Randomized Positioning DSSS for Anti-Jamming Wireless Communications

Ahmad Alagil, Meshari Alotaibi, Yao Liu
{alagil, meshari}@mail.usf.edu, yliu@cse.usf.edu
University of South Florida, Tampa, FL 33620

***Abstract*—Traditional anti-jamming approaches like Frequency Hopping Spread Spectrum (FHSS) and Direct Sequence Spread Spectrum (DSSS) require the sender and the receiver to share a secret key prior to their communication. If this key is compromised by the jammer, the jammer can then generate the frequency hopping patterns or the spreading codes used by the communicators to disrupt the wireless communication. In recent years, DSSS based schemes have been proposed to provide the anti-jamming communication without the shared key. In particular, Randomized Differential DSSS (RD-DSSS) was developed to spread messages based on the indices of public known spreading code sequences. RD-DSSS can effectively mitigate reactive jamming attacks and do not need a shared key, but it appends the indices, which are critical to enable the decoding at the receiver, to the end of the spread messages. As a result, the indices can easily become the jamming target of adversaries. To solve this problem, we propose the Randomized Positioning DSSS (RP-DSSS) scheme that randomly relocate the index codes for each message. Compared to RD-DSSS, the randomization hides the indices from the adversaries and thus achieves the enhanced security.**

I. Introduction

Jamming attacks have been considered as a significant threat to wireless communications in the past few decades. A jammer can simply send noise signals over the wireless channel to interfere the legitimate communications between the sender and the receiver. As a result, the receiver cannot correctly decode the original signals from the sender, and thus the communication is blocked. To achieve the reliable wireless communication, it is critical to provide wireless users with the resilience against jamming attacks.

Spread spectrum techniques such as Frequency Hopping Spread Spectrum (FHSS), and Direct Sequence Spread Spectrum (DSSS) have been used most frequently to defend against jamming attacks (e.g., [4], [6], [7], [10]). In recent years, researchers have been concerned that these techniques require both the sender and the receiver to share a secret key, which can make the anti-jamming system vulnerable [7]. For example, in FHSS, the sender and the receiver avoid the interference caused by jamming attacks through switching to different channels from time to time. These channels are random and generated using a shared secret key. If the attacker knows this key, she can then switch to the same channels to jam the FHSS communication. In DSSS, both the sender and the receiver achieve the anti-jamming purpose by spreading the original message using spreading codes, which are selected based on a shared key. If the attacker knows the key and accordingly the spread codes, she can spread a random message and send it along with the transmission of the original message to confuse the despreading at the receiver.

Due to the security concern of the shared key, some research has been done to remove the requirement of a pre-share key for anti-jamming communication systems (e.g., [1], [5], [9], [10]. In particular, Popper et al. proposed the Uncoordinated DSSS (UDSSS) schemes that enable the anti-jamming communication without a shared secret key [7]. The sender randomly picks spreading code sequences from a set of public code sequences to encode messages. However, if the jammer has enough computational power to infer the spreading code sequences chosen by the sender before the transmission is complete, the jammer can then jam the rest of the message transmission.

To overcome this problem, Randomized Differential DSSS (RD-DSSS) scheme was proposed [4]. The sender spreads messages based on the indices of the public spreading code sequences, and the receiver uses these indices to identify the spreading code sequences chosen by the sender for despreading. The indices are randomly chosen among a public pseudorandom code set and thus they are refereed as *index codes*. Although RD-DSSS can effectively mitigate the aforementioned attacks against UDSSS and do not need a shared key, it is still vulnerable to attacks. The index codes are essential to enable the despreading at the receiver and they are appended at the end of the spread messages. As such, they easily become the jamming target of adversaries.

In this paper, we propose the Randomized Positioning DSSS (RP-DSSS) scheme as an extension to improve the security of RD-DSSS. The vulnerability of index codes roots from the fact that they are located at a fixed position of a spread message (i.e., end of the message). The fixed position makes them completely exposed to the adversaries. To protect index codes, we therefore propose to randomly relocate the index codes for each message. Specifically, we insert the index codes into a random position of a spread message instead of appending it at the end of the spread message. To enable the receiver to find the position of the index code, we design an "onion" spreading mechanism that treats this random position as the payload of a new message and encodes them using RP-DSSS in a recursive way. The receiver can de-spread the received messages by applying the reverse operations done by the sender. To remove the requirement of a shared key, similar to RD-DSSS, we also take advantage of public sets of spreading codes and code sequences. For an original message, the sender and the receiver use a random code sequence from the set of code sequences to spread and despread.

The contribution of this paper is three-fold. First, we identify the vulnerability on the index codes of the RD-DSSS scheme. To mitigate this vulnerability, we accordingly propose the RP-DSSS scheme that places index codes at random positions of a spread message to prevent adversaries discovering these index codes. Second, we develop the techniques that can inform the receiver the positions of index codes in the presence of jammers, and we integrate this technique into the RD-DSSS to achieve a security enhanced anti-jamming system, which does not require the shared secret keys. Third, we perform computer simulations to validate the performance of the proposed approaches.

The rest of this paper is organized as follows. Section II discusses the background information on DSSS. Section III presents the proposed RP-DSSS scheme. Section IV provides the performance evaluation results. Sections V and VII describe the related work and conclude this paper, respectively.

II. Background on DSSS

DSSS is a modulation method applied to digital signals [2]. It increases the signal bandwidth to a value much larger than needed to transmit the underlying information [2]. 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 [8].

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 $1 \leq i \leq k$, 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 [8]. 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).

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. Randomized Positioning DSSS

Similar to DSSS, RP-DSSS achieves the anti-jamming capability by using spreading codes to obtain the spreading gain. However, unlike DSSS, RP-DSSS relies on the correlation between two spreading codes to encode each bit of an original message. Pre-shared keys are not required for RP-DSSS communications. Because RP-DSSS is an extension of the RD-DSSS scheme, we first introduce how RD-DSSS works to facilitate readers' understanding, and then we present our proposed RP-DSSS scheme.

A. Basic Scheme of RD-DSSS

We assume that both the sender and the receiver share a set of spreading codes. According to the property of spreading codes, the auto-correlation between two identical codes is high, and the cross-correlation between two different codes is low. The sender encodes each bit of the message separately. Specifically, bit "0" is encoded using a pair of different codes and bit "1" is encoded using a pair of identical codes. The receiver decodes the spread message by calculating the correlation between two codes. If the result of the correlation is high, then the encoding of the corresponding bit uses two identical codes and thus the recovered bit is "1". If the correlation result is not high, then the recovered bit is "0" because two different codes are used. The codes are randomly chosen from the spreading codes set.

Spreading: Figure 1 demonstrates an example of the RD-DSSS scheme. The original message M is 1010. The sender randomly chooses four codes from the spreading code set to decode M. Let p_1 ,p_2, p_4, and p_8 denote these codes. The first bit of M is 1, the sender repeats using p_1. The second bit is 0, the sender randomly picks a different spreading code from the code set. Assume p_3 is selected. The rest of bits in M can be encoded in the same way and the spreading outcome is $p_1||p_2||p_4||p_8||p_1||p_3||p_4||p_5$.

De-spreading: The receiver calculates the correlation between the i-th and $i+L$-th codes in a received message, where L is the length of the message. In this example, the receiver computes $cor(p_1, p_1)$, $cor(p_2, p_3)$, $cor(p_4, p_4)$, and $cor(p_8, p_5)$, and converts the correlation output to bit 1 or 0.

RD-DSSS reduces the communication overhead by using spreading code sequences, which are also publicly known. A spreading code sequence is formed by concatenating the codes from the spreading code set, and is associated with an index code, which is a pseudorandom noise (PN) code to identify the spreading code sequence chosen by the sender. The index codes also form an index code set. As shown in Figure 1, the sender randomly chooses a spreading sequence from the

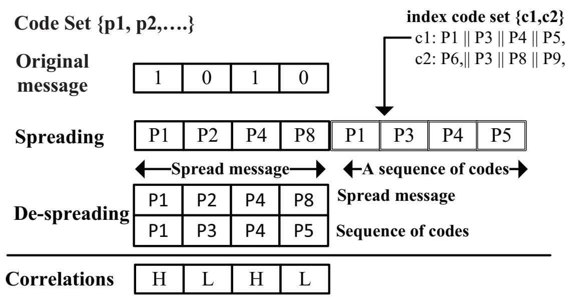


Fig. 1: The Basic scheme of DSSS

spreading sequence set and it is the sequence $p_1||p_3||p_4||p_5$. The sender then spreads the message using this sequence and appends the index code c_1 to the end of the spreading result, and thus the ultimate spreading output is $p_1||p_2||p_4||p_8||c_1$.

B. The Limitation

As mentioned earlier, RD-DSSS scheme has an obvious limitation. The index codes are always appended to the end of the message. This makes them easily become the target for the attacker. The attacker can compute the correlation between the subsequence formed by the first several bits of the index code and those of all possible index codes to identify which index code was selected by the sender. The attacker can jam the rest of the index codes by simply transmitting multiple index codes to deceive the receiver to obtain multiple possibilities of the index code, thereby boosting the computational cost for de-spreading at the receiver.

C. RP-DSSS

We create a new technique to overcome the limitation of the RD-DSSS. Intuitively, the spreading codes have the random-look because they are indeed generated by pseudorandom generators [8], and naturally a message spread by these codes exhibit the randomness as well. Thus, inserting a pseudorandom index code at any positions of a spread message would still make the entire message look like a random sequence of -1 and $+1$. We hence propose to randomly relocate the index codes. For each spread message, we insert the corresponding index code into a random position, and the random-looking property of the spread message enables the camouflage of the index code.

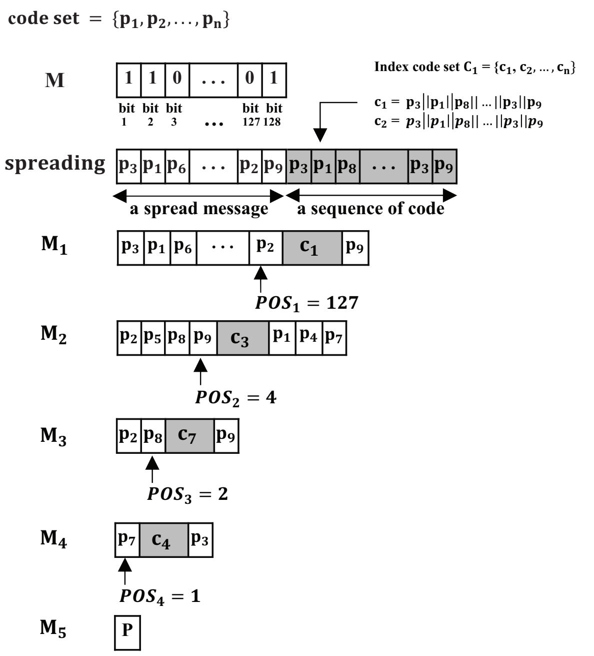
Onion spreading: An important by-product question is how can the receiver identify the index codes, such that it can find the corresponding spreading code sequence for the de-spreading. The receiver and the adversary have the same knowledge about a spread message. It seems if the index code is hidden from the adversary, then it is also hidden from the receiver. Nevertheless, the communication roles of the receiver and the adversary are totally different. The receiver aims to correctly decode a message after the message is received, whereas the adversary aims to jam the message before the message transmission is complete. Thus, compared with the receiver, the adversary has the timing constraint, i.e., it must jam the message within the message transmission time. In contrast, the receiver can buffer received data content and then take time to despread.

Based on this observation, we herein propose the idea of "onion" spreading, in which we treat the index code as a new message and spread it using the RP-DSSS recursively. The jammer has to exhaustively try all the possible positions of all messages to identify the index code before the transmission completes, while the receiver has no such a time constraint. Moreover, the exhaustive search is not imposed at the receiver, because all index codes are revealed upon the finish of the transmission. Assume that the original message is of m bits. The index code can be inserted into m positions, and thus the length of the second new message is $\log_2 m$. The spreading of the second new message also results in a new index code, which yields a third new message of length $\log_2(\log_2 m)$. In this way, we can generate multiple new messages of lengths m, $\log_2 m$, $\log_2(\log_2 m)$,..., and 1. How do we transmit the original and all additional new messages in the presence of jamming?

Although the message length gets smaller, the message itself becomes more important for the despreading. Because the correct despreading of the i-th message will rely on the correct despreading of the next (i.e., i+1-th) message. For security, we adopt different levels of spreading codes. Specifically, code sets $\mathcal{P}_1$, $\mathcal{P}_2$,..., and $\mathcal{P}_n$ are used to spread the first, second, ..., and the n-th message, where the code set $\mathcal{P}_i$ contains stronger spreading codes of a higher spreading gain than the codes in the set $\mathcal{P}_{i-1}$. To achieve this, the length of a code in $\mathcal{P}_i$ should be larger than that of a code in $\mathcal{P}_{i-1}$. Note that the probability that an arbitrary position holds the index code increases as the message length decreases. We thus pad additional bits to each message to restrain this probability under a desired value. In what follows, we present the details of the presented scheme.

Spreading code sets and code sequence set: Both the sender and the receiver share multiple sets of the spreading codes and spreading code sequences. Let $\{\mathcal{P}_1, \mathcal{P}_2, ..., \mathcal{P}_n\}$ denote the publicly known spreading code sets. In this paper, we assume that $f_i \geq f_{i-1}$, where f_i is the code length for the code set $\mathcal{P}_i$. RP-DSSS does not limit to certain spreading codes and any type of spreading codes can be adopted as long as it has the low cross-correlation property. Typical codes include Walsh-Hadamard codes and Gold codes [8]. Let $\mathcal{C}_1, \mathcal{C}_2, ..., \mathcal{C}_n$ denote the sets of code sequences. $\mathcal{C}_i$ is used to spread the i-th message and each element of $\mathcal{C}_i$ is formed by the concatenation of spreading codes from $\mathcal{P}_i$. Assume that the length of the i-th message is l_i. $\mathcal{C}_i$ can be represented by $\mathcal{C}_i = p_{i_1}||p_{i_2}||...||p_{i_{l_k}}$. Each code sequence is associated with an index code, and we use $\mathcal{I}_1, \mathcal{I}_2, ..., \mathcal{I}_n$ to represent the index code sets that identify the code sequences for $\mathcal{C}_1, \mathcal{C}_2, ..., \mathcal{C}_n$.

Spreading: As an example shown in Figure 2, the original message M to be sent is of 128 bits and we use $m_1||m_2||, ..., ||m_{128}$ to represent M. To generate the first spread message M_1, the sender randomly picks a code sequence from $\mathcal{C}_1$. According to the chosen code sequence, the sender creates the spreading result in a way that is similar to RD-DSSS, i.e., two identical codes are used to spread "1" and two different codes are used to spread "0". The original message M has 128 bits and thus M_1 is formed by 128 spreading codes from $\mathcal{C}_1$. In this example, the length of the first message M_1 is 128. The sender inserts the index code of M_1 after the 127-th spreading code. So the insertion position $POS_1 = 0111111$. The second message M_2 is thus generated by spreading POS_1 using the code sequence set $\mathcal{C}_2$. The length of M_2 is 7 (i.e., $\log_2 128$). The sender inserts the index code of M_2 after the 4-rd spreading code and $POS_2 = 100$.



code set = {p1, p2, ..., pn}
M
Index code set C1 = {c1, c2, ..., cn}
bit 1 bit 2 bit 3 ... bit 127 bit 128
c1 = p3||p1||p8||...||p3||p9
c2 = p3||p1||p8||...||p3||p9
spreading
a spread message
a sequence of code
M1
POS1 = 127
M2
POS2 = 4
M3
POS3 = 2
M4
POS4 = 1
M5

Fig. 2: Randomized Postioning DSSS

De-spreading: The receiver receives messages M_1, M_2, M_3, M_4, and M_5. To recover the original message, the receiver despreads the last message M_5 first and the despreading order is reverse to the spreading order. To despread message M_i, the receiver needs to know two main things: (1) the position of the index code, and (2) the code sequence that was used to spread M_i. As shown in Figure 2, M_5 is spreaded by one spreading code, which indicates POS_4. In this example, POS_4 represents two possibilities and it is either 1 or 0. After POS_4 is identified, the receiver can obtain the index code and then continues to despread M_4. Assume that the obtained index code in M_4 is $\hat{c}$. The receiver computes the correlation between each index code in $\mathcal{I}_4$ and $\hat{c}$, finds the index code that can result in the highest correlation with $\hat{c}$, and uses the code sequence that is associated with this index code to despread M_4.

Let $D = d_1||d_2||...||d_1$ denote the spreading code sequence identified by the sender. Further let $S = s_1||s_2||...||s_l$ denote the concatenation of the spreading codes in M_4. The receiver calculates the correlation between S and D (i.e., $cor(s_1, d_1)$, $cor(s_2, d_2)$,..., $cor(s_{l_4}, d_{l_4})$) to find the index code of M_3. If $cor(s_i, d_i)$ is greater than a empirically per-defined threshold t, then the recover bit is 1. Otherwise, the recovered bit is 0. The recover message leads to the reveal of POS_3, which can be used to despread M_3. In a similar way, the receiver identifies POS_2 and POS_1 to despread M_2 and M_1, respectively.

Security enhancement by padding: As mentioned earlier, due to the decreased message length, the probability that the jammer can find the index codes increases as we approach to the end of the message chain. We propose to pad additional spreading codes to the end of spread messages to control this probability. Specifically, for the i-th message M_i, we randomly select L_{P_i} codes from the spreading code set $\mathcal{P}_i$ and pad them to the end of M_i. The index code can be inserted between any two consecutive spreading codes of the padded M_i. Lemma 1 gives the probability that an arbitrary insertion position (i.e., the position between two consecutive spreading codes) points to the beginning of the index code.

Lemma 1. *Assume insertion positions are equally likely to be inserted with the index codes. For $1 \leq i \leq n$, the probability $\mathbb{P}_i$ that an arbitrary insertion position of the i-th message M_i points to the beginning of the index code is $\frac{1}{\log_2(L_{i-1}+L_{P_{i-1}})+L_{P_i}}$, where L_{i-1} is the number spreading codes in the $i-1$-th message M_{i-1} before padding (L_0 is the number binary bits in the original message M), $L_{P_{i-1}}$ and L_{P_i} are the numbers of spreading codes padded to the end of M_{i-1} and M_i respectively.*

Proof. M_{i-1} is spreaded by L_{i-1} spreading codes and $L_{P_{i-1}}$ additional padding codes are appended to the end of M_{i-1}. Thus, the total number of spreading codes in M_{i-1} is $L_{i-1} + L_{P_{i-1}}$. The index code can be inserted between any two consecutive spreading codes of M_{i-1}, and thus the total number of candidate insertion positions is $L_{i-1} + L_{P_{i-1}}$, which leads to the next i-th message M_i that is composed by $\log_2(L_{i-1}+L_{P_{i-1}})$ spreading codes. After padding extra L_{P_i} spreading codes to the end of M_i, the message size increases to $\log_2(L_{i-1}+L_{P_{i-1}})+L_{P_i}$. Because an index code is inserted into the candidate insertion positions with equal probability, the probability $\mathbb{P}_i$ is therefore $\frac{1}{\log_2(L_{i-1}+L_{P_{i-1}})+L_{P_i}}$. $\square$

To find how many spreading codes should be padded, the sender can treat the padded bits L_{P_i} as the unknown variable and solve it from $\mathbb{P}_i \leq t_p$, where t_p is the desired upper bound of $\mathbb{P}_i$. Note that $L_{P_i} = 0$ indicates that no additional spreading codes are padded into M_i, and the security enhancement scheme degenerates to the original one without padding. Figure 3 plots the impact of the number L_{P_i} of padded spreading codes on the probability $\mathbb{P}_i$. The original message length is set to 128 and L_{P_i} ranges between 10% and 100% of the original message length. We can see that $\mathbb{P}_i$ significantly decreases as the number of padded spreading codes increases. In particular, for the last message M_5, $\mathbb{P}_i = 0.0015$ when the number of padded spreading codes is equal to the length of the original message.

Despreading cost: To despread, the receiver starts from the last received message M_n to identify the index code of M_{n-1}. M_n consists of $L_{n-1} + L_{P_n}$ spreading codes, and thus

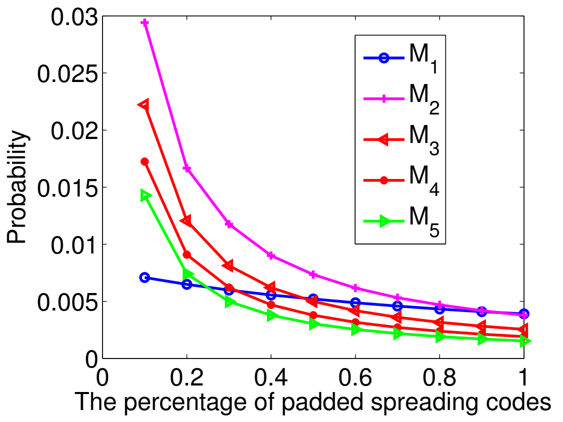


Fig. 3: $\mathbb{P}_i$ as a function of the number of padded spreading codes.

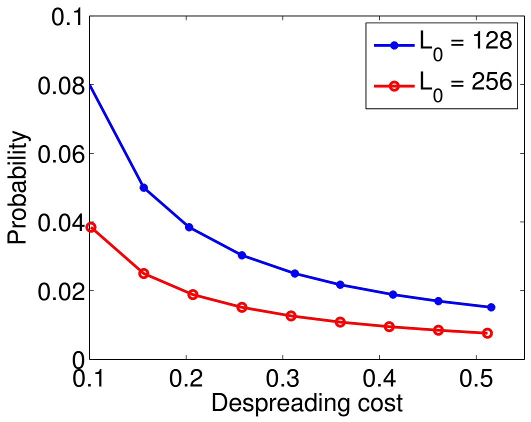


Fig. 4: $\mathbb{P}_i$ v.s. the despreading for message length L_0 of 128 and 256.

the receiver needs to examine $L_{n-1} + L_{P_n}$ positions, each of which yields a potential index code. To verify, the receiver correlate a potential index code with all the codes in $\mathcal{I}_n$ and a high correlation value that pass a certain threshold suggests a correct index code. We point out that the decoding cost of the index codes depends on the length of M_n. A large L_{P_n} of padded spreading codes leads to high level of security (low $\mathbb{P}$) but an increased decoding cost. Thus, L_{P_n} should be carefully controlled to satisfy the trade-off between security and the decoding cost.

Because the decoding cost depends on the number of spreading codes in M_n, we quantify the decoding cost using the ratio of the number of candidate positions in M_n to the length L_0 of the original message. This quantization will enable us to observe how much spreading cost is required as compared to a naive RD-DSSS improvement, which does not advocate the "onion" spreading but simply inserts the index code into a random position of a spread message. The decoding cost for this extension is thus the number of spreading codes in the spread message, and it is exactly the length of the original message. Figure 4 shows the jamming probability as a function of the decoding cost, we can see that RP-DSSS can reduce the probability $\mathbb{P}$ to small values while maintaining a low decoding cost compared to the naive RD-DSSS improvement. For example, with a spreading cost that is only 20% of that of the naive RD-DSSS improvement, the probability $\mathbb{P}$ is smaller than 0.02 and 0.04 for message length of 128 and 256, respectively.

IV. Simulation Result

We performed computer simulations using MATLAB to validate the performance of RP-DSSS. The Type I attack discussed in RD-DSSS causes the worst probability for a message being jammed. Thus, we consider the Type I attack in our simulation. In Type I attacks, the jammer randomly selects codes from the code set, and transmits them to interfere with the original message transmission. A type I attacker cannot flip a bit of 1 to 0, because no matter which code it transmits, two identical codes are always involved in the despreading process and the correlation is always high. Nevertheless, the attacker can flip a bit of 0 to 1. Let p_{s_1} and p_{s_2} denote the i-th codes of a spread message and the corresponding spreading code sequence respectively, where $p_{s_1} \neq p_{s_2}$. Further let p_{a_1} and p_{a_2} denote the i-th codes transmitted by the jammer along with the transmissions of p_{s_1} and p_{s_2}. We can see that a high correlation can be caused if $p_{p_1} = p_{a_1}$, or $p_{p_2} = p_{a_1}$, or $p_{p_1} = p_{a_2}$, or $p_{p_2} = p_{a_2}$. If any of the four conditions holds, 0 is decoded as 1 and a bit error occurs. In practice, 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 [11]. If the ECC can correct a maximum of δ bit errors of the original message but the jammer can flip more than δ bits, then jammer is successful and the receiver cannot reconstruct the original message.

In our simulation, we set the padded spread message lengths to be 64 and 128, and set the error correction code capability ECC_{Cap} to be 1% - 3%, which means that the number of tolerable bit errors is $1\% - 3\%$ of the message length. Specifically, $ECC_{Cap} = 1\%$ for length-128 message and $ECC_{Cap} = 3\%$ for length-64 message. We perform 500 trials. In each trial, we spread the message by picking a random code sequence from $\mathcal{C}$. We also generate random codes for the jammer. To calculate the probability that the message to be jammed, we count the number of error bits that affected by the jammer. If this number is larger than $ECC_{Cap} \times L_0$, then the trial is considered as failed. The probability that the message is jammed is finally calculated by $\frac{\text{number of failed trials}}{\text{total number of trials}}$. Figure 5 shows the simulation results for RP-DSSS. The jamming probability decreases significantly as the code set size increases. In particular, the jamming probability is around 0.01 for the code set size of 60.

V. Related Work

The jamming problem in wireless communication has been widely studied during the past few decades, and spread spec-

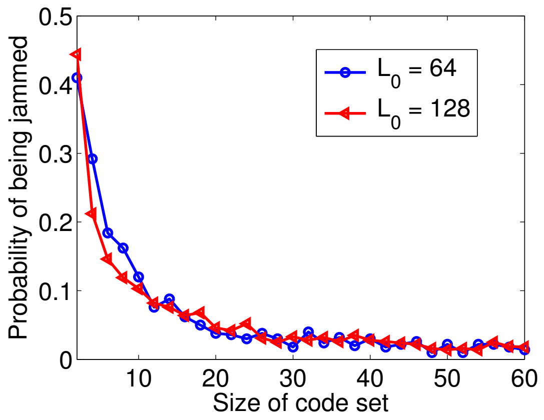


Fig. 5: The probability that the attacker jams the communication when the length of the message is 64 and 128 bits respectively.

trum such as DSSS and FHSS are traditional anti-jamming techniques [8]. However, as discussed earlier, those techniques require that senders and receivers to pre-share secret keys. Some researchers recently investigated how to enable jamming-resistant broadcast communication without shared keys [1], [4], [7]. 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]. Although this method works efficiently with short pulses in the time domain, it cannot be directly extended to DSSS or FHSS systems [4]. To support FHSS systems, Strasser et al. proposed Uncoordinated Frequency Hopping (UFH) schemes ([9], [10]) that allow the sender and the receiver that do not share a secret key to establish the jamming resilient wireless communication.

UDSSS and RD-DSSS proposed in [7] and [4] respectively are the most relevant schemes to ours. UDSSS randomly selects a spreading code sequence from a publicly known set to spread a message, and the receiver uses exhaustive search to identify the chosen sequence for despreading. As mentioned earlier, UDSSS faces an obvious vulnerability, i.e., once the jammer figures out the single chosen spreading code sequence, it can then jam the rest of the message transmission. Inspired by UDSSS, RD-DSSS tolerates this type of reactive jamming attacks by exploiting the correlations between spreading codes, but the index codes involved in essential despreading operations can easily become the target of adversaries. To improve the security of RD-DSS, we propose RP-DSSS to fix the open issue of index codes by randomizing the positions of these codes.

Recent work also consider the threats from broadband jammers, who can jam all frequency channels simultaneously and have a high transmit power to overcome the spreading gain. Specifically, Xu et al. proposed to use timing-based covert channels to address broadband jammers [12]. The covert channels are constructed by linking the inter-arrival times of a sender's corrupted packets to information bits. In addition, BitTrickle schemes are proposed by [3] to establish the wireless communication in the presence of a broadband reactive jammer. The basic idea is to utilize the short time delay caused by the channel sensing of a reactively jammer to deliver information bits. The receiver may collect information bits from the unjammed parts of received packets and assemble these bits together to obtain a meaningful message. All these approaches are complementary to ours.

VI. Acknowledgement

This work is supported by the Army Research Office under grant W911NF-14-1-0324.

VII. Conclusion

In this paper, we propose the RP-DSSS scheme to enhance the security of RD-DSSS. Instead of placing an index code at a fixed location, we hide the index code by inserting it at a random position of a spread message. To enable the receiver to find the position of the index code, we design an "onion" spreading mechanism, which recursively encodes the random positions using RP-DSSS, and pads additional spreading codes to reduce the probability that a jammer can infer these random positions. To achieve the anti-jamming without shared key, the spreading and despreading operations of RP-DSSS totally utilize publicly known sets of spreading codes and code sequences. We performed computer simulations to validate the performance of the proposed approaches.

References

[1] L. C. Baird, W. L. Bahn, M. D. Collins, M. C. Carlisle, and S. C. Butler. Keyless jam resistance. In *Information Assurance and Security Workshop, 2007. IAW'07. IEEE SMC*, pages 143–150. IEEE, 2007.

[2] A. Goldsmith. *Wireless communications*. Cambridge university press, 2005.

[3] Y. Liu and P. Ning. Bittrickle: Defending against broadband and high-power reactive jamming attacks. In *Proceedings of the 2012 IEEE INFOCOM*, 2012.

[4] Y. Liu, P. Ning, H. Dai, and A. Liu. Randomized differential dsss: Jamming-resistant wireless broadcast communication. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–9. IEEE, 2010.

[5] Y. H. Oh and D. J. Thuente. Enhanced security of random seed dsss algorithms against seed jamming attacks. In *Global Communications Conference (GLOBECOM), 2012 IEEE*, pages 801–806. IEEE, 2012.

[6] R. Poisel. *Modern Communications Jamming: Principles and Techniques*. Artech House, 2011.

[7] C. Pöpper, M. Strasser, S. Capkun, S. Capkun, and S. Capkun. Jamming-resistant broadcast communication without shared keys. In *USENIX security Symposium*, pages 231–248, 2009.

[8] 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.

[9] M. Strasser, S. Capkun, and M. Cagalj. Jamming-resistant key establishment using uncoordinated frequency hopping. In *Security and Privacy, 2008. SP 2008. IEEE Symposium on*, pages 64–78. IEEE, 2008.

[10] M. Strasser, C. Pöpper, and S. Čapkun. Efficient uncoordinated fhss anti-jamming communication. In *Proceedings of the tenth ACM international symposium on Mobile ad hoc networking and computing*, pages 207–218. ACM, 2009.

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

[12] W. Xu, W. Trappe, and Y. Zhang. Anti-jamming timing channels for wireless networks. In *WiSec '08: Proceedings of the first ACM conference on Wireless network security*, pages 203–213, New York, NY, USA, 2008. ACM.