Keyless Anti-Jamming Communication via Randomized DSSS

Ahmad Alagil

University of South Florida

Follow this and additional works at: https://digitalcommons.usf.edu/etd

Part of the Computer Engineering Commons, and the Computer Sciences Commons

Scholar Commons Citation

Alagil, Ahmad, "Keyless Anti-Jamming Communication via Randomized DSSS" (2020). USF Tampa Graduate Theses and Dissertations. https://digitalcommons.usf.edu/etd/8151

This Dissertation is brought to you for free and open access by the USF Graduate Theses and Dissertations at Digital Commons @ University of South Florida. It has been accepted for inclusion in USF Tampa Graduate Theses and Dissertations by an authorized administrator of Digital Commons @ University of South Florida. For more information, please contact digitalcommons@usf.edu.
Keyless Anti-Jamming Communication via Randomized DSSS

by

Ahmad Alagil

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Science and Engineering
Department of Computer Science and Engineering
College of Engineering
University of South Florida

Major Professor: Yao Liu, Ph.D.
   Jay Ligatti, Ph.D.
   Nasir Ghani, Ph.D.
   Ximming Ou, Ph.D.
   Kaiqi Xiong, Ph.D.

Date of Approval:
   March 24, 2020

Keywords: Security, Spread Spectrum, Correlation, Wireless Networks

Copyright © 2020, Ahmad Alagil
Dedication

To my parents and family.
Acknowledgments

First and foremost, I want to thank my advisor Dr. Yao Liu for her advice, support, patience, guidance, and encouragement throughout my graduate studies. She has taught me the skills necessary for academic research that enable me to work harder and smarter at the same time. I appreciate all of her time, efforts, and contributions that have helped to enrich my knowledge and to have further experience in my field. It has been my honor to be one of her Ph.D. students.

My grateful thanks go to the committee members, Dr. Jay Ligatti, Dr. Nasir Ghani, Dr. Xinming Ou, and Dr. Kaiqi Xiong, for their participation and the contribution. Their guidance, useful feedback has enriched my dissertation.

To my lab mates, Ian Markwood, Song Fang, Tao Wang, Yan Albright, Dakun Shen, Meshari Alotaibi, Cagri Cetin, and Jean Baptists Subils, Abed Alanazi, Xiaoshan Wang, Chengbin Hu, and Zi Li, I am grateful for the opportunity to have gotten to know each one of you, and all of your help and support. I wish all of you success in your future endeavors and look forward to your continued friendship.

To all my classmates and friends at the University of South Florida, thank you for the enjoyable and memorable moments spent with you. I appreciate the tremendous emotional support and academic advice you gave me.

I would like to sincerely thank my parents, brothers and sisters for encouraging me in all of my pursuits to achieve my goals. Their unwavering and emotional support is even present even though the distance between us is long.

Also, my heartfelt thanks go with love to my wife, Dimah Albassam, and my son, Abdullah. The successful completion of my Ph.D. study would not have been possible without their
unabiding love, fervent devotion, and constant support throughout this process. I can not imagine being on this life’s journey without them.

Last, but not least, I would like to thank my home country, Saudi Arabia, represented by the Saudi Arabian Cultural Mission (SACM) and Umm Al-Qura University (UQU) for their financial support during my graduate studies. Without their support, this dissertation would not have been possible.
Table of Contents

List of Tables iv

List of Figures v

Abstract vii

Chapter 1: Introduction 1
  1.1 Wireless Jamming Attack 3
    1.1.1 Background Information on FHSS 5
    1.1.2 Background Information on DSSS 6
    1.1.3 The limitations of FHSS and DSSS 8
      1.1.3.1 Randomized Positioning DSSS with Message Shuffling for Anti-jamming Wireless Communications 9
      1.1.3.2 Random Allocation Seed-DSSS Broadcast Communication against Jamming Attacks 10
  1.2 Summary of Contributions 10
  1.3 Dissertation Road-map 11

Chapter 2: Related Work 12
  2.1 Jamming Attacks 12
  2.2 Anti-Jamming Techniques in Wireless Communication Systems 13
  2.3 Wireless Physical Layer Security 17

Chapter 3: Randomized Positioning DSSS with Message Shuffling for Anti-Jamming Wireless Communications 19
  3.1 Basic Scheme of RD-DSSS 19
    3.1.1 Spreading 20
    3.1.2 Despreading 20
    3.1.3 The Limitation of RD-DSSS 21
  3.2 RP-DSSS 23
    3.2.1 Onion Spreading 23
    3.2.2 Spreading Code Sets and Code Sequence Set 24
    3.2.3 Spreading 25
    3.2.4 Despreading 25
    3.2.5 Security Enhancement by Padding 26
3.2.6 Padding Spreading Codes 28
3.2.7 Despreading Cost 28
3.3 The Enhanced RP-DSSS 29
3.4 Overview of Proposed Method 30
  3.4.1 Spreading Code Set and Code Sequence Set 31
  3.4.2 Spreading 32
  3.4.3 Despreading 33
  3.4.4 The Advantage of the Receiver over the Jammer 35
  3.4.5 The Probability of Hitting the Last Message 35
  3.4.6 Probability that the Receiver Can Recover the Original Message 35
  3.4.7 Storage Overhead 37
3.5 Simulation Result 37

Chapter 4: Random Allocation Seed-DSSS Broadcast Communication against Jamming Attacks 39
  4.1 Basic Scheme of Delayed Seed Disclosure-DSSS 39
    4.1.1 The Limitations of DSD-DSSS 39
  4.2 RAS-DSSS 41
    4.2.1 Code Sets 42
    4.2.2 Spreading the Original Message 42
    4.2.3 Spreading the Seed 42
    4.2.4 Randomization the Position of the Seed 43
    4.2.5 Despreading Process 43
    4.2.6 Sliding Window Approach 44
    4.2.7 Computation Overhead 46
    4.2.8 Security Enhancement 47
    4.2.9 Advantages for a Receiver over a Jammer 48
    4.2.10 How to Confuse a Reactive Jammer? 49
  4.3 Simulation Results 51
    4.3.1 Analysis 51
    4.3.2 Evaluation 53

Chapter 5: Future Work 58
  5.1 Future Possible Approaches to My Current Works 58
    5.1.1 Fragmentation the Seed into Equal Chunks within a Spread Message 58
    5.1.2 Adoption the Length of the Spreading Code 59
  5.2 Friendly Jamming 59
  5.3 Security Issues in Internet of Things 60

Chapter 6: Conclusion 62

References 64
Appendix A: Tables 74
Appendix B: Copyright Permissions 75
List of Tables

Table 2.1 Summary of the most relevant work to the research in the dissertation. 14

Table 4.1 Definition of symbol. 45

Table A1 Detection rates in the presence of attackers type 1, 2, and 3 when the length of spreading code \( (I_c) \) is 256, 512, and 1024, respectively 74
## List of Figures

| Figure 1.1 | Security requirements of the wireless networks. | 2 |
| Figure 1.2 | Wireless jamming attacks. | 4 |
| Figure 1.3 | Basic FHSS communication. | 5 |
| Figure 1.4 | Basic DSSS communication system. | 7 |
| Figure 3.1 | The basic scheme of DSSS. | 20 |
| Figure 3.2 | The limitation of RD-DSSS. | 21 |
| Figure 3.3 | Randomized positioning DSSS. | 22 |
| Figure 3.4 | $P_i$ as a function of the number of padded spreading codes. | 26 |
| Figure 3.5 | $P_i$ v.s. the despreading for message length $L_0$ of 128 and 256. | 27 |
| Figure 3.6 | Length of $M$ and number of spreading messages. | 32 |
| Figure 3.7 | The probability of hitting the last message. | 33 |
| Figure 3.8 | The probability $P$ that the receiver can recover the original message. | 34 |
| Figure 3.9 | Storage overhead when $k = 64$-bit, and $q = 5000$. | 36 |
| Figure 3.10 | Storage overhead when $k = 128$-bit, and $q = 5000$. | 36 |
| Figure 3.11 | The probability that the attacker jams the communication when the length of the message is 64 and 128 bits respectively. | 38 |
| Figure 4.1 | The limitation of DSD-DSSS. | 40 |
Figure 4.2  The proposed system (RAS-DSSS).

Figure 4.3  Locating the position of the seed.

Figure 4.4  Security enhancement.

Figure 4.5  The correlation between $SPM_1$ and $SPM_2$, and CDF of the high and low correlation of the received spreading messages in the presence of three types of attackers 1, 2 and 3.

Figure 4.6  Different situations show the effects when the seed ($s_x$) in the second spreading ($SPM_2$) message is not shifted, or shifted by 8, 16, 24 bits on the decision to identify the position of the seed in the first spreading message ($SPM_1$).

Figure 4.7  Number of success trials to find the position of the seed.

Figure 4.8  Probability for a non-reactive jammer to jam the seed.

Figure 4.9  Probability for a reactive jammer to jam the seed.

Figure 5.1  Enhanced RAS-DSSS.

Figure 5.2  A friendly jamming against relay attacks.
Abstract

Nowadays, wireless networking is ubiquitous. In wireless communication systems, multiple nodes exchange data during the transmission time. Due to the natural use of the communication channel, it is crucial to protect the physical layer to make wireless channels between nodes more reliable. Jamming attacks consider one of the most significant threats on wireless communication. Spread spectrum techniques have been widely used to mitigate the effects of the jammer. Traditional anti-jamming approaches like Frequency Hopping Spread Spectrum (FHSS) and Direct Sequence Spread Spectrum (DSSS) require a sender and a 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. This dissertation includes two works as a countermeasure of jamming attacks using DSSS without the requirement of sharing a secret key.

In the first work, we propose the Randomized Positioning DSSS (RP-DSSS) scheme as an extension to RD-DSSS. RP-DSSS randomly relocates the index codes for each message. This randomization hides the indices from the adversaries and thus achieves the enhanced security. Also in this dissertation, we design an onion spreading technique, which encodes the random positions recursively using RP-DSSS, and pads additional spreading codes to reduce the probability that a jammer can infer these random positions.

In the second work, we propose Random Allocation Seed DSSS (RAS-DSSS) which mainly focuses on concealing a random seed from the attacker by inserting it at a random position of a spreading message. In RAS-DSSS, We develop a new technique to identify the location
of a random seed at a receiver by aligning between multiple received messages. Thus, a receiver can obtain the position of a random seed, and then he can recover both a random seed then regenerating the spreading codes used to spread an original message.
Chapter 1: Introduction

The fast growth of wireless communications in the communications industry has captured the attention of the media and peoples imagination. In recent years, developed countries have witnessed the replacement of antiquated wired systems by wireless networks. For instance, many public buildings, airports, homes, hotels, businesses, and campuses have replaced wired systems by wireless systems.

Additionally, many realistic systems, such as automated highways, smart homes, and intelligent appliances, have emerged from various research ideas. The explosive growth of wireless networks in the communications industry in the future will exceed that of wired networks because of the many advantages it offers, such as convenience, mobility, expandability, and cost. Even though wireless networks have numerous benefits, they coupled with security vulnerabilities and threats.

In communication through a wireless system, the critical elements include the transmitter and receiver during transmission time of information. Due to the open share medium, wireless networks are prone to many security threats and attacks. Some of the safety threats are Denial of Service (DoS) attacks on wireless networks [1]. Other security vulnerabilities include man-in-the-middle (MITM) attack [2], message injection [3], and eavesdropping attack [4].

In wireless networks, exchanging data among legit and authorized nodes in the presence of numerous security threats is challenging. It is essential to maintain the security requirements for the wireless networks to protect the wireless channels from the attacks. Some of
the critical security requirements which make wireless networks to be more secure include confidentiality, integrity, and availability. These requirements are defined below:

1. **Confidentiality:**
   It implies that the ability to access data is available only to authorized users. Legitimate senders and receivers can transfer and receive data in a wireless communication system. Also, the data cannot disclose to unauthorized users. For instance, to assure confidentiality, the sender first encrypts the original message using a secret key known to the receiver. Then, the sender transmits the encrypted data to the receivers side. To recover the exact information, the recipient decrypts the received signal using the same secret key. An attacker cannot recover the original message without knowing the secret key. In this regard, the DiffieHellman key exchange (DH) protocol is used between nodes to generate a secret code over a channel that is public [5].

2. **Integrity:**
   In wireless communication systems, data integrity means that all transmitted data must be reliable and accurate without being altered or modified by unauthenticated
users. A compromised node can significantly damage information integrity by launching malicious attacks, such as data injection or message modification. Generally, it is difficult to identify the attacks resulting from nodes that are compromised because these malicious codes also have strong identities [6].

3. Availability:

The availability requirement means that all authorized users are capable of accessing all available resources on the network upon their requests. Data availability violation can be by the DoS attacks with capabilities of preventing both a sender and receiver from exchanging data by making the communication channel busy. For example, an unauthorized node can generate interference by launching DoS attacks, also called jamming attacks, which in turn disrupts the communication channel between legitimate users [7].

The security requirements of the wireless networks mentioned above should be the same as the one in wired networks. However, due to the open nature of the wireless communication system, it is subject to many security threats. For instance, jamming attacks affect the availability of wireless networks. As mentioned earlier, assuring all these security requirements of the wireless network can be challenging [8–14].

1.1 Wireless Jamming Attack

This section\(^1\) presents a brief introduction of jamming attacks. As we mention, in a wireless communication system, both a transmitter and a receiver share critical information during the transmission time. This makes the wireless channels to be subject to challenging aspects, such as noise and interference. Additionally, there are security concerns associated with wireless communication systems due to open nature of wireless networks, and the

limitation of the power and computation capability of each device. Jamming attacks aim to target the channel of wireless communication systems, and to prevent both a sender and a receiver from sharing data.

A jammer aims to prevent a legitimate sender and receiver from sending and receiving messages. He can simply emit noise signal to make the channel busy such that a sender is not able to transmit data. Also, a jammer can send junk data to a receiver. As a result, a receiver can not recover the received message correctly. Both non-reactive and reactive jammers are considered as the most popular types of jamming attacks. A non-reactive jammer has no knowledge about the channel usage status. On the other hand, a reactive jammer senses the channel before it starts jamming. If the channel is idle, the jammer stays quiet. If a jammer detects transmission on the channel between a sender and a receiver, it starts emitting noise signal on the channel.

In fact, it is important to make the wireless communication between a sender and a receiver reliable by developing new schemes to mitigate the effects of jamming attacks. Frequency Hopping Spread Spectrum (FHSS) and Direct Sequence Spread Spectrum (DSSS) are examples of spread spectrum techniques that have been widely used to defend against jamming attacks for wireless communication systems (e.g., [16, 17]). Traditional FHSS and DSSS assume that the sender and the receiver share the same secret key to encode messages.
The shared secret key enables the communicators to generate the same frequency hopping patterns or spreading code sequences [18]. In the following, I give background information about traditional FHSS and DSSS, respectively. Also, I point out the limitations of these techniques.

1.1.1 Background Information on FHSS

FHSS transmits the wireless signal over multiple channels to avoid interference. Specifically, the transmitter and the receiver hop from one frequency to another based on a shared frequency hopping pattern generated by a pseudo-noise (PN) generator. The transmitter
multiplies the transmit signal with a carrier of different frequencies, and the receiver searches signals according to the corresponding frequencies.

A receiver must be able to align with the signal sent by the transmitter. Towards this goal, both the transmitter and the receiver may stay on a certain channel without hopping until they hear from each other, or the transmitter may stay on a certain channel, and the receiver searches over different channels until it discovers the signal from the transmitter. Once the signal alignment is done, the transmitter and the receiver can start hopping over different frequencies according to the shared frequency hopping pattern. Strasser et. al. proposed a new way for signal alignment by using uncoordinated hopping and public key cryptography [19]. The basic idea is to let the transmitter and receiver randomly hop over different frequencies but at different hopping rates. Once the communicators are on the same channel, the transmitter sends the receiver a secret value that is encrypted by the public key of the receiver, who will decrypt this value using the corresponding private key. The transmitter and the receiver then generate shared hopping pattern based on the secret value. The advantage of this method is that the transmitter and the receiver are no longer required to have a shared secret key prior to the wireless communication.

1.1.2 Background Information on DSSS

The concept of DSSS is more complicated than FHSS. DSSS is a modulation technique that can be applied to a base band signal to increase its bandwidth. This can be achieved by multiplying an original signal by a spreading code formed by chips. This spreading code is considered as a pre-shared secret key between a sender and a receiver to expand a signal. Each spreading code is represented by a sequence of bits with values of 1 and -1, or 1 and 0 as polar or non-polar representations, respectively ([20,21]). Each bit of an original message is multiplied with a spreading code and the result is the spreading message to be transmitted through the wireless channel. A receiver does the reverse steps of a sender to despread a
message. Figure 1.4 shows a basic DSSS communication system with two functions, including spreading and de-spreading.

To illustrate how to spread and de-spread a message, assume that “10” represents the original message, and +1-1-1+1 represents the spreading code used to spread the original message. The sender first converts the original message “10” from un-polar to polar form, i.e., “10” is converted to +1-1. Then, the sender multiplies each bit of the original message with a spreading code to generate the spreading result. For this example, the result is +1-1-1+1-1+1+1-1.

To de-spread a received message, the receiver needs to correlate the received message with a local replica of the spreading code. In this example, the received message is +111+11+1+11 and the local replica of the spreading code is +1-1-1+1. The receiver aligns +1-1-1+1 with the rst 4 chips of the received message (i.e., +111+1) and correlates them to get bit +1 (i.e., “1” in non-polar form), and then aligns +1-1-1+1 with the next 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).
Regarding the synchronization, in DSSS, identifying the beginning of a received message from the incoming bit stream is important for a receiver to recover the original message. This can be achieved by taking advantage of the auto-correlation property of the spreading code. A receiver computes the correlation between the received message and the spreading code in a sliding window way, and high correlation indicates the beginning of a message \[21,22\].

1.1.3 The limitations of FHSS and DSSS

Both FHSS and DSSS have obvious limitations. In FHSS and DSSS, senders and receivers need to share the same secret key to establish anti-jamming communication, for instance, frequency hopping patterns in FH and spreading codes in DSSS. Additionally, if a jammer knows the secret key, then he can jam the communication channel by using the same frequency hopping patterns and spreading code.

The previous limitations of the classical anti-jamming techniques have developed in recent years. Research work has been done to overcome the concern of establishing jamming-resilient communication with a pre-shared secret key. Strasser et al. proposed an anti-jamming system based on an Uncoordinated Frequency Hopping (UFH), which enables two wireless devices to establish a secret key during the transmission time in the presence of a jammer (\([19]\)). Pöpper et al. created a scheme called Uncoordinated Direct Sequence Spread Spectrum (UDSSS), which can remove the requirement of a shared secret key for DSSS systems \([18]\), and a sender always selects random spreading code sequences from a public set to spread a message.

Randomized Differential DSSS (RD-DSSS) is another anti-jamming system developed to remove the requirement of a shared secret key. In RD-DSSS, both a sender and a receiver share public spreading code sequences. A sender spreads a message based on a chosen index code, which is appended to the end of a spread message to facilitate the decoding at the receiver \([17]\). Liu et al. developed a new scheme, named Delayed Seed-Disclosure DSSS
(DSD-DSSS), against jamming attacks with no need of sharing a secret key. They achieve anti-jamming wireless communication by generating a random seed, which can be used to generate random code sequences to spread each message. To disclose a seed, a sender spreads a seed using publicly known code sequence set and positions it to the end of a spreading message [23]. This dissertation presents two systems as an extension to DSD-DSSS in [23] and RD-DSSS in [17] to enhance their security issues. The following is a brief introduction of the proposed techniques.

1.1.3.1 Randomized Positioning DSSS with Message Shuffling for Anti-jamming Wireless Communications

The Randomized Positioning DSSS (RP-DSSS) scheme was proposed as an extension to improve the security of Randomized Differential DSSS (RD-DSSS). In RD-DSSS, 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. 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 despread 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.

1.1.3.2 Random Allocation Seed-DSSS Broadcast Communication against Jamming Attacks

We propose a new technique, named Random Allocation Seed-DSSS (RAS-DSSS), based on randomly allocating a position of a seed within a spreading message, which is an extension to improve the security of Delayed Seed-Disclosure DSSS (DSD-DSSS). In DSD-DSSS, if the size of a spreading message is fixed, a jammer can target a random seed within the transmission time. Also if the size of a spreading message is variable, a jammer can send noise signal for a long time to disrupt the data at a receiver. As a result, to protect a random seed, we propose to randomly insert the seed into a random position of a spreading message. To enable the receiver to obtain the exact position of a random seed, we propose a scheme based on aligning two spreading messages within two transmissions. For example, after receiving two spreading messages, a receiver uses a sliding window to identify the position of a random seed. If the receiver can successfully determine the position of a seed, then he can recover the original message by doing the reverse steps conducted by the sender.

1.2 Summary of Contributions

The contributions of this dissertation are summarized below:

- Randomized Positioning DSSS with Message Shuffling for Anti-Jamming Wireless Communications: The contribution of this work is three-fold. First, we identify the vulnerability of 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 from 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.

- Random Allocation Seed-DSSS Broadcast Communication against Jamming Attacks: The contribution of this work is three-fold. First, we identify the vulnerabilities on existing DSSS based anti-jamming systems, and we propose a new scheme to enhance the security of the most recent DSD-DSSS method, via positioning a random seed into a random position on a spreading message. Second, we develop a novel technique to identify the beginning of a random seed to enable the receiver to find the seed while concealing it from the jammer. Third, we perform simulations to validate our proposed system in the presence of different kind of jammers. The simulation results show that a receiver has a higher chance to find the seed compared with a jammer who needs to perform much efforts to find the seed within a transmission time. Furthermore, our simulations demonstrate how RAS-DSSS can reduce the probability that the random seed to be jammed. For instance, figure 4.8 and 4.9 illustrate the probability that the seed to be jammed in the presence of a non-reactive and a reactive jammer, respectively. The probability can be reduced significantly when the size of $C_e$ and $C_p$ is 7000.

1.3 Dissertation Road-map

The rest of this dissertation is organized as follows. Chapter 2 presents the related works. Chapter 3 demonstrates the proposed RP-DSSS scheme. Chapter 4 introduces the the second proposed system, RAS-DSSS. Chapters 5 and 6 describe the future work and the conclusion of this dissertation, respectively.
Chapter 2: Related Work

Wireless communications have been used widely to exchange data between multiple devices. In a wireless communication system, both a transmitter and a receiver exchange critical information during the transmission time. The adversaries can take advantage of the open network and may launch several security attacks including Denial of service (DoS) attack [1], man-in-the-middle (MITM) attack [2], eavesdropping attack [3], and message injection attack [4] to disrupt the transmission of data. Jamming attack is one of the most significant emerging threats in wireless communications. Different studies have been carried out in the last few decades (e.g., [21,24–28]). The following sections define jamming attacks and summarize previous and relevant works for anti-jamming techniques.

2.1 Jamming Attacks

Jamming attack is considered as one of the most serious threats to wireless communication systems as it not only blocks the communication between the nodes but also consume the energy of nodes. It disrupts the communication at the lowest layer, i.e. the physical layer, which makes it hard to mitigate. Jamming is used by the military as a tool to interrupt the communication of enemies [29]. Nevertheless, this threat applied to block the availability of using networks for civilian use (e.g., [1,30–33]). A jammer aims to prevent a legitimate sender and a receiver from sending and receiving messages. He can emit noise signal to make the channel busy, such that a sender can not transmit data wireless. Also, a jammer can send junk data to the receiver, and as a result, the receiver can not correctly recover the received
message. Thus, the major intention behind jamming attack is to disrupt the transmission of information by blocking the channel. As these attacks degrade the performance drastically, so the defense against them has been an increasing concern. Effective mechanisms are essential to avoid jamming attacks and to detect them. In last few decades, many studies have been conducted to introduce the countermeasures of jamming attacks (e.g., [2, 20, 25, 34–46]).

2.2 Anti-Jamming Techniques in Wireless Communication Systems

This section explores traditional and recently-developed anti-jamming systems on wireless communication networks. Table 2.1 gives a summary of the most relevant works in this field of study.

Spread spectrum techniques such as DSSS and FHSS are traditional anti-jamming techniques. These techniques can mitigate the effects of the jammer if a small part of the communication channels is jammed or if the transmission power of the jammer is low. There are some obvious limitations of the classic DSSS and FHSS. In these techniques both the sender and receiver agree on a pre-determined secret key. For instance, in FHSS and DSSS, the same frequency hopping pattern and Pseudo-Noise (PN) code are shared between the sender and the receiver, respectively. So, if the attacker somehow gets the knowledge of the frequency hopping pattern or the spreading codes used, he can easily jam the communication channel and prevent the legitimate nodes from communication [24, 40, 47–55].

Significant researches have been carried out, in the recent years, to overcome the limitations of the traditional FHSS and DSSS and to eliminate the requirement of a pre-define secret key. Strasser et al. [19] proposed an anti-jamming communication system, called as Uncoordinated Frequency Hopping (UFH) system. UFH allows the sender and the receiver who do not share a secret key to establish a secret key for the jamming resilient wireless communication. In this technique, two nodes communicate by randomly hopping between a set of known frequency channels. The packet would be transmitted successfully if the two
nodes send and receive at the same frequency. Later on, this system was advanced and improvements were made in [48,56].

Though UFH and its developments in [48,56] removed the requirement of sharing a secret key in wireless communication, they still have some limitations. One of these limitations is that each message was split into multiple packets due to which the resemblance process increased the overall time required by the system. Another limitation is that these techniques can establish a secret key between two nodes only, which makes there use limited in broadcast communication where the number of nodes is more.

Table 2.1: Summary of the most relevant work to the research in the dissertation.

<table>
<thead>
<tr>
<th>System</th>
<th>Description</th>
<th>Problem addressed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uncoordinated Frequency Hopping (UFH) [19]</td>
<td>a sender and a receiver hop randomly between a set of known frequency channels.</td>
<td>a jammer can sense the channel and sends message within the transmission time.</td>
</tr>
<tr>
<td>Uncoordinated DSSS (UD-DSSS) [18]</td>
<td>a sender spreads a message by randomly selecting a spreading code sequence from a publicly known set.</td>
<td>a jammer can jam the transmitted signal by guessing the spreading code.</td>
</tr>
<tr>
<td>Randomized Differential DSSS (RD-DSSS) [17]</td>
<td>the message spread based on the chosen index code from the public spreading code sequences.</td>
<td>a jammer targets the index code to jam the message.</td>
</tr>
<tr>
<td>Delayed seed-disclosure DSSS (DSD-DSSS) [23]</td>
<td>a sender generates a random seed to produce multiple spreading code sequences to spread a message.</td>
<td>a sender positions a random seed to the end of the spreading message which can be targeted by the attacker.</td>
</tr>
</tbody>
</table>

Baird et al. proposed a coding approach to encode the data into marks (e.g., short pulses at different times) that can be decoded without any prior knowledge of keys [57]. However, this method only works efficiently with short pulses in the time domain, and it cannot be
directly extended to DSSS or FHSS systems [17]. To support DSSS systems, UDSSS and RD-DSSS were proposed in [18] and [17] for jamming-resistant broadcast communication which does not require sharing of a secret key.

In UDSSS, the sender randomly selects a spreading code sequence, from a publicly known set, to spread the message and the receiver uses exhaustive search to identify the chosen sequence for de-spreading. Compared to UFH, UDSSS does not depend on fragmentation. However, it requires high proficiency to de-spread the received message. UDSSS is vulnerable as the jammer can jam the transmitted signal by guessing the code used to spread the original message. Also, if the jammer has enough computational power, it can de-spread the beginning of the spread signal to identify the spreading code used. Moreover, if it successfully identifies the spreading code, it can easily jam the rest of the spreading message within the transmission time [35].

To overcome the vulnerabilities of UDSSS, Randomized Differential DSSS (RD-DSSS) scheme was proposed [7]. In this technique, 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. The indices are randomly chosen from a public pseudo random code set and thus they are referred 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 de-spreading at the receiver and they are appended at the end of the spread messages, so they can easily become the jamming target of adversaries.

Later, Liu et al. developed a new technique, named delayed seed-disclosure DSSS (DSD-DSSS), against jamming attacks. This method achieves the anti-jamming wireless communication by generating a random seed which can be used to produce multiple spreading code sequences to spread a message [23]. To disclose a seed, a sender spreads a seed using publicly known code sequence set and positions it to the end of a spreading message. The receiver
buffers the received message, decodes the random seed in order to regenerate the spreading codes, and using the spreading codes to recover an original message.

DSD-DSSS also has some restrictions as the sender appends the spread seed to the end of the spreading message. Assume that the size of a message is fixed, a jammer can easily localize the seed to regenerate the spreading codes. They can use this code to spread the jamming signal to prevent a receiver from recovering an original message. Even if the size of the message is dynamic, the computation overhead required at the receiver side to recover the seed would be high. Moreover, the jammer can still jam the seed by sending random codes to confuse the receiver.

DSD-DSSS was later enhanced in [58]. Oh et al. developed an Advanced Random Seed DSSS (ARS-DSSS) system as an extension of DSD-DSSS scheme. They developed a new technique against seed jamming attacks. ARS-DSSS reduces the computation overhead to compute the random seed within a variable size message. It spreads the last bit of the random seed using a very long spreading code. However, the random seed is still vulnerable to jamming attacks because it is also positioned at the end of the spreading message. Although the size of a message may not be fixed, the jammer can estimate the approximate location range of the seed to focus jamming the end of the message.

Recent work also considers 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 [59]. 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 [60] 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 reactive 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.

In this dissertation, the major focus was on developing anti-jamming techniques using spread spectrum systems, especially DSSS. This study proposes RP-DSSS and RAS-DSSS techniques as an extension to DSD-DSSS in [23] and RD-DSSS in [17] to compensate the security issues in wireless networks. In RP-DSSS, we protect the index code by inserting it at a random position within a spread message instead of appending it at a fixed position. In the same way, in DSD-DSSS, we protect the seed by using it at a random position in a spread message.

Thus, there are many approaches and techniques proposed for jamming detection (e.g. [36, 61, 62, 62–65]), and minimizing the effects of jamming in wireless sensor networks (e.g. [2, 66]). All these approaches are complementary to ours.

2.3 Wireless Physical Layer Security

Security in the physical layer is an emerging topic in wireless communication networks [11, 67]. Adversaries who are located within the rage of legitimate communication constitute a significant threat to the security of the communication [68]. For example, the eavesdropper intercepts data transmission between authenticated users, and the jammer disrupts the legitimate communication between a sender and a receiver [69–72].

Other than traditional cryptography approaches, the physical layer characteristics, including radio channel properties and modulation errors, can be utilized to achieve security in the communication system [73]. Previous studies have proven that physical layer properties can be successfully used to ensure the security of wireless networks. For instance, information-theoretical security can be achieved by protecting the confidentiality of wireless communication against the eavesdropper. Additionally, the physical layer properties can be used to develop and design upper-level security algorithms for authentication and
key generation purposes (e.g., [2,74–81]). Also, several physical-layer security systems have been proposed to protect wireless information, including cooperative jamming [82–84], and beamforming [85–87]. These techniques are also complementary to the technique used in this dissertation.
Chapter 3: Randomized Positioning DSSS with Message Shuffling for Anti-Jamming Wireless Communications

This chapter\(^1\) introduces the first work in this dissertation. 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.

### 3.1 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.

\(^2\)This chapter was published in IEEE Conference on Computing, Networking and Communications 2016 [88] and IEEE Conference on Dependable and Secure Computing 2019 [89]. Permissions are included in Appendix A.
A sequence of codes
Index code set \( I = \{c_1, c_2, \ldots, c_n\} \)

\( c_1 : p_1 \| p_3 \| p_4 \| p_5 \)
\( c_2 : p_1 \| p_7 \| p_2 \| p_9 \)
\( \ldots \)

Codes set \{ \( p_1, p_2, \ldots, p_n \) \}

The original message \( M = 4 \)-bit

<table>
<thead>
<tr>
<th>( p_1 )</th>
<th>( p_2 )</th>
<th>( p_4 )</th>
<th>( p_8 )</th>
<th>( p_1 )</th>
<th>( p_3 )</th>
<th>( p_4 )</th>
<th>( p_5 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

A spread message
A sequence of codes

Figure 3.1: The basic scheme of DSSS.

3.1.1 Spreading

Figure 3.1 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 \( p_1, p_3, p_4, \) and \( p_5 \) 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 \( p_2 \) is selected. The rest of bits in \( M \) can be encoded in the same way and the spreading outcome is \( p_1 \| p_2 \| p_4 \| p_8 \| p_1 \| p_3 \| p_4 \| p_5 \).

3.1.2 Despreading

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 \( \text{cor}(p_1, p_1), \text{cor}(p_2, p_3), \text{cor}(p_4, p_4), \) and \( \text{cor}(p_8, p_5) \), 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 3.1, the sender randomly chooses a spreading sequence from the spreading sequence set and it is the sequence \( p_1\|
abla p_3\|
abla p_4\|
abla p_5 \). 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 \( p_1\|
abla p_2\|
abla p_4\|
abla p_8\|
abla c_1 \).

3.1.3 The Limitation of RD-DSSS

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 despreading at the receiver.

Figure 3.2: The limitation of RD-DSSS.
Figure 3.3: Randomized positioning DSSS.
3.2 RP-DSSS

To overcome the limitation of the RD-DSSS, we propose that the spreading codes have the random-look because they are indeed generated by pseudorandom generators [21], 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.

3.2.1 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 despreading. 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 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 generate multiple new
messages of lengths \( m, \log_2 m, \log_2 (\log_2 m), \ldots \), and 1. How do we transmit the original and
all additional 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 \( P_1, P_2, \ldots, P_n \) are used to spread the first,
second, \( \ldots \), and the \( n \)-th message, where the code set \( P_i \) contains stronger spreading codes
of a higher spreading gain than the codes in the set \( P_{i-1} \). To achieve this, the length of a
code in \( P_i \) should be larger than that of a code in \( P_{i-1} \). 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.

3.2.2 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 \( \mathcal{P} = \{ P_1, P_2, \ldots, P_n \} \) 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 \( P_i \).
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 [21]. Let \( C_1, C_2, \ldots, C_n \) denote the sets of code sequences.
\( C_i \) is used to spread the \( i \)-th message and each element of \( C_i \) 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 $\mathcal{C}_i = p_{i_1}||p_{i_2}||...||p_{i_k}$. 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 $\mathcal{C}_1, \mathcal{C}_2, ..., \mathcal{C}_n$.

3.2.3 Spreading

As an example shown in Figure 3.3, 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 $POS_2 = 100$.

3.2.4 Despreading

The receiver receives messages $M_1, M_2, M_3, M_4, \text{ and } M_5$. To recover the original message, the receiver first despreads the last message $M_5$ 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 3.3, $M_5$ 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 $I_4$ and $\hat{c}$, finds the index code that leads to the highest correlation with $\hat{c}$, and uses the code sequence 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 $S = s_1||s_2||...||s_1$ denote the concatenation of the spreading codes in $M_4$. The receiver calculates the correlation between $S$ and $D$ (i.e., $\text{cor}(s_1, d_1)$, $\text{cor}(s_2, d_2)$,..., $\text{cor}(s_{14}, d_{14})$) to find the index code of $M_3$. If $\text{cor}(s_i, d_i)$ 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.

![Figure 3.4: $P_i$ as a function of the number of padded spreading codes.](image)

3.2.5 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 $L_{Pi}$ 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 $P_i$ that an arbitrary insertion position of the $i$-th message $M_i$ points to the beginning of the index code is

$$P_i = \frac{1}{\log_2(L_{i-1} + L_{P_{i-1}}) + L_{P_i}},$$

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.

![Figure 3.5: $P_i$ v.s. the despreading for message length $L_0$ of 128 and 256.](image)

**Proof.** $M_{i-1}$ is spread 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 $P_i$ is therefore $\frac{1}{\log_2(L_{i-1} + L_{P_{i-1}}) + L_{P_i}}$.

3.2.6 Padding Spreading Codes

The sender treats the padded bits $L_{P_i}$ as the unknown variable and solve it from $P_i \leq t_p$, where $t_p$ is the desired upper bound of $P_i$. 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 3.4 plots the impact of the number $L_{P_i}$ of padded spreading codes on the probability $P_i$. 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 $P_i$ significantly decreases as the number of padded spreading codes increases. In particular, for the last message $M_5$, $P_i = 0.0015$ when the number of padded spreading codes is equal to the length of the original message.

3.2.7 Despreading Cost

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 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 $I_n$ 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 $P$) 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
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 3.5 shows the jamming probability as a function of the decoding cost, we can see that RP-DSSS can reduce the probability $P$ 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 $P$ is smaller than 0.02 and 0.04 for message length of 128 and 256, respectively.

3.3 The Enhanced RP-DSSS

In RP-DSSS, the probability that a jammer can identify the index code increases as the length of the i-th message $M_i$ decreases. Specifically, the last message $M_n$ has the most significant information for a receiver to despread $M_n - 1$. The receiver despreads $M_n$ to identify the index code used to spread $M_n - 1$. If the receiver can successfully identify the index code in $M_n - 1$, then he can identify the index code of $M_n - 2$ and so on for the rest of messages. Consequently, the attacker can target $M_n$ in order to prevent the receiver from recovering the original message.

We propose enhanced RP-DSSS to address this limitation. The enhanced technique uses the same way as RP-DSSS technique to spread and despread a message, but permutations of messages are used to enhance the security. Similar to RP-DSSS, it does not require sharing a secret key between the sender and the receiver and more bits at the end of the spreading messages to confuse jammers.
3.4 Overview of Proposed Method

The successful decoding of the $i$-th message depends on that of the $i+1$-th message. If we can hide the last message from the jammer, then the probability that the jammer can identify and jam the index code of this messages and all other massages will be reduced. To hide the last message, we propose to randomize the transmission order of all messages. After generating the five spreading messages, the sender sends these messages in a random order (e.g., $M_3, M_2, M_1, M_5$, and $M_4$). Note that these messages vary in length, because the $i$-th message encodes the index code of the $i-1$-th message and accordingly the size of the $i$-th message is much shorter than the previous one. To prevent the jammer identifying the original order of spreading messages, we propose to pad extra bits so that all messages have the same length as the first message. For example, $M_2, ..., M_5$ are of the same length as $M_1$.

To despread the messages, both the receiver and the attacker need to check all possible combinations. For example, if an original message is encoded into five spreading messages, then they need to check 120 possible message combinations (i.e., 5!). The correct message combination should allow the successful verification of the checksum and cryptographic signatures. However, the receiver has a significant advantage over the attacker in that the receiver can first buffer all messages and then try to decode and verify each possible message combination. Compared with the receiver, the jammer has to finish the decoding and verification of all 120 combinations before the transmission of the five spreading messages ends. Otherwise, the jammer fails the jam goal.

It seems that the number of message combinations increases exponentially as the number of spreading messages increases, and the decoding could be impossible if the original message length is high. Nevertheless, for a message of length $n$, the number of spreading messages actually increases very slightly in a logarithm way as the length of original message increases.
For example, when the original message length increases from 128 bits to 4096 bytes (the maximum transmission unit size of 802.11), the number of required spreading messages is still 5. To further reduce the computational overhead at the receiver, instead of transmitting a super large packet, the sender can divide large files and data into smaller standard packets and spread each packet and send resulting spreading messages to the wireless channel.

3.4.1 Spreading Code Set and Code Sequence Set

Similar to basic RP-DSSS, both the sender and the receiver share a spreading code set $\mathcal{P}$. Let $\mathcal{P} = \{p_1, p_2, ..., p_n\}$ denote the share spreading code set of $n$ codes. Unlike the basic RP-DSSS, the enhance RP-DSSS assumes that codes in $\mathcal{P}$ have the same length, because we would like to reduce the chance that the jammer can discover the original order of spreading messages by looking into message length. The code sequence set can be generated using the following two generation schemes.

1. **Basic generation scheme:**

   A dedicated set of code sequences is generated for each message. For instance, $\mathcal{C}_1, \mathcal{C}_2, ..., \mathcal{C}_n$ are used to spread messages $M_1, M_2, ..., M_n$ respectively. Each element of $\mathcal{C}_i$ is formed by the concatenation of spreading codes from $\mathcal{P}$, and each code sequence $\mathcal{C}_i$ is associated with an index code $I_i$.

2. **Overhead reduction generation scheme:**

   To reduce the storage overhead, we can use one code sequence set $\mathcal{C}$ for all messages. Assume that $\mathcal{C} = \{c_1, c_2, ..., c_n\}$ denotes the set of code sequences. If the length of the original message is $m$, then each $c_i$ in $\mathcal{C}$ is formed by concatenating $m$ random codes from $\mathcal{P}$. Assume $\mathcal{C}$ has $q$ code sequences established in advance. Also assume that $\mathcal{I}$ represents the index code set, and each index code is associated with a code sequence $c_i$. Like RP-DSSS, $\mathcal{P}$, $\mathcal{C}$, and $\mathcal{I}$ are publicly known.
3.4.2 Spreading

The sender spreads the original message $M$ using the same spreading method as proposed for RP-DSSS. One important extra step is that the sender needs to pad messages so that they have the same length. The sender may do padding either before or after spreading. Assume that the length of an original message is $m$. If the sender decides to add padding data before spreading, the sender can pad random bits to messages so that they are of the same length equal to $m$. If overhead reduction generation scheme is used, there is only one code sequence set $C$ and the length of a code sequence in $C$ is exactly the same as that of the original message, i.e., $m$. Thus, for the $i$-th message $M_i$, the sender can simply picks one code sequence from $C$ in a random way and uses it to spread $M_i$. Note that the basic generation scheme generates multiple different code sequence sets with different code sequence lengths, but a unique code sequence set with the same code sequence length is sufficient for padding before spreading. To simplify calculation, we do padding before spreading only with the overhead reduction generation scheme.
If the sender decides to do padding after spreading, for the overhead reduction generation scheme, the sender uses the first $l_i$ codes to spread message $M_i$ and then appends the remaining $m - l_i$ codes to the end of the spreading results, where $l_i$ is the length of $M_i$. For example, the length of $M_2$ is $\log_2(m)$ and the first $\log_2(m)$ spreading codes in the chosen code sequence are used to spread $M_2$ and the remaining $m - \log_2(m)$ codes are used as padding bits. For the basic generation scheme, the sender randomly chooses a code sequence from $C_i$ to spread $M_i$, randomly chooses $m - \log_2(m)$ codes from the code set $P$, and then append these codes to the end of the spreading result.

![Figure 3.7: The probability of hitting the last message.](image)

3.4.3 Despreading

After receiving all $n$ messages from the sender, the receiver needs to despread all possible combinations to identify the correct message order and recover the original message. For example, if the original message results in 3 spreading messages, the first, second, third received spreading messages can be mapped to one of the 6 possibilities, including $(M_1, M_2, M_3), (M_1, M_3, M_2), (M_2, M_1, M_3), (M_2, M_3, M_1), (M_3, M_1, M_2), (M_3, M_2, M_1)$. For each possibility, the receiver first removes the padded data from the messages, and then uses
RP-DSSS to despread. The correct combination will enable the receiver to see a meaningful
decoded message that can pass message authentication or integrality check.

![Figure 3.8: The probability $P$ that the receiver can recover the original message.](image)

Let $\log_2^n[x]$ denote the recursive logarithm calculation of the ceiling of $x$ for $n$ times. For example, $\log_2^3[x] = (\log_2[\log_2[\log_2[x]])])$. The length of $M_1$, $M_2$, $M_3$, ..., $M_i$ is thus $m$, $\log_2[m]$, $\log_2^2[m]$, ..., $\log_2^{i-1}[m]$, respectively. The last message $M_n$ has only one bit and hence the total number $n$ of messages can be calculated by solving $n$ from equation $\log_2^{n-1}[m] = 1$. We use $f(\log_2[m],1)$ to represent $n$. The total number of combinations that the receiver needs to verify is thus $f(\log_2[m],1)!$. As discussed earlier, $f(\log_2[m],1)$ increases very slightly as $m$ increases and a large file can be divided into smaller standard network packets for transmission. Therefore, the total number of combinations that the receiver needs to verify can maintain a small value even if the original message size goes up dynamically. Figure 3.6 shows that the number of spreading messages is 6 spreading messages even the length of original message is $10^5$-bit.
3.4.4 The Advantage of the Receiver over the Jammer

The receiver has advantage over the jammer that he can buffer the received messages and take his time to despread them. In contrast, the jammer has to do exhaustive search to obtain the chosen index code sequences used by the sender during transmission time. For example, if the length of the original message $M$ is 512-bit, and the number of chips in each spreading code is 128. Also if the wireless device used is 802.11n with data rate of 50Mbps then the transmission time will be $\frac{\text{number of bits}}{\text{data rate}} = \frac{512 \times 128}{50 \times 10^6} = 0.00128$ second. As a result, the jammer will find difficulties to do the computational process within the transmission time.

3.4.5 The Probability of Hitting the Last Message

Figure 3.7 shows the impact of the number of the spreading messages $SP_M$ and the number of re-transmission $n$ of these messages on the probability that the jammer hits the last message $m_i$. As $n$ and $SP_M$ increase, $P$ decreases.

3.4.6 Probability that the Receiver Can Recover the Original Message

The main goal of the jammer is to corrupt the legitimate transmission of data between the sender and the receiver. Both the sender and the receiver can mitigate the effects of corrupting data by retransmitting data at the sender side. Retransmitting the spread messages plays an important role to reduce the probability that one of the spreading messages is jammed. Assume that the probability $P$ that one of the spreading messages is jammed. Further assume the number of spreading messages is $i$-message. Then the probability that one spreading message is not jammed is $(1-P)$. As a result, the probability that all spreading messages are not jammed is $(1 - P)^i$. Assuming $n$ is the number of re-transmission of these spreading messages, the probability that all spreading messages are jammed is $(1 - (1 - P)^i)^n$. As a result, the probability $P_{Rx}$ that the receiver can receive correct spreading messages and
recover the original message \( M \) is \([1 - (1 - (1 - P)^i)^n]\). Figure 3.8 shows that when \( P=0.4 \), \( P_{Rx} \) is increased from 0.077 to 0.33 after increasing the number of re-transmission from 1 to 5. Thus, \( P_{Rx} \) significantly increases as \( P \) decreases and \( n \) increases. In other words, we can increase the probability that the receiver can successfully recovered \( M \) through increasing the number of re-transmissions of the spreading messages.

![Graph showing storage overhead](image1)

Figure 3.9: Storage overhead when \( k = 64\)-bit, and \( q = 5000 \).

![Graph showing storage overhead](image2)

Figure 3.10: Storage overhead when \( k = 128\)-bit, and \( q = 5000 \).
3.4.7 Storage Overhead

The enhance RP-DSSS systems require to store $P$, $C$, and $I$. At the receiver side, the receiver has to buffer the received messages in order to decode them. Figure 3.9 and 3.10 show a comparison of the storage overhead when RP-DSSS system has one code sequence set and multiple code sequences sets. In figure Figure 3.9 and 3.10, we can see that when the number of the code sequences $q$ is ranges between 1 to 5000, and the number of bit in each chips code $k = 64$-bit and 128-bit respectively, one code sequence set success in reducing the storage overhead compared with multiple code sequence sets.

3.5 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 despread process and the correlation is always high. Nevertheless, the attacker can flip a bit of 0 to 1. Let $p_{s1}$ and $p_{s2}$ denote the $i$-th codes of a spread message and the corresponding spreading code sequence respectively, where $p_{s1} \neq p_{s2}$. Further let $p_{a1}$ and $p_{a2}$ denote the $i$-th codes transmitted by the jammer along with the transmissions of $p_{s1}$ and $p_{s2}$. We can see that a high correlation can be caused if $p_{p1} = p_{a1}$, or $p_{p2} = p_{a1}$, or $p_{p1} = p_{a2}$, or $p_{p2} = p_{a2}$. 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 [90]. If the ECC can correct
a maximum of $\delta$ bit errors of the original message but the jammer can flip more than $\delta$ 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 $C$. 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 the number of failed trials divided by the total number of trials. Figure 3.11 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.
Chapter 4: Random Allocation Seed-DSSS Broadcast Communication against Jamming Attacks

In this chapter\(^1\), we propose a new scheme that we refer to as RAS-DSSS against jamming attacks. Similar to the traditional DSSS, RAS-DSSS spreads messages using spreading code sequences, and to remove the requirement of a pre-define secret key, RAS-DSSS uses multiple sets of code sequences to spread a random seed and the original message. In the following, we first introduce DSD-DSSS and its vulnerabilities, and then we present the proposed RAS-DSSS scheme that enhances the security of DSD-DSSS.

4.1 Basic Scheme of Delayed Seed Disclosure-DSSS

DSD-DSSS achieves the anti-jamming capability by spreading an original message based on a random seed. This random seed is used to generate random spreading codes to spread the original message. To disclose a random seed, DSD-DSSS appends it to the end of the spreading message. The receiver buffers the received message, decodes the random seed to regenerate the spreading codes, and then recover an original message using the spreading codes.

4.1.1 The Limitations of DSD-DSSS

DSD-DSSS has some key limitations that can make it vulnerable to the jammer. In the following, we discuss these limitations and security issues of DSD-DSSS:

\(^1\)This chapter was published in SecureComm 2019 [91]. Permission is included in Appendix A.
In DSD-DSSS, a sender first spreads the last bit of a seed using a random code sequence. Then he spreads the rest of bits of a seed one by one from the end to the beginning based on a specific function. At the end, the sender appends the spread seed to the end of the spreading message. Note that the sender always appends the spreading seed at the end of the spreading message. If the size of a message is fixed, the jammer can focus on jamming the seed only to prevent the receiver from identifying the seed and recover the original message. On the other hand, if the size of the message is dynamic, the computation overhead for despreading will increase at the receiver, and a jammer can still send random codes from to confuse the receiver.

In terms of the computation overhead, in DSD-DSSS, a receiver needs to de-spread a random seed in order to regenerate the spreading codes used to spread an original message. To achieve this, a receiver buffers the received message. Then he needs to locate a seed from the received message. This can take a non-trivial computational time. Assume that a random seed is of $l_s$-bit, the length of the spreading code is $l_c$, and the receiver applies a sliding window approach to identify a seed. The size of the sliding window should be $l_s \times l$. 
4.2 RAS-DSSS

We create a new scheme to overcome the limitations of DSD-DSSS. In DSD-DSSS, a sender always discloses a seed at the end of a spread message. Although, a sender spreads a seed using random codes. A jammer may target the seed only to prevent the receiver from recovering an original message. Instead of appending a seed to the end of a spread
message, we propose to allocate a seed at a random position of a spread message to reduce the probability that a jammer can identify the location of the seed. We also design a new technique that assists a receiver to determine the position of a random seed with low computation overhead.

4.2.1 Code Sets

Both a sender and a receiver share two publicly known spreading code sequences sets $C_p$ and $C_e$ which are used to spread an original message $M$ and a random seed $s_i$, respectively. $C_p$ and $C_e$ do not overlap with each other, and the cross-correlation between spreading codes are low, and each spreading code has its unique index code. We assume that each code in $C_p$ and $C_e$ is of $l_c$ bits.

4.2.2 Spreading the Original Message

If the original message $M$ is of $l_m$-bit. The sender spreads each bit in $M$ separately based on the chosen spreading code from the spreading code set $C_p$. Figure 4.2 shows an example of generating the spreading message. First, a sender needs to generate a random seed $s_i$, which is used with a pseudo-random generator to generate $l_m$ random indexes $mid_1, mid_2, ..., mid_{l_m}$, where $1 \leq mid_i \leq |C_p|$. Then a sender uses these indexes to spread each bit in $M$ with its corresponding spreading code in $C_p$. For example, the first bit in $M$ is spread using $C_p[mid_1]$ and so on for the rest of the bits in $M$. We denote $S(C_p, M)$ as the spreading message.

4.2.3 Spreading the Seed

To protect a seed from being jammed, a sender spreads a seed using the spreading codes in $C_e$. Assume that a seed $s$ is of $l_s$-bit. The sender spreads each bit of $s$ by drawing codes from $C_e$ based on the random choices of indexes $sid_1||sid_2||...||sid_{l_s}$ in $C_e$. The sender transmits the same spreading message twice. In each transmission, a sender spreads a seed
using different codes from $C_e$. Assume that $s$ with length $l_s$ equal to 3 bits is spread in the first transmission using $c_1$, $c_{10}$, and $c_8$. Also assume that $S(C_e, s)$ represents the result of the spreading outcome of a seed in the first transmission. In the second transmission, this seed will be spread using codes different from those used in the first transmission (e.g. $c_3$, $c_7$, and $c_6$). The result of this spreading is denoted as $S_x(C_e, s)$. The reason for this is to make sure there is no repetition between the chosen codes used to spread $s$, and $s_x$. As a result, this reduces the result of the computed correlation between $S(C_e, s)$ and $S_x(C_e, s)$ to the lower level.

4.2.4 Randomization the Position of the Seed

As mentioned earlier, a sender uses the random seed to generate the indexes of the spreading codes to spread an original message $M$. If a seed is disclosed at the end of the spreading message, the attacker can easily target a seed to prevent a receiver from recovering an original message. To address this concern, we propose to randomize the position of a seed in the spreading message to confuse the attacker and reduce the probability that a seed is jammed. In figure 4.2, we denote the random position of a seed as $pos$. Then a seed $S(C_e, s_i)$ is positioned between $S(C_p, M_{1:pos-1})$, and $S(C_p, M_{pos:s})$.

4.2.5 Despreading Process

If a receiver can identify the position of a seed from the received spreading messages, he can successfully recover an original message by regenerating the indexes used by a sender to spread $M$. As we mentioned earlier, multiple spreading messages will be received by the receiver. In each transmission, a random seed is inserted at a random position different from the one in the previous transmission. Figure 4.3 gives an example for the de-spreading process. After the first transmission, a random seed inserted at $pos_1$ in the spread message. However, in the second transmission, a random seed positioned at $pos_2$, and we assume that
pos₁ ≠ pos₂. A receiver buffers both received messages and then scans both messages using a sliding window approach.

![Figure 4.3: Locating the position of the seed.](image)

**4.2.6 Sliding Window Approach**

In Figure 4.3, to obtain the position of the seed, a receiver needs to compare between the two received spreading messages $SPM_1$ and $SPM_2$ using a sliding window approach. Assume that $W$ denotes the window size of a sliding window, and $W = l_s \times l_c$. Where $l_s$ and $l_c$ represent the length of a random seed and the length of the spreading code, respectively. If $x = x_1, x_2, ..., x_k$, and $y = y_1, y_2, ..., y_k$ represent the first $W$ in $SPM_1$ and $SPM_2$, respectively. A receiver computes the correlation between $x$ and $y$ which can be calculated by $\frac{1}{k} \sum_{i=1}^{k} x_i y_i$. If the result of this computed correlation is above the threshold value $\gamma$, this means High ($H$). If not, that means Low ($L$). In the second time, a receiver shifts the sliding window to the right by one chip and this sliding window can be denoted as $2W$. Following that a receiver computes the correlation between $x$ and $y$ within $2W$, and so on to the end of $SPM_1$ and $SPM_2$. After performing $l_m + l_s$ comparisons, the receiver can obtain the position of a random seed within $SPM_1$. 
Table 4.1: Definition of symbol.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M$</td>
<td>The original message</td>
</tr>
<tr>
<td>$s_i$</td>
<td>The random seed</td>
</tr>
<tr>
<td>$l_m$</td>
<td>The length of $M$</td>
</tr>
<tr>
<td>$l_s$</td>
<td>The length of $s_i$</td>
</tr>
<tr>
<td>$l_c$</td>
<td>The length of each spreading code</td>
</tr>
<tr>
<td>$C_e$</td>
<td>Spreading codes set to spread $M$</td>
</tr>
<tr>
<td>$C_p$</td>
<td>Spreading codes set to spread $s_i$</td>
</tr>
<tr>
<td>$i$</td>
<td>Number of spreading codes in $C_e$</td>
</tr>
<tr>
<td>$S(C_e, M)$</td>
<td>The result of spreading $M$</td>
</tr>
<tr>
<td>$S(C_p, s_i)$</td>
<td>The result of spreading $s_i$</td>
</tr>
<tr>
<td>$POSI_1$</td>
<td>The chosen random position</td>
</tr>
<tr>
<td>$C_x$</td>
<td>A long spreading code</td>
</tr>
<tr>
<td>$T_{seed}$</td>
<td>The time required by the receiver to obtain $s_i$</td>
</tr>
<tr>
<td>$P_{nrj}$</td>
<td>The probability for a non-reactive to jam $s_i$</td>
</tr>
<tr>
<td>$P_{rj}$</td>
<td>The probability for a reactive to jam $s_i$</td>
</tr>
</tbody>
</table>

Once the seed is localized, the receiver can start de-spreading the seed to regenerate the indexes used for spreading the original message by trying all possible spreading codes in $C_e$. The reason for this is because the seed is spread using random codes from $C_e$. As a result, the receiver applies a sliding window approach with window size $l_c$. If $c_i$ and $\hat{s}$ represent the chosen spreading code from $C_e$ and the first $l_c$ chips from the seed, the receiver uses a sliding window approach to compute the correlation between $c_i$ and $\hat{s}$. If the receiver gets a maximum correlation, this means that $c_i$ was used to spread $\hat{s}$ in the seed. The receiver applies this using all spreading codes in $C_e$ to retrieve the seed. Indeed, the receiver performs the reverse processes done by a sender to recover an original message.
4.2.7 Computation Overhead

A receiver needs to despread a random seed to regenerate the spreading codes used to spread the original message. In DSD-DSSS, the receiver needs to synchronize and decode the random seed to regenerate the spreading codes used to spread the original message. This means that the computation overhead is high compared with RAS-DSSS. As a result, a receiver may try all possible spreading codes in $C_e$ to spread chips in the received message and recover the random seed. Assume that $T_{seed}$ is the time required by the receiver to obtain the random seed within the received message.

$$T_{seed} = ((l_m \times l_c) + (l_s \times l_c)) \times i$$

In RAS-DSSS, a receiver uses a sliding window approach to identify the position of a seed. He computes the correlation between the two spreading messages with window size $l_s \times l_c$ to obtain the beginning of a seed. In RAS-DSSS, since the length of the spreading message is fixed, the computation overhead mainly depends on obtaining the position of a random seed. If a seed is positioned at the beginning of the spread message, the computation overhead will be $O(1)$, which is the best case. In the worst case, the computation overhead will be $O(n)$, because a receiver performs the comparison for $n$ time until he finds the low correlation that indicates the position of a seed. Thus the time required by the receiver to obtain the random seed in the received message is:

$$T_{seed} = ((POS_1 - 1) \times l_c) + (l_s \times l_c \times i)$$

In general, RAS-DSSS is more secure compared with DSD-DSSS, and it reduces the computation time to a lower level. This is because the seed is always inserted within the spreading
message. If the seed is inserted in $POS_1$, then $(POS_1 - 1) < l_m$ which reduces the computation time to obtain the position of the seed by $(l_m - POS_1)$.

![Diagram](begining of the random seed)

Figure 4.4: Security enhancement.

4.2.8 Security Enhancement

As discussed earlier, because RAS-DSSS randomizes the position of a random seed, the jammer needs to first obtain the location of the seed before he can launch targeted jamming against the seed. However, the attacker may spread random bits and transmit them to the receiver to confuse the despreading. Consequently, a receiver may not recover a random seed correctly. To mitigate such threat, we propose to spread the beginning of a random seed using long spreading codes. The sender can choose the length of the spreading codes based on the observations on the received messages. For example, if the receiver identifies multiples received spreading messages have been jammed, the sender spreads the first 4-bit of the random seed using long spreading codes. This will increase the chance to identify the position of the random seed within the spreading message. If $C_{long}$ represents the set of the long spreading codes compared with those $C_e$. For example, the length of the spreading codes $C_{long}$ is 2048-chip compared with 1024-chip in $C_e$. Figure 4.4 gives an example of spreading a random seed using long spreading codes. If $C_{x1}$, $C_{x2}$, $C_{x3}$, and $C_{x4}$ represent these long spreading codes form $C_{long}$ to spread the first 4-bit of the random seed. This will mitigate the effects of a jammer who sends continuously random spreading codes from $C_e$. Also this provides an efficient way against DoS attacks.
4.2.9 Advantages for a Receiver over a Jammer

In RAS-DSSS, a receiver has some advantages over a jammer to mitigate the effects of the jammer. These advantages are summarized below:

1. The receiver can fully buffer the received messages and take his time to localize and despread a random seed. While a jammer has to perform real time analysis to know the position of the seed and the spreading codes used to spread a seed for jamming the seed. Moreover, the jammer has to search over all spreading codes to jam the seed within the transmission time of one message. Assume that the length of an original message is of 1024 bits, the length of a random seed is of 16 bits, and the spreading code is of 128 bits. The transmission time of the spreading message at a 11-Mbps data rate is 
\[ \frac{(1024+16) \times 128}{11 \times 10^6} = 0.0121 \text{ ms} \]. This means that a jammer must identify the spreading codes within 0.0121 ms, whereas the receiver does not have the time limitation.

2. A reactive jammer does not have enough information regarding the position of a random seed. As a result, a jammer can not target the part of the spreading message that contains a seed. Instead, he may guess the position of a seed only. The probability that a reactive jammer can successfully guess the position of a random seed is low. Assume \( l_m = 1024 - \text{bit} \), and a random seed is going to be inserted within \( M \). Then the probability that a reactive jammer can guess the correct position of a random seed is \( \frac{1}{1024} \). On the contrary, as we explain in the previous point, the receiver can buffer both messages and take his time to obtain and despread a random seed.

3. A reactive jammer can not effectively jam a bit of a seed or the message unless he has the prior knowledge about the spreading code used for this bit. For example, if the \( i \)-th bit of the random seed is spread using spreading code \( c_{10} \), then a jammer can not
effectively jam this bit unless he knows which spreading code is used to spread this bit. It is infeasible for a reactive jammer to guess or know which spreading code used to spread a particular bit of a random seed or message within a transmission time. Again, the receiver does not suffer from this time constraint issue.

4.2.10 How to Confuse a Reactive Jammer?

As we explained earlier, a jammer always targets the most important part of the spreading message which is a seed. If a jammer can successfully jam a seed, a receiver can not recover an original message. RSA-DSSS confuses the jammer from identifying the position of the seed to reduce the probability that a seed is jammed in the following ways:

1. We use a large size of $C_p$ and $C_e$ to discourage a reactive jammer. Specifically, in RAS-DSSS, both a sender and receiver use publicly known $C_p$ and $C_e$ sets to spread both an original message and a seed. By adopting large sets, a jammer needs to perform a huge amount of computations. This means the number of the correlations required by a jammer is $(l_m \times C_p) + (l_s \times C_e)$. For example, if the size of $C_p$ and $C_e$ is 1000, $l_m$ is 128 bits, and $l_s$ is 16-bit bits. The number of correlation needs to be calculated by the attacker within the transmission time of one message is $(128 \times 1000) + (16 \times 1000) = 144,000$.

2. In RAS-DSSS, a sender inserts a seed in a random position. This position is only known to a sender. For example, in the first transmission of the spreading message, we denote ($POS_1$) as a random position chosen by a sender to hide a seed from a jammer. In the second transmission of the same spreading message, a sender spreads a seed using different spreading codes that do not overlap with those chosen for spreading the first transmission. This will result in a low correlation between the spreading seed in both transmissions.
Figure 4.5: The correlation between $SPM_1$ and $SPM_2$, and CDF of the high and low correlation of the received spreading messages in the presence of three types of attackers 1, 2 and 3.
4.3 Simulation Results

4.3.1 Analysis

RAS-DSSS provides a countermeasure against a jammer who targets either the body of the spreading message or a random seed to prevent a receiver from recovering an original message \( M \). We assume the following types of attacks:

- In attack type 1, a jammer injects random noise or random bits to disrupt the wireless communication between the sender and the receiver. If the jammer can successfully jam the seed, then the receiver cannot recover a random seed and regenerate the sequence of spreading codes used to spread the original message \( M \).

- In attack type 2, the jammer keeps sending random spreading codes from \( C_e \) to the receiver. As a result, the receiver needs to de-spread a seed using all codes in \( C_e \). This means that the receiver has to try all possible combinations to recover the original message. This attack is thus a Denial-of-service (DoS) attack.

- In attack type 3, the jammer knows the spreading codes used by the sender to spread an original message. In this type of the attack, the jammer spreads random bits using the same spreading codes used by the sender, and sends them to the receiver.

We consider the following jamming strategies:

1. Non-reactive jamming:

   Non-reactive jammers have no knowledge about the legitimate channel status. They need to guess what spreading codes are used by the sender to jam the wireless channel. Jamming interval \( I \) (where \( I > 0 \)) represents the period of the time between the pulses of jamming produced by a jammer, and it also defines the behavior of a jammer. If
the attacker decides to jam the wireless channel for a long time, $I$ should be a small value [92].

![Diagram showing POS shifts](image)

**Figure 4.6**: Different situations show the effects when the seed ($s_x$) in the second spreading ($SPM_2$) message is not shifted, or shifted by 8, 16, 24 bits on the decision to identify the position of the seed in the first spreading message ($SPM_1$).
2. Reactive jamming:

Sensing channel is essential for a reactive jammer. If the channel is idle, a jammer will be quiet. If a jammer detects wireless communication between the sender and the receiver, he starts emitting a noise signal. We point out that the reactive jammer has a time constraint compared with the receiver. As mentioned earlier, the jammer must de-spread a message within the transmission time of the message, whereas a receiver can record the message and take his time to de-spread it. Due to the time constraint, it is difficult for a reactive jammer to launch the real-time analysis to compute which code sequences are used by a sender to spread a random seed, because the jammer needs to obtain the correct spreading codes by searching on all possible code sequences within a very short time.

4.3.2 Evaluation

We validate the performance of RSA-DSSS using simulations on Matlab. In our simulation, we adopt the Additive White Gaussian Noise (AWGN) simulation channel. A sender sends the spreading messages through a AWGN channel, the simulation of which is performed by adding white gaussian noise to the transmitted signal. A jammer emits interference signals to prevent the receiver from recovering an original message. In our experiment, we use Hadamard function to generate spreading codes for $C_p$ and $C_e$ to spread the first message $M$ and a random seed $s_i$, respectively. We set the length of an original message $M$ to be 128 bits, the length of a random seed $s_i$ to be 16 bits, and the length of each spreading code to be 512 bits. Also, in our simulation, we assumed that the transmission is in the presence of the three aforementioned attackers, namely, types 1, 2, and 3 attackers.

As mentioned previously, a sender positions a seed at a random location within a spreading message, and uses two transmissions to send the spreading messages to a receiver. A receiver takes advantages of buffering the received messages to obtain and recover a seed.
As a result, in our simulation, we built a comparison function that computes, using a sliding window approach, the correlation between the received messages. The result of this comparison reveals the position of the seed. For example, if the outcome of this comparison is higher than a threshold, we consider that no seed is identified within the window size.

Figure 4.5 represents the cumulative distribution function (CDF) and the result of the correlation between two received spreading messages $SPM_1$ and $SPM_2$ in the presence of jammers type 1, 2 and 3. CDF gives us the cumulative of the probability associated with the computed correlation between $SPM_1$ and $SPM_2$, and also helps us to decide the appropriate threshold that indicates the correct position of a random seed within $SPM_1$. From figures 4.5 (a), (b), and (c), the dashed line represents the low correlation between $SPM_1$ and $SPM_2$, where a random seed is localized by the receiver, and the solid line represents the high correlation. From figure 4.5 (a), for example, 94% low correlations are smaller than or equal to 0.55, but only 0.1% high correlations are smaller than or equal to 0.55. This means that if we use 0.55 as the threshold, we will obtain the detection rate of a seed at the probability of 0.94 and this threshold also gives a false alarm rate of 0.1%.

Figure 4.6 shows four situations, when we calculate the correlation between the two spreading messages to identify the location of the seed. In each situation, we shift the position of the seed by multiplying the number of shifted bits by $l_c$. In the first situation, the random seed is at the same position for both messages, and we can see that the low correlation results in a small segment only $\approx 0.56$. As we explained previously, the sender spreads a random seed in the second spreading message using different spreading codes from the one used in the first transmission, and the result of the spreading is $S_x(C_e, s)$. Thus, low correlation results from the situation where both $S(C_e, s)$ and $S_x(C_e, s)$ are in the same position in the spreading message. As we can see, the blue line represents the computed correlation between $SPM_1$ and $SPM_2$ when the seed is inserted within $SPM_1$ and $SPM_2$ at the same position (within bit 41). When there is no shifting, the computed high correlation
remains stable at 0.62 before reaching bit number 41 where the seed is inserted. After that, there is a noticeable drop in the blue line from 0.62 to 0.48 ~ 0.52 which is resulted from the reduction of the computed correlation. This almost remains until bit number 57 where the seed end. From this, we can observe that when the random seed is inserted in the same position within $SPM_1$ and $SPM_2$, the low correlation points exactly to the position of the random seed. Also, the low correlation boundary is the length of the random seed.

On the other hand, when the seed is inserted at different positions (i.e., $POS_1$ and $POS_2$), the result of the low correlation varies based on the position shift. This results in increasing the window size where the result of the correlation is low. This indicates that the position of the seed is shifted in the second spreading message. As a result, the cost of the search on the receiver will increase as by the amount of the shift.

![Figure 4.7: Number of success trials to find the position of the seed.](image)

We run the simulation program for 1000 times with different lengths of the spreading codes to discover the impact of attacker types 1, 2, and 3 on a receiver to identify the position of a seed. Figure 4.7 shows the number of successful trials to find the location of a random seed. For instance, when the length of a spreading code is 256, the receiver can successfully identify a random seed with detection rates 0.643, 0.672, and 0.031 in the
presence of attack type 1, attack type 2, and attack type 3, respectively. On the other hand, when the length of the spreading code is 1024, the receiver can successfully identify the position of a random seed with detection rates 0.860, 0.891, and 0.11 in the presence of these attackers. We observe that increasing the length of the spreading codes can significantly improve the detection rate.

Figure 4.8: Probability for a non-reactive jammer to jam the seed.

Figure 4.9: Probability for a reactive jammer to jam the seed.
Also, similar to ([23], Theorem 2 and 3), we compute the probabilities for both a non-reactive and reactive jammers to jam a random seed during the transmission time when $C_e$ and $C_p$ have different sizes of spreading codes using equations 4.1 and 4.2, respectively.

\[
P_{nrj} = 1 - (1 - \left( \frac{1}{l_c \times (C_p + C_e)^l_m} \right)) \tag{4.1}
\]

\[
P_{rj} = 1 - (1 - \frac{1}{C_p})^l_m \times (1 - \frac{1}{C_e})^l_s \tag{4.2}
\]

Intuitively, increasing the size of $C_e$ and $C_p$ can reduce the probability that the seed is jammed. Figure 4.8 shows the probability for a non-reactive jammer to jam a random seed. As we can see from this figure, $p=0.015$ when the size of $C_e$ and $C_p$ is 1000, and $p$ significantly reduces to 0.001 when the size $C_e$ and $C_p$ is increased to 7000. On the other hand, figure 4.9 represents the probability for a reactive jammer to jam a random seed. As we can see from the figure, $p=0.019$ when the size of $C_e$ and $C_p$ is 7000. The size of $C_e$ and $C_p$ play an important role in reducing the probability that the jammer can jam the seed. The reason for this is that it is hard for the jammer to exhaustively search over all possible spreading codes in $C_e$ and $C_p$ when their sizes are big. Furthermore, it is impossible for the jammer to do real-time analysis within a transmission time.
Chapter 5: Future Work

I discuss three directions for my future work. First, I introduce new techniques that would increase the robustness and effectiveness of my current works in chapters 3 and 4. In the second direction for my future studies, I argue how we can use a friendly jammer to mitigate the effects of a relay attack. Lastly, I describe my plan on investigating vulnerabilities associated with the internet of things.

5.1 Future Possible Approaches to My Current Works

There are a couple of approaches that I plan to develop and enhance the performance and security concerns in both systems in Chapter 3 and 4.

5.1.1 Fragmentation the Seed into Equal Chunks within a Spread Message

Figure 5.1: Enhanced RAS-DSSS.
In chapter 3, to improve the security of RAS-DSSS, I plan to break the seed into equal chunks, i.e., two fragments that can be inserted randomly within the spread message by applying the same steps of RAS-DSSS. In each transmission, the two chunks of the seed will be positioned in a random different position. To recover the seed, we apply two steps of a sliding window approach, as we can see in 5.2. The first step is to obtain the position of the first part of the seed withing spreading message one. In the second step, the receiver obtains the position of the first part of the seed within the second spreading message. To obtain the second part of the seed, we repeat these steps. This will reduce the probability the attacker can jam the seed. However, it will increase the computation overhead since the receiver needs to identify the correct positions of these two fragments of the seed.

5.1.2 Adoption the Length of the Spreading Code

Form the evaluations result in Chapters 3 and 4, in RP-DSSS and RAS-DSSS, as the length of the spreading code increases the probability that the spreading message is jammed decreases. From this point, I plan to adapt the length of the spreading code, which uses to spread the original message, based on the capability of the jammer, which can be observed based on the received message. This can play an important role in reducing the probability that the received message is jammed, and reducing the computation overhead at the receiver side.

5.2 Friendly Jamming

Another direction for my future work is using a friendly jammer to protect the confidentiality of data during the communication, and message authentication purposes. In the presence of friendly jamming, "jamming is used for the good and strives to be minimally invasive," [93]. Recent works have been conducted in this area (e.g., [2,61,94–101]). Particularly, some of these researches use jamming to block harmful communication (e.g., [2,102–106]).
and others use jamming to improve the security of communications (e.g., [107–112]). Intentional interference considers as a valuable resource to secure wireless communications. For instance, increasing the level of the noise at the attacker side (e.g., the eavesdropper) prevents him/her from recovering the network messages attacking the communication.

![Figure 5.2: A friendly jamming against relay attacks.](image)

In my future studies, I plan to implement a system against a relay attack using a friendly jammer. For example, In [113], when the car owner steps away from his/her car, the attacker could relay the vehicles key fob to open and run a vehicle. To mitigate the effects of this kind of attack, I plan to use friendly jamming signals to be a wall between the user and the attacker. For example, when the driver locked his/her care, and walk away from the car, a simple jamming single can be used to prevent the attacker from relaying the vehicles key fob to the car end. A friendly jamming signal will closer to the owner than the car.

5.3 Security Issues in Internet of Things

Nowadays, there has been a dramatic increase in using IoT. However, it must be recognized that the security threats and issues have been rapidly involved in IoT. As a result, it is crucial to pay more attention and put appropriate considerations on these IoT security vulnerabilities and challenges. For example, Denial of service attacks can affect all legitimate
users and nodes from accessing all available resources of the network. Also, this type of attacker may destroy configuration information. For this, to protect the IoT form this type of attacker, I will focus and put much effort into developing techniques to monitor the system and detect any indications that the system is affected by DoS attacks. Recent studies have focused on this attack (e.g., [61, 72, 114–116]).
Chapter 6: Conclusion

This dissertation presents two works against jamming attacks on wireless communication systems.

In the first work, 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 also propose new schemes to enhance the security of RP-DSSS scheme by padding and randomizing the message transmission orders. We performed both theoretical analysis and computer simulations to validate the performance of the proposed approaches.

In the second work, we propose RAS-DSSS system to increase the security of DSD-DSSS. RAS-DSSS positions a seed at a random location of a spreading message to reduce the chance that a seed to be jammed. We also design a new mechanism that enables a receiver to locate the position of a seed using a sliding window. This can reduce the computational overhead of the proposed system compared with a DSD-DSSS system. We performed computer simulations to validate the effectiveness of RAS-DSSS, and the results show that RAS-DSSS can successfully obtain the position of a seed. Our experiment results show that our proposed
system can successfully obtain the position of the random seed, and the detection rates can be increased successfully when we increase the length of the spreading codes.
References


### Appendix A: Tables

Table A1: Detection rates in the presence of attackers type 1, 2, and 3 when the length of spreading code ($I_c$) is 256, 512, and 1024, respectively

<table>
<thead>
<tr>
<th>$I_c$</th>
<th>Attackers</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>type 1</td>
<td>type 2</td>
<td>type 3</td>
</tr>
<tr>
<td>256</td>
<td>27.54</td>
<td>13.23</td>
<td>6.80</td>
<td></td>
</tr>
<tr>
<td>512</td>
<td>29.55</td>
<td>9.61</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>1024</td>
<td>25.99</td>
<td>11.49</td>
<td>6.15</td>
<td></td>
</tr>
</tbody>
</table>
Appendix B: Copyright Permissions

- The following is permission to use the content in Chapter 1.

**Title:** Wireless Jamming Attack  
**Author:** Ahmad Alagil, Yao Liu  
**Publication:** Springer eBook  
**Publisher:** Springer Nature  
**Date:** Jan 1, 2019  
Copyright © 2019, Springer Nature Switzerland AG

**Order Completed**

Thank you for your order.

This Agreement between Year -- Alagil Alagil ("You") and Springer Nature ("Springer Nature") consists of your license details and the terms and conditions provided by Springer Nature and Copyright Clearance Center.

Your confirmation email will contain your order number for future reference.

**printable details**

License Number: 4664381407522  
License date: Sep 08, 2019  
Licensed Content Publisher: Springer Nature  
Licensed Content Publication: Springer eBook  
Licensed Content Title: Wireless Jamming Attack  
Licensed Content Author: Ahmad Alagil, Yao Liu  
Licensed Content Date: Jan 1, 2019  
Type of Use: Thesis/Dissertation  
Requestor type: academic/university or research institute  
Format: print and electronic  
Portion: full article/chapter  
Will you be translating?: no  
Circulation/distribution: 30 - 99  
Author of this Springer Nature content: yes  
Title: Student  
Institution name: University of South Florida  
Expected presentation date: Dec 2019  
Requestor Location: Year: 4202 E. Fowler Avenue  
TAMPA, FL 33620  
United States  
Attn.: Year  
Total: 0.00 USD

Copyright © 2019 Copyright Clearance Center, Inc. All Rights Reserved. Privacy statement, Terms and Conditions.
The following two permissions are for the content in Chapter 3.

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author’s approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/credit notice should be placed prominently in the references: © [Year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author’s university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity’s name goes here]’s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.
Randomized Positioning DSSS with Message Shuffling for Anti-Jamming Wireless Communications

Author: Ahmad Alagil
Publisher: IEEE
Date: Nov. 2019

Copyright © 2019, IEEE

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2019 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author’s approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis online.
3) In placing the thesis on the author’s university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity’s name goes here]’s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.
The following is permission to use the content in Chapter 4.

SPRINGER NATURE LICENSE
TERMS AND CONDITIONS

Dec 16, 2019

This Agreement between Year -- Alagil Alagil ("You") and Springer Nature ("Springer Nature") consists of your license details and the terms and conditions provided by Springer Nature and Copyright Clearance Center.

License Number: 4730911262601
License date: Dec 16, 2019
Licensed Content Publisher: Springer Nature
Licensed Content Publication: Springer eBook
Licensed Content Title: Random Allocation Seed-DSSS Broadcast Communication Against Jamming Attacks
Licensed Content Author: Ahmad Alagil, Yao Liu
Licensed Content Date: Jan 1, 2019
Type of Use: Thesis/Dissertation
Requestor type: academic/university or research institute
Format: print and electronic
Portion: full article/chapter
Will you be translating? no
Circulation/distribution: 200 - 499
Author of this Springer Nature content: yes
Title: Student
Institution name: University of South Florida
Expected presentation date: May 2020
Requestor Location: Year
4202 E. Fowler Avenue
TAMPA, FL 33620
United States
Attn: Year

Total: 0.00 USD

Terms and Conditions