Randomized Differential DSSS: Jamming-Resistant Wireless Broadcast Communication

Yao Liu, Peng Ning, Huaiyu Dai, An Liu
North Carolina State University, Raleigh, NC 27695
{yliu20, pning, hdai, aliu3}@ncsu.edu

***Abstract*—Jamming resistance is crucial for applications where reliable wireless communication is required. Spread spectrum techniques such as Frequency Hopping Spread Spectrum (FHSS) and Direct Sequence Spread Spectrum (DSSS) have been used as countermeasures against jamming attacks. Traditional anti-jamming techniques require that senders and receivers share a secret key in order to communicate with each other. However, such a requirement prevents these techniques from being effective for anti-jamming *broadcast* communication, where a jammer may learn the shared key from a compromised or malicious receiver and disrupt the reception at normal receivers.**

In this paper, we propose a Randomized Differential DSSS (RD-DSSS) scheme to achieve anti-jamming broadcast communication without shared keys. RD-DSSS encodes each bit of data using the correlation of unpredictable spreading codes. Specifically, bit "0" is encoded using two different spreading codes, which have low correlation with each other, while bit "1" is encoded using two identical spreading codes, which have high correlation. To defeat reactive jamming attacks, RD-DSSS uses multiple spreading code sequences to spread each message and rearranges the spread output before transmitting it. Our theoretical analysis and simulation results show that RD-DSSS can effectively defeat jamming attacks for anti-jamming broadcast communication without shared keys.

I. Introduction

Wireless communications is vulnerable to jamming attacks due to the shared use of wireless medium. A jammer can simply take advantage of a radio frequency (RF) device (e.g., a waveform generator) to transmit signals in the wireless channel. As a result, signals of the jammer and the sender collide at the receiver and the signal reception process is disrupted. Therefore, jamming resistance is crucial for applications where reliable wireless communications is required.

Spread spectrum techniques have been used as countermeasures against jamming attacks. Direct Sequence Spread Spectrum (DSSS), Frequency Hopping Spread Spectrum (FHSS), and Chirp Spread Spectrum (CSS) are three common forms of spread spectrum techniques [12]. In classic spread spectrum techniques, senders and receivers need to pre-share a secret key, with which they can generate identical hopping patterns, spreading codes, or timing of pulses for communication. However, if a jammer knows the secret key, the jammer can easily jam the communication by following the hopping patterns, spreading codes, or timing of pulses used by the sender.

There have been a few recent attempts to remove the dependency of jamming-resistant communications on pre-shared keys [1], [13], [16]–[18]. Strasser et al. developed an Uncoordinated Frequency Hopping (UFH) technique to allow two nodes that do not have any common secret to establish a secret key for future FHSS communication in presence of a jammer [17]. Strasser et al. [18] and Slater et al. [16] later independently proposed to use similar coding techniques to improve the robustness and efficiency in UFH. These works successfully remove the requirement of pre-shared keys in point-to-point FHSS communication.

Unfortunately, UFH and its variations [16]–[18] cannot be directly used for *broadcast* communication, since their primary objective is to establish a pairwise key between two parties. Indeed, any spread spectrum communication system that requires a shared key, either pre-shared or established at the initial stage of the communication, cannot be used for broadcast communication where there may be insider jammers. Any malicious receiver, who knows the shared key, may use the key to jam the communication.

To address this problem, researchers recently investigated how to enable jamming-resistant broadcast communication without shared keys [1], [13]. Baird et al. proposed a coding approach to encode data to be transmitted into "marks" (e.g., short pulses at different times) that can be decoded without any prior knowledge of keys [1]. However, the decoding process of the method is inherently sequential (i.e., the decoding of the next bit depends on the decoded values of the previous bits). Though it works with short pulses in the time domain, the method cannot be extended to DSSS or FHSS without significantly increasing the decoding cost.

Pöpper et al. developed an Uncoordinated Direct Sequence Spread Spectrum (UDSSS) approach, which avoids jamming by randomly selecting a spreading code sequence from a pool of code sequences. However, as indicated in [13], UDSSS is vulnerable to reactive jamming attacks. It is demonstrated in [13] that when the jammer does not have sufficient computational power to infer the spreading sequence quickly enough, UDSSS still provides good enough jamming resistance. However, when the jammer has sufficient computational power, UDSSS fails to provide strong guarantee of jamming resistance.

In this paper, we propose a Randomized Differential DSSS (RD-DSSS) scheme for DSSS-based broadcast communication. RD-DSSS relies completely on publicly known spreading codes, and thus does not require any shared key among the sender and the receivers. It does not suffer from the vulnerabilities of previous solutions, and thus is a good candidate to enable anti-jamming broadcast communication even when there are potentially compromised or malicious receivers.

RD-DSSS employs spreading codes of traditional DSSS systems to spread a message for reducing the impacts of jamming signals. However, unlike traditional DSSS, RD-DSSS encodes each bit of data using the correlation of unpredictable spreading codes. Specifically, bit “0” is encoded using two different spreading codes, which have low correlation with each other, while bit “1” is encoded using two identical spreading codes, which have high correlation. As a result, sender and receivers do not need to share any common key for communication.

In addition, RD-DSSS uses a pool of spreading code sequences to enhance its reliability and tolerate reactive jamming attacks. A sender spreads each message using multiple spreading code sequences and rearranges the spread result before transmitting it. A receiver, after receiving the entire message, can reverse the rearrangement of the spread result and then recover the original message. However, a jammer has to disrupt the communication at the same time as the message transmission. It is thus very difficult for a jammer to derive the correct spreading sequences on the fly and jam the message transmission accordingly.

The contribution of this paper is two-fold: First, we develop a new RD-DSSS scheme to both remove the requirement of shared keys for DSSS communication and overcome the weaknesses of previous solutions (e.g., vulnerability to reactive jamming attacks). Second, we evaluate the performance and effectiveness of RD-DSSS in presence of various kinds of jamming attacks through both theoretical analysis and simulation.

The rest of the paper is organized as follows. Section II gives background information on DSSS. Section III discusses the system and threat models. Section IV presents the proposed RD-DSSS scheme. Sections V and VI provide the performance and security evaluation, respectively. Section VII describes related work, and Section VIII concludes this paper.

II. Background on DSSS

DSSS is a modulation method applied to digital signals [6]. It increases the signal bandwidth to a value much larger than needed to transmit the underlying information [6]. In DSSS, spreading codes that are independent of the original signal are used to achieve the goal of bandwidth expansion. Both a sender and a receiver agree on a spreading code, which is regarded as a shared secret between them. A spreading code is usually a sequence of bits valued 1 and -1 (polar) or 1 and 0 (non-polar), which has noise-like properties. In this paper, without loss of generality, we consider spreading codes with polarity. Typical spreading codes are pseudo-random codes, Walsh-Hadamard codes and Gold codes [15]. Figure 1 shows a simple communication framework of DSSS.

Transmission Process: A sender usually first encodes the original message using an error correction code (ECC) to enhance the reliability of the communication. The sender then uses DSSS to spread the encoded message into a binary bitstream called *chips*, and uses a D/A converter to transform the chips into analog square wave signal called baseband signal. The baseband signal is multiplied by a cosine signal with a certain frequency, resulting in the RF signal. The RF signal is fed to the antenna to transmit in the wireless channel.

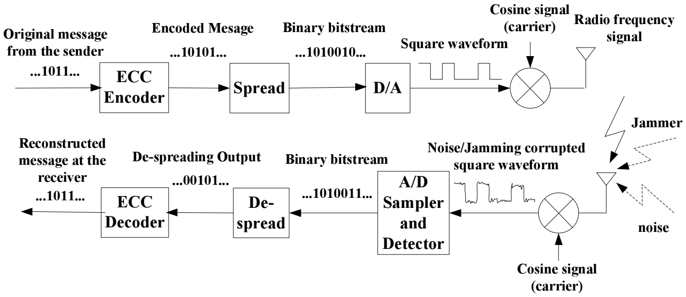

Fig. 1. A simple communication framework of DSSS

Reception Process: Upon hearing the RF signal, the receiver performs similar tasks in the reverse order. The receiver first recovers the baseband signal by multiplying a cosine signal as used by the sender, applies an A/D sampler and a detector to transform the baseband signal into chips, and uses DSSS to de-spread the chips. The receiver finally decodes the de-spread result to reconstruct the original message.

Spreading and De-spreading: Spreading and de-spreading are two important functions of a DSSS system. In spreading, a sender multiplies each bit of the original message with a spreading code to get the spread message. For example, if the original message is “01” and the spreading code is $-1+1+1-1$, then the sender converts the original message “01” into the polar form $-1+1$, and multiplies -1 and $+1$ with spreading code $-1+1+1-1$, respectively. The spread message is thus $+1-1-1+1-1+1+1-1$.

It is necessary to understand the notion of correlation to see how de-spreading works. Given two spreading codes $\mathbf{f} = f_1, .., f_k$ and $\mathbf{g} = g_1, .., g_k$, where f_i and g_i are valued -1 or 1 for $1 \le i \le k$, the correlation of $\mathbf{f}$ and $\mathbf{g}$ is $\mathbf{f}\cdot\mathbf{g} = \frac{1}{k}\sum_{i=1}^{k} f_i g_i$. Note that the correlation of two identical spreading codes is 1.

In de-spreading, the receiver uses a local replica of the spreading code and synchronizes it with the received message [15]. Then the receiver correlates the received message with the replica to generate the de-spreading output. For example, suppose the received message is $+1-1-1+1-1+1+1-1$ and the local replica of the spreading code is $-1+1+1-1$ at the receiver side. The receiver aligns $-1+1+1-1$ with the first 4 chips of the received message (i.e., $+1-1-1+1$) and correlates them to get bit -1 (i.e., “0” in non-polar form).

Synchronization: In DSSS systems, a receiver needs to identify the beginning of a message sent by the sender from the received signal. In general, the sender and the receiver agree on a known code such as Barker Code [2] that has good autocorrelation property (i.e., the correlation between a code and its shifted value is low). The sender transmits the code just before the spread message. The receiver correlates the received signal with the code using a sliding window approach; the position where the correlation is maximum indicates the beginning of the message [4], [7], [15].

DSSS allows receivers to reconstruct the desired signal with efficiency and at the same time distributes the energy of wireless interferences (e.g., narrow band jamming signals) to the entire bandwidth. Therefore, DSSS provides good anti-jam protection for wireless communications.

III. System and Threat Models

Our system consists of a sender and multiple receivers. The sender and receivers are wireless devices that can transmit and receive RF signals. We assume that there are jammers that inject noise signals into the wireless channel. The goal of the jammers is to prevent communication between the sender and the receivers. We assume a jammer has the following capabilities: (1) He is aware of the target communication systems (e.g., protocols and anti-jamming strategy); (2) he can eavesdrop the communication between the sender and the receivers; (3) he can transmit on the wireless channel to interfere with the physical transmission and reception of the desired wireless signals; (4) he can inject fake messages into the channel; and (5) he can perform real-time analysis to identify the spreading code used to spread each bit data right after its transmission. However, we assume that if a jammer does not know the code for spreading any 1-bit data, he cannot jam the transmission of it.

IV. Randomized Differential DSSS

Similar to traditional DSSS, RD-DSSS takes advantage of the correlation properties of spreading codes to achieve anti-jamming communication. The transmission, reception, and synchronization of a RD-DSSS system are the same as those of a traditional DSSS system except for the spreading and de-spreading processes. Due to the change in these processes, RD-DSSS does not require a sender and its receivers to share a secret key. In the following, we focus on spreading and de-spreading processes of RD-DSSS.

A. Basic Scheme

In RD-DSSS, the sender and the receivers share a set of spreading codes, which we call the *spreading code set*. There should be low correlation between any two codes in the spreading code set. A sender encodes each bit of data using the correlation of two unpredictable spreading codes. Specifically, bit "0" is encoded using two different spreading codes, which have low correlation with each other, while bit "1" is encoded using two identical spreading codes, which have high correlation. A sender can randomly choose different pairs of codes from the code set for different bits in a message. A receiver de-spreads a received message by computing correlation of the two codes for each bit. High correlation and low correlation are translated into "1" and "0", respectively.

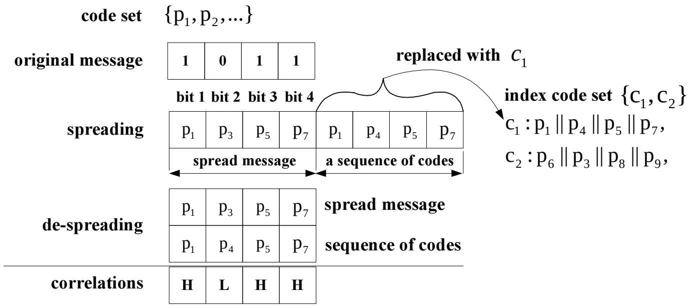

Fig. 2. An example of the basic RD-DSSS scheme.

Figure 2 shows an example, in which a sender transmits a 4-bit message "1011" to a receiver. The sender randomly chooses codes $\mathbf{p}_1$, $\mathbf{p}_4$, $\mathbf{p}_5$, and $\mathbf{p}_7$ from the spreading code set. Since bit 1 of the message is "1", the sender uses $\mathbf{p}_1$ twice to encode it. The second bit of the original message is "0". Thus, the sender uses any code different from $\mathbf{p}_4$ (i.e., $\mathbf{p}_3$ as shown in Figure 2) and $\mathbf{p}_4$ to encode it. Bits 3 and 4 are encoded similarly. As a result, the sender gets an encoded message $\underline{\mathbf{p}_1||\mathbf{p}_3||\mathbf{p}_5||\mathbf{p}_7}||\underline{\mathbf{p}_1||\mathbf{p}_4||\mathbf{p}_5||\mathbf{p}_7}$, in which the second half of the message are the codes selected by the sender earlier. For de-spreading, the receiver computes the correlations between the corresponding codes in the first and the second halves of the spread message (i.e., $\mathbf{p}_1 \cdot \mathbf{p}_1$, $\mathbf{p}_3 \cdot \mathbf{p}_4$, $\mathbf{p}_5 \cdot \mathbf{p}_5$, and $\mathbf{p}_7 \cdot \mathbf{p}_7$). High correlation and low correlation are translated into "1" and "0", respectively.

To reduce the communication overhead, we propose to have the sender and the receivers share a set of pre-defined spreading code sequences, which are formed by the concatenations of codes in the spreading code set. We associate each code sequence with a special spreading code called *index code*. The collection of all index codes is referred to as the *index code set*. We require that the correlation between two different index codes is low. Intuitively, a sender can transmit an index code (instead of the actual code sequence) to indicate the code sequence for spreading. For example, Figure 2 shows that there are two code sequences, which are represented by index codes $\mathbf{c}_1$ and $\mathbf{c}_2$, respectively. Instead of sending $\mathbf{p}_1||\mathbf{p}_4||\mathbf{p}_5||\mathbf{p}_7$ as the second half of the spread message, the sender simply transmits $\mathbf{c}_1$. The index code set and the spreading code set should have no overlap so that a receiver can easily distinguish between an index code and a regular spreading code.

In the following, we present the basic scheme in detail.

1) Code Set and Code Sequence Set: Let $\mathcal{P} = \{\mathbf{p}_1, ..., \mathbf{p}_n\}$ denote the spreading code set with n codes. As mentioned earlier, these codes should have low correlation with each other. There are multiple candidate codes for our scheme, such as Gold codes, Walsh-Hadamard codes, m sequences, and Kasami codes [15]. We assume each code in $\mathcal{P}$ is of length f and $\mathcal{P}$ is publicly known.

Let $\mathcal{C} = \{\mathbf{p}_{11}||...||\mathbf{p}_{1l}, ..., \mathbf{p}_{q1}||...||\mathbf{p}_{ql}\}$ denote the set of q pre-defined code sequences, where $\mathbf{p}_{ij}$ is a code randomly selected from $\mathcal{P}$ for $1 \leq i \leq q$ and $1 \leq j \leq l$. We assume $\mathcal{C}$ is known to the public. We associate an index code $\mathbf{c}_i$ with the i-th code sequence $\mathbf{p}_{i1}||...||\mathbf{p}_{il}$. Let $\mathcal{I} = \{\mathbf{c}_1, ..., \mathbf{c}_q\}$ be the set of index codes. We assume each index codes is of length g. Similar to $\mathcal{P}$ and $\mathcal{C}$, $\mathcal{I}$ is also known to the public.

To reduce storage overhead, we only store the index codes and generate each code sequence in $\mathcal{C}$ using its index code. One possible way is to use a pseudo-random generator (PRG) with $\mathbf{c}_i$ as the input to generate a sequence of indexes, which are then used to select codes from $\mathcal{P}$ to form a code sequence. For example, assume $\mathbf{c}_i$ is the index code of $\mathbf{p}_{i1}||\mathbf{p}_{i2}$, $n = 4$, and $PRG(\mathbf{c}_i) = (\underline{01}\ \underline{11}...)_2$. Thus, $\mathbf{p}_{i1} = \mathbf{p}_1$ and $\mathbf{p}_{i2} = \mathbf{p}_3$.

2) Spreading: Let $\mathbf{m} = m_1||...||m_l$ denote the original message to be transmitted. For spreading, a sender randomly chooses a code sequence from $\mathcal{C}$. Let $\mathbf{S} = \mathbf{s}_1||...||\mathbf{s}_l$ denote the chosen code sequence. The sender then generates the spread message based on the chosen code sequence $\mathbf{S}$. Let $\mathbf{F} = \mathbf{f}_1||...||\mathbf{f}_l$ denote the spread message. For $1 \leq i \leq l$, the sender generates $\mathbf{f}_i$ according to the following rule: if $m_i = 1$, $\mathbf{f}_i$ is the same as $\mathbf{s}_i$. Otherwise, $\mathbf{f}_i$ is an arbitrary code in $\mathcal{P}$ other than $\mathbf{s}_i$. Assume the index code of the chosen code sequence is $\mathbf{c}$. The sender appends $\mathbf{c}$ to the end of the spread message and transmits $\mathbf{F}||\mathbf{c}$ to the receiver.

3) De-spreading: For de-spreading, a receiver needs to identify the chosen code sequence. Suppose the received message is $\hat{\mathbf{F}}||\hat{\mathbf{c}}$, where $\hat{\mathbf{F}} = \hat{\mathbf{f}}_1||\hat{\mathbf{f}}_2||...||\hat{\mathbf{f}}_l$. The receiver computes correlations between each index code and $\hat{\mathbf{c}}$. Note that the correlation between two identical codes is the highest and reaches correlation peak. Thus, the receiver marks the index code that results in the highest correlation with $\hat{\mathbf{c}}$ as identified. Let $\mathbf{S} = \mathbf{s}_1||\mathbf{s}_2||...||\mathbf{s}_l$ denote the code sequence associated with the identified index code. The receiver then uses $\mathbf{S}$ to de-spread the received message.

To de-spread the received message, a receiver needs to compute the following correlations: $(\hat{\mathbf{f}}_1 \cdot \mathbf{s}_1)$, $(\hat{\mathbf{f}}_2 \cdot \mathbf{s}_2)$,..., $(\hat{\mathbf{f}}_l \cdot \mathbf{s}_l)$. Let $\hat{\mathbf{m}} = \hat{m}_1||...||\hat{m}_l$ denote the de-spreading output. $\hat{\mathbf{m}}$ can be generated according to the following rule: if $(\hat{\mathbf{f}}_i \cdot \mathbf{s}_i)$ is larger than or equal to a threshold t, $\hat{m}_i = 1$. Otherwise, $\hat{m}_i = 0$.

The threshold t is an important parameter. In this paper, we derive the threshold t as the value that minimizes the probability of decision error when the transmitted signal is polluted by additive white Gaussian noise signal, which is a generally assumed jamming signal in wireless communications [15], [20]. This threshold is given in lemma 1.

Lemma 1: Given additive Gaussian noise, the probability of decision error is minimized for threshold $t = \frac{1}{2}(1+\rho)$, where $\rho = \frac{1}{\binom{n}{2}} \sum_{\forall \mathbf{p}_i, \mathbf{p}_j \in \mathcal{P}} (\mathbf{p}_i \cdot \mathbf{p}_j)$ (i.e., ρ is the average of the correlations between two codes in $\mathcal{P}$).

Proof: Consider $X = (x_1, ..., x_l)$ and $\hat{X} = (\hat{x}_1, ..., \hat{x}_l)$, where $x_i = \mathbf{f}_i \cdot \mathbf{s}_i$ and $\hat{x}_i = \hat{\mathbf{f}}_i \cdot \mathbf{s}_i$ for $1 \leq i \leq l$. Let $\mathbf{n}_i = (n_{i1}, ..., n_{if})$ denote the errors introduced by wireless interference. Thus, $\hat{\mathbf{f}}_i = \mathbf{f}_i + \mathbf{n}_i$. We assume the elements of $\mathbf{n}_i$ are independent and identically distributed (i.i.d) Gaussian random variables with mean value 0 and variance σ^2. Let $\mathbf{e} = (e_1, ..., e_l)$ denote $X - \hat{X}$, where $e_i = x_i - \hat{x_i}$, and let $\mathbf{s}_i = s_{i1}||s_{i2}||...||s_{if}$, where s_{ij} is valued -1 or $+1$. Note that $\hat{x}_i = \hat{\mathbf{f}}_i \cdot \mathbf{s}_i = (\mathbf{f}_i + \mathbf{n}_i) \cdot \mathbf{s}_i = x_i + \mathbf{n}_i \cdot \mathbf{s}_i$. Thus, $e_i = \mathbf{n}_i \cdot \mathbf{s}_i = \frac{1}{f} \sum_{j=1}^{f} n_{ij} s_{ij}$. According to the properties of Gaussian variables [10], e_i is i.i.d. Gaussian random variables with mean value 0 and variance σ^2.

When $m_i = 1$, $\hat{x}_i = x_i + e_i = 1 + e_i$. The probability density function (pdf) of $1 + e_i$ is $f_1(\hat{x}_i) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(\hat{x}_i - 1)^2}{2\sigma^2}}$. When $m_i = 0$, $\hat{x}_i = x_i + e_i = \rho + e_i$, the pdf of $\rho + e_i$ is $f_0(\hat{x}_i) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(\hat{x}_i - \rho)^2}{2\sigma^2}}$. Assume the sender transmits “1” or “0” with probability $\frac{1}{2}$. Thus, the probability of decision error is $p_e = \frac{1}{2}(\int_{-\infty}^{t} f_1(\hat{x}_i) d\hat{x}_i + \int_{t}^{\infty} f_0(\hat{x}_i) d\hat{x}_i) = \frac{1}{2}(1 + \frac{1}{2} erf(\frac{t-1}{\sqrt{2}\sigma}) - \frac{1}{2} erf(\frac{t-\rho}{\sqrt{2}\sigma}))$, where erf is the Error Function $erf(w) = \frac{2}{\sqrt{\pi}} \int_0^w e^{-y^2} dy$ [15]. To minimize p_e, we let the derivation of p_e be 0 (i.e.,$\frac{dp_e}{dt} = 0$) and solves t. We can obtain $t = \frac{1}{2}(1+\rho)$ and the minimized probability of decision error is $p_{emin} = \frac{1}{2} + \frac{1}{2} erf(\frac{\rho - 1}{2\sqrt{2}\sigma})$. ■

4) Pros and Cons: The basic scheme of RD-DSSS provides resistance against wireless interference for broadcast communication. The sender randomly chooses a code sequence to spread each original message, and no one except for the sender knows the code sequence before the communication. Thus, the requirement of shared keys is removed, gaining reliability and scalability for broadcast communication systems. Furthermore, the sender associates an index code with each code sequence and transmits the index code instead of the actual code sequence. Thus, the communication overhead is reduced.

However, the basic scheme is still vulnerable to reactive jamming attacks. Recall that bit “1” is encoded by two identical codes. Thus, the spread message and the chosen code sequence may share many similar codes, and the correlation between them may be high. After observing the first r codes of a message being transmitted, the jammer can compute the correlation between the observed r codes and the first r codes of each code sequence in $\mathcal{C}$. The code sequence resulting in the highest correlation is probably the code sequence chosen by the sender. The jammer can then spread a fake message using the identified code sequence to jam the transmission of the remaining message.

B. Enhanced Scheme: Defending against Reactive Jamming

The basic scheme is vulnerable to reactive jamming attacks, since the correlation between the spread message and the chosen code sequence is usually high. In the enhanced scheme, we propose two mechanisms to reduce the correlation and improve the basic scheme.

First, after generating the spread message as in the basic scheme, the sender permutes all codes of the spread message. Thus, even if the i-th bit of the original message is 1, the i-th observed code is not the same as the i-th code of the chosen code sequence, resulting in reduced correlation. For example, in Figure 2, the spread message is $\mathbf{p}_1||\mathbf{p}_3||\mathbf{p}_5||\mathbf{p}_7$ and the chosen code sequence is $\mathbf{p}_1||\mathbf{p}_4||\mathbf{p}_5||\mathbf{p}_7$. Assume the correlation between two different codes is 0. Thus, the correlation between the spread message and the chosen code sequence is $\frac{1+0+1+1}{4} = 0.75$. However, if we permute the spread message and assume the result is $\mathbf{p}_7||\mathbf{p}_5||\mathbf{p}_1||\mathbf{p}_3$, then the correlation between the permuted spread message and the chosen code sequence is reduced to 0. Second, the sender spreads the message using k code sequences in $\mathcal{C}$. As a result, a reactive jammer must know all k code sequences in order to launch reactive jamming successfully.

In the following, we present the enhanced scheme in detail.

1) Spreading: A sender randomly chooses k code sequences from $\mathcal{C}$ and uses them to generate the spread message. Suppose code sequences $\mathbf{c}_{i_1} = \mathbf{p}_{i_1 1}||...||\mathbf{p}_{i_1 l}, ..., \mathbf{c}_{i_k} = \mathbf{p}_{i_k 1}||...||\mathbf{p}_{i_k l}$ are selected. For $1 \leq j \leq l$, let $\mathcal{A}_j = \{\mathbf{p}_{i_1 j}, ..., \mathbf{p}_{i_k j}\}$. That is, $\mathcal{A}_j$ consists of the j-th codes of all chosen code sequences. The sender generates the spread message $\mathbf{F} = \mathbf{f}_1||\mathbf{f}_2||...||\mathbf{f}_l$ according to the following rule: If $m_j = 1$, chooses $\mathbf{f}_j$ as any code in $\mathcal{A}_j$ randomly. Otherwise, chooses $\mathbf{f}_j$ as any code not in $\mathcal{A}_j$.

The sender then maps each set $\mathcal{A}_j$ to an integer number and sorts those numbers in descending order to generate a permutation. For example, let $h(\mathcal{A}_j)$ denote the mapping function that converts set $\mathcal{A}_j$ into an integer. Assume $l = 3$ (i.e., $\mathbf{F} = \mathbf{f}_1||\mathbf{f}_2||\mathbf{f}_3$), and $h(\mathcal{A}_1) = 55$, $h(\mathcal{A}_2) = 67$, and $h(\mathcal{A}_3) = 23$. The sorting result is $h(\mathcal{A}_2), h(\mathcal{A}_1), h(\mathcal{A}_3)$, and thus we use $\mathbf{f}_2||\mathbf{f}_1||\mathbf{f}_3$ as the permuted message.

Given $\mathcal{A}_j = \{\mathbf{p}_{i_1 j}, ..., \mathbf{p}_{i_k j}\}$, in this paper, we simply compute $h(\mathcal{A}_j)$ as $h(\mathcal{A}_j) = ind(\mathbf{p}_{i_1 j}) \times 10^{k-1} + ind(\mathbf{p}_{i_2 j}) \times 10^{k-2} ... + ind(\mathbf{p}_{i_k j})$, where $ind(\mathbf{p})$ is the index of the code $\mathbf{p}$. For example, if $k = 2$ and $\mathcal{A}_j = \{\mathbf{p}_3, \mathbf{p}_1\}$, then $h(\mathcal{A}_j) = 31$.

Finally, the sender appends index codes $\mathbf{c}_{i_1}, ..., \mathbf{c}_{i_k}$ to the end of the permuted spread message $\mathbf{F}_p$, and transmits $\mathbf{F}_p||\mathbf{c}_{i_1}||...||\mathbf{c}_{i_k}$ to the receivers.

2) De-spreading: A receiver recovers the original message in two steps: (1) identify the index codes appended by the sender, and (2) process the received message.

Let $\hat{\mathbf{F}}_p||\hat{\mathbf{c}}_{i_1}||...||\hat{\mathbf{c}}_{i_k}$ denote the received message, where $\hat{\mathbf{F}}_p$ is the permuted spread message and $\hat{\mathbf{c}}_{i_1}, ..., \hat{\mathbf{c}}_{i_k}$ are the index codes. The receiver needs to identify the index codes that are appended by the sender. For $1 \leq u \leq k$, the receiver computes the correlations between each index code and $\hat{\mathbf{c}}_{i_u}$ (i.e., $\mathbf{c}_1 \cdot \hat{\mathbf{c}}_{i_u}, ..., \mathbf{c}_q \cdot \hat{\mathbf{c}}_{i_u}$). The index code that results in the highest correlation with $\hat{\mathbf{c}}_{i_u}$ is marked as identified.

Let $\mathbf{c}_{i_1{}'}, ..., \mathbf{c}_{i_k{}'}$ denote the identified index codes and $\mathbf{p}_{i_1{}'1}||...||\mathbf{p}_{i_1{}'l}, ..., \mathbf{p}_{i_k{}'1}||...||\mathbf{p}_{i_k{}'l}$ denote the corresponding code sequences, where $1 \leq i_1{}', ..., i_k{}' \leq q$. For $1 \leq j \leq l$, let $\mathcal{A}'_j = \{\mathbf{p}_{i_1{}'j}, ..., \mathbf{p}_{i_k{}'j}\}$. The receiver computes the permutation based on $h(\mathcal{A}'_1), ..., h(\mathcal{A}'_l)$, and then recovers the original code order of the spread message. Let $\hat{\mathbf{f}}'_1||...||\hat{\mathbf{f}}'_l$ denote the result. The receiver then computes $\hat{\mathbf{m}}$ as follows: If $\exists\ \mathbf{p} \in \mathcal{A}'_j$ s.t. $\hat{\mathbf{f}}_j \cdot \mathbf{p} \geq t$, let $\hat{m}_j = 1$. Otherwise, let $\hat{m}_j = 0$.

3) Ability to Deal with Reactive Jamming: To perform reactive jamming attacks, a jammer needs to find code sequences used for message transmission. However, the correlation between the observed codes and any of code sequences chosen by the sender is not guaranteed to be high, since the code order in the spread message is rearranged.

The jammer may do an exhaustive search over all possible combinations of k code sequences in $\mathcal{C}$ to find those chosen by the user. The number of correlations the jammer should compute is $k\binom{q}{k}$. However, even with all the correlation results, the jammer still cannot determine which code sequences have been chosen, because (1) transmission of bit "0" can use any code other than those in the corresponding position of the selected code sequences and (2) multiple code sequences could be used for encoding. Moreover, the jammer must finish the computation within the transmission time of a single message in order to launch reactive jamming. For example, if $q = 13,500$ and $k = 5$, the number of correlations the attacker needs to compute is $5 \times \binom{13,500}{5} > 2^{64}$. Assume the length of the original message is $1,024$ bits and that of a spreading code is 256 chips. For a 802.11 wireless device with 11 Mbps data rate, the time for transmitting a spread message is $\frac{1,024 \times 256}{11 \times 10^6} \approx 0.024$ second. This means the jammer must finish such a huge amount of computation within 0.024 second. We provide detailed analysis to show the effectiveness of RD-DSSS in tolerating reactive jamming attacks in Section VI-B2.

TABLE I
COMPUTATION TIME (MILLISECONDS)

Operations	# of computations	computation time
Mapping	1,024	$t_h = 0.114$
Sort	1	$t_s = 4.654$
Permutation	1	$t_p = 0.005$
Inverse Permutation	1	$t_{ip} = 1.100$
Correlation	$13,500 \times 3$	$t_c = 8.989$
Total	41527	$t_a = 14.862$

V. Performance Overheads

Computational Overhead: RD-DSSS systems require additional operations compared with traditional DSSS systems (e.g., computing correlations to identify index codes). Let t_h, t_s, t_p, t_{ip}, and t_c denote the computation time for performing mapping functions, sorting, permutation, inverse permutation, and computing correlations to identify index codes. The additional computation overhead t_a introduced by RD-DSSS is $t_a = t_h + t_s + t_p + t_{ip} + t_c$.

To get an intuitive feeling of the computational overhead, we did an experiment to test the time required by these additional operations. The system settings in the experiment are as follows: The message length is 1,024 bits, the size of $\mathcal{P}$ is 100, the size of $\mathcal{C}$ is 13,500, the length of each index code is 512 chips, the number of code sequences chosen by the sender is 3, and the length of each code in $\mathcal{P}$ is 256 chips. Table I shows the computation time of those operations performed on a computer with a 3.40 GHz Intel Pentium 4 CPU and 1.5 GB memory. All those operations together can be finished within 15 milliseconds. In practice, the correlation operation is inherently parallel (i.e., dot product of two vectors) and be finished efficiently with special purpose hardware.

Note that a reactive jammer must compute $3 \times \binom{13,500}{3}$ correlations in order to gather information of the chosen code sequences used by RD-DSSS with the settings in the above experiment. Assume the computation power of the jammer is 100 times of a normal receiver. Let t_j and r denote the computational overhead for the jammer to crack a single message and the ratio of t_j to t_a, respectively. Thus, $t_j = \frac{3 \times \binom{13,500}{3} \times t_c}{13,500 \times 3 \times 100}$ and $r = \frac{3 \times \binom{13,500}{3} \times t_c}{13,500 \times 3 \times 100 \times (t_h + t_s + t_p + t_{ip} + t_c)}$. In particular, using the parameters in Table I, we can get the overhead $t_j = 2,729.8$ seconds, which is much larger than the transmission time of a single message, and the ratio $r = 183,680$. When k increases, the jammer's computation overhead t_j increases exponentially, since the jammer needs to check all combinations of k code sequences in $\mathcal{C}$ (i.e., $k\binom{q}{k}$). However, the receiver's computation overhead t_a increases linearly, since the receiver only needs to check $k \times q$ correlations for identifying the index codes.

Storage Overhead: RD-DSSS systems need to store the code set $\mathcal{P}$ and all the index codes. In our experiment, the number of index codes is $13,500$. Assume the size of $\mathcal{P}$ is 100. Thus, the required storage capacity is $100 \times 256 + 13,500 \times 512$ bits $= 0.83$ MB, which can be afforded by notebook- or handheld-class devices nowadays.

Communication Overhead: RD-DSSS systems slightly increase communication overhead compared with traditional DSSS systems, since a sender needs to append k index codes to the end of the message body. However, this communication overhead is negligible compared with the cost of transmitting the message body. For example, in the settings of the earlier experiment, the communication overhead introduced by the index codes is 3×512 chips for each message, which has $256 \times 1,024$ chips. The overhead is less than 0.6%.

VI. Security Analysis and Simulation

We perform theoretical analysis to show the effectiveness of RD-DSSS in defending against various jamming attacks. We also perform simulation to confirm our analytical results. The simulation is done in MATLAB 7.4.0 on a computer with a 3.40 GHz Intel Pentium 4 CPU and 1.5 GB memory.

A. Classification of Jamming Attacks

To facilitate the analysis, we first give a classification of jamming attacks. According to [15], jammers are classified into five types: *broadband noise jammers*, *partial-band noise jammers*, *continuous wave jammers*, *multitone jammers*, and *pulse jammers*. These jammers intentionally transmit random noise signals to cause wireless interferences. They differ from each other in energy distributions of their jamming signals. In practice, there exist more complicated jammers. For example, a "smart" jammer can not only transmit random noise signals, but can also inject fake messages that are encapsulated in the packet format used by the communicators to mislead the reception process [25].

Based on the previous work [15], [25], we generalize jamming attacks into two categories: *non-intelligent jamming attacks* and *intelligent jamming attacks*. In the former attacks, the jammer disrupts wireless communication by sending random noise signals. In the latter attacks, the jammer transmits jamming signals that are generated based on his knowledge of the communication systems (e.g., the signal patterns, anti-jamming strategies, and communication protocols).

Note that conventional DSSS is capable of mitigating non-intelligent jamming attacks [15]. Therefore, inheriting from conventional DSSS, RD-DSSS is also non-intelligent jamming resilient. Thus, in this section, we focus our analysis on the intelligent jamming attacks.

B. Intelligent Jamming Attacks

In intelligent jamming attacks, a jammer is aware of the target communication systems and transmits meaningful messages to undermine the communication. In RD-DSSS systems, a significant threat caused by intelligent jamming attacks is that a jammer may take advantage of the publicly known spreading code set $\mathcal{P}$, code sequence set $\mathcal{C}$, and the index code set $\mathcal{I}$ to jam the communication. Therefore, in our analysis, we focus on the ways that an intelligent jammer can use the publicly known information to disrupt the wireless communication.

By exploiting $\mathcal{P}$, $\mathcal{C}$, and $\mathcal{I}$, an intelligent jammer can choose the following strategies to attack RD-DSSS systems:

- Type I: He can randomly choose codes from $\mathcal{P}$ and tries to jam the communication by transmitting those spreading codes.
- Type II: He can perform reactive jamming discussed in Section IV-A4. That is, the jammer tries to find the code sequences used by a sender by analyzing the first portion of a message being transmitted, and then spreads a fake message using the identified code sequence to jam the rest of the message transmission.
- Type III: He can perform Denial of Service (DoS) attacks targeting the index codes, in which he randomly picks several index codes and transmits them along with the legal index codes to force receivers to deal with a large number of candidate index codes.

In the following we show the effectiveness of RD-DSSS in defending against these attacks.

1) Type I Attacks: In Type I attacks, the jammer randomly selects codes from $\mathcal{P}$, and transmits them to interfere with the original message transmission. We first derive the probability that the jammer can disrupt the wireless communication.

Analysis: Suppose the i-th code transmitted by the sender is $\mathbf{p}_i$ and $\mathcal{A}_i$ is the set that generates the i-th code of the permuted spread message. If the bit spread by $\mathcal{A}_i$ is "0" and $\mathbf{p}_i$ happens to be in $\mathcal{A}_i$, then the receiver cannot despread the original bit correctly. However, a few bit errors can be tolerated by ECC. For example, the standard (255, 223) Reed-Solomon code is capable of correcting up to 16 bit errors among every 223 information bits of a message [23]. If the ECC can correct a maximum of M bit errors of the original message but the jammer can collapse more than M bits, then the receiver cannot reconstruct the original message. In the following, we derive the probability that the jammer can disrupt the transmission of a message (i.e., the probability that the jammer can jam more than M bits).

Assume the sender transmits "1" or "0" with probability $\frac{1}{2}$. Let $|\mathcal{A}_i|$ be the number of codes in the set $\mathcal{A}_i$. The probability that the i-th code transmitted by the jammer is one code in $\mathcal{A}_i$ is $\mathbb{P}(\mathbf{p}_i \in \mathcal{A}_i) = \sum_{j=1}^{k} \mathbb{P}(|\mathcal{A}_i| = j)\frac{j}{n}$, where n is the size of the code set $\mathcal{P}$.

The number of ways of picking j codes from $\mathcal{P}$ is $\binom{n}{j}$, and the number of ways of putting k codes from $\mathcal{P}$ into $\mathcal{A}_i$ is n^k. Let $N(j)$ denote the number of ways of putting j distinct codes into $\mathcal{A}_i$ such that $|\mathcal{A}_i| = j$. Thus, the recurrence equation of $N(j)$ is $N(j) = j^k - \sum_{w=2}^{j} \binom{j}{w-1} N(w-1)$. According to classical probability model, $\mathbb{P}(|\mathcal{A}_i| = j) = \frac{\binom{n}{j}N(j)}{n^k}$. Thus, $\mathbb{P}(\mathbf{p}_i \in \mathcal{A}_i) = \sum_{j=1}^{k} \frac{\binom{n}{j}N(j)j}{n^{k+1}}$ and the probability that the bit spread by $\mathcal{A}_i$ is jammed is $\frac{\mathbb{P}(\mathbf{p}_i \in \mathcal{A}_i)}{2}$. Therefore, the probability that more than M bits are corrupted by the jammer (i.e., the probability that the communication is jammed) is $p_J = \sum_{i=M+1}^{l} \binom{l}{i} \left(\frac{\mathbb{P}(\mathbf{p}_i \in \mathcal{A}_i)}{2}\right)^{i} \left(1 - \frac{\mathbb{P}(\mathbf{p}_i \in \mathcal{A}_i)}{2}\right)^{l-i}$.

Figures 3 shows the jamming probability p_J when each message has 1,024 bits. It is easy to see that p_J decreases as the size of the spreading code set (n) increases. When n is larger than 70 and $M = 60$, the probability is less than 10^{-4}.

Simulation: We use simulation to further examine the

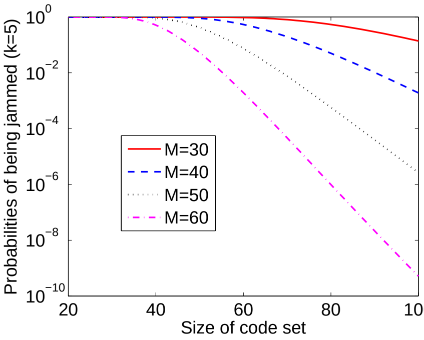

Fig. 3. Theoretical probability that an attacker jams communication for $k = 5$

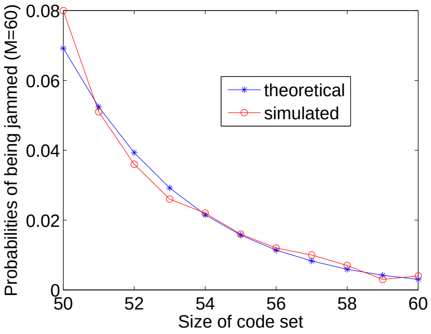

Fig. 4. Simulated and theoretical probabilities that an attacker jams communication when $M = 60$

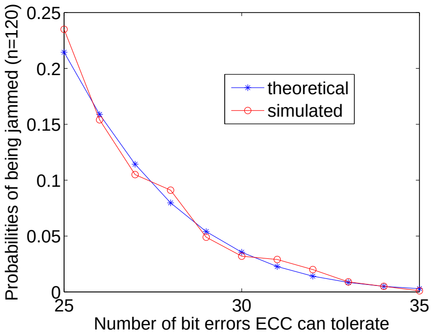

Fig. 5. Simulated and theoretical probabilities that an attacker jams communication when $n = 120$

effectiveness of RD-DSSS and confirm the theoretical results. In the simulation, the length of the message is 1,024 bits and the size of $\mathcal{C}$ is 10,000.

We let $M = 60$ and n range from 50 to 60. For each n, we randomly generate a message and spread it using 5 code sequences randomly selected from $\mathcal{C}$. We also form the jammer's code sequence by randomly picking l codes from $\mathcal{P}$. Then we count the number of bits that can be disrupted by the jammer, and mark the trial as successful if the number is larger than M. We repeat this process for 1,000 times and estimate the *simulated* p_J (i.e., the probability that a message is jammed) as $p_J = \frac{\#\ successful\ trails}{\#\ trials}$. Figure 4 shows both the simulated and the theoretical probabilities when $M = 60$. We can see that both probabilities are less than 0.01 if the size of the code set is larger than 60.

We then let $n = 120$ and M range from 25 to 35. For each M, we perform the same simulation as we did for each n. Figure 5 shows the simulated and theoretical jamming probabilities when we change the number of bit errors ECC can tolerate. We can see that both probabilities decrease as the number of bit errors ECC can tolerate increases. When $M = 35$, the jamming probabilities are less than 0.01.

2) Type II Attacks: We now evaluate the feasibility of reactive jamming attacks. We use the correlation between the jammer's observation and each code sequence in $\mathcal{C}$ as the evaluation metric. In other words, we assess the possibility for a jammer to gather enough information for reactive jamming.

In reactive jamming attacks, a jammer infers the code sequences chosen by the sender based on the first few codes that have already been transmitted by the sender, and then uses the identified code sequences to jam the transmission of the rest codes. However, in RD-DSSS the sender permutes each spread message before transmission. Thus, the correlation between the transmitted message and the code sequences chosen by the sender is reduced, and it is hard for the jammer to correctly identify the code sequences chosen by the sender by analyzing the correlation between the observed codes and each code sequence in $\mathcal{C}$.

In the following, we derive the correlation between the observed codes and a code sequence not selected by the sender. We also derive the correlation between the observed codes and a code sequence selected by the sender. We then compare both correlations and show that they are very close to each other.

Analysis: Let p_1 and p_0 denote the probability that the sender transmits "1" and "0", respectively. Let $\mathbf{f} = \mathbf{f}_1||...||\mathbf{f}_r$ denote the r codes transmitted by the sender and observed by the jammer, where $1 \leq r \leq l$. Let $\mathbf{g} = \mathbf{g}_1||...||\mathbf{g}_l$ denote a code sequence not selected by the sender. Assume $p_1 = p_0 = \frac{1}{2}$. The probability that $\mathbf{f}_i = \mathbf{g}_i$ is given in Lemma 2

Lemma 2: The probability that $\mathbf{f}_i = \mathbf{g}_i$ is $\mathbb{P}(\mathbf{f}_i = \mathbf{g}_i) = \frac{1}{2n} + \frac{1}{2n^k} \sum_{j=1}^{k} \binom{n}{j} N(j) \frac{(n-1)^j}{n^j(n-j)}$.

Proof: Assume the original position of $\mathbf{f}_i$ is i_o. If $m_{i_o} = 1$, the probability that $\mathbf{f}_i = \mathbf{g}_i$ is $\sum_{j=1}^{k} \frac{1}{j}\mathbb{P}(|\mathcal{A}_{i_o}| = j)\frac{j}{n} = \frac{1}{n}$. If $m_{i_o} = 0$, the probability that $\mathbf{f}_i = \mathbf{g}_i$ is $\sum_{j=1}^{k} \frac{1}{n-j}\mathbb{P}(|\mathcal{A}_{i_o}| = j)(1 - \frac{1}{n})^j$. Hence, $\mathbb{P}(\mathbf{f}_i = \mathbf{g}_i) = \frac{1}{2n} + \frac{1}{2}\sum_{j=1}^{k} \frac{1}{n-j}\mathbb{P}(|\mathcal{A}_{i_o}| = j)(1 - \frac{1}{n})^j = \frac{1}{2n} + \frac{1}{2n^k} \sum_{j=1}^{k} \binom{n}{j} N(j) \frac{(n-1)^j}{n^j(n-j)}$. ■

Let $\mathbf{h} = \mathbf{h}_1||...||\mathbf{h}_l$ denote a code sequence selected by the sender. The probability that $\mathbf{f}_i = \mathbf{h}_i$ is given in Lemma 3

Lemma 3: The probability that $\mathbf{f}_i = \mathbf{h}_i$ is $\mathbb{P}(\mathbf{f}_i = \mathbf{h}_i) = \frac{1}{l}((\frac{1}{2n^k} \sum_{j=1}^{k} \binom{n}{j} N(j)(\frac{1}{j} - \frac{(n-1)^j}{n^j(n-j)})) - \frac{1}{2n}) + \frac{1}{2n} + \frac{1}{2n^k} \sum_{j=1}^{k}.$

Proof: If $i = i_o$ (i.e., the i_o-th code of the original spread message does not change its position after permutation), the probability that the i-th observed code $\mathbf{f}_i$ is the same as $\mathbf{h}_i$ is $\mathbb{P}(\mathbf{f}_i = \mathbf{h}_i | i = i_o) = \frac{1}{2}\sum_{j=1}^{k} \mathbb{P}(|\mathcal{A}_{i_o}| = j)\frac{1}{j} = \frac{1}{2n^k} \sum_{j=1}^{k} \binom{n}{j} N(j)\frac{1}{j}$. If $i \neq i_o$ (i.e., the i-th code $\mathbf{f}_i$ of the original spread message changes its position after permutation), the probability that $\mathbf{f}_i = \mathbf{h}_i$ is the same as the probability that $\mathbf{f}_i = \mathbf{g}_i$, since both $\mathbf{h}_i$ and $\mathbf{g}_i$ can be regarded as an arbitrary code in $\mathcal{P}$. Therefore, $\mathbb{P}(\mathbf{f}_i = \mathbf{h}_i | i \neq i_o) = \mathbb{P}(\mathbf{f}_i = \mathbf{g}_i)$. For $1 \leq i_o, j_o \leq l$, the probability that $\mathcal{A}_{i_o} = \mathcal{A}_{j_o}$ is $\frac{1}{\binom{n}{k}}$, which is a very small value if n is relatively large and k is relatively small (e.g., $\frac{1}{\binom{n}{k}} \approx 10^{-7}$ when $n = 50$ and $k = 5$). Therefore, the probability that $\mathcal{A}_{i_o} = \mathcal{A}_{j_o}$ is approximately 0. Note that for different inputs, the outputs of mapping function h are also different. Thus, the possible values of $h(\mathcal{A}_1), ..., h(\mathcal{A}_l)$ are distinct from each other. We consider $h(\mathcal{A}_1), ..., h(\mathcal{A}_l)$ as l random variables. The probability that $h(\mathcal{A}_{i_o})$ is just the i_o-th smallest/largest element among all values is $\frac{1}{l}$. Therefore, the probability that $i = i_o$ is $\frac{1}{l}$ and the probability that $i \neq i_o$ is $1 - \frac{1}{l}$. As a result, $\mathbb{P}(\mathbf{f}_i = \mathbf{h}_i) = \mathbb{P}(i = i_o)\mathbb{P}(\mathbf{f}_i = \mathbf{h}_i | i = i_o) + \mathbb{P}(i \neq i_o)\mathbb{P}(\mathbf{f}_i = \mathbf{h}_i | i \neq i_o) =$

$\frac{1}{l}((\frac{1}{2n^k}\sum_{j=1}^{k}\binom{n}{j}N(j)(\frac{1}{j}-\frac{(n-1)^j}{n^j(n-j)}))-\frac{1}{2n})+\frac{1}{2n}+\frac{1}{2n^k}\sum_{j=1}^{k}.$ ■

Assume the correlation of two identical codes is 1 and that of two different codes is 0. Based on Lemmas 2 and 3, the expectation of the correlation between $\mathbf{f}_1||...||\mathbf{f}_r$ and $\mathbf{g}_1||...||\mathbf{g}_r$ is $\mathbb{E}(\mathbf{f}\cdot\mathbf{g}) = \frac{1}{r}\sum_{i=1}^{r}\mathbb{P}(\mathbf{f}_i = \mathbf{g}_i) = \mathbb{P}(\mathbf{f}_i = \mathbf{g}_i)$, and the expectation of the correlation between $\mathbf{f}_1||...||\mathbf{f}_r$ and $\mathbf{h}_1||...||\mathbf{h}_r$ is $E(\mathbf{f}\cdot\mathbf{h}) = \frac{1}{r}\sum_{i=1}^{r}\mathbb{P}(\mathbf{f}_i = \mathbf{h}_i) = \mathbb{P}(\mathbf{f}_i = \mathbf{h}_i)$.

Figure 6 shows the expectation of the correlation between the observed codes and a code sequence selected by the sender (i.e., $\mathbb{E}(\mathbf{f}\cdot\mathbf{h})$), as well as the expectation of the correlations between the observed codes and a code sequence not selected by the sender (i.e., $\mathbb{E}(\mathbf{f}\cdot\mathbf{g})$). We can see that $\mathbb{E}(\mathbf{f}\cdot\mathbf{h})$ approaches $\mathbb{E}(\mathbf{f}\cdot\mathbf{g})$ as the length l of the message increases. When l is larger than 100, $\mathbb{E}(\mathbf{f}\cdot\mathbf{h})$ and $\mathbb{E}(\mathbf{f}\cdot\mathbf{g})$ are almost the same. Thus, it is hard for the jammer to distinguish the code sequences of the sender by analyzing correlations between the observed codes and each code sequence in $\mathcal{C}$.

Figure 7 shows both $\mathbb{E}(\mathbf{f}\cdot\mathbf{h})$ and $\mathbb{E}(\mathbf{f}\cdot\mathbf{g})$ when the size of the spreading code set increases. We can see that $\mathbb{E}(\mathbf{f}\cdot\mathbf{h})$ is very close to $\mathbb{E}(\mathbf{f}\cdot\mathbf{g})$. Although both $\mathbb{E}(\mathbf{f}\cdot\mathbf{h})$ and $\mathbb{E}(\mathbf{f}\cdot\mathbf{g})$ decrease as n increases, the distance between them is small.

Simulation: We use simulation to confirm the theoretical results. In the simulation, $\mathcal{P}$ has 30 codes, and $\mathcal{C}$ has 10,000 code sequences. All codes in $\mathcal{P}$ are Walsh-Hadamard codes.

We let l range from 1 to 100. For each l, we randomly pick k code sequences from $\mathcal{C}$ and generate a message. We spread and permute the message based on the chosen code sequences. Assume $r = l$. We compute and record the average of the correlations between the observed codes and each code sequence selected by the sender. We repeat this process 1,000 times and estimate the *simulated* $\mathbb{E}(\mathbf{f}\cdot\mathbf{h})$ as the average of the 1,000 recorded values.

Figure 8 shows simulated and theoretical $\mathbb{E}(\mathbf{f}\cdot\mathbf{h})$ for $k = 5$. The correlation between the observed codes and the code sequences selected by the sender decreases as n increases. The correlation is less than 0.02 when $n = 80$ and $l \geq 20$.

In addition, we obtain $\mathbb{E}(\mathbf{f}\cdot\mathbf{h})$ through simulation when the size of $\mathcal{P}$ increases, with $k = 1$ and $k = 5$. In the simulation, $l = 300$ and other parameter settings are the same as those in the simulation of Figure 8. We let n range from 50 to 100. For each n, we perform the same process as we did for each l in the simulation of Figure 8, and estimate the simulated $\mathbb{E}(\mathbf{f}\cdot\mathbf{h})$. Figure 9 shows both simulated and theoretical $\mathbb{E}(\mathbf{f}\cdot\mathbf{h})$. For both $k = 1$ and $k = 5$, we can see that the correlation between the observed codes and the code sequences selected by the sender is quite small ($\mathbb{E}(\mathbf{f}\cdot\mathbf{h}) \leq 0.022$).

The simulation results are very close to the theoretical results, as shown in Figures 8 and 9. Both figures demonstrate that there is very small correlation between the observed codes and the code sequences selected by the sender. Thus, RD-DSSS can prevent jammers from learning the chosen code sequences and launch reactive jamming attacks. In contrast, UDSSS allows a reactive jammer to derive the code sequence directly by observing the first code in the sequence.

3) Type III Attacks: In RD-DSSS systems, a receiver needs to identify index codes of a received message. Without the knowledge of the index codes, the receiver cannot correctly de-spread the received message. Therefore, a jammer may launch DoS attacks targeting the index codes.

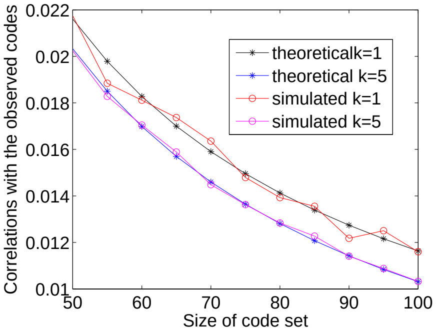

Fig. 9. Simulated and theoretical $\mathbb{E}(\mathbf{f}\cdot\mathbf{h})$ for $k = 1$ and $k = 5$ ($l = 300$)

Specifically, a jammer may randomly choose k index codes, get synchronized with the sender, and transmit the chosen index codes to interfere with the k index codes from the sender. As a result, the receiver will have to deal with 2^k combinations of index codes (i.e., using each combination to de-spread the received message and authenticating the de-spreading output).

However, as discussed earlier, the number of code sequences chosen by the sender is small, i.e., $k \leq 5$. This means DoS attacks against index codes can be tolerated. For example, if $k = 3$ and $q = 13,500$, the receiver only needs to deal with $2^3 = 8$ combinations of index codes under DoS attacks. However, with the same parameter setting, a reactive jammer is forced to compute $3\times\binom{13,500}{3} > 2^{40}$ correlations on average within a very short period of time (i.e., the transmission time of a single message).

VII. Related Work

The jamming problem in wireless communication has been widely studied during the past few decades (e.g., [5], [11], [12], [14], [15], [21], [22]), and spread spectrum such as DSSS and FHSS are traditional anti-jamming techniques [15], [20]. However, as discussed earlier, those techniques require that senders and receivers pre-share secret keys. There have been a few recent attempts to enable the establishment of pre-shared secret keys [16]–[18]. Unfortunately, all those works cannot be used for broadcast communication where there exist malicious receivers that may also act as jammers. An insider jammer who knows the shared key can use the key to prevent other receivers from receiving the messages.

To address this problem, some researchers recently investigated how to enable jamming-resistant broadcast communication without shared keys [1], [13]. As discussed in Section I, the decoding process of [1] cannot be extended to DSSS or FHSS. UDSSS proposed in [13] is the most relevant to ours. However, the broadcast communication provided by UDSSS is still vulnerable if an insider jammer with sufficient computational power launches reactive jamming attacks as indicated in [13]. Inspired by UDSSS, RD-DSSS tolerates reactive jamming attacks by using multiple code sequences and permutation to protect each message.

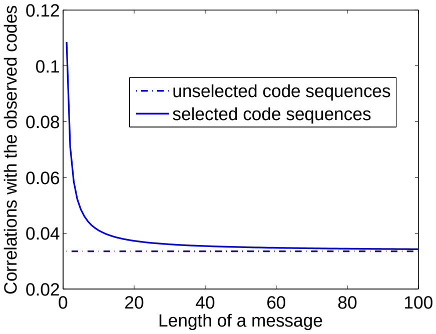

Fig. 6. $\mathbb{E}(\mathbf{f} \cdot \mathbf{h})$ and $\mathbb{E}(\mathbf{f} \cdot \mathbf{g})$ ($k = 5$, $n = 30$)

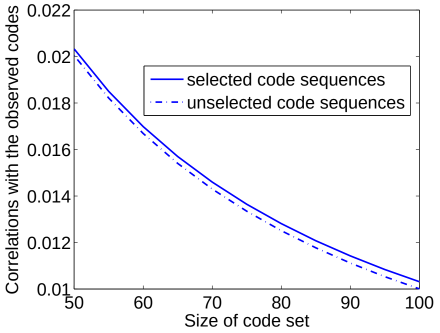

Fig. 7. $\mathbb{E}(\mathbf{f} \cdot \mathbf{h})$ and $\mathbb{E}(\mathbf{f} \cdot \mathbf{g})$ ($k = 5$, $l = 300$)

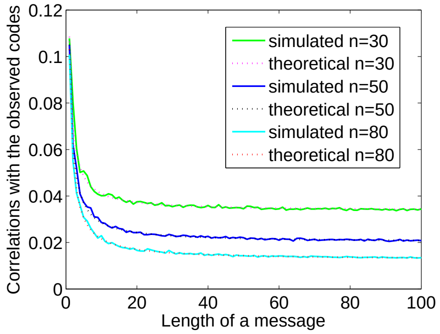

Fig. 8. Simulated and theoretical $E(\mathbf{f} \cdot \mathbf{h})$ for $n = 30$, $n = 50$, and $n = 80$

Besides the literature discussed above, there exist other related works. For example, a code tree based technique that enables the system to identify insider jammers was proposed in [3]. Xu et al. proposed to employ consistency checking for detecting jamming attacks [25]. The problems such as how to mitigate jamming on control channels [8], [19] and sensor networks [9], [24] were also studied by previous researchers. These approaches are complementary to ours.

VIII. Conclusion

In this paper, we proposed RD-DSSS to enable anti-jam broadcast communication without shared secret keys. RD-DSSS encodes each bit of data using the correlation of unpredictable spreading codes. Thus, a sender and a receiver do not need to share any common key to communicate with each other. In addition, RD-DSSS spreads a message using multiple code sequences and permutes each spread message before transmitting it. As a result, the chance that a reactive jammer can correctly guess which code sequences are used by the sender is greatly reduced. We use both theoretical analysis and simulation to evaluate the performance and jamming resistance of the proposed RD-DSSS scheme. The results demonstrate that RD-DSSS can effectively defend against jamming attacks.

Acknowledgment

This work is supported by the National Science Foundation under grants CAREER-0447761 and CNS-0721424.

References

[1] L. C. Baird, W. L. Bahn, M. D. Collins, M. C. Carlisle, and S. C. Butler. Keyless jam resistance. In *Proceedings of the IEEE Information Assurance and Security Workshop*, pages 143–150, June 2007.

[2] R. H. Barker. Group synchronization of binary digital systems. *Communication Theory*, pages 273–287, 1953.

[3] J. Chiang and Y. Hu. Dynamic jamming mitigation for wireless boradcast networks. In *Proceedings of IEEE International Conference on Computer Communications (INFOCOM)*, 2008.

[4] M. Chiani and M. G. Martini. Analysis of optimum frame synchronization based on periodically embedded sync words. *IEEE Transaction on Communications*, 55:2056–2060, Nov. 2007.

[5] R. A. Dillard and G. M. Dillard. *Detectability of Spread-spectrum Signals*. Artech House Publishers, 1989.

[6] A. Goldsmith. *Wireless Communications*. Cambridge University Press, New York, NY, USA, 2005.

[7] S. Gollakota and D. Katabi. Zigzag decoding: Combating hidden terminals in wireless networks. In *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication*, pages 159–170, Seattle, WA, USA, April 2008.

[8] L. Lazos, S. Liu, and M. Krunz. Mitigating control-channel jamming attacks in multi-channel ad hoc networks. In *Proceedings of 2nd ACM Conference on Wireless Networking Security (WiSec '09)*, March 2009.

[9] M. Li, I. Koutsopoulos, and R. Poovendran. Optimal jamming attacks and network defense policies in wireless sensor networks. In *Proceedings of IEEE International Conference on Computer Communications (INFOCOM)*, 2007.

[10] A. Papoulis. *Probability, Random Variables, and Stochastic Processes*. Mc-Graw Hill, 1984.

[11] R. L. Peterson, R. E. Ziemer, and D. E. Borth. *Introduction to Spread Spectrum Communications*. Prentice Hall, 1995.

[12] R. A. Poisel. *Modern Communications Jamming Principles and Techniques*. Artech House Publishers, 2006.

[13] C. Pöpper, M. Strasser, and S. Čapkun. Jamming-resistant broadcast communication without shared keys. In *Proceedings of the USENIX Security Symposium*, 2009. To appear.

[14] L. A. Rusch and H. V. Poor. Narrowband interference suppression in cdma spread spectrum communications. *IEEE Transaction on Communications*, 42:1969–1979, Feb. 1994.

[15] M. K. Simon, J. K. Omura, R. A. Scholtz, and B. K. Levitt. *Spread Spectrum Communications Handbook, Revised Edition*. New York: McGraw-Hill, Inc., 1994.

[16] D. Slater, P. Tague, R. Poovendran, and B. Matt. A coding-theoretic approach for efficient message verification over insecure channels. In *Procceedings of the 2nd ACM Conference on Wireless Networking Security (WiSec '09)*, pages 151–160, March 2009.

[17] M. Strasser, C. Pöper, S. Čapkun, and M. Čagalj. Jamming-resistant key establishment using uncoordinated frequency hopping. In *Proceedings of the 2008 IEEE Symposium on Security and Privacy*, pages 64–78, 2008.

[18] M. Strasser, C. Pöpper, and S. Čapkun. Efficient uncoordinated FHSS anti-jamming communication. In *Procceedings of MobiHoc'09*, May 2009. To appear.

[19] P. Tague, M. Li, and R. Poovendran. Probabilistic mitigation of control channel jamming via random key distribution. In *Proceedings of IEEE 18th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC '07)*, pages 1–5, 2007.

[20] D. Torrieri. *Principles of Spread-Spectrum Communication Systems*. Springer, 2004.

[21] D. J. Torrieri. *Principles of Military Communication Systems*. Artech House Publishers, 1981.

[22] D. J. Torrieri. The performance of five different metrics against pulsed jamming. *IEEE Transaction on Communications*, 34:200–207, Feb. 1986.

[23] S.B. Wicker and V.K. Bhargava. *Reed-Solomon Codes and Their Applications*. IEEE Press, 1994.

[24] W. Xu, W. Trappe, and Y. Zhang. Channel surfing: Defending wireless sensor networks from jamming and interference. In *Proceedings of the 6th International Conference on Information Processing in Sensor Networks (IPSN '07)*, 2007.

[25] W. Xu, W. Trappe, Y. Zhang, and T. Wood. The feasibility of launching and detecting jamming attacks in wireless networks. In *Proceedings of ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc '05)*, 2005.