadband and high-power reactive jamming attacks.

Note that our goal is to raise the wireless communication from non-existence in extremely hostile environments (e.g., battlefield) to being available, rather than support high-speed applications like video streaming in benign environments. When FHSS and DSSS cannot deliver a single bit, the proposed techniques can still maintain the wireless communication.

The contribution of this paper is three-fold: (1) we developed novel techniques to harness the reaction time of a reactive jammer for anti-jamming communication; (2) we designed a communication system that integrates the proposed techniques to enable information exchange between wireless devices under broadband and high-power reactive jamming attacks; (3) we implemented a prototype using the USRP platform [20], and evaluated the performance on top of the prototype implementation.

2 Technical Challenges

To facilitate the readers' understanding of the technical challenges, we first describe a basic design of the sender for achieving such an anti-jamming system. To transmit a message, the sender may take a random backoff before each transmission, as shown in Fig. 1. This makes it hard for the reactive jammer to predict when the sender will start the next transmission. The jammer may attempt to jam the communication for longer time periods. However, this will increase the chance for the reactive jammer to be detected and removed. The sender transmits each bit of the message for multiple times (e.g., three times in Fig. 1) to increase the chance that the receiver receives this bit.

Note that there may exist other ways to design the sender. For example, if the sender resides in the power range of the jammer, the need of random backoffs can be removed. Before each transmission, the sender may perform channel sensing to determine whether or not the jammer is transmitting. If not, the sender immediately sends bits without waiting for the backoff time to expire, and hence the system throughput can be greatly improved. Nevertheless, as our initial investigation, we will focus on the basic design that utilizes random backoffs and retransmissions.

Although the design of the sender can be simple and straightforward, the design of the corresponding receiver is difficult and complicated. To use the unjammed bits survived in the jammer's reaction time to reconstruct an original message, a receiver should have the following essential capabilities as shown in Fig. 2. First, the receiver receives a series of bits from the wireless channel. Among these received bits, the receiver should be able to extract unjammed bits that carry useful information about original messages. Thus, the receiver needs to process each received bit with a *jamming detector*, which checks if this bit is jammed, and discard all jammed bits. Second, to assemble unjammed bits into an original message, the receiver should be able to achieve bit synchronization, i.e., to identify the correct positions of received unjammed bits in an original message. Accordingly, the receiver needs to feed the output of the jamming detector to a *bit synchronization decoder* to achieve this goal.

Finally, if a smart jammer knows that such an anti-jamming system that collects unjammed bits for communication is being used, in addition to reactive jamming, the jammer may try to defeat the system by transmitting fake bits to the receiver when the sender is not transmitting. These fake bits can mislead the receiver to incorrectly decode a message. In this paper, we refer to such attacks as *pollution attacks*. The receiver should be able to address pollution attacks to make the decoding of the original message feasible. Thus, the receiver needs to enforce defending techniques against pollution attacks throughout the communication. However, there exist significant technical challenges in achieving these essential capabilities:

- *Detecting jammed bits.* Traditional jamming detection aims to find out if wireless communication is jammed (e.g., [32], [41]). To detect jamming, a receiver usually analyzes received signal samples to obtain statistical values like packet loss rate and bit error rate. These values enable the receiver to make a decision regarding whether or not the communication system is under jamming attacks. However, traditional techniques cannot be directly applied to achieve anti-jamming systems that reconstruct messages by assembling unjammed bits together, because in such systems the receiver needs to distinguish jammed bits from unjammed bits rather than merely detecting the existence of jamming. Therefore, reliable jamming detection techniques that can identify unjammed bits should be created to realize such anti-jamming systems.
- *Bit synchronization.* Bit synchronization errors will prevent the receiver from correctly assembling the unjammed bits from the sender into an original message. Bit synchronization errors are mainly caused by lost, duplicated, and inserted bits. A bit of the sender's original message may get lost when it is jammed by the jammer. Also, if the sender transmits each bit of a message for multiple times, the receiver will receive extra bits. With the presence of lost and extra bits, the receiver cannot know the correct positions of received unjammed bits in the original message, and thus it fails to recover this message. Therefore, techniques should be created to establish the correct bit synchronization between the sender and the receiver in the presence of bit synchronization errors.
- *Dealing with pollution attacks.* One possible way to distinguish the sender's bits from the jammer's fake

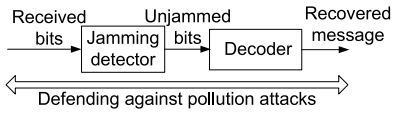


Fig. 2. Receiver design.

bits is to use existing physical layer fingerprinting techniques such as radio-metrics (e.g., [5]) and radio frequency fingerprints (e.g., [22], [43]). However, recent research discoveries (e.g., [10], [18]) reveal that physical layer fingerprinting techniques are vulnerable to certain security threats (e.g., mimicry attacks [18]), and it has been demonstrated that an attacker can easily forge physical layer fingerprints to impersonate a target wireless device (e.g., [10], [18]). Thus, secure and reliable countermeasures against pollution attacks should be designed to make the use of unjammed bits for anti-jamming communication feasible.

In what follows, we present the proposed techniques to deal with these challenges. We made the following clarifications of the jammer. First, a general reactive jammer jams the channel when it detects the sender's transmission and stops jamming when the sender ends transmission. In this paper, however, we assume a more challenging reactive jammer model with unpredictable jamming behavior, i.e., the jammer jams the channel when it detects the sender's transmission, but after the sender stops transmission, instead of immediately stopping, the jammer chooses a random value δ ($\delta \geq 0$), and lasts jamming for δ second(s).

Second, multiple jammers may collaborate to jam the communication channel. The impact of these jammers can be reduced to that of one jammer. Specifically, if the jammers take turns to jam the channel in a seamless way (i.e., all time slots are jammed), then the jamming impact is similar to that caused by one constant jammer. If the jammers take turns to jam the channel in an unseamless way (i.e., some time slots are not jammed), then the jamming impact is similar to that caused by one random jammer who jams the channel at a random time and lasts jamming for a random duration. For a constant jammer that jams all channels, overwhelms the spreading gain, and never stops, no existing methods can be used to beat such a jammer. Localization or social engineering approaches might be used to physically find the jammers so that they can be disabled. However, as long as unjammed time slots exist, the proposed techniques can decode bits received during the unjammed time slots into a meaningful message. The details are shown below.

3 Detection of (Un)Jammed Bits

In this section, we develop a novel technique that utilizes physical layer modulation properties to identify (un) jammed bits.

3.1 Preliminaries on Modulation

I/Q modulation has been widely used in modern wireless systems, including WCDMA, WiMax, ZigBee, WiFi, and digital video broadcasting (DVB). I/Q modulation encodes data bits into physical layer symbols, which are the transmission units in the physical layer. In the following, we use quadrature phase-shift keying (QPSK) modulation, a typical I/Q modulation, to illustrate how I/Q modulation works.

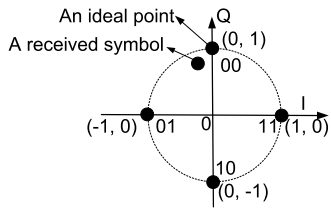


Fig. 3. QPSK modulation/demodulation.

QPSK—An example I/Q modulation. QPSK encodes 2 bits into one symbol at a time. In Fig. 3, bits 00, 01, 10, and 11 are represented by points whose coordinates are $(0, 1)$, $(-1, 0)$, $(0, -1)$, and $(1, 0)$ in an I/Q plane, respectively. The I/Q plane is called a *constellation diagram*. A symbol is the coordinate of a point in the constellation diagram. For a bit sequence 0010, the modulation output are two symbols: $(0, 1)$ and $(0, -1)$. A received symbol is not exactly the same as the original symbol sent by the sender, since wireless channels usually introduce noise to signals that pass through them [11]. To demodulate, the receiver finds the point that is closest to the received symbol in the constellation diagram. For example, in Fig. 3, the point closest to the received symbol is $(0, 1)$. Thus, the demodulation output is 00.

3.2 Observation

Intuitively, jamming signals can introduce a large distortion to signals transmitted by the sender, since the goal of the jammer is to corrupt the signals. If a received symbol is jammed, it may greatly deviate from its ideal point in the constellation diagram and can hardly be recovered. To get more insights in this process, we perform experiments to examine the impacts of jamming on symbol locations.

We collect the received symbols using USRPs [20], which are radio frequency front ends equipped with analog to digital (AD) and digital to analog (DA) converters. In our experiments, three USRPs are used as the sender, the receiver, and the jammer, respectively, each of which is connected to a computer. Automatic gain control (AGC) is employed by USRPs. We set the bit rate as 1 Mbps, carrier frequency as 5 GHz, and modulator as QPSK.

We consider a normal scenario and a jamming scenario. In the first one, only the sender transmits randomly generated packets to the receiver, while in the second one, both the sender and the jammer transmit random packets to the receiver concurrently. The receiver record the coordinates of the received symbols in the constellation diagram.

In the normal scenario, as shown in Fig. 4, the received symbols form four clusters, each of which centers around an ideal point of QPSK. However, in the jamming scenario, as shown in Fig. 5, the received symbols randomly spread over the constellation diagram. Thus, it is hard to identify the ideal points for the received symbols, and demodulation errors may happen frequently.

3.3 Detection Method

Let d_{unjam} (or d_{jam}) be the distance between an unjammed (or a jammed) symbol and the origin in the constellation

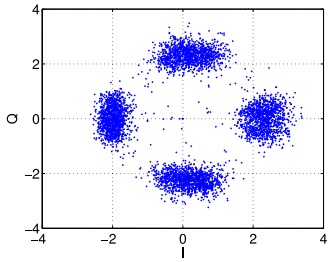


Fig. 4. Normal scenario.

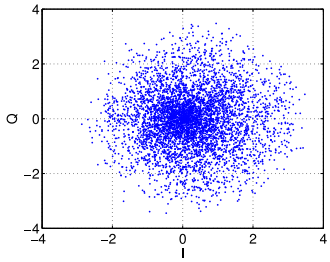


Fig. 5. Jamming scenario.

diagram. As shown in the above experiment, unjammed symbols are close to their ideal constellation points, and thus d_{unjam} approximately equals to the distance between an ideal point and the origin. In contrast, jammed symbols deviate from their ideal points. Due to AGC, such deviation is actually a convergence from ideal points toward the origin rather than an expansion out of the constellation diagram range. Hence, unlike unjammed symbols, jammed symbols are randomly distributed within the constellation diagram, and the expected value of d_{jam} is smaller than that of d_{unjam}. For example, in Figs. 4 and 5, the average distance between a received symbol and the origin is 2.2524 and 1.2628, respectively.

We propose to use the distance d between a received symbol and the origin of the constellation diagram as a metric to detect the existence of jammed symbols. For each received symbol, we compute the corresponding distance d, and compare d with a threshold t. If $d > t$, the received symbol is marked as unjammed. Otherwise, it is jammed and we discard it.

Note that different metrics can be explored to accommodate different variants of I/Q modulation. For example, rectangular based I/Q modulation (e.g., 64 QAM) may use the distance between a received symbol and the closest constellation point as the detection metric. We choose the metric d_{unjam} (or d_{jam}) due to its simplicity. This metric serves as an example to illustrate how our observation can be utilized for detecting jammed and unjammed symbols.

The detection accuracy can be enhanced by using the temporal correlation of adjacent symbols. Let s_i and d_i denote the ith received symbol and its distance from the origin, respectively. We determine whether s_i is jammed or not by examining it along with its neighbor symbols $s_{i-N}, \ldots, s_{i-1}, s_i, s_{i+1}, \ldots, s_{i+N}$, where N is system parameter. Symbol s_i is marked as unjammed, if all symbols in this sequence have distances larger than the threshold. As we will show below, this method can enhance the detection accuracy.

3.4 False Positives (FPs) and False Negatives (FNs)

False positives and false negatives are two types of errors that may happen in the detection. In a false positive, d_{unjam} of at least one symbol in the temporal sequence is less than or equal to t, and thus an unjammed symbol is incorrectly classified as a jammed symbol. In a false negative, d_{jam} of all symbols in the temporal sequences are larger than t, and thus a jammed symbol is incorrectly classified as an unjammed symbol. In Theorems 1 and 2 we derive both probabilities of false negative and positive.

Theorem 1 (Probability of false positive). *The probability P_{fp} that an unjammed symbol is classified as a jammed symbol is $1-(M_1(\frac{v}{\sigma_N},\frac{t}{\sigma_N}))^{2N+1}$, where M_1 is the Marcum Q-function, v is the distance between an ideal point and the origin of the constellation diagram, t is the threshold, $2N+1$ is the length of the temporal sequence, and ${\sigma_N}^2$ is the variance of the jamming signal.*

Proof: Let (I, Q) and (I_i, Q_i) denote the coordinate of a received symbol and its closest ideal point in a constellation diagram, respectively. Due to jamming, $I \neq I_i$ and $Q \neq Q_i$. For an unjammed symbol, we assume additive white Gaussian noise, and thus I and Q can be represented as $I = I_i + \delta_I$ and $Q = Q_i + \delta_Q$, where δ_I and δ_Q are independent and identically distributed (i.i.d) Gaussian random variables with mean value 0 and variance ${\sigma_N}^2$. According to the properties of Gaussian variables [21], $I = I_i + \delta_I$ and $Q = Q_i + \delta_Q$ are also i.i.d. Gaussian random variables. The mean values of I and Q equal to those of I_i and Q_i, respectively, and the variances of them are all ${\sigma_N}^2$. Let d denote the distance between the received symbol and the origin of the constellation diagram. Thus, $d = \sqrt{I^2 + Q^2}$. According to [24], d follows Rice distribution and the cumulative distribution function $F_d(t)$ of d is $F_d(t) = \mathbb{P}(d \leq t) = 1 - M_1(\frac{v}{\sigma_N},\frac{t}{\sigma_N})$, where $\mathbb{P}(d \leq t)$ denote the probability that d is less than or equal to t, $v = \sqrt{{I_i}^2 + {Q_i}^2}$, and M_1 is the Marcum Q-function. Note that the probability P_{fp} of false positive equals to the probability that d of at least one symbol in the temporal sequence is less than or equal to t. Thus, $P_{fp} = 1-(1-\mathbb{P}(d \leq t))^{2N+1} = 1-(M_1(\frac{v}{\sigma_N},\frac{t}{\sigma_N}))^{2N+1}$. □

Theorem 2 (Probability of false negative). *Given that each ideal point in the constellation is jammed with equal probability, the probability P_{fn} that a jammed symbol is classified as an unjammed symbol is $(e^{\frac{-t^2}{2\sigma^2}})^{2N+1}$, where t is the threshold, $2N+1$ is the length of the temporal sequence, and σ^2 is the variance of the I/Q coordinate of a received symbol.*

Proof: Let (I, Q) denote the coordinate of a received symbol. I and Q can be represented as $I = I_s + I_j + \delta_I$ and $Q = Q_s + Q_j + \delta_Q$, where $(I_s,\ Q_s)$ and $(I_j,\ \ Q_j)$ are the symbols transmitted by the sender and the jammer, respectively, and δ_I and δ_Q are additive white Gaussian noise. Assume that the sender transmits each ideal point in the constellation diagram with equal probability. Hence, I_s, Q_s, I_j, Q_j are i.i.d random variables. According to central limit theorem, the probability distribution of the average of i.i.d random variables converges to

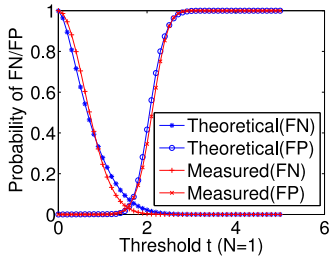


Fig. 6. Theoretical/measured probability of false positive/negative when $N = 1$.

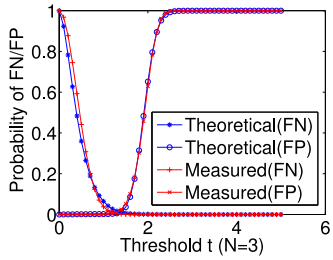


Fig. 7. Theoretical/measured probability of false positive/negative when $N = 3$.

Gaussian distribution as the number of random variables increases. Therefore, we use Gaussian distribution to approximate the distribution of $(I_s + I_j)/2$ and $(Q_s + Q_j)/2$. According to the properties of Gaussian variables [21], $I = I_s + I_j + \delta_I$ and $Q = Q_s + Q_j + \delta_Q$ are also approximately Gaussian distributed, where δ_I and δ_Q are i.i.d Gaussian random variables with mean 0. Ideal points center around the origin, and thus the mean values of I_s, Q_s, I_j, and Q_j are 0.

Let d denote the distance between the received symbol and the origin of the constellation diagram. Thus, $d = \sqrt{I^2 + Q^2}$. Assume that I and Q have the same variance, which is denoted by σ^2. According to [11], d follows Rayleigh distribution and the cumulative distribution function $F_d(t)$ of d is $F_d(t) = \mathbb{P}(d \leq t) = 1 - e^{\frac{-t^2}{2\sigma^2}}$. The probability P_{fn} of false negative equals to the probability that d of all symbols in the temporal sequence are larger than t. Therefore, $P_{fn} = (1 - \mathbb{P}(d \leq t))^{2N+1} = (e^{\frac{-t^2}{2\sigma^2}})^{2N+1}$. □

Experimental validation. To verify the theoretical results, we run the temporal check enhanced method to detect unjammed symbols from symbols collected for normal and jamming scenarios in our earlier experiment. The measured false positive probability P_{fp} and false negative probability P_{fn} are computed by $P_{fp} = 1 - \frac{\#\text{detected symbols}}{\#\text{total symbols}}$ and $P_{fn} = \frac{\#\text{detected symbols}}{\#\text{total symbols}}$, respectively. Meanwhile, we compute P_{fp} and P_{fn} using Theorems 1 and 2. The computation results are shown in Figs. 6 and 7. Note that statistic parameters v, σ_N, and σ are determined based on our earlier experiment.[1] Both theoretical and real measured results are in close consistency. A large N can result in both small P_{fn} and P_{fp}. When $N = 1$, both real measured P_{fn} and P_{fp} can be as low as 0.0444 by using a threshold t that equals to 1.6. If we increase N to 3, we can achieve even lower error rate.

3.5 Determining the Threshold

The threshold t can be determined based on the system requirement for P_{fn} and P_{fp}. For example, if the false negative probability P_{fn} is required to be less than α, we have $(e^{\frac{-t^2}{2\sigma^2}})^{2N+1} < \alpha$. By treating t as an unknown and solve the inequality, we can get $t > \sqrt{2\sigma^2 \ln \alpha^{2N+1}}$. As the threshold t increases, P_{fp} increases but P_{fn} decreases. If the goal is to minimize both P_{fp} and P_{fn}, as shown in Figs. 6 and 7, the minimization result and the corresponding t form the intersection point of the P_{fp} and P_{fn} curves.

In practice, the receiver may use the heuristic search to determine an appropriate threshold that achieves the desired performance in a self-adaptive way. Specifically, under the jamming attacks, a high bit error rate is more likely to be caused by a high false negative rate and hence a small threshold, whereas a low throughput is more likely to be caused by a high false alarm rate and hence a large threshold. Based on the observation of the bit error rate and the throughput, the receiver can adjust the threshold until a desired performance is achieved. To improve the efficiency, the adjudication process can be further implemented using the binary exponential backoff algorithm, which has been widely employed by the computer network design. The receiver may multiplicatively decrease or increase the current threshold to gradually find an acceptable value.

4 Bit Synchronization

The original message is first encoded with a traditional ECC (e.g., Reed-Solomon (RS) codes). ECC corrects substitution errors (i.e., bit "1" is replaced by "0" and vice-versa). The proposed bit synchronization encoding scheme further encodes the ECC-coded message to allow a receiver to decode the correct positions of received bits and recover from synchronization errors.

4.1 Basic Idea

Bit synchronization encoding. The sender and the receiver agree on a sequence that is formed by n integers, where n is the length of the message (a long message may be spited into several short messages of n bits). We call such an integer sequence a *positioning code* and each integer in the sequence a *label*. As shown in Fig. 8, the message is 10110 and the positioning code is 03572. For $1 \leq i \leq 5$, the sender labels the ith bit of the message using the ith label in the positioning code (e.g., the second bit is 0 and its label is 3). In the labeling, the sender uses one symbol to represent both a bit and its label. (Details of labeling will be presented in Section 4.2.) Note that a symbol is the transmission unit of physical layer. Once a receiver receives a symbol, the

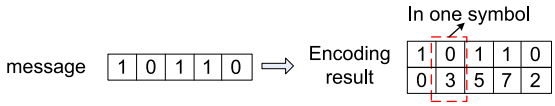


Fig. 8. Bit synchronization encoding at the sender.

1. $v = 2.3949$, $\sigma_N = 0.3838$, and $\sigma = 1.0344$.

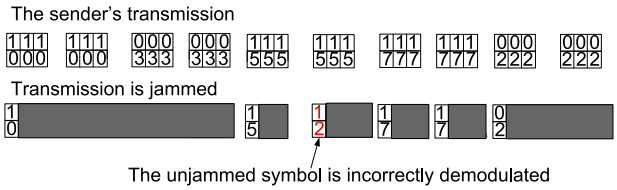


Fig. 9. Bit synchronization errors.

receiver knows both the bit and its label. The encoding results are shown in Fig. 8.

Transmission errors. After encoding, the sender transmits each symbol for multiple times. As an example shown in Fig. 9, the sender transmits the first symbol for three times in the first transmission, and transmits this symbol again for three times in the second transmission. The sender repeats its transmission until the last symbol is transmitted. Due to jamming and retransmissions, a symbol may get lost or duplicated. Also, channel noise may introduce a small number of incorrectly demodulated symbols, and thus extra inserted symbols can be resulted. Fig. 9 shows an example of bit synchronization errors, all copies of the second symbol in the original message are lost, the third received symbol is an extra inserted symbol caused by the incorrect demodulation of a retransmitted unjammed symbol (the original unjammed symbol of "1,5" is incorrectly demodulated as "1,2"), and the fourth and fifth received symbols are duplications of each other.

Bit synchronization decoding. The receiver demodulates each received symbol to extract the bit and corresponding label carried by this symbol. Fig. 10 shows an example following Fig. 9. The extracted bits and labels are 111110 and 052772, respectively. The receiver then takes two steps to correct synchronization errors.

The first step is merging, in which bits are merged into a single bit if they are identical and have the same label. As shown in Fig. 10, the fourth and the fifth received bit are identical (i.e., both of them are 1), and have the same label 7. Thus, they are merged together. The merging result is 11110 and the corresponding labels are 05272. An incorrect merging may happen if multiple bits in the BTmessage are identical and use the same label. In Section 4.3, we give the analytical upper bound of the error probability, and show that the upper bound decreases quickly as configurable parameters such as the number of retransmissions increases. The second step is alignment, which consists of two substeps:

(1) *Dealing with false negatives*. Although most unjammed symbols can be correctly demodulated by the existing demodulation techniques, the demodulation of a small amount of them may be incorrect due to the channel noise. These wrong symbols are actually random incoherent pieces and the correlation between their labels and the positioning code is weak. Therefore, to filter out inserted bits, we perform alignment on the most correlated part between the positioning code and the merged received labels. We find the largest common subsequence (LCS) between the positioning code and received labels, and align the LCS with the positioning code. For example, in Fig. 10, the received labels are $05\underline{2}72$, where the underlined 2 is the inserted label from the jammer. The

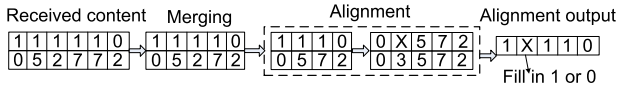


Fig. 10. Bit synchronization decoding at the receiver.

LCS between $05\underline{2}72$ and the positioning code 03572 is 0572. Thus, the inserted label is filtered.

LCS is not necessarily unique. Finding all LCSs requires exponential time complexity in the worst case, whereas finding one LCS is solvable in polynomial time by dynamic programming [9]. Therefore, we utilize existing dynamic programming method to only find one LCS. It is possible that there exist multiple LCSs and the LCS returned by dynamic programming contains inserted labels. However, as shown in the appendix of [17], such probability decreases quickly with the increase of the percentage of the sender's labels. Since incorrect demodulations are rare events, the sender's labels comprise the great majority of total received labels, and thus there is a high chance that the sender's labels form the LCS of received labels and positioning code. Retransmissions further increase this chance. For example, the sender may transmit a message for 3 times. For each transmission, the receiver can obtain a LCS. Given a 0.1 probability that a LCS contains inserted bits, the chance that at least one LCS does not contain inserted bits is 0.999.

(2) *Generating alignment output*. In the LCS 0572, the labels 0, 5, 7, and 2 match the first, third, fourth, and the last label in the positioning code, respectively. Thus, the receiver knows that the second bit is lost, and corrects synchronization errors by filling a bit that can be either 1 or 0 in the position shown in Fig. 10. The alignment output is further processed by traditional ECC to recover the original message. There may exist multiple alignment outputs. In Section 4.3, we develop a fast alignment approach that not only achieves desired alignment accuracy, but also reduces the overhead by only trying a subset of all combinations.

Diversity degree of a positioning code. To avoid incorrect merging/alignment, we require that consecutive labels in the positioning code to be different. Specifically, the i-th label in the positioning code does not equal to any of its previous d labels (i.e., $i-1$ th,..., $i-d$ th label) and successive d labels (i.e., $i+1$ th, ..., $i+d$th label), where $d \geq 1$ is an adjustable parameter, referred to as *diversity degree* of the positioning code. For example, when diversity degree is 2, the eigth label should not be the same as the seventh, sixth, ninth and 10th label.

4.2 Encoding at Sender

The sender adds special data content (e.g., 11111) to both the beginning and the end of a message, so that a receiver can recognize the boundary of a message. We refer to the special data content as a *message delimitation code (MDC)*.

Afterwards, the sender labels the ith bit of the message by packing the ith bit and the ith label of the positioning code into one physical layer symbol. For example, assume that ith bit is 1 and its label is 2. The sender appends 10 (i.e, binary form of 2) to the data bit 1, and the result is $1\underline{10}$, which are modulated into one symbol (e.g., a 8PSK symbol). To improve efficiency, bits in the MDC are not labeled. For an M-ary modulator that encodes $\log_2 M$ bits by

one symbol, the maximum value of a label of the positioning code should be $2^{\log_2 M-1}-1=\frac{M}{2}-1$. For example, an 8PSK symbol uses 1 bit to carry data information and 2 bits to carry the label. Hence, a label is less than or equal to 3 (i.e., 11). Packing a data bit and its label in one symbol achieves atomicity: data bits are always associated with their labels. Upon receiving a symbol, the receiver knows both the data bit and its label.

4.3 Decoding at Receiver

Before decoding, the receiver searches for boundaries of a message. The boundary of the message is identified if the receiver can observe an MDC or a certain data pattern that is a part of MDC. For example, assume that the MDC equals to 1111111, the receiver identifies the beginning or end of a message if the receiver receives 1111111, multiple consecutive 1's (e.g., 1111), or multiple consecutive 1's interleaved with quite a few 0's (e.g., 1110111). The third condition deals with bits inserted by false negatives.

To reduces the chances that the entire MDC is jammed, the sender and the receiver can increase the length of the MDC according to the severity of jamming attacks, so that the receiver can observe at least a part of the MDC. Alternatively, they may take a backoff time between transmitting two consecutive symbols of an MDC.

The receiver then demodulates the symbols of the received message, extracts a data bit and a label from each symbol, and takes two steps to correct synchronization errors.

Merging. Bits are identified as duplicated and merged into a single bit if they are consecutive, identical and have the same label. To detect and merge duplicated bits, the receiver points a cursor to the first bit/label of the received message. Then, the receiver compares the bit/label pointed by the cursor and each of the following N_r-1 bits/labels, where N_r denote the number of retransmissions for a single bit. If inequality occur (e.g, two bits are not equal or have different labels), the receiver merges all equal bits/labels together and points the cursor to the next bit/label. The receiver repeats until all bits/labels of the received message are scanned.

Merging errors. Different bits may be incorrectly merged together, introducing extra lost bits. We first give Lemma 1, based on which we then derive the upper bound of the probability of merging errors in Theorem 3.

Lemma 1. *The probability that two labels in a positioning code equal to each other is less than or equal to $\frac{1}{R-d}$, where d is the diversity degree of the positioning code and R is the number of possible values for each label (e.g., $R=4$ for 8PSK).*

Proof. Let $S=s_1||...||s_n$ denote the positioning code, where n is the number of labels in it. Let p_{eq} denote the probability that the ith label s_i equals to the jth label s_j, where $1\le i,j\le n$ and $i\neq j$. Without loss of generality, we assume that $j>i$. According to the diversity requirement of a positioning code (See Section 4.1), $s_j\neq s_{j-1},\ldots,s_{j-d}$ and $s_i\neq s_{i+1},\ldots,s_{i+d}$. If $j-i\le d$, s_j is one of the previous d labels of s_i. Hence, $p_{eq}=0$. If $j-i=d+1$, s_j's previous d labels are actually s_i's successive d labels, and thus $s_j\neq s_{j-1},\ldots,s_{j-d}$ and $s_i\neq s_{j-1},\ldots,s_{j-d}$. Therefore, $p_e=\frac{1}{R-d}$. If $d+1<j-i\le 2d$, there exists an overlap between s_j's previous d labels and s_i's successive d labels, but the length of the overlap is less than d. Thus, $p_e<\frac{1}{R-d}$. If $j-i>2d$, there is no overlap and $p_e=\frac{1}{R}$. Thus, p_e is at most $\frac{1}{R-d}$. □

Theorem 3 (Probability of merging errors). *The probability p_e that a received message is merged incorrectly is less than $1-(1-\frac{p^{rd}-p^{r(n-n(1-p^r)+1)}}{2(R-d)})^{n(1-p^r)-1}$, where n is the length of a positioning code, R is the number of possible values for each label, r is the number of retransmissions for each bit in the message, d is the diversity degree of the positioning code, and p is the probability that a bit transmitted is lost.*

Proof. Let $S=s_1||...||s_n$ and $M=m_1||...||m_n$ denote the positioning code and original message, respectively. Let $M_r=m_{i_1}^{r_1}||...||m_{i_g}^{r_g}$ denote the received message, where m_{i_j} is the i_jth element of the original message and $m_{i_j}^{r_j}$ means that the receiver receives r_j retransmitted copies of m_{i_j}. $m_{i_j}^{r_j}$ may be incorrectly merged with $m_{i_{j+1}}^{r_{j+1}}$ if they are identical and have the same label (i.e., $m_{i_j}=m_{i_{j+1}}$ and $s_{i_j}=s_{i_{j+1}}$).

An information bit (i.e., a bit in the original message and all its retransmitted copies) may get lost. Let p denote the probability that a transmitted bit is lost. The probability that an information bit is lost equals to p^r, where r is the number of retransmissions. If $i_{j+1}-i_j\le d$ (i.e., less than d information bits between $m_{i_{j+1}}$ and m_{i_j} are lost), then $m_{i_{j+1}}$ is among the successive d elements of m_{i_j} and their labels are always different. Thus, the probability of merging errors is 0. If $i_{j+1}-i_j>d$, $m_{i_j}^{r_j}$ and $m_{i_{j+1}}^{r_{j+1}}$ will be identified as duplicate bits when $m_{i_j}=m_{i_{j+1}}$ and $s_{i_j}=s_{i_{j+1}}$. Let p_{eq} be the probability that the i-th element s_i equals to the j-th element s_j, where $1\le i,j\le n$ and $i\neq j$. The overall probability p_{e_j} that $m_{i_j}^{r_j}$ is incorrectly merged with $m_{i_{j+1}}^{r_{j+1}}$ is $\frac{p_{eq}}{2}\mathbb{P}(i_{j+1}-i_j>d)=\frac{p_{eq}}{2}\sum_{k=d}^{n-n(1-p^r)}p^{rk}(1-p^r)$. According to Lemma 1, $p_{eq}\le\frac{1}{(R-d)}$. Therefore, $p_{e_j}\le\frac{(p^{rd}-p^{r(n-n(1-p^r)+1)})}{2(R-d)}$, where $n(1-p^r)$ is the expected number of information bits received by the receiver. For the received message $M_r=m_{i_1}^{r_1}||...||m_{i_g}^{r_g}$, the probability p_e of merging incorrectly is $1-\prod_{j=1}^{j=n(1-p^r)-1}(1-p_{e_j})$, and thus $p_e\le 1-(1-\frac{p^{rd}-p^{r(n-n(1-p^r)+1)}}{2(R-d)})^{n(1-p^r)-1}$. □

We use simulations to validate the analytical upper bound. All simulations are done in MATLAB 7.7.0. We let $R=32$ and $p=0.95$, and perform 10,000 trails. In each trial, we randomly generate a message and a positioning code of length 155, and label the message using the positioning code. We retransmit each bit of the message for r times ($10\le r\le 30$), and delete each retransmitted bit with probability p. We then merge the remaining bits, and compare the result with the correct result obtained based on the original generated message. If both results are not equal, a merging error happens and we mark this trial as failed. We compute the simulated probability of

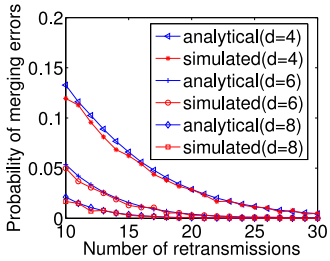


Fig. 11. Merging errors.

merging error and its analytical upper bound using $(\frac{\#\text{failed trials}}{\#\text{total trials}})$ and Theorem 3, respectively.

As shown in Fig. 11, the simulated probability is only slightly less than its analytical upper bound, which indicates that the result Theorem 3 is a tight upper bound. A larger diversity degree d can achieve smaller error probability. As the number r of retransmissions increases, both the simulated probability and its upper bound decrease and approach to 0. In particular, when $d = 8$ and $r = 20$, the simulated probability and the analytical upper bound are 0.0005 and 0.0006.

Merging errors will generate additional lost bits during message recovery. In the following, we develop a method to recover lost bits through alignment and ECC.

Alignment. The goal of alignment is to find the actual position of each received bit in the original BTmessage. Let S denote the positioning code and L the merged labels (e.g., in Fig. 10, the merged labels are 0572).

(1) *Basic alignment method*. If the length of the positioning code is small, we can do alignment in a brute force way. Specifically, assume that the length of L is q. The receiver can find all length-q subsequences of S, and compare each of them with L. For each subsequence that equals to L, the receiver generates an alignment output by padding 1's or 0's into the positions of lost bits. For example, assume that padding bits are 1's and the received message after merging is 00. For $L = 17$ and $S = 1{,}317$, the alignment outputs are 0110 and 1100. Each alignment output is further processed by traditional ECC decoding, where replacement errors (i.e., $1 \to 0$ or $0 \to 1$) are corrected. Since there may exist multiple alignment outputs, the receiver may obtain multiple decoding results, among which the one that can pass cyclic redundancy check (CRC) or authentication is the recovered message. If the length of S is large, this method is time consuming. We develop a fast alignment approach below to reduce the overhead.

(2) *A fast alignment method*. To achieve fast alignment, we propose to only find one alignment. We further show that given proper configurations, this single alignment leads to a very small error probability. We use a simple greedy strategy to obtain a single alignment. Specifically, the receiver compares labels of L with those of the positioning code S, trying to find S's leftmost or rightmost subsequence that equals to L. For example, if $L = 17$ and $S = 1{,}177$, the S's leftmost and rightmost subsequence that equals to L is underlined in $\underline{1}{,}1\underline{77}$ and $1{,}\underline{17}\underline{7}$, respectively. The positions of the leftmost/rightmost subsequence is 13/24, and thus the corresponding decision is that the first and the third bits of the message are received (or the second and the last bits are received).

Alignment errors. For basic alignment, the probability that alignment errors happen is 0. For fast alignment, alignment errors may happen if the positions of the leftmost/rightmost subsequence are not formed by the correct positions of the received bits. We derive the upper bound of the probability of alignment errors in Theorem 4.

Theorem 4 (Probability of alignment errors). *The probability p_e that the receiver fails to generate correct alignments is $1 - \sum_{k=q}^{n} \frac{\binom{k}{q}}{\sum_{w=q}^{n}\binom{w}{q}} (1 - \frac{p^{rd} - p^{r(n-q+1)}}{R-d})^{k-q}$, where n is the length of a positioning code, R is the number of possible values for each label, r is the number of retransmissions for each bit in the message, d is the diversity degree of the positioning code, and p is the probability that a bit is lost.*

Proof. Let $S = s_1||...||s_n$ denote the sequence formed by the positioning code and p_{eq} be the probability that the ith element s_i equals to the jth element s_j, where $1 \le i, j \le n$ and $i \ne j$. According to Lemma 1, $p_{eq} \le \frac{1}{R-d}$, where R and d are the number of labels and the diversity degree of the positioning code, respectively. Let $F = f_1||...||f_q$ denote the sequence formed by the actual positions of received bits (i.e., the receiver receives the f_1th, ..., f_qth bits), and L denote the sequence formed by merged labels. F is the positions of S's leftmost subsequence that equals L if two conditions are satisfied: (1) For $1 \le i < f_1$, $s_i \ne s_{f_1}$. (2) For $1 \le j \le q-1$ and $f_j < i < f_{j+1}$, $s_i \ne s_{f_{j+1}}$. □

Therefore, the probability p_{min} that F is the positions of S's leftmost subsequence is $\prod_{i=1}^{f_1-1}(1 - \mathbb{P}(s_i = s_{f_1})) \prod_{j=1}^{q-1} \prod_{i=f_j+1}^{f_{j+1}-1}(1 - \mathbb{P}(s_i = s_{f_{j+1}}))$. Assume that $f_j < i < f_{j+1}$. Thus, $\mathbb{P}(s_i = s_{f_{j+1}}) = p_{eq}\mathbb{P}(f_{j+1} - i > d)$. According to Lemma 1, $p_{eq} \le \frac{1}{(R-d)}$. Thus, $\mathbb{P}(s_i = s_{f_{j+1}}) \le \frac{\mathbb{P}(f_{j+1} - f_j > d)}{R-d}$. Note that $f_{j+1} - f_j > d$ indicates that at least d labels between s_{f_j} and $s_{f_{j+1}}$ are lost. Therefore, $\mathbb{P}(s_i = s_{f_{j+1}}) \le \frac{p^{rd} - p^{r(n-q+1)}}{R-d}$ Similarly, for $1 \le i \le f_1$, $\mathbb{P}(s_i = s_{f_1}) \le \frac{p^{rd} - p^{r(n-q+1)}}{R-d}$ Thus, $p_{min} \ge (1 - \frac{p^{rd} - p^{r(n-q+1)}}{R-d})^{f_q - q}$, where f_q is a random variable ranging from q to n. According to total probability formula, $p_{min} \ge \sum_{k=q}^{n} \frac{\binom{k}{q}}{\sum_{w=q}^{n}\binom{w}{q}} (1 - \frac{p^{rd} - p^{r(n-q+1)}}{R-d})^{k-q}$. The probability p_e that the alignment is incorrect equals to the probability that F is not the positions of S's leftmost subsequence. Hence, $p_e = (1 - p_{min}) \le 1 - \sum_{k=q}^{n} \frac{\binom{k}{q}}{\sum_{w=q}^{n}\binom{w}{q}} (1 - \frac{p^{rd} - p^{r(n-q+1)}}{R-d})^{k-q}$.

We also use simulation to validate the analytical upper bound of alignment errors. The parameters are the same as those used in the simulation for merging errors (i.e., $R = 32$, $p = 0.95$, and 10,000 trials). In each trial, we randomly generate a positioning code of length 155, retransmit each label for r times ($10 \le r \le 30$), and delete each retransmitted label with probability p. The remaining labels are merged together. Then we find the positions of received bits (labels) using the fast alignment alignment approach, and compare the result with the true positions. If they are not equal, an alignment error happens and we mark this trial as failed. We compute the simulated

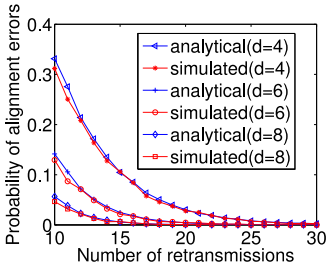


Fig. 12. Alignment errors.

probability of alignment error and its analytical upper bound using $(\frac{\#\text{failed trials}}{\#\text{total trials}})$ and Theorem 4.

Fig. 12 shows that the error probability decreases as diversity degree d and the number of retransmissions increase. From Fig. 12, we can observe that Theorem 4 gives a tight upper bound of the error probability. In particular, when $d = 8$ and $r = 20$, both the simulated probability and the upper bound are about 0.0006.

5 Dealing with Pollution Attacks

At first glance, the receiver can use the physical layer fingerprints (e.g., [5], [22]) to distinguish between the sender's and the jammer's bits. However, as discussed earlier, an attacker may forge these fingerprints to impersonate a target wireless device. The root reason for such forging attacks is that physical layer fingerprints are exposed to the public, and anyone who receives the sender's signals can obtain these fingerprints. The easy access of physical layer fingerprints provides opportunities for attackers to perform reverse engineering to crack these fingerprints. In this section, instead of relying on the publicly-known fingerprints, we aim to create techniques that enable the use of a shared secret key between the sender and the receiver to combat pollution attacks.

5.1 Overview of the Technique

As indicated in Section 2, the sender may take a random backoff prior to each transmission to reduce chances of being jammed. We explore such random backoffs to deal with pollution attacks. Intuitively, assume the receiver knows when the sender is in backoffs, we can create a "dilemma" for the jammer: if the jammer sends fake bits when the sender is in backoffs, these bits will be simply discarded by the receiver because the receiver knows that the sender is silent. On the other hand, if the jammer sends fake bits when the sender is in non-backoffs, the jammer actually jams the transmission and the receiver can remove jammed bits by using a jamming detector. In this paper, we refer to a non-backoff interval as a *transmission interval*.

Based on this intuition, we propose to establish common transmission intervals between the sender and the receiver to deal with pollution attacks. Note that the common transmission intervals should be confidential to the jammer, such that the jammer cannot follow them to jam the communication. In traditional FHSS and DSSS systems, a shared secret key is used to generate common frequency hopping patterns or spread spectrum sequences only known to the communicators for the anti-jamming purpose. Similarly, the sender and the receiver can utilize such a shared secret key to generate their transmission intervals.

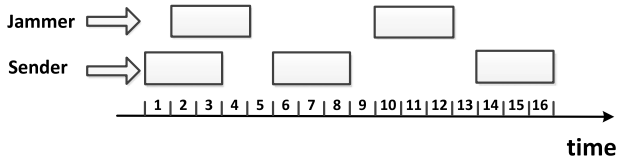


Fig. 13. An example of the defense against pollution attacks.

Fig. 13 shows an example of the countermeasure against pollution attacks. For the sake of the presentation, we assume a time slotted system, where continuous time is divided by multiple time slots. As shown in Fig. 13, three transmission intervals are generated by the shared secret, and they start at time slots 1, 6, and 14. Each transmission interval lasts for three time slots. The sender (receiver) only transmits (receives) on the transmission intervals. As aforementioned, if the jammer transmits fake bits during the transmission interval (e.g., slots 1, 2, and 3), the jammer actually jams the transmission, and the receiver will detect and remove jammed bits by using a jamming detector; If the jammer transmits fake bits during the backoffs (e.g., slots 10, 11, and 12), the receiver will simply discard these fake bits.

One important difference between the physical layer fingerprinting techniques and the proposed scheme is that the former detects fake bits after fake bits have already been received by the receiver, whereas the latter prevents the receiver from receiving fake bits at the beginning. Thus, compared to the physical layer fingerprinting techniques, the proposed scheme makes a dramatic shift from *tolerating the disaster after it happens* to *preventing the disaster from happening*.

To make such a preventive defense scheme against pollution attacks feasible, two key technical questions need to be addressed. First, how can the sender and the receiver generate the transmission pattern? Second, due to clock discrepancy and transmission delay, the receiver's generated pattern may not exactly synchronize with the sender's generated pattern. As shown in Fig. 14, the first transmission interval of the sender starts at t_0, but that of the receiver starts at t_1 in the sender's clock. Thus, the second question is how the sender and the receiver can achieve transmission synchronization, such that the receiver captures the sender's transmission intervals. In what follows, we create techniques that can solve both challenges.

5.2 Generation of Transmission Intervals

To prevent a jammer from predicting the transmission schedule, a transmission interval should happen at a random time. The sender and the receiver use the shared secret

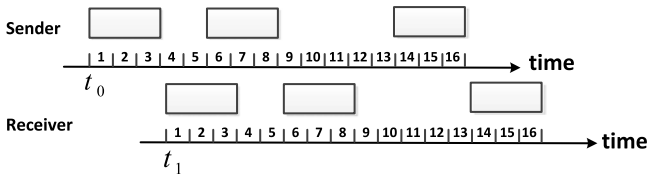


Fig. 14. Example of the transmission synchronization.

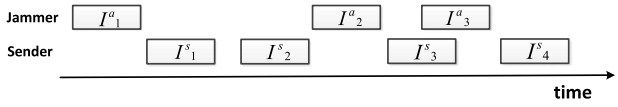


Fig. 15. Observed transmission intervals.

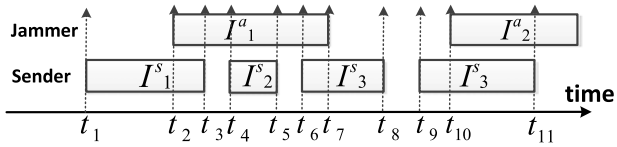


Fig. 16. Start times of the sender's and the jammer's transmission intervals.

to generate random start times for transmission intervals. Assume the communicators expect m transmission intervals. Let t_i denote the start time of the ith transmission interval for $1 \leq i \leq m$. Let $h(\cdot)$ denote utilizing a cryptographic hash function. Let s and l denote the secret key and the duration of a transmission interval, respectively. Further let n denote the maximum interval between the end time of a transmission interval and the start time of the next transmission interval. Both l and n are adjustable and can be selected based on the system requirement. To generate t_i, the sender and the receiver use their current local times to initialize t_0, and then for $1 \leq i \leq m$ they compute t_i as $t_i = t_{i-1} + l + h^{m-i+1}(s) \bmod n$, where mod denotes the modular operation, $t_{i-1} + l$ denotes the finishing time of the $i-1$th transmission interval, and $h^{m-i+1}(s)$ denotes applying the hash function $h(\cdot)$ on the secret key s for $m-i+1$ times. For example, $t_m = t_{m-1} + l + h(s) \bmod l$, $t_{m-1} = t_{m-2} + l + h(h(s)) \bmod l$, and $t_1 = t_0 + l + h^m(s) \bmod l$.

5.3 Transmission Synchronization

The receiver needs to identify the sender's transmission intervals, such that it can capture the full picture of the transmitted content. In the ideal case, the receiver can simply use existing channel sensing techniques (e.g., [29], [30]) to detect the sender's transmission intervals. For example, if the received signal strength is larger than a threshold, then a transmission activity is detected and the receiver can know the corresponding start time. However, in practice, the jammer can inject fake transmission intervals to the channel, but the channel sensing techniques may not distinguish between fake transmission intervals and the true ones. As an example shown in Fig. 15, received transmission intervals consist of the sender's transmission intervals $I^s{}_1$, $I^s{}_2$, $I^s{}_3$, and $I^s{}_4$, and the jammer's fake transmission intervals $I^a{}_1$, $I^a{}_2$, $I^a{}_3$, and $I^a{}_4$. Note that $I^a{}_3$ jams $I^s{}_3$ and they overlap each other. The receiver needs to identify the sender's transmission intervals from the received jammed intervals.

We propose to use two steps to identify the sender's transmission intervals. First, the receiver coarsely finds all possible start times of the sender's transmission intervals. Then, the receiver uses its local generated start times of transmission intervals to refine the coarse result and determine the correct start times of the sender's transmission intervals. Note that the duration of each transmission intervals can be either fixed or generated using the shared key. With the start time t_i and the transmission duration l_i, the receiver identifies the corresponding transmission interval I_i.

Coarse transmission synchronization. Intuitively, the event that jamming occurs indicates that a second party (i.e., a jammer or the sender) begins to transmit. Thus, a jamming detector can be utilized to identify the sender's start times. A receiver identifies a possible start time of the sender if it detects a new transmission activity or jamming. Fig. 16 shows a synchronization example. At time t_1, the receiver detects that a new transmission starts and it records t_1 as an identified start time. At time t_2, the receiver detects that a new jamming starts, and thus it knows that either the sender or the jammer begins to transmit at t_2. So the receiver also records t_2 as an identified start time. Similarly, the receiver records t_4, t_6, and t_{10} because new jamming starts at these time points, and records t_9 because a new transmission is detected. Note that the time points t_3, t_5, t_7, t_8, and t_{11} are not recorded as start times because the receiver neither detects new transmissions nor new jamming.

Fine-grained transmission synchronization. After the receiver finds out the coarse set of the start times of the sender's transmission intervals, the receiver will refine this coarse set to precisely determine the sender's start times. Assume the coarse set includes all the start times of the sender. We compare identified start times in the set to the receiver's local generated start times to find the sender's start times. Specifically, let $t^s{}_i$ and $t^r{}_i$ denote the ith $(1 \leq i \leq m)$ start time generated by the sender and the receiver, respectively. Because the sender and the receiver use a shared secret key to generate $t^s{}_i$ and $t^r{}_i$, we have $\Delta^s{}_i = \Delta^r{}_i$, where $\Delta^s{}_i = t^s{}_i - t^s{}_{i-1}$ and $\Delta^r{}_i = t^r{}_i - t^r{}_{i-1}$ for $1 \leq i \leq m$.

Assume that the receiver identifies k $(k >= m)$ start times during a certain time window. Let $\mathcal{T}$ denote the set formed by the k start times and $\mathcal{T} = \{t^o{}_1, t^o{}_2, ..., t^o{}_k\}$. If $t^o{}_1$ is the start time of the first transmission interval of the sender (i.e., $t^o{}_1 = t^s{}_1$), then the start times of the second, third, ..., and the ith transmission intervals of the sender should be $t^o{}_1 + \Delta^r{}_2$, $t^o{}_1 + \Delta^r{}_2 + \Delta^r{}_3, \ldots,$ and $t^o{}_1 + \sum_{j=2}^{j=i} \Delta^r{}_j$, respectively. Let $t_i = t^o{}_1 + \sum_{j=2}^{j=i} \Delta^r{}_j$. Thus, the receiver can check if $t_2, t_3, \ldots, t_m$ are elements in the set $\mathcal{T}$. If all of them are elements in $\mathcal{T}$, then $t^o{}_1$ is the start time of the first transmission interval of the sender, and t_i for $2 \leq i \leq m$ is the start time of the ith transmission interval. If any of the t_is are not in $\mathcal{T}$, then $t^o{}_1$ is not the start time of the first transmission interval of the sender, and the receiver uses the same method to check if $t^o{}_2$ is the start time of the first transmission interval, i.e., verifying whether or not $t^o{}_2 + \sum_{j=2}^{j=i} \Delta^r{}_j$ are elements of $\mathcal{T}$ for $2 \leq i \leq m$. Once the receiver finds the start time t $(t \in \mathcal{T})$ of the first transmission interval, the receiver can find the start times of all the other transmission intervals (e.g., the start time of the ith $(2 \leq i \leq m)$ transmission interval is $t + \sum_{j=2}^{j=i} \Delta^r{}_j$.).

5.4 Synchronization Errors

When false negatives of the jamming detector occur, the coarse set may not include all the sender's start times. For example, if the false negative happens at time t_4, the corresponding jamming event will not be detected and thus the

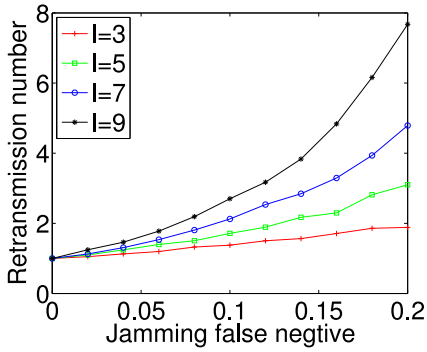


Fig. 17. Retransmission number versus false negative rate.

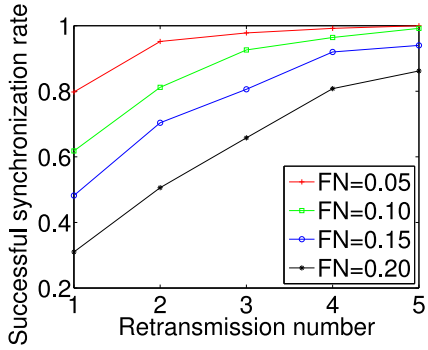


Fig. 18. Successful synchronization rate versus retransmission number.

start time of the second transmission interval is missing in the coarse set. However, the sender and the receiver can always use retransmissions to eventually establish the transmission synchronization. We perform the following simulation to examine the number of retransmissions required for synchronization under different probabilities of false negative.

We perform 1,000 trails in the simulation. In each trail, we randomly generate transmission intervals and the jamming events, and then perform coarse synchronization to form the coarse set. Let FN denote the probability of false negative of the jamming detector, where FN ranges between 0 and 0.2 in the simulation. For each of the sender's start times in the coarse set, we delete it at the probability of FN. Afterwards, we input the coarse set into the fine-grained synchronization algorithm. If the fine-grained synchronization fails, we let the sender retransmit (i.e., we repeat the same procedure) until the synchronization is successful. We record the average of the number of retransmissions of the 1,000 trails.

As shown in Fig. 17, the average ranges between 3 and 9 for different lengths of the transmission duration l when FN reaches a high value of 0.2. We can observe that a smaller l leads to reduced number of retransmissions, and thus lessens the communication overhead. In particular, when $l = 5$, the average number of retransmissions can be as low as 3.

Fig. 18 shows the relationship between the retransmission number and the successful synchronization rate when the length l of the transmission duration is set to 5. We can see that the rate increases as the number of retransmissions increases. Specifically, when the jamming detection false negative rate is as high as 0.2, we can still achieve a successful synchronization rate that is above 0.85 after using five retransmissions.

False alarms can be caused by the jamming detector and the transmission detection algorithms. Fig. 19 shows the relationship between the successful synchronization rate and the false alarm rate, when the retransmission number and the length l of the transmission are 3 and 5, respectively. We can see that the false alarm rate has no effect on the successful synchronization rate for different false negatives. When we increase the false alarm rate (i.e., more false starting times are inserted into the coarse set), the fine-grained transmission synchronization will encounter increased running time, but the synchronization accuracy remains the same as there are no false alarms.

Furthermore, we explore the relationship between the reaction time of the jammer and the number of retransmissions for different bit rates. We plot the results in Fig. 20. The message size M is set to 1,024 bytes. We can observe that the retransmission number always decreases as the reaction time increases. This observation implies that the sender must utilize more retransmissions to defend against the jamming attacks when the jammer has a reduced action time. Specifically, when the reaction time r_a is 0.6 ms and the bit rate R is 1 Mbps, the sender can only transmit a very small portion ($r_a * R/M = 0.007\%$) of the total message within the reaction time. For such a powerful jammer, as shown in Fig. 20a, the receiver can still successfully achieve synchronization with the sender after using two retransmissions when the false negative rate is 0.05. Figs. 20b and 20c show the number of retransmission for different bit rates when the false negative rates are 0.1 and 0.15, respectively.

6 Implementation and Evaluation

We develop a prototype anti-jamming communication system, which we name as *BitTrickle*, to facilitate the experimental evaluation of the proposed techniques under reactive jamming. The prototype system consists of a sender and a receiver, both implemented as a USRP connected to a commodity PC that runs the sender (receiver) program. The USRPs uses XCVR2450 daughter boards operating in the 2.4 GHZ range as RF front ends. The software toolkit is based on GNURadio [1].

Reactive jammer: We setup a high power and sensitive reactive jammer to test the performance. The jammer is implemented on USRPs using GNURadio [1]. We employ energy detection to achieve a lower channel sensing time

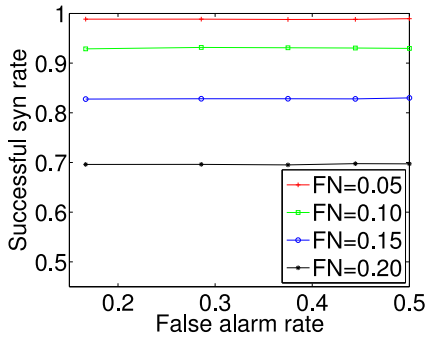
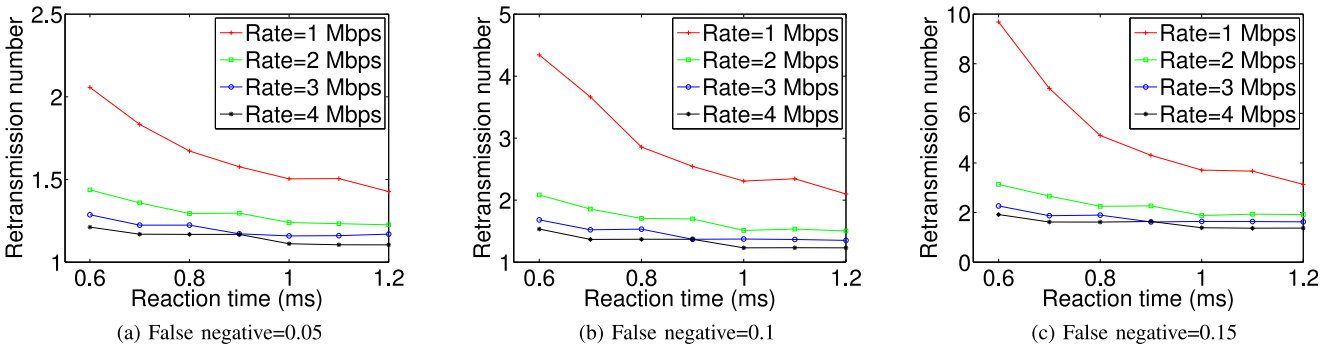


Fig. 19. Successful synchronization rate versus false alarm rate.



(a) False negative=0.05 (b) False negative=0.1 (c) False negative=0.15

Fig. 20. Retransmission number versus the reaction time of the jammer.

(i.e., a signal is detected if received signal strength exceeds a configurable threshold). In our design of the jammer, we equip the jammer with two RFX2400 daughter boards that are used as a transmitter and a receiver, respectively. For both the transmitter and receiver component, we set the parameter "samples per symbol" the minimum value supported by GNURadio to reduce the processing delay (i.e., 2 and 4 for transmitter and receiver, respectively). Also, to maximize the impact of the jammer on the BitTrickle receiver, we let the jammer transmits with maximum gain and place the jammer very close to the receiver (i.e., within 0.1 meter range of the receiver). Parameters of the jammer is shown in Table 1. Fig. 21 shows the topology of the experiment system.

Compared schemes. We compare the following schemes:

- *BitTrickle*. The prototype anti-jamming implementation. This approach uses Reed-Solomon error correction codes, and differential 8PSK modulator/demodulater. The prototype system supports two RS coding rates, which are RS(155, 55) and RS(60, 36).
- *GNURadio benchmark*. The communication tool provided by GNURadio for data transmission and file transfer between two USRPs. The source codes are located at gnuradio/gnuradio-examples/python/digital.
- *802.11 DSSS*. IEEE 802.11 protocol running at direct-sequence spread spectrum mode on 802.11 wireless cards. This approach uses a 11-bits barker code for spreading, carrier sense multiple access with collision avoidance mechanism (CSMA/CA) to resolve collisions on shared channels, and forward error correction (FEC) to enable the reconstruction of the original data.

Evaluation metrics. A jammer aims to prevent the communication between legitimate users. Therefore, how well the sender and the receiver can communicate under jamming attacks is a primary concern to assess anti-jamming systems. We use the following metrics to evaluate the performance: (1) *Packet delivery ratio*. The ratio of the number of correctly received packets to the total number of packets transmitted by the sender. We consider a packet to be received correctly if the packet passes CRC check. (2) *Throughput*. This is the number of successfully delivered bits normalized by time unit. We use bits per second to measure the throughput.

TABLE 1
Technical Details of the Reactive Jammer

Parameter	Value
Frequency range	2.3-2.9 GHz
Channel sensing time	0.6 ms
Transmit power	50 mW
Interpolation/Decimation rate	64/32
Maximum receiving RF bandwidth	16 MHz

6.1 Jamming Detector Parameters Selection

The function of jamming detector is to remove jammed symbols. We implement the temporal based detection method discussed in Section 3.3 to detect jammed symbols. This method involves two key parameters: *temporal sequence length* and *jamming detector threshold*. Thus, we use a training phase to learn appropriate values of both parameters in order to guarantee a good performance of the jamming detector. The training phase collects jammed and unjammed symbols and adjusts both parameters to find a plausible pair. For example, Fig. 22 shows the result for a temporal sequence length of 5. We can see that a threshold of 0.3 balances the probabilities of false negative and false positive and both probabilities are about 0.1. By "probing" possible values, the communication system can decide temporal sequence length and jamming detector threshold to achieve the expected false error probabilities.

6.2 Performance of BitTrickle

We set the bit rate of the sender, the jammer, and the receiver to be 1 Mbps. The sender transmits 100 data packets, each with 1,500 bytes. Positioning codes are randomly generated, and the diversity degree is set to 2 throughout the evaluation. Since the size of a data packet (1,500 bytes) is too long to be directly used with the positioning code and ECC, we divide it into multiple blocks and append a CRC checksum to each block. We use block size 36 or 55 bits, then RS (60,36) or RS (155,55) for ECC, and finally a positioning code of 60 or 155 bits.

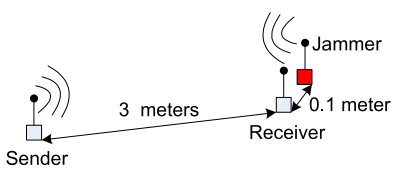


Fig. 21. Topology of the experiment system.

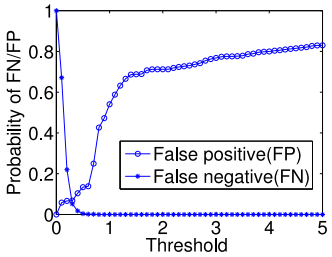


Fig. 22. False negative or positive of jamming detector.

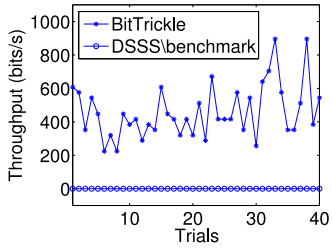


Fig. 24. Throughput.

Packet delivery ratio. We consider different jamming intensities. We use a probabilistic reactive jammer that jams at probability p for $0 \leq p \leq 1$ once detects a sender's signal. The jamming duration is set to be 10 times of the transmission time of a single packet. We compute packet delivery ratio as $\frac{\#\text{correct packets (blocks)}}{\#\text{total transmitted packets (blocks)}}$. Fig. 23 shows the result.

(1) *802.11 DSSS and GNURadio benchmark*. Packet delivery ratio decreases as jamming probability increases. Due to the lack of ECC and retransmission mechanism, the packet delivery ratio of GNURadio benchmark decreases at a rate linearly proportional to the jamming probability. Although 802.11 DSSS achieves a higher packet delivery ratio than GNURadio benchmark, when jamming probability exceeds 0.7, the performance of 802.11 DSSS degrades dramatically. For both 802.11 and benchmark, when the jamming probability equal to 1, the packet delivery ratio drops to 0.

(2) *BitTrickle*. We use a random backoff ranging between 150-200 ms and set the number of bit retransmissions to be 15. Fig. 23 shows that BitTrickle achieves a stable packet delivery ratio that is around 1 no matter how the jamming probability varies. We then reduce the backoff time to 0 ms and increase the bit retransmissions to 60. Fig. 23 shows that the packet delivery ratio of BitTrickle decreases as jamming probability increases. This is because the reduced backoff time increases the chance that the sender's signal collides with the jammer's signal. The modulator used by the BitTrickle prototype has a higher bit error rate (i.e., BER) than that used by GNURadio benchmark (i.e., GFSK). Therefore, the packet delivery ratio of BitTrickle is less than that of benchmark when jamming probability is small (e.g., ≤ 0.7). However, when the probability is 1, unlike benchmark and 802.11, BitTrickle with zero backoff still achieves a non-zero delivery ratio.

Throughput. We consider a common jamming scenario, where the reactive jammer jams the channel as long as it hears the target signal (i.e., $p = 1$). To be conservative, we set the backoff time of BitTrickle to be 0 ms. We performs 40 trials. In each trial, the number of bit retransmissions is set to 60 and the sender transmits 100 data packets to the receiver. We compute throughput as $\frac{\#\text{correct packets (blocks)} \times \text{packet (block) length}}{\text{transmission time}}$.

Fig. 24 plots the computed throughput for each trial. The GNURadio benchmark and 802.11 DSSS fail to send any packet, whereas BitTrickle still achieves a throughput that ranges between 200-900 bits/s, allowing communication to continue.

We also test the BitTrickle throughput under different ECC coding rate (i.e., RS (155,55) and RS (60,36)). Fig. 25 plots the throughput as a function of signal-to-jamming ratio (SJR) (i.e., the ratio of the reaction time to jamming duration). As shown in Fig. 25, RS (60,36) leads to a higher throughput than RS (155,55) when SJR is less than 0.25. That's because RS (60,36) requires a shorter positioning code than RS(155,55), which reduces the chance of synchronization errors. As SJR increases, the receiver gets more information from the sender, and thus the probability of synchronization errors decreases. The error correction capability of RS (155,55) is stronger than that of RS (60,36). For small SJRs, RS (155,55) does not suffer from severe synchronization errors, and thus it can correct more substitution errors and achieve a better throughput.

7 Related Work

The jamming problem in wireless communication has been widely studied during the past few decades (e.g., [4], [11], [12], [19], [25], [26], [27], [28], [31], [32], [33], [34], [35], [36]), and FHSS and DSSS (e.g., [19], [25], [28], [33], [34], [35]) have been widely used for defending against jamming attacks. However, as discussed earlier, FHSS, DSSS, and their variations fail to maintain the wireless communication if the jammer is broadband and has a high transmit power. The proposed research targets at broadband and high-power reactive jammers, and will create anti-jamming techniques that allow wireless devices to exchange information when attacked by such jammers.

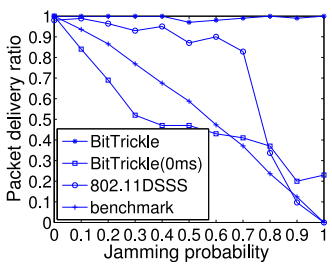


Fig. 23. Packet delivery ratio.

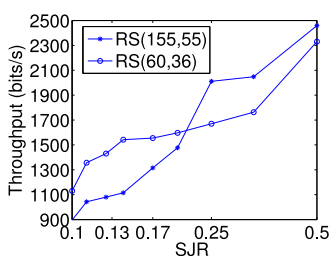


Fig. 25. Throughput for different coding rate.

A recent work considers threats from broadband jammers, and proposes to use timing-based covert channels to address broadband jammers [40]. The idea is to map the inter-arrival times of a sender's corrupted packets into information bits [14]. However, this method fails if the jammer launches pollution attacks or transmits with high power to overwhelm transmitted packets. The proposed research considers both broadband and high power jammers, as well as pollution attacks. There exist other related work, including methods for identifying insider jammers [7], [8], mitigating jamming of control channels [14], [37], jamming avoidance and evasion [3], [39], [42], and mitigating jamming in sensor networks [15], [39]. These work are complementary to the proposed research.

8 Conclusion

We developed an anti-jamming system that can enable wireless communication when a broadband and high power reactive jammer is present. The designed system delivers information by harnessing the reaction time of a reactive jammer. It does not assume a reactive jammer with limited spectrum coverage and transmit power, and thus can be used in scenarios where traditional approaches fail. We implemented a prototype of such system based on GNU Radio. Our results showed that the prototype achieved a reasonable throughput when 802.11 DSSS and GNURadio benchmark were completely disabled by the jammer.

Acknowledgments

This work is supported by the Army Research Office under grant W911NF-14-1-0324. Yao Liu is the corresponding author. An earlier version of the work was published in the 31st IEEE Conference on Computer Communications (INFOCOM'12).

References

[1] GNU Radio-The GNU Software Radio. (2014). [Online]. Available: http://www.gnu.org/software/gnuradio/

[2] Reactive jamming technologies. (2012). [Online]. Available: http://www.ece.gatech.edu/academic/courses/ece4007/08fall/ece4007l02/lm5/jammer.doc

[3] L. Baird, W. Bahn, and M. Collins, "Jam-resistant communication without shared secrets through the use of concurrent codes," US Air Force Academy, Colorado Springs, CO, USA, Tech. Rep. USAFA-TR-2007-01, U.S. Air Force Academy, 2007.

[4] A. J. Berni and W. D. Greeg, "On the utility of chirp modulation for digital signaling," *IEEE Trans. Commun.*, vol. C-21, no. 6, pp. 748–751, Jun. 1973.

[5] V. Brik, S. Banerjee, M. Gruteser, and S. Oh, "Wireless device identification with radiometric signatures," in *Proc. 14th ACM Int. Conf. Mobile Comput. Netw.*, 2008, pp. 116–127.

[6] D. Cabric, A. Tkachenko, and R. W. Brodersen, "Experimental study of spectrum sensing based on energy detection and network cooperation," in *Proc. 1st Int. Workshop Technol. Policy Accessing Spectrum*, 2006, p. 12.

[7] J. Chiang and Y. Hu, "Extended abstract: Cross-layer jamming detection and mitigation in wireless broadcast networks," in *Proc. ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, 2007, pp. 346–349.

[8] J. Chiang and Y. Hu, "Dynamic jamming mitigation for wireless boradcast networks," in *Proc. IEEE Int. Conf. Comput. Commun.*, 2008, pp. 1211–1219.

[9] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed. Cambridge, MA, USA: MIT Press, 2001.

[10] B. Danev, H. Luecken, S. Capkun, and K. E. Defrawy, "Attacks on physical-layer identification," in *Proc. 3nd ACM Conf. Wireless Netw. Secur.*, Mar. 2010, pp. 89–98.

[11] A. Goldsmith, *Wireless Communications*. Cambridge, U.K.: Cambridge Univ. Press, Aug. 2005.

[12] S. Hengstler, D. P. Kasilingam, and A. H. Costa, "A novel chirp modulation spread spectrum technique for multiple access," in *Proc. IEEE Int. Symp. Spread Spectrum Techn. Appl.*, Sep. 2002, pp. 73–77.

[13] H. Kim and K. G. Shin, "In-band spectrum sensing in cognitive radio networks: Energy detection or feature detection?" in *Proc. 14th ACM Int. Conf. Mobile Comput. Netw.*, 2008, pp. 14–25.

[14] L. Lazos, S. Liu, and M. Krunz, "Mitigating control-channel jamming attacks in multi-channel ad hoc networks," in *Proc. 2nd ACM Conf. Wireless Netw. Secur.*, Mar. 2009, pp. 169–180.

[15] M. Li, I. Koutsopoulos, and R. Poovendran, "Optimal jamming attacks and network defense policies in wireless sensor networks," in *Proc. IEEE Int. Conf. Comput. Commun.*, 2007, pp. 1307–1305.

[16] A. Liu, P. Ning, H. Dai, Y. Liu, and C. Wang, "Defending DSSS-based broadcast communication against insider jammers via delayed seed-disclosure," in *Proc. 26th Annu. Comput. Secur. Appl. Conf.*, Dec. 2010, pp. 367–376.

[17] Y. Liu and P. Ning, "BitTrickle: Defending against broadband and high-power reactive jamming attacks," Comput. Sci. Dept., NC State University, Raleigh, NC, USA, Tech. Rep. TR-2011-17, Jul. 2011.

[18] Y. Liu and P. Ning, "Enhanced wireless channel authentication using time-synched link signature," in *Proc., Mini-Conf.*, 2012, pp. 2636–2640.

[19] Y. Liu, P. Ning, H. Dai, and A. Liu, "Randomized differential DSSS: Jamming-resistant wireless broadcast communication," in *Proc. IEEE INFOCOM*, 2010, pp. 1–9.

[20] Ettus Research LLC, The USRP product family products and daughter boards. (2011). [Online]. Available: http://www.ettus.com/products

[21] A. Papoulis, *Probability, Random Variables, and Stochastic Processes*. New York, NY, USA: Mc-Graw Hill, 1984.

[22] N. Patwari and S. K. Kasera, "Robust location distinction using temporal link signatures," in *Proc. 13th Annu. ACM Int. Conf. Mobile Comput. Netw.*, New York, NY, USA, 2007, pp. 111–122.

[23] R. Poisel, *Modern Communications Jamming Principles Techniques*. Norwood, MA, USA: Artech House Publishers, 2006.

[24] S. O. Rice, "Mathematical analysis of random noise," *Bell Syst. Tech. J.*, vol. 24, pp. 46–156, 1945.

[25] C. Pöpper, M. Strasser, and S. Čapkun, "Jamming-resistant broadcast communication without shared keys," in *Proc. USENIX Security Symp.*, 2009, pp. 231–248.

[26] C. Pöpper, M. Strasser, and S. Čapkun, "Anti-jamming broadcast communication using uncoordinated spread spectrum techniques," *IEEE J. Sel. Areas Commun.*, vol. 28, vol. 5, pp. 703–715, Jun. 2010.

[27] R. A. Scholtz, "Multiple access with time hopping impulse modulation," in *Proc. IEEE MILCOM Conf.*, 1993, pp. 447–450.

[28] R. A. Scholtz, *Spread Spectrum Communications Handbook*. New York, NY, USA: McGraw-Hill, 2001.

[29] S. Shellhammer, *An ATSC Detector Using Peak Combining*, IEEE 802.22-06/0243r0, Nov. 2006.

[30] S. Shellhammer, S. Shankar, N. R. Tandra, and J. Tomcik, "Performance of power detector sensors of DTV signals in IEEE 802.22 WRANs," presented at the 1st Int. Workshop Technology and Policy Accessing Spectrum, New York, NY, USA, 2006.

[31] A. Springer, W. Gugler, M. Huemer, L. Reindl, C. C. W. Ruppel, and R. Weigel, "Spread spectrum communication using chirp signals," in *Proc. IEEE/AFCEA EUROCOMM Conf.*, May 2000, pp. 166–170.

[32] M. Strasser, B. Danve, and S. Capkun, "Detection of reactive jamming in sensor networks," *ACM Trans. Sensor Netw.*, vol. 7, pp. 1–29, Aug. 2010.

[33] M. Strasser, C. Pöper, S. Čapkun, and M. Čagalj, "Jamming-resistant key establishment using uncoordinated frequency hopping," in *Proc. IEEE Symp. Secur. Privacy*, 2008, pp. 64–78.

[34] M. Strasser, C. Pöpper, and S. Čapkun, "Efficient uncoordinated FHSS anti-jamming communication," in *Proc. 10th ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, 2009, pp. 207–218.

[35] D. Torrieri, *Principles of Spread-Spectrum Communication Systems*. New York, NY, USA: Springer, 2004.

[36] M. Win and R. Scholtz, "Impulse radio: How it works," *IEEE Commun. Lett.*, vol. 2, no. 2, pp. 36–38, Feb. 1998.

[37] P. Tague, M. Li, and R. Poovendran, "Probabilistic mitigation of control channel jamming via random key distribution," in *Proc. IEEE 18th Int. Symp. Personal, Indoor Mobile Radio Commun.*, 2007, pp. 1–5.
[38] W. Xu, K. Ma, W. Trappe, and Y. Zhang, "Jamming sensor networks: Attack and defense strategies," *IEEE Netw.*, vol. 20, no. 3, pp. 41–47, May/Jun. 2006.
[39] W. Xu, W. Trappe, and Y. Zhang, "Channel surfing: Defending wireless sensor networks from jamming and interference," in *Proc. 6th Int. Conf. Inf. Process. Sen. Netw.*, 2007, pp. 499–508.
[40] W. Xu, W. Trappe, and Y. Zhang, "Anti-jamming timing channels for wireless networks," in *Proc. 1st ACM Conf. Wireless Netw. Secur.*, 2008, pp. 203–213.
[41] W. Xu, W. Trappe, Y. Zhang, and T. Wood, "The feasibility of launching and detecting jamming attacks in wireless networks," in *Proc. ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, 2005, pp. 46–57.
[42] W. Xu, T. Wood, W. Trappe, and Y. Zhang, "Channel surfing and spatial retreats: Defenses against wireless denial of service," in *Proc. 3rd ACM Workshop Wireless Secur.*, 2004, pp. 80–89.
[43] J. Zhang, M. H. Firooz, N. Patwari, and S. K. Kasera, "Advancing wireless link signatures for location distinction," in *Proc. 14th ACM Int. Conf. Mobile Comput. Netw.*, New York, NY, USA, 2008, pp. 26–37.

Song Fang received the BS degree in information engineering from the South China University of Technology, Guangzhou, China, in July 2011, and the MS degree in communication and information engineering from the Beijing University of Posts and Telecommunications, Beijing, China, in March 2014. From August 2013, he is working toward the PhD degree in computer science at the University of South Florida, Tampa, FL. His research interests are in the area of network and system security. His current research mainly focuses on utilizing novel physical layer techniques to improve the security in wireless networks.

Yao Liu received the PhD degree in computer science from North Carolina State University in 2012. She is currently an assistant professor in the Department of Computer Science and Engineering, University of South Florida, Tampa, FL. Her research is related to computer and network security, with an emphasis on designing and implementing defense approaches that protect emerging wireless technologies from being undermined by adversaries. Her research interest also lies in the security of cyber-physical systems, especially in smart grid security. Her research work has appeared in premier journals and conferences including the *ACM Transactions on Information and Systems Security*, IEEE Symposium on Security and Privacy (IEEE S&P), ACM Conference on Computer and Communications Security (CCS), and IEEE International Conference on Computer Communications (INFOCOM). She received the Best Paper Award for the Seventh IEEE International Conference on Mobile Ad-Hoc and Sensor Systems.

Peng Ning (M'01-SM'12) received the BS degree in information sciences from the University of Science and Technology of China (USTC), Hefei, China, in 1994, the ME degree in communications and electronics systems from USTC, Graduate School in Beijing, Beijing, China, in 1997, and the PhD degree in information technology from George Mason University, Fairfax, VA, in 2001. He is a professor of computer science in NC State University, where he also serves as the technical director for Secure Open Systems, a recipient of the US National Science Foundation (NSF) Initiative (SOSI). He received the US NSF CAREER Award in 2005. He is currently the secretary/treasurer of the ACM Special Interest Group on Security, Auditing, and Control (SIGSAC), and is on the executive committee of ACM SIGSAC. He is an editor for *Springer Briefs in Computer Science*, responsible for Briefs on information security. He has served or is serving on the editorial boards of several international journals, including the *ACM Transactions on Sensor Networks*, *Journal of Computer Security*, *Ad-Hoc Networks*, *Ad-Hoc & Sensor Networks: An International Journal*, *International Journal of Security and Networks*, and IET Proceedings Information Security. He also served as the program chair or co-chair for ACM SASN '05, ICICS '06 and ESORICS '09, ICDCS-SPCC '10, and NDSS '13, the general chair of ACM CCS '07 & '08, and the program vice chair for ICDCS '09 & '10-Security and Privacy Track. He served on the Steering Committee of ACM CCS from 2007 to 2011, and is a founding Steering Committee member of ACM WiSec and ICDCS SPCC. His research has been supported by the US NSF, Army Research Office (ARO), the Advanced Research and Development Activity (ARDA), IBM Research, SRI International, and the NCSU/Duke Center for Advanced Computing and Communication (CACC). He is a senior member of the ACM, ACM SIGSAC, and the IEEE.