ABSTRACT

LIU, YAO. Wireless Physical Layer Security. (Under the direction of Peng Ning.)

Wireless communication is ubiquitous today. Various wireless applications like WiFi, bluetooth, and cellular networks have been widely used in people's daily life. How to protect the physical layer of wireless communication is crucial for applications where reliable information exchange is required. On the other hand, characteristics of wireless physical layer create new opportunities for us to come up with novel and efficient approaches that can guarantee the security of wireless physical layer. This dissertation includes four work toward the protection of wireless physical layer. The first and second work propose solutions to combat jamming attacks, which are well known threats to wireless communications. The third work utilizes physical layer characteristics to authenticate wireless signal for cognitive radio networks (CRNs), and the lats work identifies and addresses vulnerabilities existing in the link signature authentication techniques.

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

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

The second work targets at reactive jamming attacks, which are the most effective jamming attacks against wireless communication. A reactive jammer jams the channel only when the target devices are transmitting. Therefore, it is more energy efficient and harder to detect. Frequency Hopping Spread Spectrum (FHSS) and Direct Sequence Spread Spectrum (DSSS) have been widely used as countermeasures against (reactive) jamming attacks. However, both FHSS and DSSS will fail if the jammer can jam all frequency channels or has large transmit power.

In this work, we present BitTrickle, an anti-jamming wireless communication scheme that allows

communication in the presence of a broadband and high power reactive jammer. BitTrickle transmits messages by taking advantage of the subtle opportunity that arises from the reaction time of a reactive jammer. Unlike FHSS and DSSS, BitTrickle does not assume a reactive jammer with limited spectrum coverage and transmit power, and thus can be used in scenarios where traditional approaches fail. We develop a prototype of BitTrickle using GNURadio to evaluate the performance of BitTrickle. Our results show that when a reactive jammer is turned on, BitTrickle still maintains wireless communication, whereas other schemes such as 802.11 DSSS fails to deliver packets.

The third work targets at wireless signal authentication in cognitive radio networks. Cognitive radio networks have been proposed to increase the efficiency of channel utilization and address the increasing demand for wireless bandwidth; they enable the sharing of channels among secondary (unlicensed) and primary (licensed) users on a non-interference basis. A secondary user in a CRN should constantly monitor for the presence of a primary user's signal to avoid interfering with the primary user. However, to gain unfair share of radio channels, an attacker (e.g., a selfish secondary user) may mimic a primary user's signal to evict other secondary users. Therefore, a secure primary user detection method that can distinguish a primary user's signal from an attacker's signal is needed. A unique challenge in addressing this problem is that Federal Communications Commission (FCC) prohibits any modification to primary users. Consequently, existing cryptographic techniques cannot be used directly.

In this work, we develop a novel approach for authenticating primary users' signals in CRNs, which conforms to FCC's requirement. The proposed approach integrates cryptographic signatures and wireless link signatures (derived from physical radio channel characteristics) to enable primary user detection in the presence of attackers. Essential to the proposed approach is a *helper node* placed physically close to a primary user. The helper node serves as a "bridge" to enable a secondary user to verify cryptographic signatures carried by the helper node's signals and then obtain the helper node's authentic link signatures to verify the primary user's signals. A key contribution is a novel physical layer authentication technique that enables the helper node to authenticate signals from its associated primary user. Unlike previous techniques for link signatures, the proposed approach explores the geographical proximity of the helper node to the primary user, and thus does not require any training process.

The last work targets at vulnerability analysis of wireless link signature authentication, which is a physical layer authentication mechanism that uses the unique wireless channel characteristics between a transmitter and a receiver to provide authentication of wireless channels. We identified a vulnerability of existing link signature schemes by introducing a new attack, called *mimicry attack*. It was assumed that "an attacker cannot 'spoof' an arbitrary link signature" and that the attacker "will not have the same link signature at the receiver unless it is at exactly the same location as the legitimate transmitter". However, this work shows that an attacker *can* forge an *arbitrary* link signature as long as it roughly knows or can estimate the legitimate signal at the receiver's location. To defend against the mimicry attack, we proposed a novel construction for wireless link signature, called *time-synched link signature*, by integrating cryptographic protection and time factor into traditional link signatures.

Wireless Physical Layer Security

by
Yao Liu

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Computer Science

Raleigh, North Carolina

2012

APPROVED BY:

Douglas S. Reeves	Huaiyu Dai
Xuxian Jiang	Peng Ning Chair of Advisory Committee

DEDICATION

To my parents and family.

BIOGRAPHY

Yao Liu earned her B.S. degree in Computer Science and M.S. degree in Communication and Information System from Xidian University in 2004 and 2007, respectively. In August 2007, she started her Ph.D. study at North Carolina State University in the Department of Computer Science. Her research focuses on designing and implementing innovative defense approaches that can effectively prevent wireless networks from being undermined by malicious attackers.

ACKNOWLEDGEMENTS

I would like to thank all those who have supported and encouraged me during my Ph.D. study. First, I would like to thank my advisor, Dr. Peng Ning. I have benefited a lot from his guidance and suggestions of my work. I would like to thank him for his patience, encouragement, and financial support. I would like to thank my committee members, Dr. Douglas S. Reeves, Dr. Huaiyu Dai, and Dr. Xuxian Jiang, for their valuable feedback and comments on my research. Second, I would like to thank National Science Foundation (NSF) for its funding support. Third, I would like to give my thanks to my friends for their help during my Ph.D. study: Ahmed Moneeb Azab, Juan Du, Wenbo Shen, Ruowen Wang, Xianqing Yu, Barry Peddycord III, Wu Zhou, Sangwon Hyun, Chongkyung Kil, Mihui Kim, An Liu, Emre Can Sezer, Attila Altay Yavuz, Qinghua Zhang, Yi Zhang, and Jason Gionta. Finally, I would like to give my special thanks to my families for their encouragement and support during my Ph.D. study.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

Wireless communication is ubiquitous today. Various wireless applications like WiFi, bluetooth, and cellular networks have been widely used in people's daily life. There exist unique security needs in wireless physical layer. For example, wireless medium is exposed to the public, which makes it easy for an attacker to eavesdrop the communication or block the communication by transmitting jamming signals. As another example, an attacker or a selfish user may inject fake signals into the wireless channel to undermine the normal operations of legitimate users. How to protect the physical layer of wireless communication is crucial for applications where reliable information exchange is required. On the other hand, characteristics of wireless physical layer create new opportunities for us to come up with novel and efficient approaches that can guarantee the security of wireless physical layer. Prior work have demonstrated successes in utilizing physical layer properties like radio channel characteristic and modulation errors as fingerprints to authenticate wireless devices [12, 65, 112].

This dissertation includes my four work toward the protection of wireless physical layer. The first and second work propose solutions to combat jamming attacks, which are well known threats to wireless communications. The third work utilize physical layer characteristics to authenticate wireless signal for cognitive radio networks (CRNs), and the last work identifies and addresses vulnerabilities existing in the link signature authentication techniques. Details of those work will be shown in Chapters 2, 3, 4 and 5. In the following, I give the motivations of the four work respectively.

1.1 Randomized Differential Direct Sequence Spread Spectrum

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

Spread spectrum techniques have been used as countermeasures against jamming attacks. Direct Sequence Spread Spectrum (DSSS), Frequency Hopping Spread Spectrum (FHSS), and Chirp Spread Spectrum (CSS) are three common forms of spread spectrum techniques [68]. In classic spread spectrum techniques, senders and receivers need to pre-share a secret key, with which they can generate identical hopping patterns, spreading codes, or timing of pulses for communication. However, if a jammer knows the secret key, the jammer can easily jam the communication by following the hopping patterns, spreading codes, or timing of pulses used by the sender.

There have been a few recent attempts to remove the dependency of jamming-resistant communications on pre-shared keys [8, 69, 87, 92, 93]. Strasser et al. developed an Uncoordinated Frequency Hopping (UFH) technique to allow two nodes that do not have any common secret to establish a secret key for future FHSS communication in presence of a jammer [92]. Strasser et al. [93] and Slater et al. [87] later independently proposed to use similar coding techniques to improve the robustness and efficiency in UFH. These works successfully remove the requirement of pre-shared keys in point-to-point FHSS communication.

Unfortunately, UFH and its variations [87, 92, 93] cannot be directly used for *broadcast* communication, since their primary objective is to establish a pairwise key between two parties. Indeed, any spread spectrum communication system that requires a shared key, either pre-shared or established at the initial stage of the communication, cannot be used for broadcast communication where there may be insider jammers. Any malicious receiver, who knows the shared key, may use the key to jam the communication.

To address this problem, researchers recently investigated how to enable jamming-resistant broadcast communication without shared keys [8, 69]. Baird et al. proposed a coding approach to encode data to be transmitted into "marks" (e.g., short pulses at different times) that can be decoded without any prior knowledge of keys [8]. However, the decoding process of the method is inherently sequential (i.e., the decoding of the next bit depends on the decoded values of the previous bits). Though it works with short pulses in the time domain, the method cannot be extended to DSSS or FHSS without significantly increasing the decoding cost.

Pöpper et al. developed an Uncoordinated Direct Sequence Spread Spectrum (UDSSS) approach, which avoids jamming by randomly selecting a spreading code sequence from a pool of code sequences. However, as indicated in [69], UDSSS is vulnerable to reactive jamming attacks. It is demonstrated in [69] that when the jammer does not have sufficient computational power to infer the spreading sequence quickly enough, UDSSS still provides good enough jamming resistance. However, when the jammer has sufficient computational power, UDSSS fails to provide strong guarantee of jamming resistance.

In this dissertation, we propose a Randomized Differential DSSS (RD-DSSS) scheme for DSSS-based broadcast communication. RD-DSSS relies completely on publicly known spreading codes, and thus does not require any shared key among the sender and the receivers. It does not suffer from the

vulnerabilities of previous solutions, and thus is a good candidate to enable anti-jamming broadcast communication even when there are potentially compromised or malicious receivers.

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

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

1.2 BitTrickle: Wireless Communication under Broadband and High Power Reactive Jamming

Reactive jamming is one of the most effective jamming attacks [5]. A reactive jammer remains quiet when the target sender is not transmitting, but jams the channel when it detects transmission from the target device. Compared with constant jamming, reactive jamming is harder to track and much more energy efficient [108]. Reactive jamming has been widely used in military applications to cut off the wireless communication among opponent armies or disable the opponent's use of radio-controlled devices [5].

Current countermeasures against (reactive) jamming attacks mainly rely on spread spectrum techniques, which can be categorized as Frequency Hopping Spread Spectrum (FHSS) (e.g., [39, 80, 92, 93, 97]), Direct Sequence Spread Spectrum (DSSS) (e.g., [39, 56, 69, 80, 97]), Time Hopping Spread Spectrum (THSS) (e.g., [79, 106]), and Chirp Spread Spectrum (CSS) (e.g., [10, 43, 90]). Among all those spread spectrum techniques, FHSS and DSSS are dominantly used for the purpose of anti-jamming, whereas THSS and CSS require specific hardware (e.g., a pulse/chirp signal generator) [79, 101], and thus are less popular than FHSS and DSSS.

In FHSS, the sender and the receiver switch a frequency channel among a pool of candidate channels from time to time. As long as the jammer and the communicators (i.e., the sender and the receiver) are not operating on the same channel, the jammer cannot jam ongoing communication. In DSSS, the sender multiplies the original data with a pseudo-random sequence to obtain spreading gain. If the jammer's

power is not strong enough to overwhelm the DSSS signals with spreading gain, the receiver can use the same pseudo-random sequence to recover the original message.

However, both FHSS and DSSS will fail to protect the communication when attacked by a broadband jammer who is capable of jamming all frequency channels or a high power jammer who beats DSSS spreading gain. In our work, we endeavor to find a way to enable wireless communication in the presence of such jammers. Ideally, we would like to achieve this anti-jamming purpose without adding specific demands on hardware.

In this dissertation, we develop BitTrickle, a novel communication scheme that allows wireless devices to exchange information when attacked by broadband and high power reactive jammers. BitTrickle requires no special hardware. Even wireless devices that are not equipped with spread spectrum capability can employ BitTrickle as a countermeasure against reactive jamming attacks.

BitTrickle achieves the anti-jamming capability by harnessing a subtle opportunity arising from a basic feature of reactive jamming, i.e., "the jammer stays quiet when the channel is idle, but starts transmitting a radio signal as soon as it senses activity on the channel" [111].

Channel sensing is an indispensable function for a reactive jammer to determine if a target sender is transmitting. Channel sensing causes a short time delay. For example, energy detection, the most popular channel sensing approach with very small sensing time [46], requires more than 1 millisecond to detect the existences of target signals for a 0.6 detection probability and -110dBm signal strength level, when implemented in a fully parallel pipelined FPGA (field programmable gate array) architecture for fast speed [15]. Therefore, before the jammer detects the sender's transmission and starts jamming, the sender has already transmitted one or several bits. As shown in Figure 1.1, though the packet transmitted by the sender is corrupted and cannot be fully recovered, the first few bits of this packet are unjammed due to channel sensing delay Δt at the jammer.

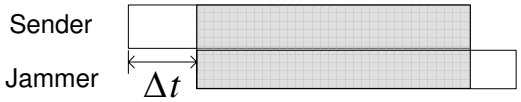

Figure 1.1: Reactive jamming: The first ΔtR bits of the sender's packet are jamming-free, where Δt and R are channel sensing delay of the jammer and the transmission bit rate of the sender, respectively.

BitTrickle takes advantage of unjammmed bits in corrupted packets to establish jamming-resilient communications. In BitTrickle, the receiver collects bits that are transmitted by the sender but not jammed by the reactive jammer, and assembles them together to construct the original message.

Two technical challenges are addressed during the development of BitTrickle. First, the receiver should be able to extract unjammed bits from received bit stream. We developed a jamming detector

that utilizes modulation properties to identify unjammed bits. In addition, an error recovery mechanism is required to tolerate synchronization errors (e.g., lost bits), and guarantee the performance of traditional error correction codes (ECC). We proposed BitTrickle encoding/decoding techniques, which can locate the original position for each received bit in the original message and enable the use of ECC with high efficiency.

1.3 Authenticating Primary Users' Signals in CRNs

The proliferation of emerging wireless applications requires a better utilization of radio channels [17]. To address the increasing demand for wireless bandwidth, cognitive radio networks (CRNs) have been proposed to increase the efficiency of channel utilization under the current static channel allocation policy [46]. They enable unlicensed users to use licensed channels on a non-interference basis, thus serve as a solution to the current low usage of radio channels [25]. For example, IEEE 802.22 Standard on Wireless Regional Area Networks (WRANs) employs cognitive radio to allow the sharing of geographically unused channels allocated to television broadcast services, and therefore bring broadband access to hard-to-reach low-population-density areas (e.g., rural environments) [27].

In CRNs, there are two types of users: *primary users* and *secondary users* [46]. Primary users are licensed users who are assigned with certain channels, and secondary users are unlicensed users who are allowed to use the channels assigned to a primary user only when they do not cause any harmful interference to the primary user [46]. For example, in IEEE 802.22 WRANs, TV transmission towers are primary users, and radio devices that use TV channels for communication are secondary users.

An essential issue in CRNs is *primary user detection*, in which a secondary user monitors for the presence of a primary user's signal on target channels [17]. If a primary user's signal is detected, the secondary user should not use those channels to avoid interfering with the transmission of the primary user.

Existing methods for primary user detection can be categorized as *energy detection* and *feature detection* [46]. In energy detection methods (e.g., [85]), any captured signal whose energy exceeds a threshold is identified as a primary user's signal. In feature detection methods (e.g., [38,70,78,84,107]), secondary users attempt to find a specific feature of a captured signal, such as a pilot, a synchronization word, and cyclostationarity. If a feature is detected, then the captured signal is identified as a primary user's signal.

Due to the open nature of wireless communications and the increasingly available software defined radio platforms (e.g., Universal Software Radio Peripherals (USRPs) [34]), it is necessary to consider potential threats to normal operations of CRNs. Indeed, CRNs do face several threats. In particular, an attacker may transmit with high power or mimic specific features of a primary user's signal (e.g., use the same pilots or synchronization words) to bypass the existing primary user detection methods [17]. Consequently, secondary users may incorrectly identify the attacker's signal as a primary user's signal

and do not use relevant channels. Such attacks are called *primary user emulation (PUE) attacks* [17].

It is necessary to have a secure primary user detection method that can identify a primary user's signal in the presence of attackers. At first glance, a cryptographic signature seems to be a good candidate for this task. Unfortunately, CRNs face a unique constraint that prevents it from being employed. Specifically, Federal Communications Commission (FCC) states that "no modification to the incumbent system (i.e., primary user) should be required to accommodate opportunistic use of the spectrum by secondary users" [23]. As a result, any solution that requires changes to primary users, such as enhancing primary users' signals with cryptographic signatures, is not desirable.

There has been a recent attempt that uses a location distinction approach to distinguish between a primary user's signal and an attacker's signal [17]. Specifically, this approach uses received signal strength (RSS) measurements to estimate the location of the source of a signal, and then determines if the signal is from the (static) primary user based on the known location of the primary user [17]. However, as indicated in [65], RSS based location distinction can be easily disrupted if an attacker uses array antennas to send different signal strengths in different directions simultaneously. Moreover, it requires multi-node collaboration, which is expensive in terms of bandwidth and energy.

Link signatures (i.e., radio channel characteristics such as channel impulse responses) have been developed recently to obtain more secure and robust location distinction [65, 112]. Unfortunately, it remains non-trivial to exploit link signature based location distinction approach for primary user detection in the presence of attackers. In particular, a receiver needs to know a transmitter's historical link signatures in order to verify if a newly received signal is from the transmitter. In CRNs, however, it is impossible for a secondary user to know a primary user's historical link signatures, unless the secondary user can first authenticate whether a signal is from the primary user or not.

In this dissertation, we develop a novel approach that integrates traditional cryptographic signatures and link signatures to enable primary user detection in the presence of attackers. The proposed approach does not require any change to primary users, and thus follows the FCC constraint properly.

A key component of the approach is a *helper node* placed close (and physically bound) to a primary user. Though we cannot modify any primary user due to the FCC constraint, we can put necessary mechanisms on each helper node, including the use of cryptographic signatures. Moreover, since the helper node is placed very close to the primary user, their link signatures observed by a secondary user are very similar to each other. The helper node thus serves as a "bridge" that enables a secondary user to first verify the cryptographic signatures included in the helper node's signals, then learn the helper node's authentic link signatures, and finally verify the primary user's link signatures. In other words, the proposed approach properly integrates cryptographic signatures and wireless link signatures to enable primary user detection in CRNs in the presence of attackers.

1.4 Mimicry Attacks against Wireless Link Signature and Defense using Time-Synched Link Signature

Wireless physical layer security is becoming increasingly important as wireless devices are more and more pervasive and adopted in critical applications. For example, implantable medical devices (IMD) such as pacemaker may grant access to an external control device only when it is close enough [72], thus making it critical to verify the physical proximity of the control device. There have been multiple proposals recently to provide enhanced wireless security using physical layer characteristics, including fingerprinting wireless devices (e.g., [13]), authenticating and identifying wireless channels (e.g., [65, 112]), and deriving secret keys from wireless channel features only observable to the communicating parties (e.g., [58]).

Among the recent advances in wireless physical layer security is (wireless) link signature. Link signature uses the unique wireless channel characteristics (e.g., the multi-path effect) between a transmitter and a receiver to provide authentication of the wireless channel. Three link signature schemes [53, 65, 112] have been proposed so far. Since its introduction, link signature has been recognized as a wireless channel authentication mechanism for applications where wireless channel characteristics are unique (e.g., [13, 58]).

In this dissertation, we identify a vulnerability of existing link signature schemes [53, 65, 112] by introducing a new attack called *mimicry attack*. We start our investigation with the link signature scheme in [65], which claimed that an attacker "cannot 'spoof' an arbitrary link signature" and that the attacker "will not have the same link signature at the receiver unless it is at exactly the same location as the legitimate transmitter." However, we show in this dissertation that (1) an attacker *can* forge an *arbitrary* link signature as long as the attacker can roughly estimate the legitimate signal at the receiver's location, and (2) the attacker does not have to be at exactly the same location as the legitimate transmitter in order to forge its link signature. We also extend the mimicry attack to the link signature scheme in [53]. Since the link signature scheme in [112] is essentially an integration of the techniques in [65] and [53], all three link signature schemes are vulnerable to the mimicry attack.

To defend against the threat identified in this dissertation, we develop a new link signature scheme, which is called time-synched (i.e., time synchronized) link signature. Time-synched link signature integrates cryptographic protection as well as time factor into the wireless physical layer features, and provides an effective countermeasure against mimicry attacks. We also perform an extensive set of experimental evaluation of the mimicry attacks and the time-synched link signature scheme on the USRP2 platform [57] running GNURadio [1]. Our experiments confirm that the mimicry attacks against the previous link signature schemes are a real threat and demonstrate that the newly proposed time-synched link signatures are effective in mitigating those attacks.

1.5 Summary of Contributions

The contributions of this dissertation are summarized below:

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

- BitTrickle: We address the problem of anti-jamming wireless communication under high power broadband reactive jamming, where the previous approaches all fail. We develop the BitTrickle anti-jamming communication scheme by taking advantage of the reaction time of the jammer, thus allowing wireless communication even when all previous anti-jamming techniques fail. We develop solutions to two technical challenges in the development of BitTrickle. First, we develop a high quality jamming detector that utilizes modulation properties to precisely distinguish unjammed bits from jammed bits. Second, we develop a novel encoding/decoding technique that can successfully recover a transmitted message from message fragments that partially survive reactive jamming. We implement a working prototype of BitTrickle based on GNURadio [1] and evaluate it on Universal Software Radio Peripherals (USRPs) [57]. We also compare it with 802.11 DSSS and a benchmark program, both provided in GNURadio, in terms of packet delivery ratio and throughout. Our experimental results show that the BitTrickle prototype achieves a reasonable throughput that enables message exchange between victim wireless devices when 802.11 DSSS and the GNURadio benchmark is completely disabled by a reactive jammer.

- Authentication for CRNs: We develop a new primary user detection method that integrates cryptographic signatures with wireless link signatures to distinguish a primary user's signal from an attacker's signal. The proposed method conforms to the FCC's requirement of not modifying primary users. Unlike the previous approach [17], the method does not require the deployment of a monitoring network, and thus avoids the weakness of the previous approach. We develop a novel physical-layer authentication technique that enables a helper node to authenticate signals from its associated primary user. Unlike previous proposals for link signatures, the approach explores the geographical proximity of the helper node to the primary user rather than historical link signatures. A key consequence is that the method does not require any training process. We evaluate the effectiveness of our method through both theoretical analysis and experiments using real-world link signatures obtained from the CRAWDAD data set [64]. Moreover, we demonstrate the feasibility of our proposed method by a prototype implementation on a software-defined radio platform [34].

- Mimicry attacks: First, we identify the vulnerability of existing link signature schemes to mimicry attacks. Second, we develop the time-synched link signature scheme to defend against mimicry attacks and achieve enhanced capability for wireless channel authentication. Finally, we perform an extensive set of experiments to evaluate the impact of mimicry attacks and demonstrate the effectiveness of the proposed time-synched link signature.

Chapter 2

Randomized Differential Direct Sequence Spread Spectrum

2.1 Background on DSSS

DSSS is a modulation method applied to digital signals [40]. It increases the signal bandwidth to a value much larger than needed to transmit the underlying information [40]. In DSSS, spreading codes that are independent of the original signal are used to achieve the goal of bandwidth expansion. Both a sender and a receiver agree on a spreading code, which is regarded as a shared secret between them. A spreading code is usually a sequence of bits valued 1 and -1 (polar) or 1 and 0 (non-polar), which has noise-like properties. Without loss of generality, we consider spreading codes with polarity. Typical spreading codes are pseudo-random codes, Walsh-Hadamard codes and Gold codes [86]. Figure 3.16 shows a simple communication framework of DSSS.

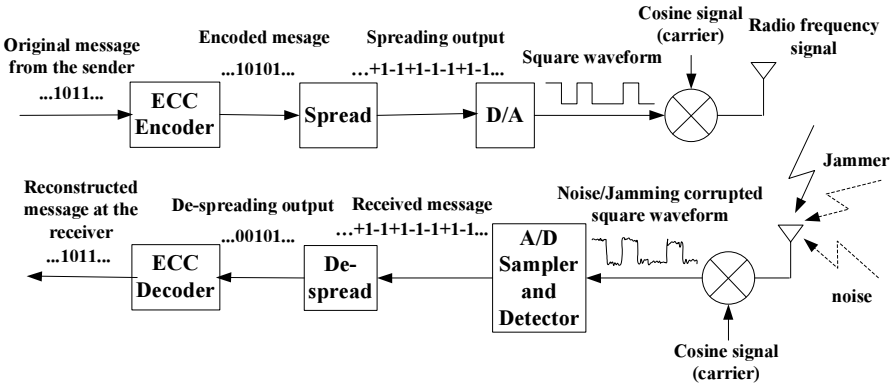

Figure 2.1: A simple communication framework of DSSS

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

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

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

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

In de-spreading, the receiver uses a local replica of the spreading code and synchronizes it with the received message [86]. Then the receiver correlates the received message with the replica to generate the de-spreading output. For example, suppose the received message is $+1-1-1+1-1+1+1-1$ and the local replica of the spreading code is $-1+1+1-1$ at the receiver side. The receiver aligns $-1+1+1-1$ with the first 4 chips of the received message (i.e., $+1-1-1+1$) and correlates them to get bit -1 (i.e., "0" in non-polar form).

Synchronization: In DSSS systems, a receiver needs to identify the beginning of a message sent by the sender from the received signal. In general, the sender and the receiver agree on a known code such as Barker Code [9] that has good autocorrelation property (i.e., the correlation between a code and its shifted value is low). The sender transmits the code just before the spread message. The receiver correlates the received signal with the code using a sliding window approach; the position where the correlation is maximum indicates the beginning of the message [21, 41, 86].

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

2.2 System and Threat Models

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

2.3 Randomized Differential DSSS

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

2.3.1 Basic Scheme

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

Figure 2.2 shows an example, in which a sender transmits a 4-bit message "1011" to a receiver. The sender randomly chooses codes $\mathbf{p}_1$, $\mathbf{p}_4$, $\mathbf{p}_5$, and $\mathbf{p}_7$ from the spreading code set. Since bit 1 of the message is "1", the sender uses $\mathbf{p}_1$ twice to encode it. The second bit of the original message is "0". Thus, the sender uses any code different from $\mathbf{p}_4$ (i.e., $\mathbf{p}_3$ as shown in Figure 2.2) and $\mathbf{p}_4$ to encode it. Bits 3 and 4 are encoded similarly. As a result, the sender gets an encoded message $\underline{\mathbf{p}_1||\mathbf{p}_3||\mathbf{p}_5||\mathbf{p}_7}||\underline{\mathbf{p}_1||\mathbf{p}_4||\mathbf{p}_5||\mathbf{p}_7}$, in which the second half of the message are the codes selected by the

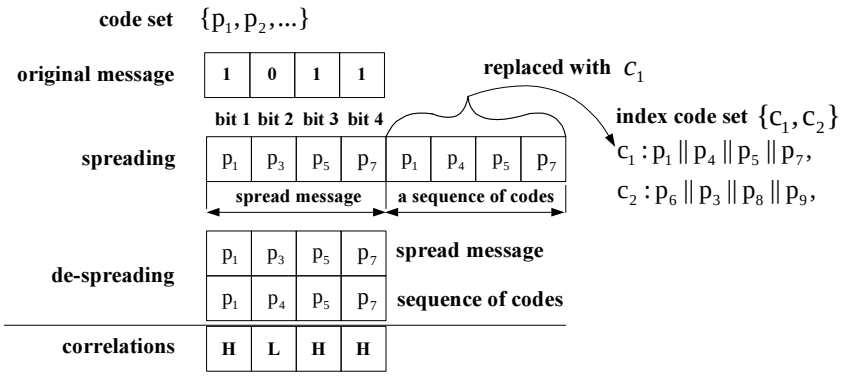

Figure 2.2: An example of the basic RD-DSSS scheme.

sender earlier. For de-spreading, the receiver computes the correlations between the corresponding codes in the first and the second halves of the spread message (i.e., $\mathbf{p}_1 \cdot \mathbf{p}_1$, $\mathbf{p}_3 \cdot \mathbf{p}_4$, $\mathbf{p}_5 \cdot \mathbf{p}_5$, and $\mathbf{p}_7 \cdot \mathbf{p}_7$). High correlation and low correlation are translated into "1" and "0", respectively.

To reduce the communication overhead, we propose to have the sender and the receivers share a set of pre-defined spreading code sequences, which are formed by the concatenations of codes in the spreading code set. We associate each code sequence with a special spreading code called *index code*. The collection of all index codes is referred to as the *index code set*. We require that the correlation between two different index codes is low. Intuitively, a sender can transmit an index code (instead of the actual code sequence) to indicate the code sequence for spreading. For example, Figure 2.2 shows that there are two code sequences, which are represented by index codes $\mathbf{c}_1$ and $\mathbf{c}_2$, respectively. Instead of sending $\mathbf{p}_1||\mathbf{p}_4||\mathbf{p}_5||\mathbf{p}_7$ as the second half of the spread message, the sender simply transmits $\mathbf{c}_1$. The index code set and the spreading code set should have no overlap so that a receiver can easily distinguish between an index code and a regular spreading code.

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

Code Set and Code Sequence Set

Let $\mathcal{P} = \{\mathbf{p}_1, ..., \mathbf{p}_n\}$ denote the spreading code set with n codes. As mentioned earlier, these codes should have low correlation with each other. There are multiple candidate codes for our scheme, such as Gold codes, Walsh-Hadamard codes, m sequences, and Kasami codes [86]. We assume each code in $\mathcal{P}$ is of length f and $\mathcal{P}$ is publicly known.

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

public.

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

Spreading

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

De-spreading

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

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

The threshold t is an important parameter. We derive the threshold t as the value that minimizes the probability of decision error when the transmitted signal is polluted by additive white Gaussian noise signal, which is a generally assumed jamming signal in wireless communications [86, 97]. This threshold is given in lemma 1.

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

Proof: Consider $X = (x_1, ..., x_l)$ and $\hat{X} = (\hat{x}_1, ..., \hat{x}_l)$, where $x_i = \mathbf{f}_i \cdot \mathbf{s}_i$ and $\hat{x}_i = \hat{\mathbf{f}}_i \cdot \mathbf{s}_i$ for $1 \leq i \leq l$. Let $\mathbf{n}_i = (n_{i1}, ..., n_{if})$ denote the errors introduced by wireless interference. Thus, $\hat{\mathbf{f}}_i = \mathbf{f}_i + \mathbf{n}_i$. We

assume the elements of $\mathbf{n}_i$ are independent and identically distributed (i.i.d) Gaussian random variables with mean value 0 and variance σ^2. Let $\mathbf{e} = (e_1, ..., e_l)$ denote $X - \hat{X}$, where $e_i = x_i - \hat{x_i}$, and let $\mathbf{s}_i = s_{i1}||s_{i2}||...||s_{if}$, where s_{ij} is valued -1 or $+1$. Note that $\hat{x}_i = \hat{\mathbf{f}}_i \cdot \mathbf{s}_i = (\mathbf{f}_i + \mathbf{n}_i) \cdot \mathbf{s}_i = x_i + \mathbf{n}_i \cdot \mathbf{s}_i$. Thus, $e_i = \mathbf{n}_i \cdot \mathbf{s}_i = \frac{1}{f} \sum_{j=1}^{f} n_{ij} s_{ij}$. According to the properties of Gaussian variables [62], e_i is i.i.d. Gaussian random variables with mean value 0 and variance σ^2.

When $m_i = 1$, $\hat{x}_i = x_i + e_i = 1 + e_i$. The probability density function (pdf) of $1 + e_i$ is $f_1(\hat{x}_i) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(\hat{x}_i - 1)^2}{2\sigma^2}}$. When $m_i = 0$, $\hat{x}_i = x_i + e_i = \rho + e_i$, the pdf of $\rho + e_i$ is $f_0(\hat{x}_i) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(\hat{x}_i - \rho)^2}{2\sigma^2}}$. Assume the sender transmits "1" or "0" with probability $\frac{1}{2}$. Thus, the probability of decision error is $p_e = \frac{1}{2}(\int_{-\infty}^{t} f_1(\hat{x}_i) d\hat{x}_i + \int_{t}^{\infty} f_0(\hat{x}_i) d\hat{x}_i) = \frac{1}{2}(1 + \frac{1}{2} erf(\frac{t-1}{\sqrt{2}\sigma}) - \frac{1}{2} erf(\frac{t-\rho}{\sqrt{2}\sigma}))$, where erf is the Error Function $erf(w) = \frac{2}{\sqrt{\pi}} \int_0^w e^{-y^2} dy$ [86].

To minimize p_e, we let the derivation of p_e be 0 (i.e., $\frac{dp_e}{dt} = 0$) and solves t. We can obtain $t = \frac{1}{2}(1 + \rho)$ and the minimized probability of decision error is $p_{emin} = \frac{1}{2} + \frac{1}{2} erf(\frac{\rho - 1}{2\sqrt{2}\sigma})$. □

Pros and Cons

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

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

2.3.2 Enhanced Scheme: Defending against Reactive Jamming

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

First, after generating the spread message as in the basic scheme, the sender permutes all codes of the spread message. Thus, even if the i-th bit of the original message is 1, the i-th observed code is not the same as the i-th code of the chosen code sequence, resulting in reduced correlation. For example, in Figure 2.2, the spread message is $\mathbf{p}_1||\mathbf{p}_3||\mathbf{p}_5||\mathbf{p}_7$ and the chosen code sequence is $\mathbf{p}_1||\mathbf{p}_4||\mathbf{p}_5||\mathbf{p}_7$.

Assume the correlation between two different codes is 0. Thus, the correlation between the spread message and the chosen code sequence is $\frac{1+0+1+1}{4} = 0.75$. However, if we permute the spread message and assume the result is $\mathbf{p}_7||\mathbf{p}_5||\mathbf{p}_1||\mathbf{p}_3$, then the correlation between the permuted spread message and the chosen code sequence is reduced to 0. Second, the sender spreads the message using k code sequences in $\mathcal{C}$. As a result, a reactive jammer must know all k code sequences in order to launch reactive jamming successfully.

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

Spreading

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

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

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

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

De-spreading

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

Let $\hat{\mathbf{F}}_p||\hat{\mathbf{c}}_{i_1}||...||\hat{\mathbf{c}}_{i_k}$ denote the received message, where $\hat{\mathbf{F}}_p$ is the permuted spread message and $\hat{\mathbf{c}}_{i_1}, ..., \hat{\mathbf{c}}_{i_k}$ are the index codes. The receiver needs to identify the index codes that are appended by the sender. For $1 \leq u \leq k$, the receiver computes the correlations between each index code and $\hat{\mathbf{c}}_{i_u}$ (i.e., $\mathbf{c}_1 \cdot \hat{\mathbf{c}}_{i_u}, ..., \mathbf{c}_q \cdot \hat{\mathbf{c}}_{i_u}$). The index code that results in the highest correlation with $\hat{\mathbf{c}}_{i_u}$ is marked as identified. Let $\mathbf{c}_{i_1{}'}, ..., \mathbf{c}_{i_k{}'}$ denote the identified index codes and $\mathbf{p}_{i_1{}'1}||...||\mathbf{p}_{i_1{}'l}, ..., \mathbf{p}_{i_k{}'1}||...||\mathbf{p}_{i_k{}'l}$ denote the corresponding code sequences, where $1 \leq i_1{}', ..., i_k{}' \leq q$. For $1 \leq j \leq l$, let $\mathcal{A}'_j = \{\mathbf{p}_{i_1{}'j}, ..., \mathbf{p}_{i_k{}'j}\}$. The receiver computes the permutation based on $h(\mathcal{A}'_1), ..., h(\mathcal{A}'_l)$, and then recovers the original code order of the spread message. Let $\hat{\mathbf{f}}'_1||...||\hat{\mathbf{f}}'_l$ denote the result. The receiver then computes $\hat{\mathbf{m}}$ as follows: If $\exists\ \mathbf{p} \in \mathcal{A}'_j$ s.t. $\hat{\mathbf{f}}_j \cdot \mathbf{p} \geq t$, let $\hat{m}_j = 1$. Otherwise, let $\hat{m}_j = 0$.

Ability to Deal with Reactive Jamming

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

The jammer may do an exhaustive search over all possible combinations of k code sequences in $\mathcal{C}$ to find those chosen by the user. The number of correlations the jammer should compute is $k\binom{q}{k}$. However, even with all the correlation results, the jammer still cannot determine which code sequences have been chosen, because (1) transmission of bit "0" can use any code other than those in the corresponding position of the selected code sequences and (2) multiple code sequences could be used for encoding. Moreover, the jammer must finish the computation within the transmission time of a single message in order to launch reactive jamming. For example, if $q = 13,500$ and $k = 5$, the number of correlations the attacker needs to compute is $5 \times \binom{13,500}{5} > 2^{64}$. Assume the length of the original message is $1,024$ bits and that of a spreading code is 256 chips. For a 802.11 wireless device with 11 Mbps data rate, the time for transmitting a spread message is $\frac{1,024 \times 256}{11 \times 10^6} \approx 0.024$ second. This means the jammer must finish such a huge amount of computation within 0.024 second. We provide detailed analysis to show the effectiveness of RD-DSSS in tolerating reactive jamming attacks in Section 2.5.2.

2.4 Performance Overheads

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

To get an intuitive feeling of the computational overhead, we did an experiment to test the time required by these additional operations. The system settings in the experiment are as follows: The message length is 1,024 bits, the size of $\mathcal{P}$ is 100, the size of $\mathcal{C}$ is 13,500, the length of each index code is 512 chips, the number of code sequences chosen by the sender is 3, and the length of each code in $\mathcal{P}$ is 256 chips. Table 2.1 shows the computation time of those operations performed on a computer with a 3.40 GHz Intel Pentium 4 CPU and 1.5 GB memory. All those operations together can be finished within 15 milliseconds. In practice, the correlation operation is inherently parallel (i.e., dot product of two vectors) and be finished efficiently with special purpose hardware.

Note that a reactive jammer must compute $3 \times \binom{13,500}{3}$ correlations in order to gather information of the chosen code sequences used by RD-DSSS with the settings in the above experiment. Assume the computation power of the jammer is 100 times of a normal receiver. Let t_j and r denote the computational overhead for the jammer to crack a single message and the ratio of t_j to t_a, respectively. Thus,

Table 2.1: computation time (milliseconds)

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

$t_j = \frac{3\times\binom{13,500}{3}\times t_c}{13{,}500\times 3\times 100}$ and $r = \frac{3\times\binom{13,500}{3}\times t_c}{13{,}500\times 3\times 100\times(t_h+t_s+t_p+t_{ip}+t_c)}$. In particular, using the parameters in Table 2.1, we can get the overhead $t_j = 2,729.8$ seconds, which is much larger than the transmission time of a single message, and the ratio $r = 183,680$. When k increases, the jammer's computation overhead t_j increases exponentially, since the jammer needs to check all combinations of k code sequences in $\mathcal{C}$ (i.e., $k\binom{q}{k}$). However, the receiver's computation overhead t_a increases linearly, since the receiver only needs to check $k \times q$ correlations for identifying the index codes.

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

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

2.5 Security Analysis and Simulation

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

2.5.1 Classification of Jamming Attacks

To facilitate the analysis, we first give a classification of jamming attacks. According to [86], jammers are classified into five types: *broadband noise jammers*, *partial-band noise jammers*, *continuous wave jammers*, *multitone jammers*, and *pulse jammers*. These jammers intentionally transmit random noise

signals to cause wireless interferences. They differ from each other in energy distributions of their jamming signals. In practice, there exist more complicated jammers. For example, a "smart" jammer can not only transmit random noise signals, but can also inject fake messages that are encapsulated in the packet format used by the communicators to mislead the reception process [111].

Based on the previous work [86, 111], we generalize jamming attacks into two categories: *non-intelligent jamming attacks* and *intelligent jamming attacks*. In the former attacks, the jammer disrupts wireless communication by sending random noise signals. In the latter attacks, the jammer transmits jamming signals that are generated based on his knowledge of the communication systems (e.g., the signal patterns, anti-jamming strategies, and communication protocols).

Note that conventional DSSS is capable of mitigating non-intelligent jamming attacks [86]. Therefore, inheriting from conventional DSSS, RD-DSSS is also non-intelligent jamming resilient. Thus, in this section, we focus our analysis on the intelligent jamming attacks.

2.5.2 Intelligent Jamming Attacks

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

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

- Type I: He can randomly choose codes from $\mathcal{P}$ and tries to jam the communication by transmitting those spreading codes.
- Type II: He can perform reactive jamming discussed in Section 2.3.1. That is, the jammer tries to find the code sequences used by a sender by analyzing the first portion of a message being transmitted, and then spreads a fake message using the identified code sequence to jam the rest of the message transmission.
- Type III: He can perform Denial of Service (DoS) attacks targeting the index codes, in which he randomly picks several index codes and transmits them along with the legal index codes to force receivers to deal with a large number of candidate index codes.

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

Type I Attacks

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

Analysis: Suppose the i-th code transmitted by the sender is $\mathbf{p}_i$ and $\mathcal{A}_i$ is the set that generates the i-th code of the permuted spread message. If the bit spread by $\mathcal{A}_i$ is "0" and $\mathbf{p}_i$ happens to be in $\mathcal{A}_i$, then the receiver cannot de-spread the original bit correctly. However, a few bit errors can be tolerated by ECC. For example, the standard (255, 223) Reed-Solomon code is capable of correcting up to 16 bit errors among every 223 information bits of a message [103]. If the ECC can correct a maximum of M bit errors of the original message but the jammer can collapse more than M bits, then the receiver cannot reconstruct the original message. In the following, we derive the probability that the jammer can disrupt the transmission of a message (i.e., the probability that the jammer can jam more than M bits).

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

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

Figures 2.3 shows the jamming probability p_J when each message has 1,024 bits. It is easy to see that p_J decreases as the size of the spreading code set (n) increases. When n is larger than 70 and $M = 60$, the probability is less than 10^{-4}.

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

We let $M = 60$ and n range from 50 to 60. For each n, we randomly generate a message and spread it using 5 code sequences randomly selected from $\mathcal{C}$. We also form the jammer's code sequence by randomly picking l codes from $\mathcal{P}$. Then we count the number of bits that can be disrupted by the jammer, and mark the trial as successful if the number is larger than M. We repeat this process for 1,000 times and estimate the *simulated* p_J (i.e., the probability that a message is jammed) as $p_J = \frac{\#\ successful\ trails}{\#\ trials}$. Figure 2.4 shows both the simulated and the theoretical probabilities when $M = 60$. We can see that both probabilities are less than 0.01 if the size of the code set is larger than 60.

We then let $n = 120$ and M range from 25 to 35. For each M, we perform the same simulation as we did for each n. Figure 2.5 shows the simulated and theoretical jamming probabilities when we

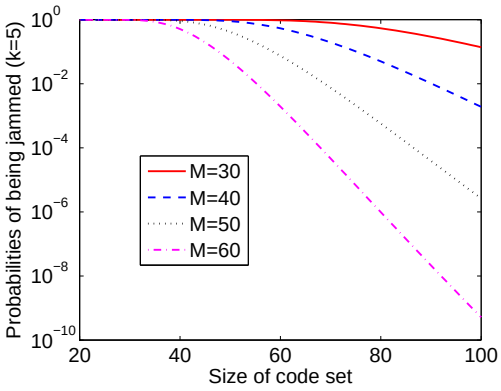

Figure 2.3: Theoretical probability that an attacker jams communication for $k = 5$

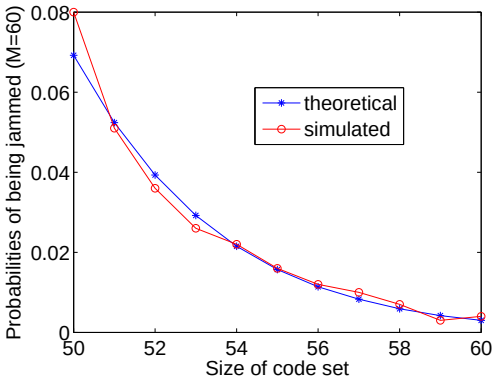

Figure 2.4: Simulated and theoretical probabilities that an attacker jams communication when $M = 60$

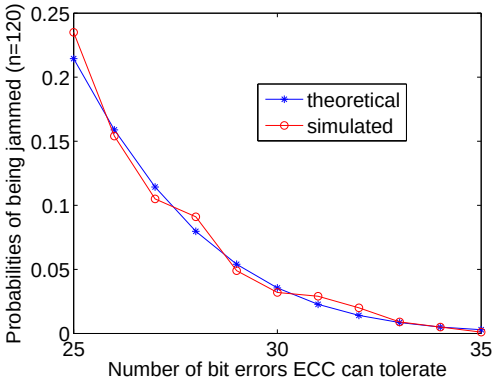

Figure 2.5: Simulated and theoretical probabilities that an attacker jams communication when $n = 120$

change the number of bit errors ECC can tolerate. We can see that both probabilities decrease as the number of bit errors ECC can tolerate increases. When $M = 35$, the jamming probabilities are less than 0.01.

Type II Attacks

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

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

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

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

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

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

$\mathbb{P}(\mathbf{f}_i = \mathbf{g}_i) = \frac{1}{2n} + \frac{1}{2}\sum_{j=1}^{k}\frac{1}{n-j}\mathbb{P}(|\mathcal{A}_{i_o}| = j)(1-\frac{1}{n})^j = \frac{1}{2n} + \frac{1}{2n^k}\sum_{j=1}^{k}\binom{n}{j}N(j)\frac{(n-1)^j}{n^j(n-j)}$. □

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

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

Proof: If $i = i_o$ (i.e., the i_o-th code of the original spread message does not change its position after permutation), the probability that the i-th observed code $\mathbf{f}_i$ is the same as $\mathbf{h}_i$ is $\mathbb{P}(\mathbf{f}_i = \mathbf{h}_i | i = i_o) = \frac{1}{2}\sum_{j=1}^{k}\mathbb{P}(|\mathcal{A}_{i_o}| = j)\frac{1}{j} = \frac{1}{2n^k}\sum_{j=1}^{k}\binom{n}{j}N(j)\frac{1}{j}$. If $i \neq i_o$ (i.e., the i-th code $\mathbf{f}_i$ of the original spread message changes its position after permutation), the probability that $\mathbf{f}_i = \mathbf{h}_i$ is the same as the probability that $\mathbf{f}_i = \mathbf{g}_i$, since both $\mathbf{h}_i$ and $\mathbf{g}_i$ can be regarded as an arbitrary code in $\mathcal{P}$.

Therefore, $\mathbb{P}(\mathbf{f}_i = \mathbf{h}_i | i \neq i_o) = \mathbb{P}(\mathbf{f}_i = \mathbf{g}_i)$. For $1 \leq i_o, j_o \leq l$, the probability that $\mathcal{A}_{i_o} = \mathcal{A}_{j_o}$ is $1/\binom{n}{k}$, which is a very small value if n is relatively large and k is relatively small (e.g., $1/\binom{n}{k} \approx 10^{-7}$ when $n = 50$ and $k = 5$). Therefore, the probability that $\mathcal{A}_{i_o} = \mathcal{A}_{j_o}$ is approximately 0. Note that for different inputs, the outputs of mapping function h are also different. Thus, the possible values of $h(\mathcal{A}_1), ..., h(\mathcal{A}_l)$ are distinct from each other. We consider $h(\mathcal{A}_1), ..., h(\mathcal{A}_l)$ as l random variables. The probability that $h(\mathcal{A}_{i_o})$ is just the i_o-th smallest/largest element among all values is $\frac{1}{l}$. Therefore, the probability that $i = i_o$ is $\frac{1}{l}$ and the probability that $i \neq i_o$ is $1 - \frac{1}{l}$. As a result, $\mathbb{P}(\mathbf{f}_i = \mathbf{h}_i) = \mathbb{P}(i = i_o)\mathbb{P}(\mathbf{f}_i = \mathbf{h}_i | i = i_o) + \mathbb{P}(i \neq i_o)\mathbb{P}(\mathbf{f}_i = \mathbf{h}_i | i \neq i_o) = \frac{1}{l}((\frac{1}{2n^k}\sum_{j=1}^{k}\binom{n}{j}N(j)(\frac{1}{j} - \frac{(n-1)^j}{n^j(n-j)})) - \frac{1}{2n}) + \frac{1}{2n} + \frac{1}{2n^k}\sum_{j=1}^{k}$. □

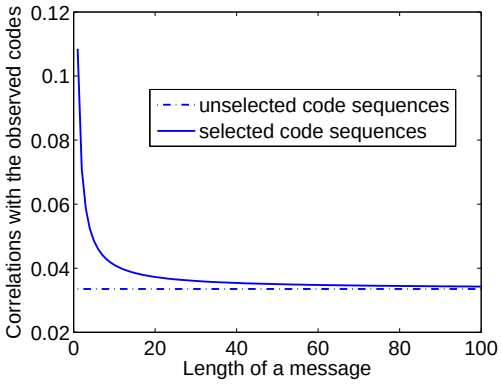

Figure 2.6: $\mathbb{E}(\mathbf{f} \cdot \mathbf{h})$ and $\mathbb{E}(\mathbf{f} \cdot \mathbf{g})$ ($k = 5$, $n = 30$)

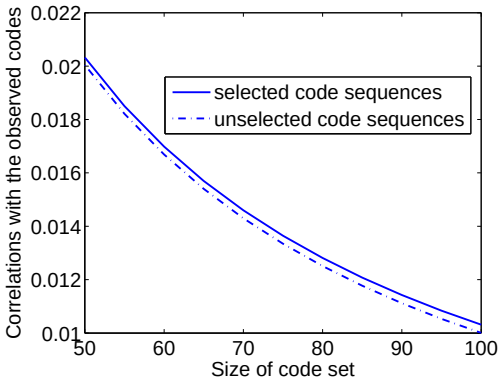

Figure 2.7: $\mathbb{E}(\mathbf{f} \cdot \mathbf{h})$ and $\mathbb{E}(\mathbf{f} \cdot \mathbf{g})$ ($k = 5$, $l = 300$)

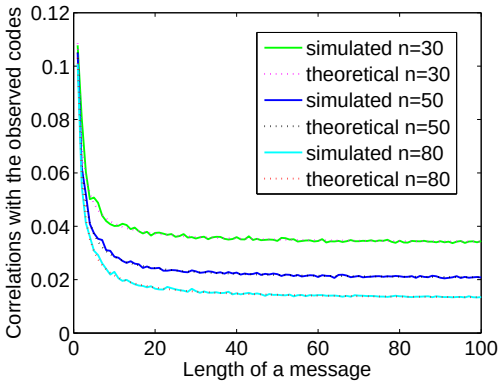

Figure 2.8: Simulated and theoretical $E(\mathbf{f} \cdot \mathbf{h})$ for $n = 30$, $n = 50$, and $n = 80$

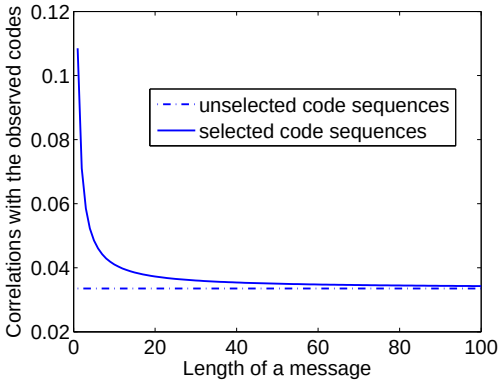

Figure 2.9: $\mathbb{E}(\mathbf{f} \cdot \mathbf{h})$ and $\mathbb{E}(\mathbf{f} \cdot \mathbf{g})$ ($k = 5$, $n = 30$)

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

Figure 2.9 shows the expectation of the correlation between the observed codes and a code sequence selected by the sender (i.e., $\mathbb{E}(\mathbf{f} \cdot \mathbf{h})$), as well as the expectation of the correlations between the observed codes and a code sequence not selected by the sender (i.e., $\mathbb{E}(\mathbf{f} \cdot \mathbf{g})$). We can see that $\mathbb{E}(\mathbf{f} \cdot \mathbf{h})$ approaches $\mathbb{E}(\mathbf{f} \cdot \mathbf{g})$ as the length l of the message increases. When l is larger than 100, $\mathbb{E}(\mathbf{f} \cdot \mathbf{h})$ and $\mathbb{E}(\mathbf{f} \cdot \mathbf{g})$ are almost the same. Thus, it is hard for the jammer to distinguish the code sequences of the sender by analyzing correlations between the observed codes and each code sequence in $\mathcal{C}$.

Figure 2.7 shows both $\mathbb{E}(\mathbf{f} \cdot \mathbf{h})$ and $\mathbb{E}(\mathbf{f} \cdot \mathbf{g})$ when the size of the spreading code set increases. We can see that $\mathbb{E}(\mathbf{f} \cdot \mathbf{h})$ is very close to $\mathbb{E}(\mathbf{f} \cdot \mathbf{g})$. Although both $\mathbb{E}(\mathbf{f} \cdot \mathbf{h})$ and $\mathbb{E}(\mathbf{f} \cdot \mathbf{g})$ decrease as n increases, the distance between them is small.

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

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

Figure 2.8 shows simulated and theoretical $\mathbb{E}(\mathbf{f} \cdot \mathbf{h})$ for $k = 5$. The correlation between the observed codes and the code sequences selected by the sender decreases as n increases. The correlation is less than 0.02 when $n = 80$ and $l \geq 20$.

In addition, we obtain $\mathbb{E}(\mathbf{f} \cdot \mathbf{h})$ through simulation when the size of $\mathcal{P}$ increases, with $k = 1$ and $k = 5$. In the simulation, $l = 300$ and other parameter settings are the same as those in the simulation of Figure 2.8. We let n range from 50 to 100. For each n, we perform the same process as we did for each l in the simulation of Figure 2.8, and estimate the simulated $\mathbb{E}(\mathbf{f} \cdot \mathbf{h})$. Figure 2.10 shows both simulated and theoretical $\mathbb{E}(\mathbf{f} \cdot \mathbf{h})$. For both $k = 1$ and $k = 5$, we can see that the correlation between the observed codes and the code sequences selected by the sender is quite small ($\mathbb{E}(\mathbf{f} \cdot \mathbf{h}) \leq 0.022$).

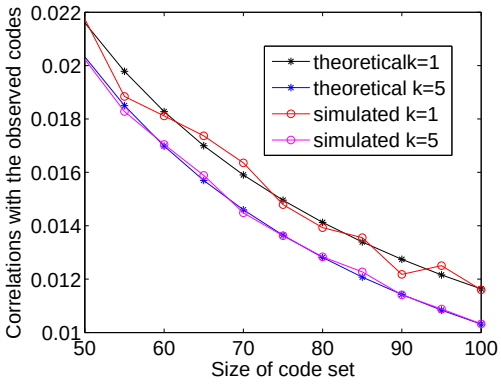

Figure 2.10: Simulated and theoretical $\mathbb{E}(\mathbf{f} \cdot \mathbf{h})$ for $k = 1$ and $k = 5$ ($l = 300$)

The simulation results are very close to the theoretical results, as shown in Figures 2.8 and 2.10. Both figures demonstrate that there is very small correlation between the observed codes and the code sequences selected by the sender. Thus, RD-DSSS can prevent jammers from learning the chosen code

sequences and launch reactive jamming attacks. In contrast, UDSSS allows a reactive jammer to derive the code sequence directly by observing the first code in the sequence.

Type III Attacks

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

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

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

Chapter 3

BitTrickle: Combating Broadband and High Power Reactive Jamming

3.1 Assumptions and Threat Model

Our system consists of a sender and a receiver, who aim to communicate in the presence of jamming attacks. We assume that both the sender and the receiver are general wireless devices that can transmit and receive wireless signals. We assume an *intelligent reactive* jammer who emits noisy signals onto wireless channels to interfere with the transmission from the sender to the receiver. The goal of the jammer is to block the communication from the sender to the receiver in an efficient way. Specifically, the jammer remains quite when the sender is not transmitting, but starts sending noise signal once it senses any transmission from the sender. We assume that the jammer has a high transmit power and can jam all channels used by the sender and the receiver.

3.2 Overview of BitTrickle

At first glance, a reactive jammer seems to be perfect and invulnerable. It transmits when the sender transmits and it stops when the sender stops. Such adaptive behavior provides a jammer with high efficiency and concealment. However, a closer examination on the working principle of a reactive jammer reveals its Achilles Heel.

Consider a scenario in Figure 3.1. The signal emitted by the sender is also received by the jammer, and it passes through channel sensing algorithm (e.g., energy detector) that determines if the sender is transmitting. If yes, the reactive jammer configures its transmitter to send jamming signals. The process of channel sensing leads to a weakness of the reactive jammer, since it takes a short period of time, which creates an opportunity for the sender to transmit information.

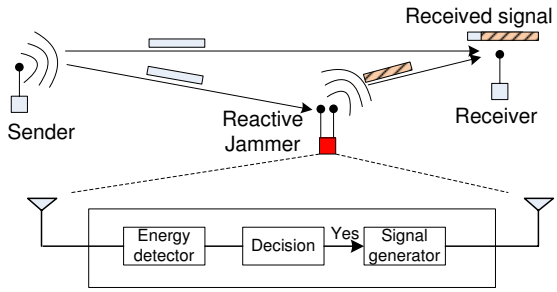

Figure 3.1: Reactive jammer starts jamming once detecting sender's transmission.

3.2.1 Transmission at the Sender

Intuitively, the sender can simply split the original message into very short packets. If each packet can be transmitted within the reaction time of the jammer, then the receiver can construct the original message. Note that a typical packet usually contains fields such as packet ID, payload, and error correction information. Thus, if a reactive jammer reacts very fast, the sender can only deliver the first few bits of the packet before the jammer starts jamming. Therefore, the receiver cannot receive the original message. To deal with fast reactive jammers, we let the sender transmit a message in a "bits-by-bits" manner rather than "packet-by-packet", and let the receiver collect all unjammed bits and assemble them together to recover the original message.

Transmission: The sender encodes a message using a *BitTrickle encoder*, which enables the reconstruction of the original data at the receiver in the presence of transmission errors (e.g., bit lost) caused by jamming, low link quality and other reasons. Encoding/Decoding details will be discussed in Section 3.4. As shown in Figure 3.2, the sender transmits each bit of the encoded message for multiple times to increase the chance that the receiver can receive this bit.

The sender also takes a random backoff before each transmission. Specifically, the sender becomes quiet for a random time before next transmission. This makes it hard for the reactive jammer to predict when the sender will start the next transmission. To disrupt the communication, the jammer may sense the channel again to determine if the sender is transmitting, which yields another reaction time and thus opportunity for the sender to transmit. Alternatively, the jammer may keep jamming for a long time or randomly jam the channel. However, long time jamming degrades the performance of the jammer (e.g., causing reduced efficiency and concealment), and the communicators can increase their backoff time to deal with it, whereas random jamming still yields idle time slots between two attempts of jamming. As long as the reaction time or idle time slots exist, the sender can always deliver information to the receiver.

Figure 3.2 shows that the sender attempts to transmit one bit a time. This is the most conservative approach that needs to be used when the sender has no information on the reception at the receiver.

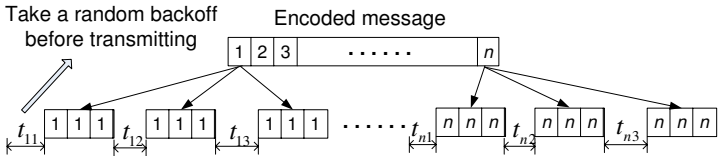

Figure 3.2: Transmission at the sender

This can be easily improved by learning how many bits can be received in one transmission within the jammer's reaction time, through, for example, detecting the point where jamming happens or using ACK packets from the receiver, which can be transmitted in a similar way. Specifically, the sender can transmit multiple bits rather than one bit a time, as shown in Figure 3.3.

In our study, we focus on the transmission of one bit a time, since it makes a qualitative change from "cannot communicate" to "can communicate" in the presence of a reactive jammer and forms the foundation of other possible improvements.

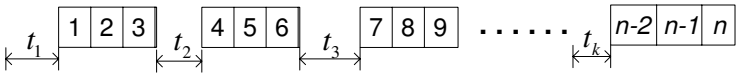

Figure 3.3: Assume the sender knows that 3 bits can be transmitted within the reaction time of the jammer. The sender can transmit 3 bits of the message between the random backoffs.

3.2.2 Reception at the Receiver

The receiver uses two steps to reconstruct the sender's message: (1) extract the unjammed bits from each transmission, and (2) decode the collection of unjammed bits into the original message.

Obtaining Unjammed Bits: To extract unjammed bits, the receiver needs to remove jammed bits and fake bits injected by the jammer. As shown in Figure 3.4, each received bit enters a *jamming detector*, which checks whether this bit is jammed or not. All jammed bits are discarded and the output of the jamming detector is a bit stream that consists of unjammed bits that are either from the sender or the jammer. The development of such a jamming detector is shown in Section 3.3. The output of the jamming detector is further fed into an *authenticator*, which distinguishes the sender's bits from the jammer's bits by using traditional physical layer authentication approaches such as radio frequency (RF) fingerprinting (e.g., [65, 112]) and radiometric techniques (e.g., [12]). Ideally, the output of the authenticator is a bit stream that formed by unjammed bits transmitted by the sender only.

Decoding Unjammed Bits: After obtaining unjammed bits, the receiver needs to decode them to re-

construct the sender's original message. Note that a bit and all its copies may be totally jammed by the jammer if the jammer keeps jamming the channel for a long time. Also, the sender transmits each bit for multiple times, and thus the receiver may receive duplicate unjammed bits. Therefore, to deal with transmission errors such as lost/duplicate bits, the receiver utilizes a *BitTrickle decoder*, which corresponds to the *BitTrickle encoder* used by the sender, as an error recovery mechanism against reception failures. The output of the decoder is the reconstructed message.

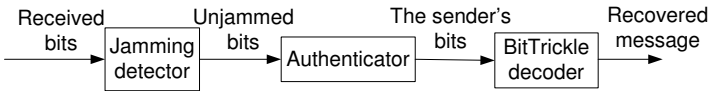

Figure 3.4: Reception at the Receiver

3.2.3 Technical Challenges

Two technical challenges need to be addressed in the process of developing BitTrickle.

Jamming Detector: The requirement of jamming detection is drastically different from traditional jamming detection, which is only to find out whether or not wireless communication or a transmitted packet is jammed (e.g., [91,111]). Instead, jamming detection in BitTrickle needs to distinguish jammed bits from unjammed ones. This requirement makes the previous jamming detectors insufficient.

It is easy to see why jamming detection based on packet delivery rate fails to meet the requirement of BitTrickle. However, it is more subtle to understand why those approaches based on received signal strength (RSS) do not work.

RSS-based jamming detection identifies jamming when the RSS is high. However, this approach will fail in situations where the distribution of RSS values are inherently time-varying. For example, wireless hardware typically employs automatic gain control (AGC) [41], which reduces the power of a strong signal and raises the power of a weak signal to keep RSS at an appropriate range. Due to such adjustment, jammed signals and unjammed signals have almost the same RSS, and thus it is hard for the receiver to identify which signal is jammed. Moreover, in practice, multiple reasons may contribute to the changes of RSS values. For example, in mobile communication, RSS values increase as a sender gets near to a receiver and decrease as the sender moves away. Similarly, for wireless devices that utilize power control technologies (e.g., [26, 48, 74]), wireless interference is not the only reason that causes a high RSS values, since the sender dynamically adjusts its transmit power for energy efficiency.

In our study, we propose a novel technique that takes advantage of modulation properties to distinguish jammed and unjammed bits. The proposed detection method does not rely on RSS values, and thus can be used in general wireless applications that have either dynamic or static RSS.

BitTrickle Decoder: Transmission errors like lost or duplicate bits may happen when there exist jamming attacks or a retransmission mechanism is employed. At first glance, an ECC might be directly used by BitTrickle to deal with transmission errors. However, a closer look reveals that this message reconstruction is quite different from traditional error recover, where both correct and error bits are in their correct positions. Indeed, in a sequence of bits received by the BitTrickle Decoder, a small number of lost/duplicate bits can make many bits mis-aligned, which greatly reduce the efficiency of ECC and exceed their correction capacity. Thus, it is likely to have a large number of bits errors in the reconstructed message. To deal with lost/duplicate bits, we develop BitTrickle decoder, which can find the original position for each received bit in the encoded message, and enable the use of ECC with high efficiency.

3.3 Jamming Detector-Identifying Jammed Bits

In this section, we design a novel jamming detector, which utilizes physical layer modulation properties to precisely distinguish between jammed and unjammed bits. To facilitate the understanding of our approach, we first give some background information on modulation.

3.3.1 Preliminaries on Modulation

I/Q modulation techniques have been widely used in modern wireless systems, including WCDMA, WiMax, ZigBee, WiFi, and DVB (Digital Video Broadcasting). In I/Q modulation, data bits are encoded into a physical layer symbol, which is the transmission unit in the physical layer. In the following, we take Quadrature Phase-Shift Keying (QPSK) modulation, a typical I/Q modulation, as an example to illustrate how I/Q modulation works.

QPSK – An Example I/Q Modulation: QPSK encodes two bits into one symbol at a time. Hence, as shown in Figure 3.5, bits 00, 01, 10, and 11 are represented by points whose coordinates are $(0, 1)$, $(-1, 0)$, $(0, -1)$, and $(1, 0)$ in an I/Q plane, respectively. The I/Q plane is called a *constellation diagram*. A symbol is the coordinate of a point in the constellation diagram. For a bit sequence 0010, the modulation output are two symbols: $(0, 1)$ and $(0, -1)$.

A symbol received by a receiver is not exactly the same as the original symbol transmitted by the sender, since wireless channels are usually noisy and introduce distortions (e.g., phase and amplitude changes) to signals that pass through them [39]. Hence, in demodulation, the receiver finds the point that is closet to the received symbol in the constellation diagram. For example, in Figure 3.5, the distance between the received symbol and the point (0, 1) is the minimum, and thus the demodulation output is 00.

We use QPSK as an example to illustrate I/Q modulation. The proposed jamming detector is not limited to QPSK and can be applied to all variations of I/Q modulation such as M-PSK and quadrature

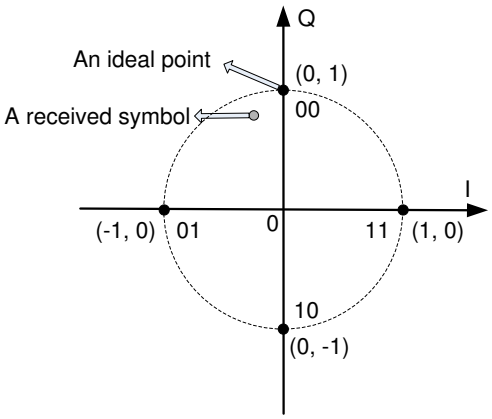

Figure 3.5: QPSK modulation/demodulation

amplitude modulation (QAM).

3.3.2 Experimental Observation

Intuitively, jamming signals can introduce a large, even unrecoverable distortion to signals transmitted by the sender, since the goal of the jammer is to interfere and corrupt desired signals. Therefore, if a received symbol is jammed, it may greatly deviate from its ideal point in the constellation diagram and can hardly be recovered. Based on this intuition, we perform experiments to examine the impacts of jamming attacks on symbol locations in a constellation diagram.

We collect the received symbols using USRPs [57], which are radio frequency (RF) front ends equipped with analog to digital (AD) and digital to analog (DA) converters. In our experiments, three USRPs are used as the sender, the receiver, and the jammer, respectively, each of which is connected to a computer. AGC is employed by USRPs. We set the bit rate as 1Mbps, carrier frequency as 5GHz, and modulator as QPSK.

We consider two communication scenarios: a normal scenario and a jamming scenario. In the first one, only the sender transmits randomly generated packets to the receiver, while in the second one, both the sender and the jammer transmit random packets to the receiver concurrently. We let the receiver records locations (i.e., coordinates in the constellation diagram) of received symbols.

Figures 3.6 and 3.7 show locations of received symbols for the normal and the jamming scenarios, respectively. In the normal scenario, as shown in Figure 3.6, received symbols form four clusters, each of which centers around an ideal point of QPSK. However, when there exist jamming attacks, as shown in Figure 3.7, all received symbols tend to be randomly spread over the constellation diagram; it is hard to determine the ideal points for most of the received symbols, and demodulation errors may happen frequently.

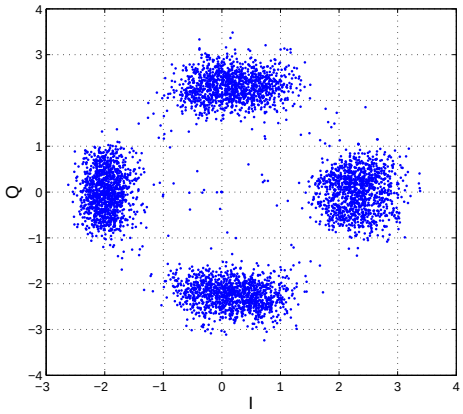

Figure 3.6: Normal scenario: Received symbols center around ideal points

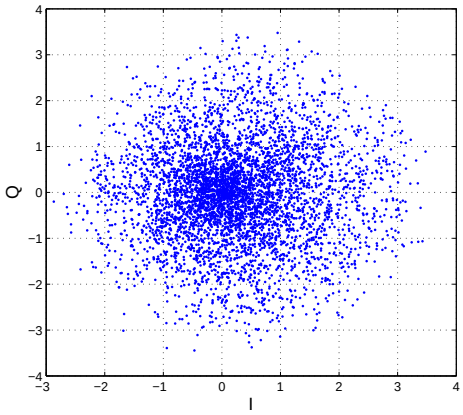

Figure 3.7: Jamming scenario: Received symbols deviate from ideal points

Note that the sender's signal maybe treated as noise and filtered by noise cancellation algorithms at the receiver if the jammer's transmit power is high enough. In this case, received symbols may behave normally, since they can be regarded as from the jammer's signal only. However, they will be further removed by physical layer or cryptographic authentication.

3.3.3 Detection Method

Let d_{unjam} denote the distance between an unjammed symbol and the origin in the constellation diagram. Further let d_{jam} denote the distance between a jammed symbol and the origin.

Detection Metric: As shown in the above experiment, unjammed symbols are close to their ideal constellation points, and thus d_{unjam} approximately equals to the distance between an ideal point and

the origin.

On the other hand, jammed symbols deviate from their ideal points. Due to AGC, such deviation is actually a convergence from ideal points toward the origin rather than an expansion out of the constellation diagram range. Hence, unlike unjammed symbols, jammed symbols are randomly distributed within the constellation diagram, and the expected value of d_{jam} is smaller than that of d_{unjam}. For example, in Figures 3.6 and 3.7 , the expected value of the distance between a received symbol and the origin is 2.2524 and 1.2628, respectively.

We propose to use the distance d between a received symbol and the origin of the constellation diagram as a metric to detect the existence of jammed symbols.

Temporal Detection: For each received symbol, we compute the corresponding distance d, and then compare d with a threshold t. If $d > t$, then the received symbol is marked as unjammed. Otherwise, it is a jammed symbol and will be discarded. To increase accuracy, we enhance the above detection approach by using a temporal check. Let s_i and d_i denote the i-th received symbol and its distance from the origin, respectively.

In the check, we determine whether s_i is jammed or not by examining a temporal symbol sequence $s_{i-N}, ..., s_{i-1}s_is_{i+1}, ..., s_{i+N}$, where N is a small positive integer (e.g., $N = 1$). Symbol s_i is marked as unjammed, if all symbols in the temporal sequence have distances that are larger than the threshold (i.e., $d_{i-j} > t$ for $0 \leq j \leq N$).

Two remaining problems need to be answered: (1) How to determine the detection threshold t, and (2) how well the detection method works. These problems turn out to be related. In the following, we first look at the quality of detection and then develop a method to determine the detection threshold t.

3.3.4 False Alarms and False Negatives

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

In the following, we derive both probabilities of false negative and false alarm.

Theorem 1 *(Probability of false alarm) The probability P_{fa} that an unjammed symbol is incorrectly classified as a jammed symbol is $1 - (M_1(\frac{v}{\sigma_N}, \frac{t}{\sigma_N}))^{2N+1}$, where M_1 is the Marcum Q-function [3], v is the distance between an ideal point and the origin of the constellation diagram, t is the threshold, $2N + 1$ is the length of the temporal sequence, and ${\sigma_N}^2$ is the variance of the jamming signal.*

Proof: Let (I, Q) and (I_i, Q_i) denote the coordinate of a received symbol and its closest ideal point in a constellation diagram, respectively. Due to the existence of jamming signal, $I \neq I_i$ and

$Q \neq Q_i$. For an unjammed symbol, we assume additive white Gaussian noise, and thus I and Q can be represented as $I = I_i + \delta_I$ and $Q = Q_i + \delta_Q$, where δ_I and δ_Q are independent and identically distributed (i.i.d) Gaussian random variables with mean value 0 and variance ${\sigma_N}^2$. According to the properties of Gaussian variables [63], $I = I_i + \delta_I$ and $Q = Q_i + \delta_Q$ are also i.i.d. Gaussian random variables. The mean values of I and Q equal to those of I_i and Q_i, respectively, and the variances of them are all ${\sigma_N}^2$.

Let d denote the distance between the received symbol and the origin of the constellation diagram. Thus, $d = \sqrt{I^2 + Q^2}$. According to [75], d follows Rice distribution and the cumulative distribution function $F_d(t)$ of d is $F_d(t) = \mathbb{P}(d \leq t) = 1 - M_1(\frac{v}{\sigma_N}, \frac{t}{\sigma_N})$, where $\mathbb{P}(d \leq t)$ denote the probability that d is less than or equal to t, $v = \sqrt{{I_i}^2 + {Q_i}^2}$, and M_1 is the Marcum Q-function. Note that the probability P_{fa} of false alarm equals to the probability that d of at least one symbol in the temporal sequence is less than or equal to t. Therefore, $P_{fa} = 1 - (1 - \mathbb{P}(d \leq t))^{2N+1} = 1 - (M_1(\frac{v}{\sigma_N}, \frac{t}{\sigma_N}))^{2N+1}$. □

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

Proof: Let (I, Q) denote the coordinate of a received symbol. A jammed symbol is the mixture of the sender's symbol and the jammer's symbol. Hence, I and Q can be represented as $I = I_s + I_j + \delta_I$ and $Q = Q_s + Q_j + \delta_Q$, where (I_s,Q_s) and (I_j, Q_j) are the symbols transmitted by the sender and the jammer, respectively, and δ_I and δ_Q are additive white Gaussian noise.

Assume that the sender transmits each ideal point in the constellation diagram with equal probability. Therefore, I_s, Q_s, I_j, Q_j are i.i.d random variables. According to central limit theorem, the probability distribution of the average of i.i.d random variables converges to Gaussian distribution as the number of random variables increases. Therefore, we use Gaussian distribution to roughly approximate the distribution of $(I_s + I_j)/2$ and $(Q_s + Q_j)/2$. According to the properties of Gaussian variables [63], $I = I_s + I_j + \delta_I$ and $Q = Q_s + Q_j + \delta_Q$ are also approximately Gaussian distributed, where δ_I and δ_Q are i.i.d Gaussian random variables with mean 0. Note that all ideal points center around the origin, and thus the mean values of I_s, Q_s, I_j, and Q_j are 0.

Let d denote the distance between the received symbol and the origin of the constellation diagram. Thus, $d = \sqrt{I^2 + Q^2}$. Assume that I and Q have the same variance, which is denoted by σ^2. According to [39], d follows Rayleigh distribution and the cumulative distribution function $F_d(t)$ of d is $F_d(t) = \mathbb{P}(d \leq t) = 1 - e^{\frac{-t^2}{2\sigma^2}}$. The probability P_{fn} of false negative equals to the probability that d of all symbols in the temporal sequence are larger than t. Therefore, $P_{fn} = (1 - \mathbb{P}(d \leq t))^{2N+1} = (e^{\frac{-t^2}{2\sigma^2}})^{2N+1}$. □

Experimental Validation: To verify the theoretical probabilities of false alarm and false negative, we let the threshold t range between 0 and 5, and for each value of t we run the temporal check enhanced

method to detect unjammed symbols from symbols collected for normal and jamming scenarios in our earlier experiment. Ratios of detected symbol number to total symbol number are used to compute the real measured probability of false alarm and false negative. (i.e., $P_{fa} = 1 - \frac{\text{\# detected symbols}}{\text{\# total symbols}}$ and $P_{fn} = \frac{\text{\# detected symbols}}{\text{\# total symbols}}$).

Figures 3.8 and 3.9 show the results for $N = 1$ and $N = 3$, respectively. Meanwhile, we compute P_{fa} and P_{fn} using Theorems 7 and 2. The computation results are also shown in Figures 3.8 and 3.9. Note that statistic parameters v, σ_N, and σ are determined based on our earlier experiment[1]. Both theoretical and real measured results are in close consistency.

A large N can result in both small P_{fn} and P_{fa}. When $N = 1$, both real measured P_{fn} and P_{fa} can be as low as 0.0444 by using a threshold t that equals to 1.6. If we increase N to 3, we can achieve even lower error rate.

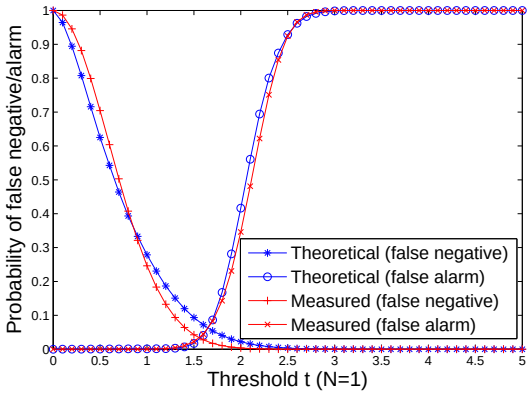

Figure 3.8: Theoretical and measured probabilities of false alarm and false negative when $N = 1$.

3.3.5 Determining the Threshold

The threshold t can be determined based on the system requirement for P_{fn} and P_{fa}. For example, if the false negative probability P_{fn} is required to be less than α, we have $(e^{\frac{-t^2}{2\sigma^2}})^{2N+1} < \alpha$. By treating t as an unknown and solve the inequality, we can get $t > \sqrt{2\sigma^2 \ln \alpha^{2N+1}}$. As the threshold t increases, P_{fa} increases but P_{fn} decreases. If the system objective is to minimize both P_{fa} and P_{fn}, as shown in Figures 3.8 and 3.9, the minimization result and the corresponding threshold t form the intersection point of the P_{fa} and P_{fn} curves.

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

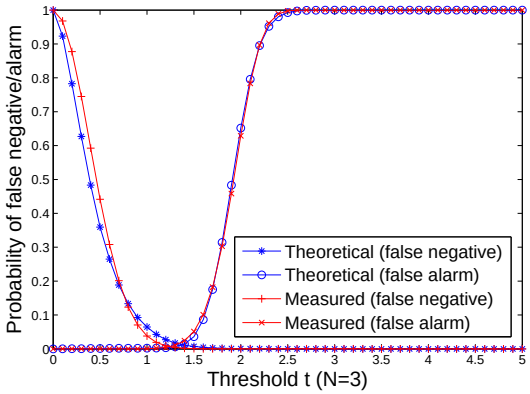

Figure 3.9: Theoretical and measured probabilities of false alarm and false negative when $N = 3$.

3.4 BitTrickle Encoding/Decoding

In this section, we develop an encoding/decoding scheme, which enables the recovery of original message in the presence of transmission errors. We assume the original message is first encoded with a traditional ECC (e.g., Reed-Solomon codes) before being processed by BitTrickle. The use of traditional ECC corrects substitution errors (i.e., bit "'1" is replaced by "0" and vice-versa). The BitTrickle encoding scheme further encodes the ECC-coded message to allow a receiver to decode the correct position of each received bit and recover from synchronization errors such as lost and duplicate bits.

3.4.1 Basic Idea

For the sake of presentation, we call the input to BitTrickle encoding (i.e., the ECC-coded message) as a BTmessage.

BitTrickle Encoding: The sender and the receiver agree on a sequence that is formed by n integers, where n is the length of the BTmessage. We call such an integer sequence a *positioning code* and each integer in the sequence a *label*. As shown in Figure 3.10, the BTmessage is 10110 and the positioning code is 03572. For $1 \leq i \leq 5$, the sender labels the i-th bit of the message using the i-th label in the positioning code (e.g., the second bit is 0 and its label is 3). In the labeling, the sender uses one symbol to represent both a bit and its label. (Details of labeling will be presented in Section 3.4.2.) Note that a symbol is the transmission unit of physical layer. Hence, if a receiver receives a symbol, the receiver knows both the bit and its label. The encoding results are 5 symbols as shown in Figure 3.10.

Transmission Errors: The sender transmits the encoded symbols using the method discussed in Section 4.3. Figure 3.11 shows an example. The sender transmits the first symbol for 3 times, takes a random backoff, and transmits this symbol again for 3 times. The sender repeats this process on all the

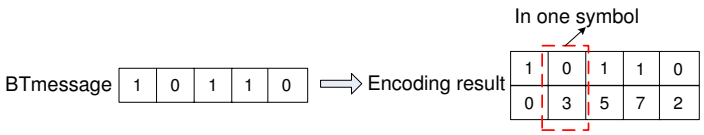

Figure 3.10: BitTrickle encoding

following symbols until the last symbol is transmitted. Due to jamming attacks and retransmissions, a symbol may get lost or duplicated. For example, in Figure 3.11, all copies of the 2nd symbol are lost and the 4th symbol is duplicated.

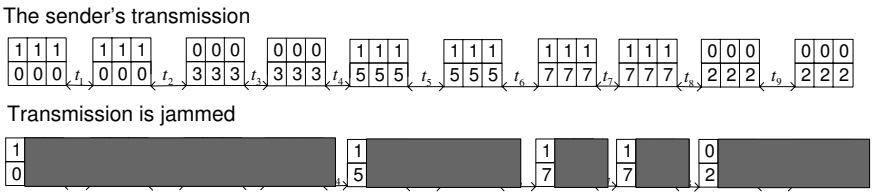

Figure 3.11: Transmission Errors

BitTrickle Decoding: The receiver demodulates each received symbol to extract the bit and corresponding label carried by this symbol. Figure 3.12 shows an example following Figure 3.11. In this example, the extracted bits and labels are 11110 and 05772, respectively. The receiver then takes two steps to correct synchronization errors.

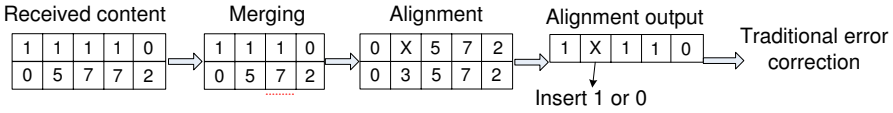

Figure 3.12: BitTrickle decoding

The first step is merging, in which bits are merged into a single bit if they are identical and have the same label. As shown in Figure 3.12, the 3rd and the 4th received bit are identical (i.e., both of them are 1), and have the same label 7. Thus, they are merged together. The merging result is 1110 and the corresponding labels are 0572. Note that an incorrect merging may happen if multiple bits in the BTmessage are identical and share the same label. In Section 3.4.3, we give the analytical upper bound of the probability of merging errors, and we show that the upper bound decreases quickly as

configurable parameters such as the total number of retransmissions of each bit change.

The second step is alignment. The merged labels are 0572 and the positioning code is 03572. The labels 0, 5, 7, and 2 match the 1st, 3rd, 4th, and the last label in the positioning code, respectively. Thus, the receiver knows that the second bit is lost, and corrects synchronization errors by inserting a bit that can be either 1 or 0 at the position shown in Figure 3.12. The alignment output is further processed by traditional ECC to recover the original message. There may exist multiple alignment outputs, since the merged labels may fit multiple combinations of positions in the positioning code. In Section 3.4.3, we develop a fast alignment approach that not only achieves desired alignment accuracy, but also reduces the overhead by only trying a subset of all combinations.

Diversity Degree of a Positioning Code: Note that consecutive bits of a BTmessage may happen to be the same, and they will be treated as duplication of a single bit and incorrectly merged together if they have the same label. To avoid such situation, we require that consecutive labels in the positioning code to be different. Specifically, the i-th label in the positioning code does not equal to any of its previous d labels (i.e., $i-1$-th, ..., $i-d$-th label) and successive d labels (i.e., $i+1$-th, ..., $i+d$-th label), where $d \geq 1$ is a parameter that can be adjusted by a user, referred to as *diversity degree* of the positioning code. For example, when diversity degree is 2, the 8th label should not be the same as the 7th, 6th, 9th and 10th label.

In the following, we present more details during BitTrickle encoding and decoding.

3.4.2 Encoding at Sender

In the encoding, the sender adds special data content (e.g., 10101010) to both the beginning and the end of a BTmessage, so that a receiver can recognize the boundary of a BTmessage. We refer to the special data content as a *message delimitation code (MDC)*.

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

For an *M*-ary modulator that encodes $\log_2 M$ bits by one symbol, the maximum value of a label of the positioning code should not exceed $2^{\log_2 M-1}-1 = \frac{M}{2}-1$. For example, an 8PSK symbol uses one bit to carry data information and two bits to carry the label. Hence, a label should be less than or equal to 3 (i.e., 11). Packing a data bit and its label in one symbol achieves atomicity: data bits are always associated with their labels. Upon receiving a symbol, the receiver knows both the data bit and its label.

3.4.3 Decoding at Receiver

During decoding, the receiver first searches for boundaries of a BTmessage. As shown in Figure 3.13, the boundary of the BTmessage is identified if the receiver can observe an MDC or a certain data pattern that is a part of MDC. For example, assume that the MDC equals to 1010101010, the receiver identifies the beginning or end of a BTmessage if the receiver receives 1010101010 or consecutive 10's (e.g., 101010).

To reduces the chances that the entire MDC is jammed, the sender and the receiver can increase the length of the MDC according to the severity of jamming attacks, so that the receiver can observe at least a part of the MDC. Alternatively, they may also use backoff time between transmitting two consecutive symbols of an MDC to reduce the chance of colliding with the jammer's signals.

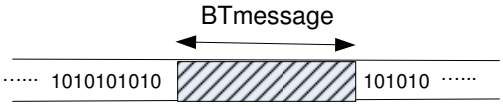

Figure 3.13: Identifying boundaries of a BTmessage

The receiver then demodulates the symbols of the received BTmessage, and extracts a data bit and a label from each symbol. The number of extracted data bits equals the number of symbols. As discussed earlier, there may be lost and/or duplicated bits, and the receiver takes two steps to correct synchronization errors.

Merging: Bits are identified as duplicated bits and merged into a single bit if they are consecutive, identical and have the same label. To detect and merge duplicated bits, the receiver points a cursor to the first bit/label of the received BTmessage. Then, the receiver compares the bit/label pointed by the cursor and each of the following $N_r - 1$ bits/labels, where N_r denote the number of retransmissions for a single bit. If inequality occur (e.g, two bits are not equal or have different labels), the receiver merges all equal bits/labels together and points the cursor to the next bit/label. The receiver repeats the same process until all bits/labels of the received BTmessage are scanned. The expected number of comparisons is $\frac{L}{(N_r-1)/2} \times \frac{N_r-1}{2} = L$, where L is the message length.

Merging Errors: Different bits in a BTmessage may be incorrectly merged together, thus introducing extra lost bits. However, the occurrences of merging errors do not mean that the entire message is unrecoverable. Lost bits can still be recovered by alignment and ECC. In Theorem 3, we derive an upper bound of the probability of merging errors. Before we show Theorem 3, we first give the following Lemma.

Lemma 4 *The probability that two labels in a positioning code equal to each other is less than or equal*

to $\frac{1}{R-d}$, where d is the diversity degree of the positioning code and R is the number of possible values for each label (e.g., R=4 for 8PSK).

Proof: Let $S = s_1||...||s_n$ denote the positioning code, where n is the number of labels in it. Let p_{eq} denote the probability that the i-th label s_i equals to the j-th label s_j, where $1 \leq i, j \leq n$ and $i \neq j$. Without loss of generality, we assume that $j > i$. According to the diversity requirement of a positioning code (See Section 3.4.1), $s_j \neq s_{j-1}, ..., s_{j-d}$ and $s_i \neq s_{i+1}, ..., s_{i+d}$. If $j - i \leq d$, s_j is one of the previous d labels of s_i. Hence, $p_{eq} = 0$. If $j - i = d + 1$, s_j's previous d labels are actually s_i's successive d labels, and thus $s_j \neq s_{j-1}, ..., s_{j-d}$ and $s_i \neq s_{j-1}, ..., s_{j-d}$. Therefore, $p_e = \frac{1}{R-d}$. If $d + 1 < j - i \leq 2d$, there exists an overlap between s_j's previous d labels and s_i's successive d labels, but the length of the overlap is less than d. Thus, $p_e < \frac{1}{R-d}$. If $j - i > 2d$, there is no overlap and $p_e = \frac{1}{R}$. Thus, p_e is at most $\frac{1}{R-d}$. □

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

Proof: Let $S = s_1||...||s_n$ and $M = m_1||...||m_n$ denote the positioning code and original BTmessage, respectively. Let $M_r = m_{i_1}^{r_1}||...||m_{i_g}^{r_g}$ denote the received BTmessage, where m_{i_j} is the i_j-th element of the original BTmessage and $m_{i_j}^{r_j}$ means that the receiver receives r_j retransmitted copies of m_{i_j} (e.g., $m_2^3 = m_2||m_2||m_2$). $m_{i_j}^{r_j}$ may be incorrectly merged with $m_{i_{j+1}}^{r_{j+1}}$ if they are identical and have the same label (i.e., $m_{i_j} = m_{i_{j+1}}$ and $s_{i_j} = s_{i_{j+1}}$).

When there are no lost bits, the receiver receives the whole message and $M_r = m_1^r||...||m_n^r$, where r is the number of retransmissions. Note that each label in the positioning code is different from any of its previous and successive d labels (i.e., $s_i \neq s_{i\pm k}$ for $1 \leq k \leq d$). Therefore, m_i^r cannot be merged with $m_{i\pm1}^r$, since the label of m_i is different from those of its predecessor and successor. Thus, m_i can only be merged with its own copies, yielding a merged message $m_1||...||m_n$, and the probability of merging errors is 0.

When there exist jamming attacks, an information bit (i.e., a bit in the original BTmessage and all its retransmitted copies) may get lost. Let p denote the probability that a bit transmitted by the sender is lost due to jamming, low link quality, and other reasons. The probability that an information bit is lost equals to p^r, where r is the number of retransmissions. If $i_{j+1} - i_j \leq d$ (i.e., less than d information bits between $m_{i_{j+1}}$ and m_{i_j} are lost), then $m_{i_{j+1}}$ is among the successive d elements of m_{i_j} and their labels are always different. Thus, the probability of merging errors is 0. If $i_{j+1} - i_j > d$ (i.e., more than or equal to d bits between $m_{i_{j+1}}$ and m_{i_j} are lost), $m_{i_j}^{r_j}$ and $m_{i_{j+1}}^{r_{j+1}}$ will be identified as duplicate bits when $m_{i_j} = m_{i_{j+1}}$ and $s_{i_j} = s_{i_{j+1}}$. Let p_{eq} be the probability that the i-th element s_i equals to the j-th

element s_j, where $1 \leq i, j \leq n$ and $i \neq j$. According to Lemma 4, the probability of merging errors is $\frac{1}{2}p_{eq} \leq \frac{1}{2(R-d)}$. The overall probability p_{e_j} that $m_{i_j}^{r_j}$ is incorrectly merged with $m_{i_{j+1}}^{r_{j+1}}$ is

$$\begin{aligned} p_{e_j} =& 0 \times \mathbb{P}(i_{j+1} - i_j \leq d) + \frac{p_{eq}}{2}\mathbb{P}(i_{j+1} - i_j > d) \\ =& \frac{p_{eq}}{2} \sum_{k=d+1}^{n-n(1-p^r)+1} \mathbb{P}(i_{j+1} - i_j = k) = \frac{p_{eq}}{2} \sum_{k=d}^{n-n(1-p^r)} p^{rk}(1-p^r) \\ \leq& \frac{(p^{rd} - p^{r(n-n(1-p^r)+1)})}{2(R-d)}, \end{aligned}$$

where $n(1-p^r)$ is the expected number of information bits received by the receiver. For the received BTmessage $M_r = m_{i_1}^{r_1}||...||m_{i_g}^{r_g}$, the probability p_e of merging incorrectly is 1-$\prod_{j=1}^{j=n(1-p^r)-1}(1-p_{e_j})$, and thus

$$p_e \leq 1 - (1 - \frac{p^{rd} - p^{r(n-n(1-p^r)+1)}}{2(R-d)})^{n(1-p^r)-1}. \tag{3.1}$$

□

We use simulation to validate the above analytical upper bound.[2] We let $R = 32$ and $p = 0.95$, and perform 10,000 trails in our simulation. In each trial, we randomly generate a message and a positioning code whose length is 155, and label the message using the positioning code. We retransmit each bit of the message for r times ($10 \leq r \leq 30$), and delete each retransmitted bit with probability p. We then merge the remaining bits, and compare the result with the correct result obtained based on the original generated message. If both results are not equal, a merging error happens and we mark this trial as failed. We compute the simulated probability of merging error ($\frac{\text{\# failed trials}}{\text{\# total trials}}$) and its analytical upper bound using and Equation (3.1), respectively.

As shown in Figure 3.14, the simulated probability of merging error is only slightly less than its analytical upper bound, which indicates that the upper bound computed by Equation (3.1) is a tight upper bound. It shows that a larger diversity degree d can achieve smaller error probability. As the number r of retransmissions increases, both the simulated probability and its upper bound decrease and approach to 0. In particular, when $d = 8$ and $r = 20$, the simulated probability and the analytical upper bound are 0.0005 and 0.0006, respectively.

Alignment: In this step, the receiver attempts to find the actual position of each received bit in the original BTmessage. Let S denote the positioning code and L the merged labels (e.g., in Figure 3.12, the merged labels are 0572). As discussed earlier, the receiver can feed L into S to determine the positions of received bits. For example, if $L = 17$ and $S = 1317$, then either the first and the last bit or the last two bits of the original BTmessage are received.

[2]Simulations are done in MATLAB 7.7.0 on a computer with a 2.30 GHz AMD CPU and 4.0 GB memory

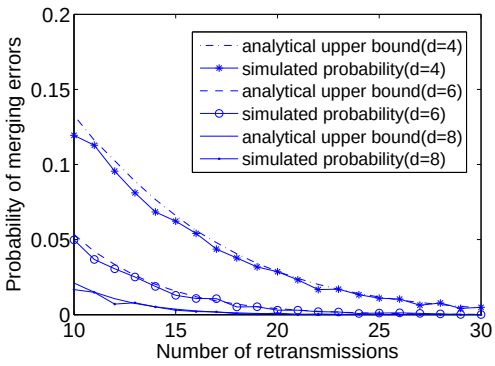

Figure 3.14: Probability of merging errors

(1) A Basic Alignment Method: If the length of the positioning code is small, we can do alignment in a brute force way. Specifically, we find all possible combinations of positions of received bits. Assume that the length of L is q. The receiver can find all length-q subsequences of S, and compare each of them with L. For each subsequence that equals to L, the receiver generates an alignment output by padding 1's or 0's into the positions of lost bits. For example, assume that padding bits are 1's and the received message after merging is 00. For $L = 17$ and $S = 1317$, the alignment outputs are 0110 and 1100. Each alignment output is further processed by traditional ECC decoding, where replacement errors (i.e., $1 \rightarrow 0$ or $0 \rightarrow 1$) are corrected. Since there may exist multiple alignment outputs, the receiver may obtain multiple decoding results, among which the one that can pass cyclic redundancy check (CRC) or authentication is the recovered message.

The number of comparisons the brute force method requires is $\binom{n}{q}$, where n is the length of S. If n is large, the brute force method will be time consuming. Hence, we develop a fast alignment approach to reduce the overhead.

(2) A Fast Alignment Method: To achieve fast alignment, we propose to only find one alignment instead of finding all possible alignments in a brute force way. We will then show that given proper configurations, this single alignment actually leads to a very small error probability.

We use a simple greedy strategy to obtain a single alignment. Specifically, the receiver compares labels of L with those of the positioning code S, trying to find S's leftmost or rightmost subsequence that equals to L. For example, if $L = 17$ and $S = 1177$, the S's leftmost and rightmost subsequence that equals to L is underlined in $\underline{1}1\underline{7}7$ and $1\underline{1}7\underline{7}$, respectively. The positions of the leftmost/rightmost subsequence is 13/24, and thus the corresponding decision is that the first and the third bits of the original BTmessage are received (or the second and the last bits are received).

Alignment Errors: For basic alignment, the probability that alignment errors happen is 0, since the basic alignment approach examines all possible combinations. For fast alignment, alignment errors may

happen if the positions of the leftmost/rightmost subsequence are not formed by the correct positions of the received bits. Without loss of generality, we assume that the fast alignment finds the leftmost subsequence of the positioning code. In Theorem 4, we derive an upper bound for the probability of alignment errors.

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

Proof: Let $S = s_1||...||s_n$ denote the sequence formed by the positioning code and p_{eq} be the probability that the i-th element s_i equals to the j-th element s_j, where $1 \leq i, j \leq n$ and $i \neq j$. According to Lemma 4, $p_{eq} \leq \frac{1}{R-d}$, where R and d are the number of labels and the diversity degree of the positioning code, respectively.

Let $F = f_1||...||f_q$ denote the sequence formed by the actual positions of received bits (i.e., the receiver receives the f_1-th,...,f_q-th bits), and L denote the sequence formed by merged labels. F is the positions of S's leftmost subsequence that equals L if two conditions are satisfied: (1) For $1 \leq i < f_1$, $s_i \neq s_{f_1}$. (2) For $1 \leq j \leq q-1$ and $f_j < i < f_{j+1}$, $s_i \neq s_{f_{j+1}}$. Therefore, the probability p_{min} that F is the positions of S's leftmost subsequence is

$$p_{min}=\prod_{i=1}^{f_1-1}(1-\mathbb{P}(s_i = s_{f_1}))\prod_{j=1}^{q-1}\prod_{i=f_j+1}^{f_{j+1}-1}(1-\mathbb{P}(s_i = s_{f_{j+1}})).$$

Assuming that $f_j < i < f_{j+1}$, the probability

$$\mathbb{P}(s_i = s_{f_{j+1}}) = p_{eq}\mathbb{P}(f_{j+1} - i > d) \leq \frac{\mathbb{P}(f_{j+1} - f_j > d)}{R-d}.$$

Note that $f_{j+1} - f_j > d$ indicates that at least d labels between s_{f_j} and $s_{f_{j+1}}$ are lost. Therefore,

$$\begin{aligned}\mathbb{P}(s_i = s_{f_{j+1}}) &\leq \frac{\sum_{k=d}^{n-q} p^{rk}(1-p^r)}{R-d} \\ &= \frac{p^{rd} - p^{r(n-q+1)}}{R-d}. \end{aligned} \tag{3.2}$$

Similarly, for $1 \leq i \leq f_1$,

$$\mathbb{P}(s_i = s_{f_1}) \leq \frac{\mathbb{P}(f_1 - 1 > d)}{R-d} = \frac{p^{rd} - p^{r(n-q+1)}}{R-d}. \tag{3.3}$$

Then, it follows from (3.2) and (3.3) that

$$p_{min} \geq (1 - \frac{p^{rd} - p^{r(n-q+1)}}{R-d})^{f_q - q}.$$

Note that f_q is a random variable ranging from q to n. According to total probability formula,

$$\begin{aligned} p_{min} &\geq \sum_{k=q}^{n} \mathbb{P}(f_q = k)(1 - \frac{p^{rd} - p^{r(n-q+1)}}{R-d})^{k-q} \\ &= \sum_{k=q}^{n} \frac{\binom{k}{q}}{\sum_{w=q}^{n}\binom{w}{q}}(1 - \frac{p^{rd} - p^{r(n-q+1)}}{R-d})^{k-q}. \end{aligned}$$

The probability p_e that the alignment is incorrect equals to the probability that F is not the positions of S's leftmost subsequence. Hence,

$$\begin{aligned} p_e &= (1 - p_{min}) \\ &\leq 1 - \sum_{k=q}^{n} \frac{\binom{k}{q}}{\sum_{w=q}^{n}\binom{w}{q}}(1 - \frac{p^{rd} - p^{r(n-q+1)}}{R-d})^{k-q}. \end{aligned} \tag{3.4}$$

□

We also use simulation to validate the analytical upper bound of alignment errors. Parameters are the same as those used in the simulation for merging errors (i.e., $R = 32$, $p = 0.95$, and 10,000 trials). In each trial, we randomly generate a positioning code of length 155, retransmit each label of the positioning code for r times ($10 \leq r \leq 30$), and delete each retransmitted label with probability p. The remaining labels are merged together. Then we find the positions of received bits (labels) using the fast alignment alignment approach, and compare the result with the true positions. If they are not equal, an alignment error happens and we mark this trial as failed. We compute the simulated probability of alignment error ($\frac{\text{\# failed trials}}{\text{\# total trials}}$) and its analytical upper bound using and Equation (3.4), respectively.

Figure 3.14 shows that the simulated probability of alignment error and the analytical upper bound decrease as the number of retransmissions increases, and a larger diversity degree d can lead to a smaller error probability. The upper bound computed by Equation (3.4) is a tight upper bound of the error probability. In particular, when $d = 8$ and $r = 20$, both the simulated probability and the analytical upper bound are about 0.0006.

3.5 Implementation and Evaluation Results

We develop a prototype system for BitTrickle to facilitate the experimental evaluation of BitTrickle performance under reactive jamming. The prototype system consists of a sender and a receiver, both

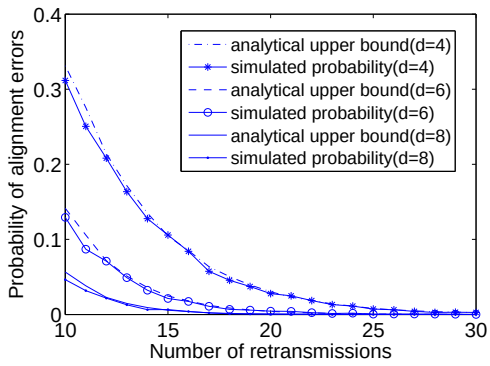

Figure 3.15: Probability of alignment errors

implemented as a USRP connected to a commodity PC that runs the sender (receiver) program. The USRPs are equipped with XCVR2450 daughter boards operating in the 2.4GHZ range, and are used as RF front ends. The software implementing BitTrickle is based on GNURadio [1].

Communication Flow: Figure 3.16 shows the communication flow of the BitTrickle prototype. The sender encodes the original message using the BitTrickle encoder, and then modulates the binary bits of the encoded message into symbols. Each symbol is further split into in-phase (I) and quadrature-phase (Q) components, and the sender multiplies I component and Q component by a cosine and sine carrier signal, respectively. The outcomes of both multiplication are superimposed, resulting in the modulated signal. Finally, the sender uses a D/A converter to transform the modulated signal into RF signal and transmit in the wireless channel.

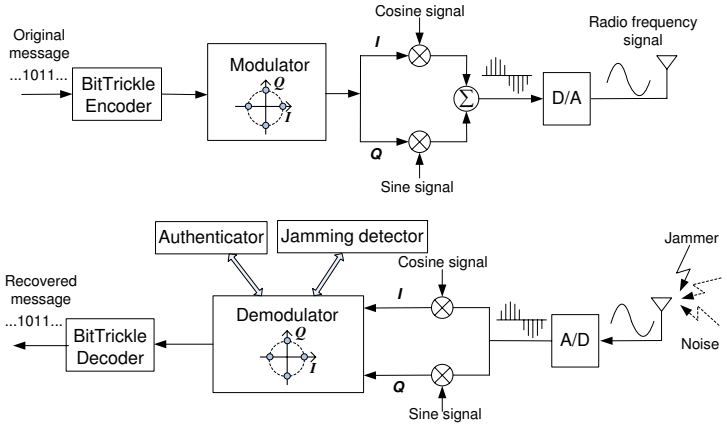

Figure 3.16: A Communication Framework of BitTrickle

In the reception process, the receiver applies an A/D sampler to transform a RF signal into a modulated signal, and then multiplies the digital modulated signal by cosine and sine carrier signals to get the I and Q components, which together represent a physical layer symbol. Each symbol enters a demodulator that can decode symbols into binary bits. A jamming detector and an authenticator run on top of the demodulator. They decide whether a symbol is jammed or fake by looking at intermediate demodulation results, and feed their decisions to the demodulator, which will discard all "bad" symbols. Finally, the receiver uses the BitTrickle decoder to process the output of the demodulator and recover the message from the sender.

Reactive Jammer: A reactive jammer senses the channel and transmits jamming signals once it detects a sender's signal. In our evaluation, we setup a high power and sensitive reactive jammer to test the performance of BitTrickle. The jammer is implemented on USRPs using GNURadio [1]. We employ energy detection to achieve a lower channel sensing time. Specifically, a sender's signal is detected if received signal strength exceeds a configurable threshold.

Note that a reactive jammer not only transmits jamming signals, but receives from the wireless channel to detect legitimate user's transmission. In our design of the jammer, we equip the jammer with two RFX2400 daughter boards that are used as a transmitter and a receiver, respectively. For both the transmitter and receiver component, we set the parameter "samples per symbol" the minimum value supported by GNURadio to reduce the processing delay (i.e., 2 and 4 for transmitter and receiver, respectively). Also, we let the jammer transmits with maximum gain and place the jammer within 0.1 meter range of the BitTrickle receiver. Parameters of the reactive jammer is shown in Table 3.1.

Table 3.1: Technical details of the reactive jammer

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

Comparison: To understand the capability of BitTrickle, we compare the following schemes under reactive jamming.

1. **BitTrickle**–The prototype implementation of BitTrickle sender and receiver. This approach uses Reed-Solomon (RS) error correction codes, and differential 8 phase shift keying (D8PSK) modulator/demodulater. The prototype system supports two RS coding rate, which are RS(155, 55)

and RS(60, 36)[3].

2. **GNURadio Benchmark**–The benchmark communication tool provided by GNURadio for data transmission and file transfer between two USRPs. The source codes are located at the directory `gnuradio/gnuradio-examples/python/digital`.

3. **802.11 DSSS**–IEEE 802.11 protocol running at direct-sequence spread spectrum (DSSS) mode on commercial 802.11 wireless cards. This approach uses a 11-bits barker code for spreading, carrier sense multiple access with collision avoidance mechanism (CSMA/CA) to resolve collisions on shared wireless channels, and forward error correction (FEC) to enable the reconstruction of the original, error-free data.

Evaluation Metrics: The jammer's goal is to prevent the communication between legitimate users. Therefore, how well the sender and the receiver can communicate under jamming attacks is a primary concern to assess anti-jamming systems. Thus we use the following metrics to evaluate the performance.

Packet delivery ratio (PDR): The ratio of the number of correctly received packets to the total number of packets transmitted by the sender. We consider a packet to be received correctly if the packet can pass CRC check.

Throughput: This is the number of successfully delivered bits normalized by time unit. Herein, we use bits per second to measure the throughput.

3.5.1 Component Evaluation

We first examine the performance of jamming detector and physical layer authenticator, which are two components included in BitTrickle.

Jamming Detector: The function of jamming detector is to remove jammed symbols. We use the temporal based detection method discussed in Section 3.3.3 to detect jammed symbols. To examine the the performance of jamming detector in terms of false negative rate and false alarm rate, we let the receiver collect the distance of each symbol from the origin of the constellation diagram and perform off-line analysis in MATLAB. Figure 3.17 shows the result for temporal sequence length $N = 5$. We can see that a threshold of 0.3 balances the false negative and false alarm. In our implementation, we set the threshold and N to be 0.3 and 5, respectively.

Physical Layer Authenticator: The authenticator aims to remove symbols inserted by the jammer. We develop a simple device authenticator based on [12], which uses modulation error metrics (i.e., frequency error, phase error, magnitude error, EVM, I/Q offset, SYNC correlation) to identify wireless devices. To simplify the implementation of the authenticator, we only choose EVM (error vector magnitude) as the metric to identify the sender. In the training stage (the jammer is turned off), we let the

[3] i.e., 55/36 bits are encoded into 155/60 bits

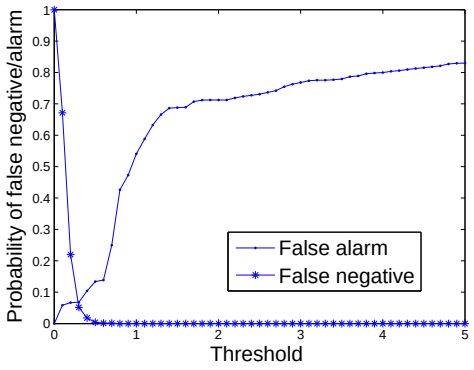

Figure 3.17: False negative/alarm of jamming detector

sender transmit and record the EVMs of received symbols. Those EVMs are used as the fingerprints of the sender's signal. The receiver computes the Euclidean distance between received k symbols and each of the fingerprints. The minimum distance is then compared with a threshold to decide whether the received symbols are transmitted by the sender.

To test false negative and false alarm rate, we collect EVMs of the sender and the jammer and perform off-line analysis in MATLAB. Figure 3.18 shows the result. For a threshold that equals to 14.5, the false negative and the false alarm rate achieved by the authenticator are 0.0970 and 0.0788, respectively. It should be noted that we use a simple physical layer authenticator in our prototype system. Certainly other advanced authenticators (e.g., a hybrid authenticator that utilizes both radiometric signatures and wireless channel properties) can be adopted in BitTrickle to achieve a even lower false alarm/negative rate.

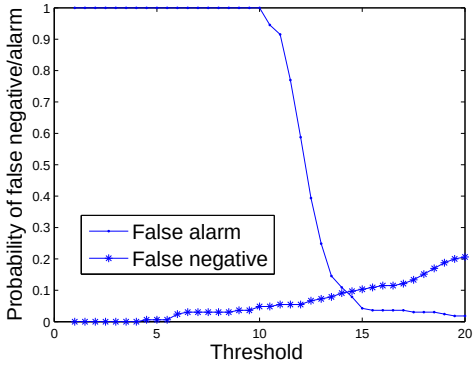

Figure 3.18: False negative/alarm of physical layer authenticator

Dealing with False Alarms/Negatives: False alarms can result in additional lost bits, and thus increase the difficulty of decoding. However, they do not directly lead to packet delivery failures, since BitTrickle uses retransmission mechanisms, basic/fast alignment and traditional error correction codes to deal with lost bits.

On the other hand, false negatives result in inserted bits. If the inserted bits do not interleave with the sender's bits, they can be directly removed by CRC checksum and upper layer cryptographic authentication. Decoding failures may occur if inserted bits interleave with the sender's bits. Note that those inserted bits are jammed bits or incoherent pieces of a fake message injected by the jammer, and thus the correlation between their labels and the positioning code is weak. Therefore, in the prototype system, to reduce decoding failures and filter out inserted bits, we perform alignment on the most correlated part between the positioning code and the (merged) received labels (e.g., the largest common subsequence between the positioning code and received labels).

3.5.2 Performance of BitTrickle

We use the scenario in Figure 3.19 in the overall evaluation of BitTrickle, where the jammer is physically much closer to the receiver than the sender.

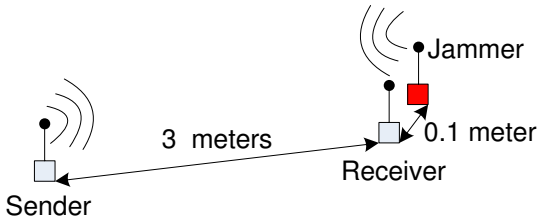

Figure 3.19: Evaluation scenario

We set the transmission bit rate of the sender, the jammer, and the receiver to be 1Mbps. The sender transmits 100 data packets, each with 1500 bytes. Positioning codes used by BitTrickle are randomly generated, and the diversity degree is set to 2 throughout the evaluation. Since the size of a data packet (1,500 bytes) is too long to be directly used with the positioning code and ECC, we divide a single packet into multiple blocks and append a CRC checksum to each block. In our experiments, we use block size 36 or 55 bits, then RS(60,36) or RS(155, 55) for ECC, and finally a positioning code of 60 or 155 bits.

Packet Delivery Ratio: To examine the performance in detail, we consider different jamming intensities. Specifically, we use a probabilistic reactive jammer, which jams at probability p for $0 \leq p \leq 1$ once detecting a sender's signal. The jamming duration is set to be 10 times of the transmission time of

a single data packet. We compute packet delivery ratio as $\frac{\text{\# correct packets (blocks)}}{\text{\# total transmitted packets (blocks)}}$. Figure 3.20 shows the result.

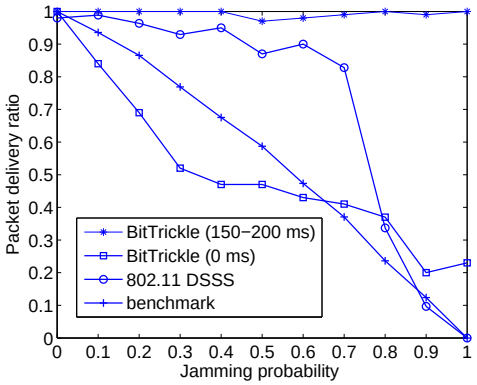

Figure 3.20: Packet delivery ratio

(1) 802.11 DSSS and GNURadio Benchmark: For 802.11 DSSS and GNURadio benchmark, packet delivery ratio decreases as jamming probability increases. Note that GNURadio benchmark does not employ ECC and retransmission mechanism, and thus the packet delivery ratio decreases at a rate linearly proportional to the jamming probability. 802.11 DSSS uses packet retransmissions and forward error correction. Hence, it achieves a higher packet delivery ratio than GNURadio benchmark. However, when jamming probability exceeds 0.7, the performance of 802.11 DSSS degrades dramatically. For both 802.11 and GNURadio benchmark, when the jamming probability equal to 1 (i.e., the jammer always jams when it senses a transmission), the packet delivery ratio drops to 0, and no data packets can be delivered.

(2) BitTrickle: For BitTrickle, we let the sender takes a random backoff ranging between 150–200 ms after it transmits every 6 bits, each bit of which is retransmitted for 15 times. Figure 3.20 shows that BitTrickle achieves a stable packet delivery ratio that fluctuates around 1 no matter how jamming probability varies.

We then reduce the backoff time to 0 ms and increase the bit retransmissions to 60. Figure 3.20 shows that the packet delivery ratio of BitTrickle decreases as jamming probability increases. This is because the reduced backoff time increases the chance that the sender's signal collides with the jammer's signal. The modulator used by the BitTrickle prototype has a higher bit error rate (i.e., BER) than that used by GNURadio benchmark (i.e., GFSK). Therefore, the packet delivery ratio of BitTrickle is less than that of benchmark when jamming probability is small (e.g., ≤ 0.7). However, when the jamming probability is 1, unlike GNURadio benchmark and 802.11 DSSS, BitTrickle with zero backoff still

achieves a non-zero packet delivery ratio (i.e., more than 0.2).

Throughput: To examine the throughput of BitTrickle, we consider a common jamming scenario, where the reactive jammer jams the channel as long as it hears the target signal(i.e., $p = 1$). To be conservative, we set the backoff time of BitTrickle to be 0 ms. Note that BitTrickle with non-zero backoff can achieve a higher throughput than BitTrickle with zero backoff, since the packet delivery ratio of the former is much higher than that of the latter. We performs 40 trials. In each trial, the sender transmits 100 data packets to the receiver and we compute throughput as $\frac{\text{\# correct packets (blocks)} \times \text{packet (block) length}}{\text{transmission time}}$. Figure 3.21 plots the computed throughput for each trial. The GNURadio benchmark and 802.11 DSSS fail to deliver any packet (0 throughput), whereas BitTrickle still achieves a throughput that ranges between 200–900 bits/s.

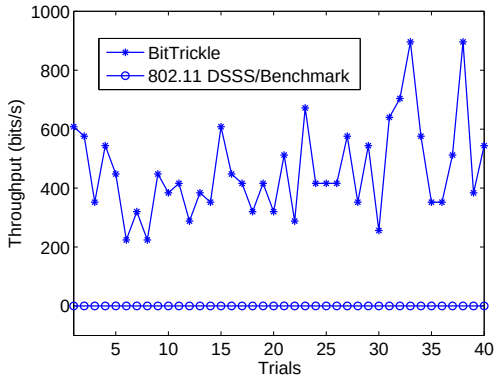

Figure 3.21: Throughput of each scheme when jamming probability is 1.

To understand how ECC coding rate affects the throughput, we test the throughput of BitTrickle using RS(155,55) and RS(60,36) respectively. Figure 3.22 plots the throughput as a function of signal-to-jamming ratio (SJR), which is the ratio of the reaction time of the jammer to the jamming duration.

Figure 3.22 shows that RS(60,36) leads to a higher throughput than RS(155,55) when SJR is less than 0.25. This is because RS(60,36) requires a shorter positioning code than RS(155,55), which reduces the chance of synchronization errors caused by alignment. As SJR increases, the receiver can get more information from the sender, and thus the probability of synchronization errors decreases. Note that the error correction capability of RS(155,55) is stronger than that of RS(60,36). For small SJRs, RS(155,55) does not suffer from severe synchronization errors, and thus it can correct more substitution errors and achieve a better throughput. When attacked by a jammer with SJR = 0.5, the throughput of the prototype system using RS(155,55) is about 2.5kbps. Figure 3.22 indicates that we can improve the throughput of BitTrickle by choosing an appropriate coding rate for different SJRs.

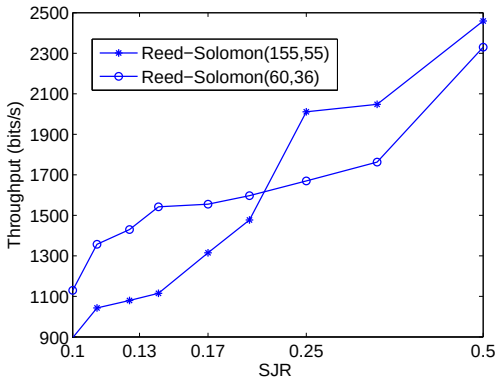

Figure 3.22: Throughput for different coding rate.

Remarks: The experiments reveal the following:

1. It is possible that a reactive jammer can defeat traditional anti-jamming approaches like FHSS and DSSS if the jammer is broadband and has high transmit power. As an example shown in the experiments, the throughput of 802.11 DSSS is zero and no packets can be delivered when it is attacked by the reactive jammer.

2. Even if traditional anti-jamming methods fail, the experiments indicate that BitTrickle can still allow wireless nodes to establish communication in the presence of a powerful reactive jammer by taking advantage of the channel sensing behavior of reactive jammers.

3. The efficiency of BitTrickle can be improved by adjusting system parameters (e.g., choosing an appropriate coding rate).

Chapter 4

Authenticating Primary Users' Signals in Cognitive Radio Networks

4.1 Preliminaries

In this section, I provide some preliminary information on link signatures, which will be used for primary user detection.

Radio signal generally propagates in the air over multiple paths due to reflection, diffraction, and scattering [65]. Therefore, a receiver usually receives multiple copies of the transmitted signal (See Figure 4.1). Since different paths have different distances and path losses, signal copies travel on multiple paths typically arrive at the receiver at different times and with different attenuations [65]. The sum of those signal copies forms the received signal. For the sake of presentation, we refer to a signal copy that travels along one path as a *multipath component*. For example, in Figure 4.1, signals s_1, s_2, s_3, and s_4 are multipath components.

Multipath effect might be reduced by using directional antennas. However, directional antennas usually cannot provide perfect laser-like radio signals. For example, the beamwidth of a 3-element Yagi Antenna, the most common type of directional antennas, is 90 degrees in the vertical plane and 54 degrees in the horizontal plane [49]. Thus, it is in general hard to completely eliminate multipath effect. For long distance transmission, the amount of multipath effect seen by a receiver may be much more due to the reduced focusing power at the receiver [4].

Note that a multipath component herein refers to a resolvable multipath component (i.e., the arrival of a multipath component does not interfere with that of its subsequent multipath component). Figure 4.2 is an example that shows the difference between resolvable and non-resolvable multipath components.

A radio channel consists of multiple paths from a transmitter to a receiver, and each path of the channel has a response (e.g., distortion and attenuation) to the multipath component traveling on it [65]. For convenience, we call the response to each multipath component a *component response*. Essen-

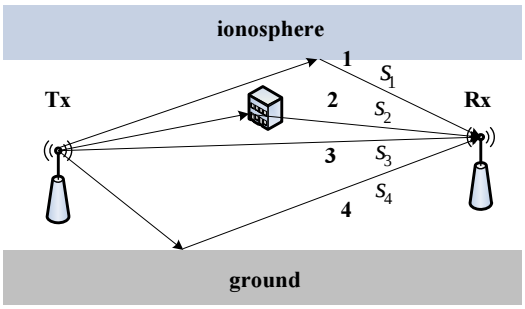

Figure 4.1: Example of a multipath effect. The wireless signal sent by transmitter Tx is reflected by the ionosphere, a building, and the ground. Thus, radio waves propagate over paths 1, 2, 3, and 4. The receiver Rx receives signal copies s_1, s_2, s_3, and s_4 from paths 1, 2, 3, and 4, and the received signal is the sum of all signal copies.

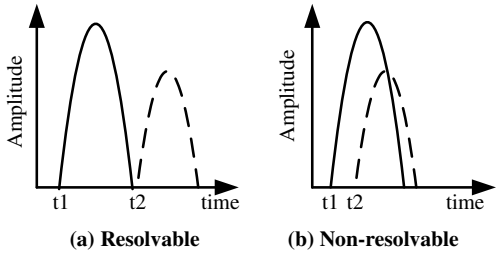

Figure 4.2: Resolvable and non-resolvable multipath components. In (a), the arrivals of two multipath components do not interfere with each other. Therefore, they are resolvable. In (b), the arrival of the second multipath component interferes with that of the first multipath component. Therefore, they are non-resolvable.

tially, the *channel impulse response* is formed by the superposition of many component responses, each representing a single path [65]. Therefore, the channel impulse response, denoted by $h(\tau)$, is given by

$$h(\tau) = \sum_{l=1}^{L} a_l e^{j\phi_l} \delta(\tau - \tau_l), \tag{4.1}$$

where L is the total number of multipaths, $\delta(\tau)$ is the Dirac delta function, and a_l, ϕ_l, and τ_l are the channel gain, the phase, and the time delay of the l-th multipath component, respectively [65].

If a transmitter moves from one place to another, the multiple paths from the transmitter to the receiver change, and thus the channel impulse response also changes. As a result, the channel impulse responses can be used to determine whether the transmitter changes its location or not. A channel impulse response is referred to as a *link signature* [65]. A location distinction algorithm using link signatures has been proposed in [65]. Specifically, a history of n link signatures are measured and stored while the transmitter is not moving. For a newly measured link signature, the receiver computes the distance between the newly measured link signature and the historical link signatures. If the distance is larger than a threshold, then a location change is detected.

4.2 Assumptions and Threat Model

Our system consists of primary users and secondary users. A primary user is assumed to be at a fixed location (e.g., a TV broadcast tower) [17]. As stated by FCC, TV stations and radio infrastructures should maintain physical security through a combination of security personnel, card restricted access, video surveillance, and other methods [81]. Thus, we assume that primary users are physically protected and any unauthorized entity cannot be physically close to a primary user due to those physical protection methods. We assume that secondary users are equipped with wireless radio devices and are allowed to transmit signals on the channels allocated to primary users only when the primary users are not transmitting.

We assume that an attacker's objective is to prevent other secondary users from using the primary users' channel and get an unfair share of the bandwidth when the primary users are not transmitting. Jamming attacks, which affect other users as well as the attackers themselves, are thus not in the scope of this work.

We assume that attackers can mimic a primary user's signal and inject their fake signals into the primary user's channel. We assume that an attacker has the following capabilities: (1) He knows the signal feature of a primary user and is able to generate a fake signal with the same feature. (2) He can transmit signals on the a primary user's channel to mislead the primary user detection process at secondary users. (3) He has a large maximum transmit power that can be several times of that of a primary user. However, we assume that an attacker cannot be physically close to a primary user due to

the physical protection.

We assume all secondary users have reliable ways to obtain the public key of each helper node, and an attacker cannot compromise the helper node.

4.3 Overview

Our goal is to provide secondary users with the ability to determine whether a received signal is from a primary user or not in the presence of attackers. One possibility is to use the link signature of the received signal. However, as discussed in the Introduction, it is non-trivial for any secondary user to obtain the historical link signatures of a primary user in an authenticated way, given FCC's restriction on (no modification of) primary users.

We develop a novel approach that integrates traditional cryptographic signatures and link signatures to enable primary user detection in the presence of attackers. Specifically, we propose to place a *helper node* close (and physically bound) to each primary user. Given the FCC requirement on the physical security of primary users such as TV stations, such helper nodes can also be physically protected. Though we cannot modify any primary user due to the FCC constraint, we do have the flexibility to put necessary mechanisms on each helper node, including the use of cryptographic signatures. Moreover, since each helper node is placed physically close to the primary user, their link signatures observed by a secondary user are very similar to each other.

To enable secondary users to authenticate signals from a primary user, we propose to use the helper node associated with the primary user as a "bridge". Specifically, we propose to have the helper node transmit messages when the target channel is vacant. These messages include cryptographic signatures, which will allow secondary users to verify their authenticity. As a result, secondary users can authenticate messages from the help node, then obtain the helper node's authentic link signatures, and finally verify the primary user's link signatures using those learned from the helper node. Note that our approach does not require any change to primary users, and thus follows the FCC constraint properly.

Issues of spacing multiple independent radio wave transmitters very close to each other (e.g., on the same mast) have been explored and demonstrated feasible [7, 51]. These techniques can be readily adopted to facilitate the deployment of helper nodes close to primary users in CRNs.

For the sake of presentation, we focus our discussion on one primary user and its associated helper node. However, all discussion in this dissertation applies to the situations where there are multiple primary users and helper nodes, as long as the association of each primary user and its helper node is clear.

4.3.1 Technical Challenges

Two technical problems need to be resolved to make the proposed approach work. First, the helper node has to have a reliable way to detect primary user's signals. In particular, the attacker may target at the helper node. Note that the proposed approach requires that the helper node transmit messages to secondary users so that the secondary users can obtain valid *training link signatures*. However, the attacker may pretend to be the primary user and inject fake signals into the target channel. This can effectively stop the helper node, and the proposed approach will fail. Thus, it is critical for the helper node to distinguish signals from the primary users and those from the attacker.

At first glance, this seems to be the same problem as what we are trying to solve, and thus put us in a "chicken-first or egg-first" situation. However, we will show that this is not the case due to the proximity of the helper node to the primary user. We will develop a novel physical-layer authentication technique to enable the helper node to properly authenticate messages from the primary users *without using any training link signatures*. This is dramatically different from traditional link signatures, where training is a necessary part of the scheme. The details will be discussed in Section 4.4.

Second, the interaction between the helper node and secondary users must be properly protected with lightweight mechanisms. In particular, the integration of cryptographic signatures and link signatures is a critical component of the proposed approach, and must be done efficiently. Moreover, there has to be a mechanism to prevent the attacker from replaying messages originally sent by the helper node. Otherwise, the attacker may simply reuse the valid cryptographic signatures to mislead secondary users into accepting invalid training link signatures. We will discuss critical design issues for the protocol between the helper node and a secondary user in Section 4.5.

4.4 Authenticating Primary User's Signal at the Helper Node

As discussed earlier, the helper node transmits signals using the channels allocated to its primary user such that secondary users can "learn" the link signatures of the primary user. To avoid interfering with the transmission of the primary user, the helper node transmits signals to secondary users only when the primary user is not transmitting. Therefore, the helper node should first sense the channel to decide whether the primary user is transmitting.

Unfortunately, the helper node cannot simply employ traditional primary detection approaches to determine the presence of the primary user's signal, since the attacker may mimic the primary user's signal and inject fake signals into the target channels.

In this section, we propose a novel physical-layer authentication approach that enables the helper node to authenticate the primary user's signal *without using any training link signatures*. Intuitively, the multipath effect exhibited by the primary user's signal and observed by the helper node has some unique properties, since the primary user is very close to the helper node. In our approach, we utilize

such unique multipath effect to enable the helper node to distinguish the primary user's signal from those transmitted by attackers.

In the following, we first give our observation behind our new technique, then describe the proposed authentication approach, and analyze the effectiveness of the proposed approach.

4.4.1 Observation

Ideally, signal strength decreases as the signal propagates farther away from the transmitter. A short propagation path results in a large received signal amplitude, whereas a long propagation path leads to a small received signal amplitude.

Assume that there is no obstacle between the primary user and the helper node. The primary user is close to the helper node. This means the first received multipath component travels on a very short path, which is a straight line between the primary user and the helper node. Unlike the first received multipath component, the second received multipath component travels along a longer path. According to [40], the length of the path over which the second received (resolvable) multipath component travels should be larger than $\frac{c}{R}$, where c is the speed of light and R is the transmission rate.

If the distance between the primary user and the helper node is much smaller than $\frac{c}{R}$, then the amplitude of the first received multipath component is much larger than that of the second received multipath component. In other words, the amplitude ratio of the first received multipath component to that of the second received multipath component is a large number, as illustrated in Figure 4.3.

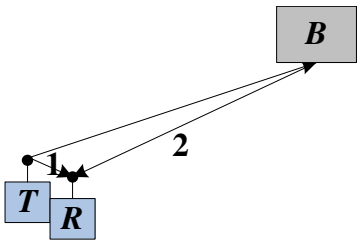

Figure 4.3: Amplitude ratio. T, R, and B is the primary user, the helper node, and an obstacle, respectively. The signal transmitted by T travels along two paths: path 1 ($T \rightarrow R$) and path 2 ($T \rightarrow B \rightarrow R$). Let $P1$ and $P2$ denote the amplitudes of the signal received from path 1 and path 2, respectively. The length of path 1 is much smaller than that of path 2, resulting in a large amplitude ratio $\frac{P_1}{P_2}$.

4.4.2 Authentication Method

Based on the above observation, we propose to use the amplitude ratio $r = \frac{P_1}{P_2}$ to authenticate the signal from the primary user, where P_1 and P_2 are the amplitude of the first and the second received multipath

components, respectively. For each newly received signal, the helper node computes the amplitude ratio r, and then compares r with a threshold w. If $r > w$, then the received signal is marked as the primary user's signal. Otherwise, the received signal is a suspicious signal that may have been sent by an attacker, and are discarded.

For the sake of presentation, we use r_a and r_p denote the amplitude ratio of the attacker' signal and that of the primary user's signal, respectively. We would like to point out that the values of r_a and r_p depend on the positions of obstacles. Due to the randomness and uncertainty of the surroundings, r_a (r_p) may not always be smaller (larger) than the pre-determined threshold w. Hence, we may have two types of possible errors: *false alarm* and *false negative*. With a false alarm, $r_p < w$, and thus the primary user's signal is incorrectly identified as the attacker's signal. With a false negative, $r_a > w$, and thus the attacker's signal is incorrectly identified as the primary user's signal.

In Sections 4.4.3 and 5.4, through both theoretical analysis and experiment evaluation, we will show that the probability of false alarm and the probability of false negative decrease quickly as the distance between the attacker and the helper node increases.

Computing the Amplitude Ratio

A helper node can first measure the channel impulse response of a received signal, and then calculate the amplitude ratio based on the measured channel impulse response. In Lemma 5, we show that the amplitude ratio of the first multipath component to the second multipath component indeed equals the amplitude ratio of h_1 to h_2, where h_1 and h_2 are the component responses for the first and the second multipath components, respectively.

Lemma 5 *Let s_1 and s_2 denote the first and the second received multipath components. The amplitude ratio r of s_1 to s_2 equals to that of h_1 to h_2, where h_1 and h_2 are the component responses for s_1 and s_2.*

Proof Recall that the channel impulse response $h(\tau)$ is $h(\tau) = \sum_{l=1}^{L} a_l e^{j\phi_l}\delta(\tau - \tau_l)$. Assume the first and the second multipath component arrives at time τ_1 and τ_2. Thus, the component responses h_1 and h_2 for the first and the second multipath components are: $h_1 = h(\tau_1) = a_1 e^{j\phi_1}\delta(0)$ and $h_2 = h(\tau_2) = a_2 e^{j\phi_2}\delta(0)$. According to [39], the amplitude ratio of h_1 and h_2 can be transformed as follows:

$$\frac{\|h_1\|}{\|h_2\|} = \frac{\|a_1 e^{j\phi_1}\delta(0)\|}{\|a_2 e^{j\phi_2}\delta(0)\|} = \frac{\|a_1(\cos\phi_1 + i\sin\phi_1)\|}{\|a_2(\cos\phi_2 + i\sin\phi_2)\|} = \frac{\|a_1\|}{\|a_2\|}$$

The channel gain a_l of the l-th multipath component is $a_l = \frac{s_l}{s_t}$, where s_l and s_t is the l-th received multipath component and the transmitted signal [39]. Therefore,

$$\frac{\|h_1\|}{\|h_2\|} = \frac{\|a_1\|}{\|a_2\|} = \frac{\|s_1\|}{\|s_2\|} = r.$$

□

Figure 4.4 shows a channel impulse responses obtained from the CRAWDAD data set [89], which contains over 9,300 channel impulse responses measured in an indoor environment with obstacles (e.g., cubicle offices and furniture) and scatters (e.g., windows and doors). The second multipath component arrives at the receiver about 100 microseconds after the arrival of the first one. Each multipath component leads to a triangle in shape with a peak (i.e., the component response) [65], and the helper node can use the first and the second peaks as $\|h_1\|$ and $\|h_2\|$ to compute the ratio r.

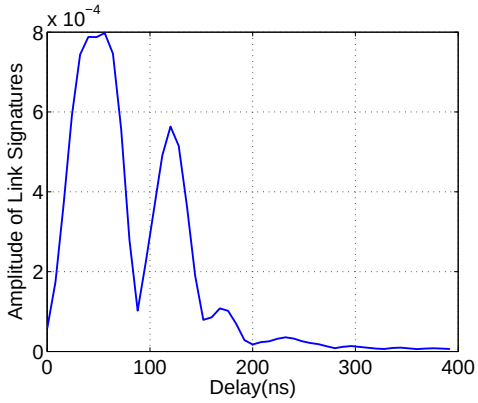

Figure 4.4: Computing the ratio r. This graph plots the amplitudes of a real measured channel impulse response (i.e., link signature) obtained from CRAWDAD for a 2.4 GHz channel, and $\|h_1\|$ and $\|h_2\|$ corresponds the first and the second rounded peak. Therefore, $\|h_1\| \approx 0.82 \times 10^{-3}$, $\|h_2\| \approx 0.55 \times 10^{-3}$, and $r = \frac{\|h_1\|}{\|h_2\|} \approx 1.49$.

Real-world Examples

Figures 4.5 and 4.6 show two real-world examples of channel impulse responses obtained from the CRAWDAD data set [89]. In Figure 4.5, the receiver is positioned 13.77 meters away from the transmitter. We can see that the corresponding amplitude ratio of the first multipath component to that of the second one is about $\frac{4}{2} = 2$. In Figure 4.6, the receiver is moved to a closer location that is 1.45 meters away from the transmitter. Now the amplitude ratio becomes $\frac{7}{0.5} = 14$.

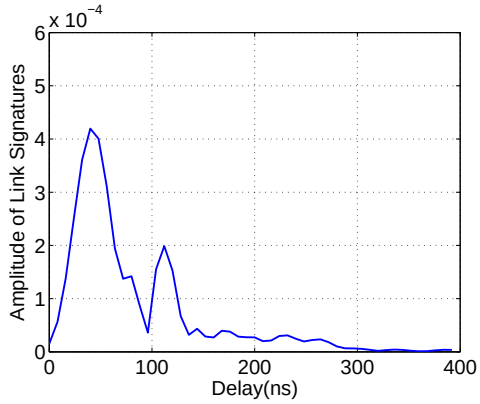

Figure 4.5: Example of amplitude ratio: The distance between the transmitter and the receiver is 13.77 meters, and the corresponding amplitude ratio is about about $\frac{4}{2} = 2$.

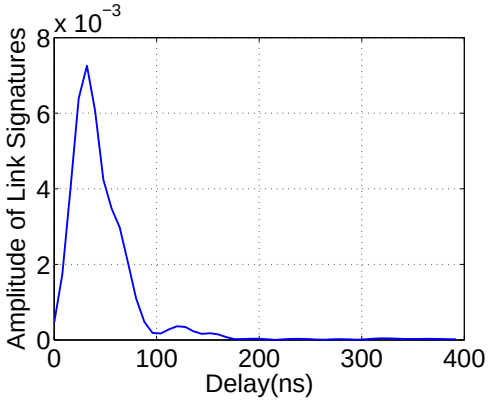

Figure 4.6: Example of amplitude ratio: The distance between the transmitter and the receiver is 1.45 meters, and the corresponding amplitude ratio is about $\frac{7}{0.5} = 14$.

4.4.3 Theoretical Analysis

In this section, we first give the mathematical model of the received signal amplitude, and then show the performance of the proposed authentication approach in terms of the probability of false negative (i.e., the attacker's signal is incorrectly identified as the primary user's signal) and the probability of false alarm (i.e., the primary user's signal is incorrectly identified as the attacker's signal).

Signal Amplitude Model

According to the simplified path loss model [39], the amplitude P_r of a received signal can be modeled as

$$P_r = \begin{cases} \sqrt{P_t k (\frac{d_0}{d})^{\gamma}} & d > d_0, \\ \sqrt{P_t k} & d \leq d_0, \end{cases} \tag{4.2}$$

where P_t is the transmit power, d is the length of the path along which the signal propagates from the transmitter to the receiver ($d > d_0$), k is a scaling factor whose value depends on the antenna characteristics and the average channel attenuation, d_0 is a reference distance for the antenna far-field, and γ is the path loss exponent. The values of k, d_0, and γ can be obtained either analytically or empirically [39].

Mathematical Analysis

We derive the probability of false negative and the probability of false alarm in Lemmas 6 and 7, respectively.

Lemma 6 *(Probability of false negative) Given a detection threshold w, the probability p_d that the attacker's signal is wrongly identified as the primary user's signal is*

$$p_d = \frac{1}{2}(1 - erf(\frac{10 \log \frac{(w^{\frac{2}{\gamma}} - 1)\sqrt{d}}{T_1 c}}{\sigma\sqrt{2}})), \tag{4.3}$$

where erf is the Error Function, d is the distance between the attacker and the helper node, c is the propagation speed of electromagnetic wave, and σ and T_1 are parameters that typically range between $2 - 6$dB and $0.1 - 1$ microsecond, respectively.

Proof: Let d_{a1} and d_{a2} be the lengths of the path along which the first and the second received multipath components of the attacker travels, respectively. Let P_{ra1} and P_{ra2} be the amplitudes of the first and the second multipath components, respectively. Assume $d_{a1} > d_0$ and $d_{a2} > d_0$. Thus, according to Equation 4.2, P_{ra1} and P_{ra1} can be approximated by

$$P_{ra1} = \sqrt{P_{ta} k (\frac{d_0}{d_{a1}})^{\gamma}},$$

$$P_{ra2} = \sqrt{P_{ta} k (\frac{d_0}{d_{a2}})^{\gamma}},$$

where P_{ta} is the transmit power of the attacker. Hence, the ratio r_a of P_{ra1} to P_{ra2} can be written as

$$r_a = \frac{\sqrt{P_{ta} k (\frac{d_0}{d_{a1}})^{\gamma}}}{\sqrt{P_{ta} k (\frac{d_0}{d_{a2}})^{\gamma}}} = \sqrt{(\frac{d_{a2}}{d_{a1}})^{\gamma}}.$$

The attacker's signal is wrongly identified as the primary user's signal if $r_a \geq w$. Thus, $p_d = 1 - \mathbb{P}(r_a < w)$. Let t_a denote the time at which the attacker's signal starts to propagate to the helper node. Let t_{a1} and t_{a2} denote the arrival times of the first and the second multipath components of the attacker, respectively. Therefore, $d_{a1} = (t_{a1} - t_a)c$ and $d_{a2} = (t_{a2} - t_a)c$, and we can have the following:

$$\begin{aligned} d_{a2} &= (t_{a2} - t_a)c \\ &= (t_{a1} - t_a)c + (t_{a2} - t_{a1})c = d_{a1} + \Delta c, \end{aligned}$$

where $\Delta = t_{a2} - t_{a1}$. According to [42], for urban, suburban, and rural areas, Δ can be statistically modeled as

$$\Delta = T_1 \sqrt{d} y,$$

where T_1 is the median value of Δ when $d = 1000$m (T_1 typically ranges from $0.1 - 1$ microsecond), and y is a lognormal variate. Specifically, $Y = 10 \log y$ is a Gaussian random with zero mean and a standard deviation that lies between $2 - 6$dB. The model parameters and their values can be found in Table III of [42]. Assume the first received multipath component travels along the straight line between the attacker and the helper node. Thus, $d_{a1} = d$ and

$$d_{a2} = (d + \Delta c) = d + T_1 \sqrt{d} y c$$

Therefore,

$$r_a = \sqrt{(\frac{d_{a2}}{d_{a1}})^{\gamma}} = \sqrt{(\frac{d + T_1 \sqrt{d} y c}{d})^{\gamma}}$$

Recall that $Y = 10\log y$ is a Gaussian random with zero mean. Let σ denote the deviation for Y. Thus,

$$\begin{aligned} p_d &= \mathbb{P}(r_a \geq w) = 1 - \mathbb{P}(r_a < w) \\ &= 1 - \mathbb{P}(\sqrt{(\frac{d + T_1\sqrt{d}yc}{d})^{\gamma}} < w) \\ &= 1 - \mathbb{P}(Y < 10\log\frac{(w^{\frac{2}{\gamma}} - 1)\sqrt{d}}{T_1 c}) \\ &= \frac{1}{2}(1 - erf(\frac{10\log\frac{(w^{\frac{2}{\gamma}}-1)\sqrt{d}}{T_1 c}}{\sigma\sqrt{2}})) \end{aligned}$$

□

Lemma 7 *(Probability of false alarm) Given a detection threshold w, the probability p_f that the primary user's signal is wrongly identified as the attacker's signal is*

$$p_f = \frac{1}{2}(1 + erf(\frac{10\log\frac{(w^{\frac{2}{\gamma}}-1)\sqrt{d_{p1}}}{T_1 c}}{\sigma\sqrt{2}})). \tag{4.4}$$

Proof: Let d_{p1} and d_{p2} be the path lengths corresponding to the first multipath component and the second multipath component of the primary user, respectively. Note that the helper node and the primary user are very close to each other. Thus, we assume that $d_{p1} \leq d_0$. Similar to d_{a1}, we assume that $d_{p2} > d_0$. Let P_{rp1} and P_{rp2} be the amplitudes of the first and the second multipath components of the primary user, respectively. According to Equation 4.2, P_{rp1} and P_{rp2} can be modeled by

$$P_{rp1} = \sqrt{P_{tp}k},$$

$$P_{rp2} = \sqrt{P_{tp}k(\frac{d_0}{d_{p2}})^{\gamma}},$$

where P_{tp} is the transmit power of the primary user. Hence, the ratio r_p of P_{rp1} to P_{rp2} can be written as

$$r_p = \frac{\sqrt{P_{tp}k}}{\sqrt{P_{tp}k(\frac{d_0}{d_{p2}})^{\gamma}}} = \sqrt{(\frac{d_{p2}}{d_0})^{\gamma}}.$$

The primary user's signal is wrongly identified as an attacker's signal if $r_p < w$. Thus, the probability p_f that the primary user's signal is rejected is $\mathbb{P}(r_p < w)$. Let t_p denote

the time at which the primary user's signal starts to propagate to the helper node. Let t_{p1} and t_{p2} denote the arrival times of the first and the second multipath components of the primary user, respectively.

Therefore,

$$d_{p2} = (t_{p2} - t_p)c = d_{p1} + T_1\sqrt{d_{p1}}yc.$$

Without loss of generality, we assume $d_{p1} = d_0$ to simplify the calculation, and thus obtain

$$r_p = \sqrt{(\frac{d_{p2}}{d_0})^\gamma} = \sqrt{(1 + \frac{T_1 yc}{\sqrt{d_{p1}}})^\gamma}.$$

Thus, we can obtain the probability p_f that the primary user's signal is wrongly identified as the attacker's signal, and

$$p_f = \mathbb{P}(r_p < w) = \mathbb{P}(\sqrt{(1 + \frac{T_1 yc}{\sqrt{d_{p1}}})^\gamma} < w)$$
$$= \frac{1}{2}(1 + erf(\frac{10\log\frac{(w^{\frac{2}{\gamma}}-1)\sqrt{d_{p1}}}{T_1 c}}{\sigma\sqrt{2}})).$$

□

Determining the Threshold w

The threshold w can be determined based on the requirement for the probability of false negative p_d or the probability of false alarm p_f. For practical applications, the IEEE 802.22 standard suggests both probabilities of false negative and false alarm be less than 0.1 in terms of detecting primary users [24]. Herein, we assume a stricter requirement that $p_d \leq 0.05$, and thus

$$p_d = \frac{1}{2}(1 - erf(\frac{10\log\frac{(w^{\frac{2}{\gamma}}-1)\sqrt{d}}{T_1 c}}{\sigma\sqrt{2}})) \leq 0.05$$

By treating w as an unknown and solve the inequality, we can get that

$$w \geq \sqrt{(1 + \frac{T_1 c \times 10^{0.11\times\sqrt{2}\sigma}}{\sqrt{d}})^\gamma}. \tag{4.5}$$

Although the helper node does not know the actual distance d between itself and the attacker, the helper node can estimate the minimum distance d_{min} from the attacker to him/her based on the physical protection policy and the approaches he/she uses. Let

$$w_{min} = \sqrt{(1 + \frac{T_1 c \times 10^{0.11\times\sqrt{2}\sigma}}{\sqrt{d_{min}}})^\gamma}.$$

w_{min} can be used as the threshold w, since Equation (4.5) holds when $w = w_{min}$. Note that the primary user and the helper node are very close to each other. Thus, we substitute $d_{p1} = 1$ and $w = w_{min}$ into Equation (4.4) and we get

$$p_f = \frac{1}{2}(1 + erf(\frac{10 \log \frac{10^{0.11 \times \sqrt{2}\sigma}}{\sqrt{d_{min}}}}{\sigma\sqrt{2}})).$$

Figure 4.7 shows that the probability p_f of false alarm decreases dramatically as the minimum distance d_{min} from the attacker to the helper increases. In particular, if the minimum distance is larger than 90 meters, the probability of false alarm is smaller than 0.05 for a constant 0.05 probability of false negative.

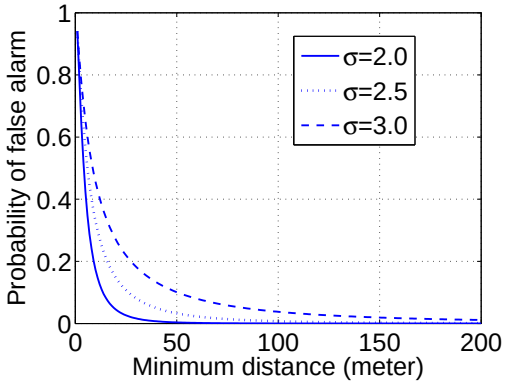

Figure 4.7: Probability of false alarm vs minimum distance from the attacker to the helper node for a constant 0.05 probability of false negative.

If we assume that $p_f < 0.05$, we can use the same method to get the threshold w and the corresponding probability p_d of false negative:

$$w = \sqrt{(1 + T_1 c \times 10^{-0.11 \times \sqrt{2}\sigma})^{\gamma}}.$$

$$p_d = \frac{1}{2}(1 - erf(\frac{10 \log 10^{-0.11 \times \sqrt{2}\sigma}\sqrt{d}}{\sigma\sqrt{2}})).$$

Figure 4.8 shows the probability of false negative for a constant 0.05 probability of false alarm.

Figure 4.9 displays the tradeoff between the probability of false alarm and the probability of false negative, when $\sigma = 2.5$ and the minimum distance between the attacker and the helper node is 50, 60, and 70 meters, respectively.

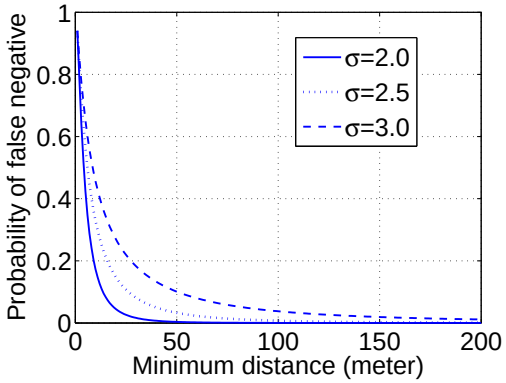

Figure 4.8: Probability of false negative vs minimum distance from the attacker to the helper node for a constant 0.05 probability of false alarm.

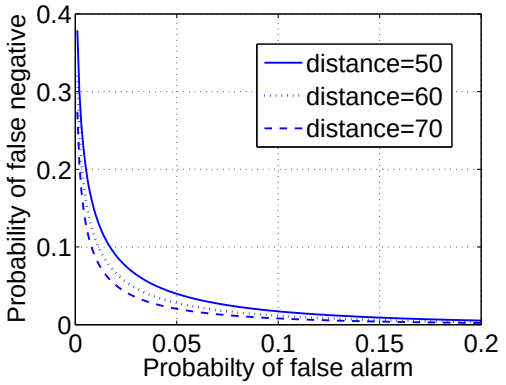

Figure 4.9: Tradeoff between probability of false alarm and the probability of false negative.

4.5 Interaction between the Helper Node and Secondary Users

Intuitively, a helper node can notify secondary users if the channel is open to them, since the helper node itself has the ability to authenticate a primary user's signal. However, we utilize link signatures to let secondary users identify a primary user's signal in a proactive way even when the helper node is sleeping.

The objective of having a secondary user interact with the helper node is to allow the secondary user to learn valid link signature from the helper node. Thus, the interaction between the secondary user and the helper node can be considered as a *training process*, during which the secondary user collects enough valid link signatures that could be used to verify future signals from the primary user. For convenience, we refer to the link signatures collected during the training process as *training link signatures*, and the packets from the helper node as *training packets*.

Note that the helper node is not required to transmit training packets all the time, and the training process may be triggered periodically or at the requests of secondary users. Our scheme allows the helper node to sleep during non-training period (e.g., the time interval between the end of a training process and the beginning of the subsequent training process). However, secondary users can still work in a proactive way even when the helper node is sleeping. As a result, the probability of interfering the transmission of the primary user is reduced. With training link signatures acquired in training processes, secondary users can directly verify whether a newly received signal is from the primary user or not.

4.5.1 Obtaining Training Link Signatures

We assume that the helper node is able to deliver training packets to secondary users. For example, the helper node may periodically sense the channel and broadcast training packets to all secondary users if the channel is open. Alternatively, we may use a request/reply protocol between secondary users and the helper node. In other words, if a secondary user does not have enough training link signatures, it sends a request to the helper node through the control channel, and the helper node then transmits training packets back upon request. Our approach is independent of the exact way training packets are triggered.

Upon receiving a packet from the helper node, a secondary user measures the link signature and verifies the cryptographic signature in the received packet. If the cryptographic signature is valid, the secondary user accepts the corresponding link signature. Otherwise, the secondary user has to discard both the link signature and the received packet.

It is well-known that public key cryptographic signatures are expensive to generate and verify. A straightforward application of cryptographic signatures will lead to substantial overheads on the helper node as well as secondary nodes. To enable efficient interaction between the helper node and a secondary user, we propose to amortize the signature generation and verification costs on both helper node and secondary users.

Note that there are known ways for signature amortization using cryptographic hash functions (e.g.,

[66, 88]). Thus, we consider our contribution here secondary (compared with the authentication method in Section 4.4).

Amortizing Cryptographic Signature Costs: The helper node randomly picks a number r_l and uses a one-way cryptographic hash function H to generates a one-way hash chain $r_0 \leftarrow r_1 \leftarrow ... \leftarrow r_l$, where $r_{i-1} = H(r_i)$ for $1 \leq i \leq l$. It is well-known that given an authenticated value r_i in this hash chain, it is easy to authenticate any later value r_j ($i < j < l$). However, it is computationally infeasible to derive any later r_j ($i < j < l$) if no value beyond r_i is known.

To reduce the signature cost, for each hash chain, the helper node generates one and only one cryptographic signature on r_0 using its private key. Let $sig(r_0)$ denote the signature. Suppose the helper node needs to authenticate the i-th packet since the generation of the hash chain. The helper node then place r_0, $sig(r_0)$, i, and r_i in the packet. (The helper node should certainly start with $i = 0$.) Thus, the helper node never needs to generate another signature for this hash chain.

Consider a secondary node that receives a packet using the above hash chain from the helper node for the first time. Note that i could be greater than 0 if this secondary node has not received any packet from the helper node recently. The secondary node then first verifies the signature $sig(r_0)$. If $i = 0$, the secondary node has successfully verified the cryptographic signature from the helper node. However, if $i > 0$, the secondary node needs to future hash r_i for i times and compare $H^{<i>}(r_i)$ with r_0. If they match, the packet is also valid. In any case, the secondary node should save r_i for future authentication.

If the secondary node has received and verified a signature from the helper node with the same hash chain previously, it must have saved an authenticated hash value r_j ($j < i$). As a result, the secondary node does not have to verify the signature $sig(r_0)$ again. Instead, it only needs to compute $H^{<i-j>}(r_j)$ and compare the result with r_i. A match indicates a successful authentication of the packet.

As we can see, the helper node needs to generate one and only one cryptographic signature for each hash chain. Similarly, each secondary node only needs to verify one cryptographic signature once for each hash chain. Thus, this amortization approach can greatly reduce the computational overheads on both the helper node and the secondary node.

Defending against Replay Attacks: As discussed earlier, a critical threat is that the attacker may replay intercepted training packets from a valid helper node at its own location. As a result, the attacker can convince secondary users to accept the attacker's link signatures as training link signatures. Since the secondary users are not guaranteed to have received the original transmission, traditional anti-replay mechanisms such as sequence numbers, which are intended for detecting replayed packet contents (rather than replayed signals), will not work.

Fortunately, there are multiple known techniques to handle replayed signals in wireless networks, such as the hardware-based, authenticated Medium Access Control (MAC) layer timestamping [94] and the method for detecting wireless signals tunneled by a malicious node [54]. These techniques can be adopted in CRNs to enable a secondary node and the helper node to detect replayed training packets.

Alternatively, we may take advantage of potentially synchronized clocks between valid secondary

users and the helper node to defend against such threats. According to IEEE 802.22 standard [27], secondary users and base stations are "required to use satellite based geo-location technology, which will also facilitate synchronization among neighboring networks by providing a global time source." We can assign each value in the above hash chain to a specific point in time. These times can be pre-scheduled such that all secondary users know when each hash value should be used. The helper node then transmits each hash value at the pre-scheduled point in time, provided that the primary user is not using the channel. When a secondary user receives a training packet, it can use its local time and the pre-scheduled time to estimate the transmission time of this packet. An overly long transmission time indicates that the packet has been replayed by the attacker.

Learning Training Link Signatures: To compute the training link signature, the secondary user samples the received signal using an A/D sampler, stores the first $\kappa + 1$ samples in a buffer, and demodulates the samples of the received signal into a packet. If the packet can pass authentication, the secondary user computes the link signature of the packet using the stored $\kappa + 1$ samples. Otherwise, the secondary user discards the stored samples. The secondary user typically needs to obtain a series of training link signatures for verifying future signals.

The methodology proposed in [65] can be used to compute the link signature of the stored $\kappa + 1$ samples. Let $\mathbf{r} = [r(0), ..., r(\kappa T_r)]$ denote the samples of the received signal $r(t)$, where T_r is the sampling rate. Based on the demodulation results, the secondary user can recreate the transmitted signal $s(t)$. Let $\mathbf{s} = [s(0), ..., s(\kappa T_r)]$ denote the corresponding $\kappa + 1$ samples of the transmitted signal $s(t)$. Let $R(iT_r)$ and $S(iT_r)$ be the discrete Fourier transform of $r(iT_r)$ and $s(iT_r)$, respectively. According to [65], the link signature $\mathbf{h} = [h(0), ..., h(\kappa T_r)]$, which are the $\kappa+1$ samples of $h(t)$, can be calculated as

$$h(iT_r) = \frac{1}{\mathcal{P}_s} F^{-1}(S^*(iT_r)R(iT_r)),$$

where $F^{-1}(\cdot)$ denote the inverse discrete Fourier transform, $S^*(iT_r)$ is the complex conjugate of $S(iT_r)$, and $\mathcal{P}_s = S^*(iT_r) * S(iT_r)$.

4.5.2 Verifying Link Signatures

For a newly received signal s_N, the secondary user first measures its link signature, which is denoted by $\mathbf{h}^{(N)}$, and then use training link signatures to verify $\mathbf{h}^{(N)}$.

Let $\mathcal{H} = \{\mathbf{h}^{(n)}\}_{n=1}^{N-1}$ denote the set of training link signatures, where $\mathbf{h}^{(n)}$ is the link signature measured from the i-th received training packet. The secondary user can verify whether s_N is transmitted by the primary user or not using the location distinction algorithm proposed in [65]. Specifically, the secondary user calculates the distance (i.e., difference) between $\mathbf{h}^N$ and the training set $\mathcal{H}$, and then compares the distance with a threshold. If the distance is less than a threshold, s_N is marked as the primary user's signal. Otherwise, s_N may be sent by the attacker and the secondary user ignores it. The method that can be used to calculate distance is discussed in [65].

4.6 Experimental Evaluation

Our approach involves two types of authentication: authentication of the primary user's signal at the helper node, and authentication of the primary user's signal at a secondary user. In this section, we report our experimental evaluation to show the effectiveness of both methods.

We validate the proposed authentication methods using the CRAWDAD data set [64], which includes over 9,300 real channel impulse response measurements (i.e., link signatures) in a 44-node wireless network [89]. There are $44 \times 43 = 1,892$ pairwise links between the nodes, and multiple measurements are provided for each link [89]. The map of the 44 node locations is shown in [65]. The measurement environment is an indoor environment with obstacles (e.g., cubicle offices and furniture) and scatters (e.g., windows and doors). More information regarding the CRAWDAD data set can be found in [64, 89].

4.6.1 Authentication at the Helper Node

To avoid interfering with the primary user's transmission, the helper node needs to first sense the channel, and verify whether a received signal is from the primary user. As discussed earlier, false alarms and false negatives may occur during the authentication process. Thus, we evaluate the performance of the authentication method in terms of the probability of false negative and the probability of false alarm.

Recall that during authentication the helper node computes the amplitude ratio of the first multipath component to the second multipath component for each received signal. If the amplitude ratio is larger than a threshold, then the received signal is considered from the primary user. Otherwise, it is considered from the attacker. Hence, false alarms happen when the primary user's amplitude ratio is less than the threshold, and false negative happens when the attacker's amplitude ratio is larger than the threshold.

Probability of False Alarm

To obtain the amplitude ratio of the primary user's signal, we perform experiments as follows. For $1 \leq i \leq 44$, we assume that node i is the helper node. For each of the remaining nodes, if there is no obstruction between itself and node i, we mark it as a *line-of-sight node*. Among all line-of-sight nodes for node i, we pick the one that is closest to node i as an approximation of the primary user p. Note that some nodes do not have line-of-sight nodes in their vicinities, and thus they are not used in our experiment (e.g., nodes 8 and 29 in the map shown by [65]). Finally, we compute the amplitude ratio using the primary user's channel impulse responses (i.e., link signatures of link (p, i)). The CRAWDAD data set has multiple measurements for each link. Thus, we can get multiple amplitude ratios for each link. We sort the collected amplitude ratios and compute empirical cumulative distribution function (CDF) for them. Let N denote the number of the collected amplitude ratios, $F(x)$ denote the empirical CDF, and $x_1, ..., x_N$ denote the sorted amplitude ratios, where $x_i \leq x_j$ for $1 \leq i \leq j \leq N$. The

empirical CDF $F(x_i)$ is given by $F(x_i) = \frac{n_{\leq x_i}}{N}$, where $n_{\leq x_i}$ is the number of amplitude ratios that are less than or equal to x_i.

Figure 4.10 shows the empirical CDF curve of the amplitude ratios computed using primary users' channel impulse responses. This CDF curve can be used to derive the probability of false alarm directly. For example, about 5% amplitude ratios are less than or equal to 5. Hence, if the threshold is set to 5, then 5% amplitude ratios are smaller than the threshold and the probability of false alarm is 0.05.

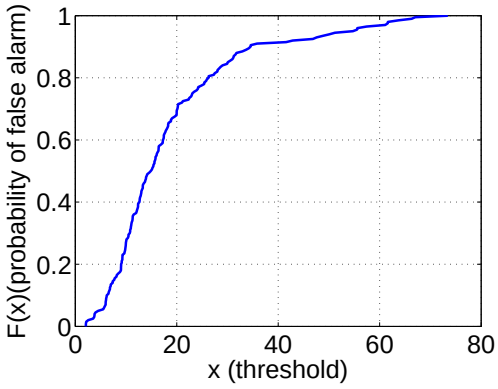

Figure 4.10: The empirical CDF curve of amplitude ratios computed using primary users' channel impulse responses

Probability of False Negative

We perform experiment to examine the amplitude ratios of attackers' signals. For $1 \leq i \leq 44$, we assume that node i is the helper node and find its primary user p using the same method as discussed in the above experiment. For each of the remaining nodes, we calculate the distance between this node and the helper node. We mark the node as the attacker if the calculated distance is larger than r times of the distance between the helper node and the primary user, where r is set to 2, 4, and 8 in our experiment. We compute the amplitude ratio for node i using the attacker's channel impulse responses (i.e., link signatures of link (a, i), where a is the node index of the attacker).

Figure 4.11 shows the empirical CDF curves of all amplitude ratios computed using attackers' channel impulse responses. In particular, about 95% amplitude ratios of attackers' signals are less than or equal to 5 for all possible values of r (i.e., $r = 2, 4, 8$). Based on the empirical CDF of the amplitude ratios, we generate Figure 4.14 to show the relationship between the probability of false negative and the threshold. For instance, the empirical CDF indicates that about 95% amplitude ratios are less than or equal to 5. Hence, about 5% amplitude ratios are larger than 5 and the probability of false negative is

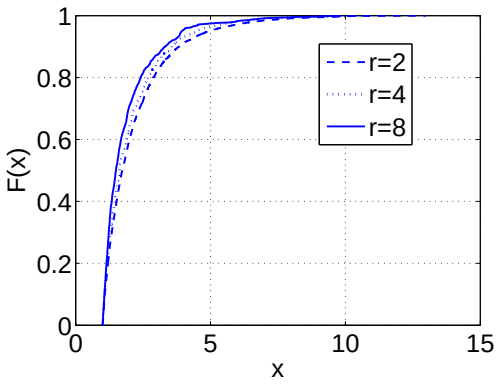

Figure 4.11: CDF curves of amplitude ratios computed using attackers' link signatures.

0.05 if the threshold is set to 5. It is shown in Figure 4.14 that the probability of false negative decreases as the distance between the attacker and the helper node increases (i.e., r gets larger).

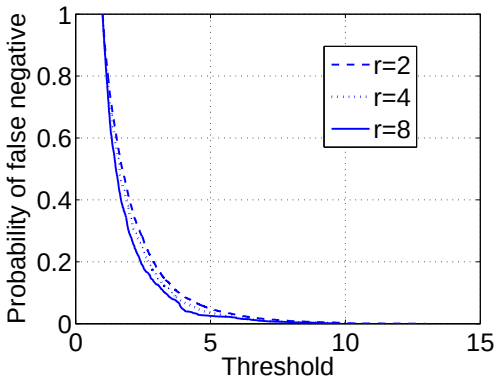

Figure 4.12: Probability of false negative vs threshold

Trade off between Probability of False Alarm and Probability of False Negative

Let P_{FA} and P_{FN} denote the probability of false alarm and false negative, respectively. We analyze the trade off between P_{FA} and P_{FN} by examining the relationship between P_{FA} and the threshold, as well as the relationship between P_{FN} and the threshold. For a particular value of threshold, the authentication approach would achieve a particular P_{FA} and P_{FN}.

Table 4.1 shows the probability P_{FN} when the probability P_{FA} of false alarm ranges between 0.05 and 0.2. If $P_{FA} = 0.05$, P_{FN} is less than 0.0655, 0.0486, and 0.0321 for $r = 2$, 4, and 8, respectively.

Table 4.1: Trade off between P_{FA} and P_{FN}

P_{FA}	P_{FN} (r=2)	P_{FN} (r=4)	P_{FN} (r=8)
0.05	≤ 0.0655	≤ 0.0486	≤ 0.0321
0.1	≤ 0.0248	≤ 0.0163	≤ 0.0155
0.15	≤ 0.0109	≤ 0.0070	≤ 0.0066
0.2	≤ 0.0053	≤ 0.0032	≤ 0.0022

For a constant P_{FA}, P_{FN} decreases as the distance between the attacker and the helper node increases (i.e., r increases). In particular, $P_{FN} = 0.0655$ when the distance between the attacker and the helper node is larger than twice of the distance between the primary user and the helper node (i.e., $r = 2$). However, P_{FN} falls to 0.0321 when the distance between the attacker and the helper node is 8 times larger than the distance between the primary user and the helper node (i.e., $r = 8$).

4.6.2 Authentication at Secondary Users

During the authentication process at a secondary user, the secondary user needs to verify whether a received signal is from the primary user or not by looking at the distance between the corresponding link signature and the training set. We refer to the distance as *link difference*. If the link difference is smaller than a threshold, then the received signal is considered from the primary user. Otherwise, the signal is considered to be sent by an attacker and the secondary user discards it.

Therefore, a false alarm happens if the link difference between the primary user's link signature and the secondary user's training set is larger than the threshold, and a false negative happens if the link difference between the attacker's link signature and the secondary user's training set is smaller than the threshold. Similar to the authentication at the helper node, we use the probability of false alarm and the probability of false negative to measure the performance of the proposed approach.

In our experiment, we compute the link differences between the primary user's link signature and the secondary user's training set, as well as the link differences between the attacker's link signature and the secondary user's training set. Based on their statistical distributions, we examine how likely false alarms and false negatives would happen.

Probability of False Alarm

To get the link differences between link signatures of the primary user and the secondary user's training set, we perform experiment as follows. We pick all nodes one by one as the primary user. Starting with node 1, we use the node closest to node 1 to approximate the helper node (i.e., node 3 in the map [65]). We further assume that all the other nodes (i.e., node 2 and nodes 4-44 on the map [65]) are secondary users. For each secondary user s, we generate its training set using all link signatures of the node pair

$(3, s)$ (i.e., the helper node's link signatures) and k ($k = 0, 1, 2$) link signatures of the node pair $(1, s)$ (i.e., the primary user's link signatures). Then, we compute link differences $d_s^1, ..., d_s^{n_p}$ between the primary user's link signatures and the training set of node s, where n_p is the number of primary user's link signatures.

We use the average value of d_s^1,... , $d_s^{n_p}$ as the link differences between the link signatures of node 1 and the training sets of each secondary user s. Similarly, we assume that nodes 2,...,44 are primary users and perform the same process to get the link differences between the link signatures of nodes 2,...,44 and the training sets of the secondary users.

Figure 4.13 shows curves of the empirical CDFs for the collected link differences, where each training set contains all measured link signatures of a helper node, and k ($k = 0, 1, 2$) measured link signatures of a primary user. Almost all link differences are less than or equal to 10 when the training set only contains the link signatures of a helper node (i.e., $k = 0$). Once a primary user's link signature is added to the training set (i.e., $k = 1$), the link differences decreases dramatically. Figure 4.14 shows the relationship between the probability of false alarm and the threshold.

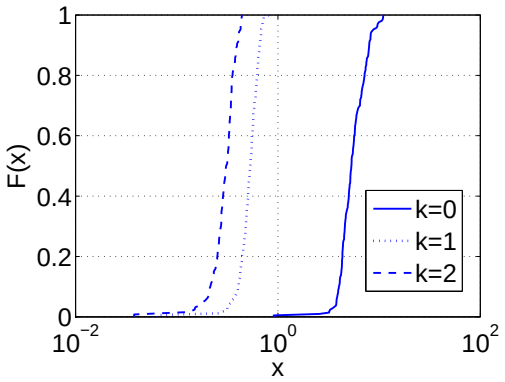

Figure 4.13: CDF curves of link differences between the link signatures of primary users and the training sets of secondary users.

Probability of False Negative

We also perform experiment to examine the link differences between link signatures of attackers and training sets of secondary users. We assume that node 1 is the attacker. We pick node p as the primary user and node s as the secondary user such that $p \neq s \neq 1$. For each combination of p and s, we first find the helper node of p. Let p_h denote the helper node. If $p_h \neq s \neq 1$, we generate the training set of s using the same approach as the first experiment. We then compute the link difference $d_{s,p}^1, ..., d_{s,p}^{n_a}$

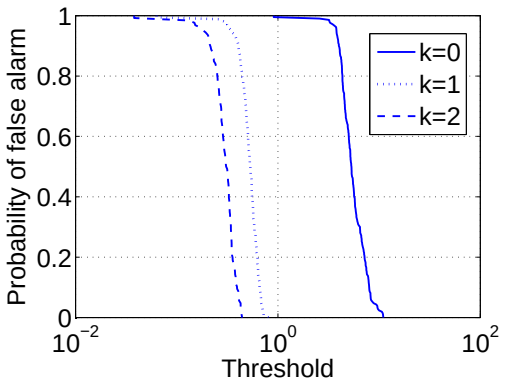

Figure 4.14: Probability of false alarm vs threshold

between the attacker's link signatures and the training set, where n_a is the number of attacker's link signatures. After scanning all combinations, we use the average value of $d^1_{s,p}$,..., the average value of $d^{n_a}_{s,p}$ as the link difference between link signatures of node 1 and the training sets of secondary users. Similarly, we assume that nodes 2,...,44 are attackers and perform the same process to get the link differences between the link signatures of nodes 2,...,44 and the training sets of secondary users.

Figure 4.15 shows the empirical CDF curves of the collected link differences for $k = 0$, $k = 1$, and $k = 2$. Note that the empirical CDF curves can be used to derive the probability of false negative directly given a threshold. For example, about 10% link differences are less than or equal to 7.5 when $k = 0$. This means the probability of false negative is 0.1 for a threshold of 2.5.

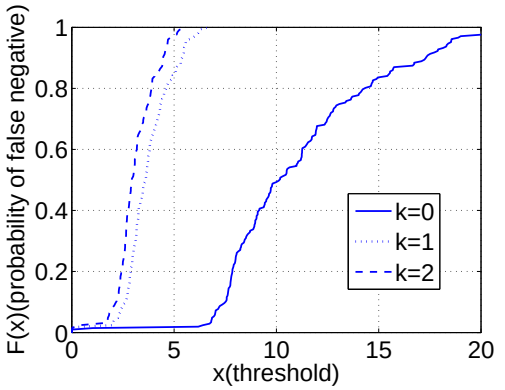

Figure 4.15: CDF curves of link differences between link signatures of attackers and the training sets of secondary users.

Table 4.2: Trade off between P_{FA} and P_{FN}: the probability P_D of false negative decreases as k increases

P_{FA} ($k = 0, 1, 2$)	P_{FN} (k=0)	P_{FN} (k=1)	P_{FN} (k=2)
0.05	≤ 0.3188	≤ 0.0241	≤ 0.0240
0.1	≤ 0.2319	≤ 0.0241	≤ 0.0240
0.15	≤ 0.1063	≤ 0.0241	≤ 0.0240
0.2	≤ 0.0821	≤ 0.0241	≤ 0.0240

Trade off between Probability of False Alarm and Probability of False Negative

We derive the trade off between the probability P_{FA} of false alarm and the probability P_{FN} of false negative by analyzing the relationship between P_{FA} (P_{FN}) and thresholds. Table 4.2 shows the result. To achieve a 0.05 probability of false alarm, the probability of false negative is less than 0.3188, which is actually a loose upper bound. In our experiment, we use the node closest to the primary user to approximate the helper node; there is indeed an unnecessarily long distance between the primary user and its helper node. Thus, the probability of false negative is unnecessarily large in our experiment.

Note that all P_{FN}'s are less than the same value 0.0241 for $k = 1$. This is because the threshold ranges between 0.6282 and 0.6816 when $0.05 \leq P_{FA} \leq 0.2$. This range is quite narrow, and we can only find a single P_{FN} from the empirical CDF of the attackers' link differences. Similarly, all P_{FN}s are less than the same value 0.0240 for $k = 2$.

4.7 Implementation

We demonstrate the feasibility of the proposed approach using a prototype implementation on Universal Software Radio Peripherals (USRPs) based on GNUradio [2]. Although wireless signals transmitted by USRPs may not exhibit the multipath properties due to low bandwidths, low power, and short range communication with USRPs, the prototype implementation nevertheless demonstrates the feasibility of integrating cryptographic signatures and link signatures for authenticating primary users' signals in CRNs.

A USRP is a radio frequency (RF) front end that has an analog to digital (AD) and a digital to analog (DA) converter, which can achieve an input and output sampling rate up to 64 Mb/s and 128 Mb/s, respectively. GNUradio is a software toolkit consisting of signal processing blocks that can be used to implement software radios on readily-available, low-cost external RF hardware and commodity processors (e.g., USRPs) [2].

We connect one USRP to a Lenovo X61 laptop (1.80 GHz Intel Core Duo CPU), and one USRP to a DELL machine (3.40 GHz Intel Pentium 4 CPU) via USB 2.0 links. Both computers are running Linux (Ubuntu 9.04) and GNUradio (version 3.2.2), and both USRPs employ XCVR2450 daughter boards

as transceivers. We implement the helper node and the secondary user applications using GNUradio toolkit, and install the helper node and the secondary user application on the laptop and the DELL machine, respectively.

The helper node application generates signed packets using the method described in Section 4.5.1, where we employed MD5 as the one-way function and RSA as the cryptographic signature algorithm. The signed packets are modulated into physical layer symbols by a differential binary phase-shift keying (DBPSK) modulator. Then all physical layer symbols enter a pulse shape filter, which transforms those symbols into baseband signals. The baseband signals are delivered to the USRP, converted into RF signals, and finally transmitted to the wireless channel through the antenna.

Upon capturing a RF signal, the secondary user application down-converts the RF signal into baseband signal. Then the baseband signal is recorded and delivered to a DBPSK demodulator. If the output of the demodulator can pass the verification, the secondary user reconstructs the transmitted signal from the demodulation output, and computes the 512-points complex Fourier transform F_1 and F_2 of the baseband signal and the transmitted signal, respectively. Finally, F_1 is multiplied by the conjugate of F_2, and the inverse Fourier transformation is used to calculate the link signature as described in the Appendix.

In our experiment, the packet length is 75 bytes, the bit rate is 2Mbit/s, and the carrier frequency is 5GHz. The laptop and the Dell machine are used as the transmitter and the receiver, respectively. We first put the transmitter about 5 meters away from the receiver, and let the transmitter send a signed packet to the receiver. Upon reception of the packet, the receiver verifies the cryptographic signature in the packet and measures the link signature. Then we move the transmitter to position a and position b, which is about 0.5 meter and 15 meters away from the old position, respectively. At both positions, we let the transmitter transmit signed packets to the receiver. Figure 4.16 displays the measured link signatures for different positions, we observe that the link signatures of the old position and position a are mixed together, and the link signature of position b greatly deviates from the mixed ones. This observation is consistent with our analytical result.

In our approach, generating signatures, verifying signatures, computing Fourier transform, and inverse Fourier transform are four major operations that are indispensable. To get an intuitive feeling of the computational overhead introduced by these operations, we did an experiment using the prototype system to test the computation time. We let the transmitter transmits 1,000 packets to the receiver every 0.1 second, and record the computation time by those operations.

Note that the transmitter (or the receiver) only needs to generate (or verify) the cryptographic signature $sig(r_0)$ in the first packet. For all the following packets, the transmitter signs them by simply appending $sig(r_0)$ and the corresponding hash values to them, and the receiver verifies them by computing and comparing hash values. Table 4.3 shows the time costs of signing (verifying) those packets. In practice, the calculation of link signatures can be performed more efficiently with Fourier transform implemented on special hardware (e.g., Virtex 2 Pro 50 Fast Fourier transform (FFT) core, which can

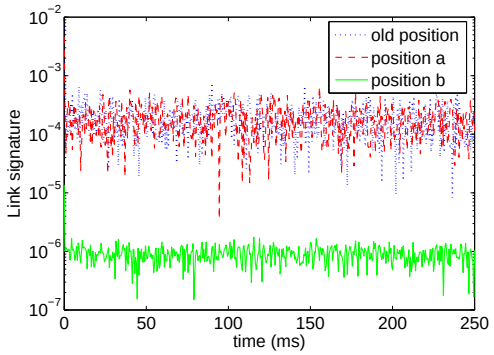

Figure 4.16: Measured link signatures for old position, position a, and position b

Table 4.3: Computation time (milliseconds)

Operations	Time range	Average
Signing	0.1699-0.0239	0.0441
Verification	0.4519-0.0781	0.1288
Fourier transform	1.4000-0.4200	0.5612
Inverse Fourier transform	0.7310-0.21901	0.2920

finish the 512 points complex Fourier transform with in less than 5.5 microseconds).

Chapter 5

Mimicry Attacks against Wireless Link Signature and Defense Design

5.1 Preliminaries

5.1.1 Multi-path Effect and Link Signature

Wireless signal usually propagates in the air along multiple paths due to reflection, diffraction, and scattering [65]. As a result, a receiver may receive multiple copies of the signal on different paths, each of which may have a different delay due to the path it traversed on. The received signal is the sum of these time delayed signal copies. Each path imposes a *response* (e.g., distortion and attenuation) on the signal traveling along it [65], and the superposition of all responses between two nodes is referred to as a *channel impulse response* [40].

The multi-path effects between different pairs of nodes are usually different, and so are the channel impulse responses [65]. Due to this reason, a channel impulse response between two nodes is also called a *link signature*, and has been proposed to provide robust location distinction and location-based authentication [65, 112]. Specifically, to determine if a received signal is from the desired location/channel of the transmitter, the receiver estimates the link signature of the received signal and compares it with *reference link signatures*, which are estimated when the receiver has known signals from the desired location/channel. The received signal is accepted only if the estimated link signature is similar to the references.

5.1.2 Link Signatures v.s. Cryptographic Signatures

Cryptographic and link signatures achieve different authentication purposes. Cryptographic authentication enables a receiver to verify if the message *content* is generated by the desired transmitter, whereas link signature enables a receiver to verify if the *signal* that carries the message is from the desired lo-

cation/channel. For example, sensors can be used to monitor wild fire and report to a remote receiver through wireless channel. An arsonist can burn the target area without being detected by moving the sensors to a different area. Cryptographic signatures cannot detect such attacks, but link signatures can alert the receiver since the signals are now transmitted on different channels due to location changes.

5.1.3 Estimating Channel Impulse Responses

Channel impulse responses are usually estimated using training sequences [77]. Specifically, the transmitter sends a training sequence (i.e., a sequence of bits) over the wireless channel, while the receiver uses the same training sequence and the corresponding received signal samples to estimate the channel impulse response. The training sequence can be pre-shared [77] or reconstructed from the received signal [65].

The physical layer channel estimation can be processed in either frequency domain (e.g. [65, 112]) or time domain (e.g., [77]), which are inter-convertible due to the linear relationship between the two domains. In the following, we describe the channel estimation method in the time domain.

Mathematical Formulation: The estimation of channel impulse responses exploits the (known) training sequence and the corresponding received samples. The transmitter converts the training sequence into M physical layer symbols (i.e., complex numbers that are transmission units at the physical layer [40]). This process is called modulation [40]. The transmitter then sends the M symbols to the wireless channel.

Let $\mathbf{x} = [x_1, x_2, ..., x_M]$ denote the transmitted symbols in the training sequence. Assume that there exist L paths. Thus, the receiver can receive L copies of $\mathbf{x}$, each traveling on one path and undergoing a response (i.e., distortion and attenuation) caused by the corresponding path. The vector $\mathbf{y}$ of received symbols is the convolution sum of the L copies of $\mathbf{x}$. Let $\mathbf{h} = [h_1, h_2, ..., h_L]^T$ be the channel impulse response, where h_i is the response of the i-th path. Assuming an additive white Gaussian noise (AWGN) channel, the received symbols $\mathbf{y}$ can be represented by [77]

$$\mathbf{y} = \mathbf{h} * \mathbf{x} + \mathbf{n}, \tag{5.1}$$

where $\mathbf{n}$ is the white Gaussian channel noise and $*$ is the convolution operator. The matrix form of

Equation (5.1) is

$$\mathbf{y} = \begin{bmatrix} x_1 & 0 & \cdot & 0 \\ x_2 & x_1 & \cdot & \cdot \\ \cdot & x_2 & \cdot & 0 \\ \cdot & \cdot & \cdot & x_1 \\ x_M & \cdot & \cdot & x_2 \\ 0 & x_M & \cdot & \cdot \\ \cdot & 0 & \cdot & \cdot \\ 0 & 0 & \cdot & x_M \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ \cdot \\ \cdot \\ h_L \end{bmatrix} + \mathbf{n} \tag{5.2}$$

Rewriting Equation (5.2) in a compact matrix form gives us

$$\mathbf{y} = \mathbf{X}\mathbf{h} + \mathbf{n}, \tag{5.3}$$

where $\mathbf{X}$ is a $(L+M-1) \times L$ Toeplitz matrix, containing L delayed versions of the transmitted symbols $\mathbf{x}$, and $\mathbf{y}$ is a vector consisting of $(L + M - 1)$ received symbols.

Estimation: Two types of estimators are generally used to estimate $\mathbf{h}$ from Equation (5.3): least-square (LS) estimator and linear minimum mean squared error (LMMSE) estimator [11]. If the statistical distribution of the channel impulse responses and noise are unknown, the LS estimator is usually used. If the statistical distribution of the channel impulse responses and noise are known, the LMMSE estimator is often used to exploit this information to decrease the estimation error [104].

For the LS estimator, the estimation result is given by $\hat{\mathbf{h}}_{LS} = (\mathbf{X}^H\mathbf{X})^{-1}\mathbf{X}^H\mathbf{y}$, where $\mathbf{X}^H$ is the conjugate transpose of $\mathbf{X}$ and $()^{-1}$ is the matrix inverse operation [83]. For the LMMSE estimator, the estimation result is $\hat{\mathbf{h}}_{LMMSE} = \mathbf{R}_h(\mathbf{R}_h + \sigma_n^2(\mathbf{X}\mathbf{X}^H)^{-1})^{-1}\hat{\mathbf{h}}_{LS}$, where $\mathbf{R}_h$ is the channel correlation matrix (i.e., the statistical expectation of $\mathbf{h}\mathbf{h}^H$) and σ_n^2 is the variance of the noise [32].

5.2 Mimicry Attack

In this section, we present the mimicry attack against link signatures. We focus on the attack against the scheme in [65] and then extend it to the schemes in [53] and [112].

The root cause of the mimicry attack is the linear relationship between the transmitted symbols $\mathbf{x}$, the received symbols $\mathbf{y}$, and the link signature $\mathbf{h}$, as indicated in Equation (5.3).

Let $\mathbf{y}_t$ and $\mathbf{y}_a$ denote the received symbols from the transmitter and the attacker, respectively. The attacker's goal in the mimicry attack is to make $\mathbf{y}_a$ approximately the same as $\mathbf{y}_t$. Thus, when the receiver attempts to extract the link signature from the attacker's symbols $\mathbf{y}_a$, it will get a link signature similar to the one estimated from $\mathbf{y}_t$.

The attacker needs to meet two requirements to launch a mimicry attack: First, the attacker needs to

roughly know the received symbols $\mathbf{y}_t$. Second, the attacker needs to manipulate its own symbols, such that when the manipulated symbols arrive at the receiver, they are similar to $\mathbf{y}_t$ (i.e., $\mathbf{y}_a \approx \mathbf{y}_t$).

5.2.1 Learning Symbols $\mathbf{y}_t$

Intuitively, an attacker should be co-located with the receiver in order to know $\mathbf{y}_t$. However, it is possible for an mimicry attacker to avoid satisfying such an extreme condition. The attacker can learn $\mathbf{y}_t$ by placing a sensing device, which we call the *symbol sensor*, close to the receiver. It records the symbols sent from the transmitter and reports them to the attacker through any available communication channel. Moreover, we also observe experimentally that the symbol sensor does not have to be at the exact location of the receiver.

Experimental Observation: We used three USRP2s as the transmitter, the receiver, and the symbol sensor. The communication frequency was 2.4GHz, and the distance D between the transmitter and the receiver was 6.5 meters. The transmitter sent a training sequence of 63 symbols for 1,000 times. We performed two sets of experiments. We first placed the symbol sensor $D = 6.5$ meters away from the receiver, and then placed the symbol sensor $0.4D = 2.6$ meters away from the receiver. In both cases, we recorded the normalized amplitude of each received symbol.

Figures 5.1 and 5.2 plot the average amplitude of received symbols. When the distance between the symbol sensor and the receiver is large, symbols received by the symbol sensor are different from those received by the receiver (Figure 5.1). However, when the symbols sensor is close to the receiver, both received symbols become similar to each other (Figure 5.2).

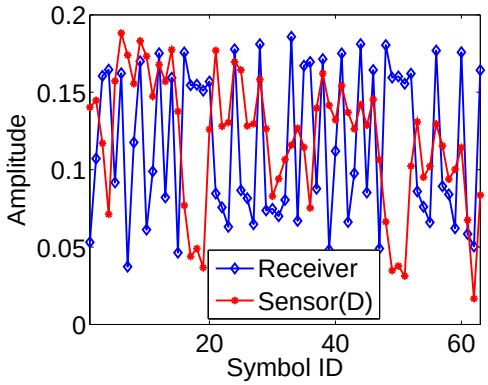

Figure 5.1: The sensor is D meters away from the receiver

Radio channels correlate within several wavelengths [31]. If the symbol sensor is within several wavelengths away from the receiver, the symbols received by the symbol sensor will be similar to those

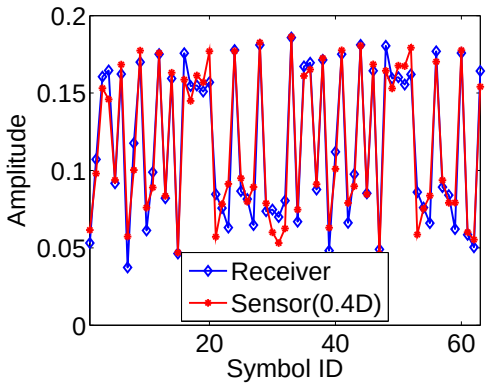

Figure 5.2: The sensor is $0.4D$ meters away from the receiver

at the receiver. Our experiment uses 2.4GHz frequency, and has a relatively short wavelength. However, as shown in US spectrum allocation chart [6], several radio systems work at MHz frequencies with much (about 1,000 times) longer wavelengths, thus allowing a much larger distance between the symbol sensor and the receiver.

Note that the attacker does not have to wait for the symbol sensor to report $\mathbf{y}_t$ in order to launch the mimicry attack. Instead, she may directly use the mathematical model in Equation (5.1) to estimate $\mathbf{y}_t$. Due to the proximity reason, the attacker can use the link signature between the transmitter and the symbol sensor as $\mathbf{h}_t$ to calculate $\mathbf{y}_t$. Thus, the symbol sensor only needs to estimate the link signatures between itself and the transmitter and report it to the attacker, and the attacker can directly calculate $\mathbf{y}_t$ whenever needed.

5.2.2 Manipulating Transmitted Symbols

The symbols $\mathbf{y}_a$ received from the attacker can be represented as $\mathbf{y}_a = \mathbf{h}_a * \mathbf{x}_a + \mathbf{n}_a$, where $\mathbf{x}_a$, $\mathbf{h}_a$, and $\mathbf{n}_a$ are the symbols transmitted by the attacker, the link signature of the attacker, and the channel noise, respectively. To make $\mathbf{y}_a$ equal to $\mathbf{y}_t$, the attacker can treat $\mathbf{x}_a$ as a unknown variable, and solve it from the following equation

$$\mathbf{h}_a * \mathbf{x}_a + \mathbf{n}_a = \mathbf{y}_t, \tag{5.4}$$

where the link signature $\mathbf{h}_a$ of the attacker can be obtained from the symbol sensor as well. The solution to this equation enables $\mathbf{y}_a$ to be similar to the transmitter's symbols $\mathbf{y}_t$. As a result, the link signatures that are estimated from $\mathbf{y}_a$ will also be close to those estimated from $\mathbf{y}_t$. In Theorem 5, we give a way to solve $\mathbf{x}_a$ from Equation (5.4).

Theorem 5 *Let $\mathbf{y}_t$ and $\mathbf{y}_a$ denote the received symbols that are sent by the transmitter and the attacker, respectively. Further let $\mathbf{H}_a$ be the Toeplitz matrix of the attacker's link signature. If $\mathbf{x}_a =$*

$(\mathbf{H}_a^H\mathbf{H}_a)^{-1}\mathbf{H}_a^H\mathbf{y}_t$, *then* $\mathbf{y}_a = \mathbf{y}_t$.

Proof: Let $\mathbf{x}_a = [x_{a1}, x_{a2}, ..., x_{aM}]^T$ denote the symbols transmitted by the attacker, and $\mathbf{h}_a = [h_{a1}, h_{a2}, ..., h_{aL}]^T$ denote the link signature of the attacker. We have

$$\begin{aligned}
\mathbf{y}_t &= \mathbf{h}_a * \mathbf{x}_a + \mathbf{n}_a = \mathbf{X}_a\mathbf{h}_a + \mathbf{n}_a \\
&= \begin{bmatrix} h_{a1} & 0 & \cdot & 0 \\ h_{a2} & h_{a1} & \cdot & \cdot \\ \cdot & h_{a2} & \cdot & 0 \\ \cdot & \cdot & \cdot & h_{a1} \\ h_{aL} & \cdot & \cdot & h_{a2} \\ 0 & h_{aL} & \cdot & \cdot \\ \cdot & 0 & \cdot & \cdot \\ 0 & 0 & \cdot & h_{aL} \end{bmatrix} \begin{bmatrix} x_{a1} \\ x_{a2} \\ \cdot \\ \cdot \\ x_{aM} \end{bmatrix} + \mathbf{n}_a \\
&= \mathbf{H}_a\mathbf{x}_a + \mathbf{n}_a.
\end{aligned}$$

Therefore, $\mathbf{y}_t = \mathbf{h}_a * \mathbf{x}_a + \mathbf{n}_a \Leftrightarrow \mathbf{y}_t = \mathbf{H}_a\mathbf{x}_a + \mathbf{n}_a$. We can solve $\mathbf{x}_a$ from $\mathbf{y}_t = \mathbf{H}_a\mathbf{x}_a + \mathbf{n}_a$. Since $\mathbf{n}_a$ is unknown, we use the standard least square approach [83] to solve $\mathbf{x}_a$. Specifically, we minimize $||\mathbf{y}_t - \mathbf{H_a}\hat{\mathbf{x}}_\mathbf{a}||^2$, where $\hat{\mathbf{x}}_a$ is the approximate solution of $\mathbf{x}_a$. The minimization yields

$$\hat{\mathbf{x}}_a = (\mathbf{H}_a^H\mathbf{H}_a)^{-1}\mathbf{H}_a^H\mathbf{y}_t. \tag{5.5}$$

□

5.2.3 Initial Validation via CRAWDAD Data Set

As an initial validation, we simulate mimicry attacks using the CRAWDAD data set [89], which contains over 9,300 link signatures measured in an indoor environment with obstacles and scatters. Our simulation is done in MATLAB 7.7.0.

We randomly pick two link signatures from CRAWDAD as the transmitter's link signature $\mathbf{h}_t$ and the attacker's link signature $\mathbf{h}_a$, respectively. We generate a training sequence of 256 bits using a pseudorandom number generator, and compute $\mathbf{y}_t$ according to Equation (5.3). To launch mimicry attacks, the attacker needs to calculate $\mathbf{x}_a$ based on Theorem 5. The corresponding received symbols $\mathbf{y}_a$ can also be computed by Equation (5.3). Finally, the receiver estimates link signatures from $\mathbf{y}_a$. Figure 5.3 shows the transmitter's link signature $\mathbf{h}_t$, the attacker's link signature $\mathbf{h}_a$, and the attacker's forged link signature estimated from $\mathbf{y}_a$. All link signatures are normalized by amplitude and each contains 50 elements. Though the attacker's original link signature is different from the transmitter's, after the mimicry attack, the forged link signature is very close to the transmitter's link signature.

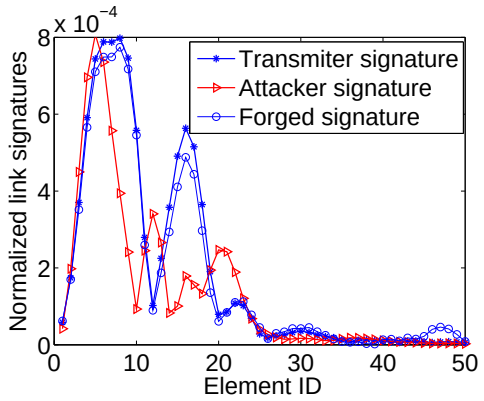

Figure 5.3: Mimicry attack using CRAWDAD data

This initial validation demonstrates the theoretical correctness of mimicry attacks. In Section 5.4, we investigate the practical impact of mimicry attacks with real world experiments.

5.2.4 Extending Attack to Multiple Tone Probing Link Signature

There are two other link signature schemes [53, 112] besides [65]. The *multiple tone probing based link signature* [53] uses complex gains at different frequencies to build a link signature, and the scheme in [112] is an integration of the techniques in [65] and [53]. We have extended the mimicry attack to compromise the multiple tone probing based link signatures in a similar way. We omit the attack here due to space limit. Details can be found in our technical report [55].

5.3 Time-synched Link Signature

In this section, we develop a novel time-synched link signature to defend against the mimicry attack. A key feature of this new mechanism is the integration of cryptographic protection and time factor into wireless link signatures.

5.3.1 Assumptions and Threat Analysis

Assumptions: We assume that there are a *Transmitter* and a *Verifier*, who share a secret key K that is only known to them. The Transmitter sends physical layer *frames* to the Verifier, who then verifies if these frames are directly transmitted by the Transmitter. We assume that the attacker can eavesdrop, overhear, and jam wireless communications. However, we assume that the attacker cannot compromise the Transmitter or the Verifier, and thus does not know their secret.

The attacker's goal is to transmit frames to the Verifier and convince it that the frames were transmitted directly by the Transmitter, so that the Verifier will derive incorrect physical layer features about the transmission (e.g., wrong Received Signal Strength, leading to incorrect estimate of distance).

Threat Analysis: Let us first understand what new challenges the mimicry attack brings given the existing network security tools. First of all, note that we can simply add digital signatures or Message Integrity Code (MIC) into each frame. As a result, the frames forged by the attacker can be easily detected through authentication of message content. Thus, the remaining threat is from the frames that are originally generated by the Transmitter but forwarded by the attacker. Note that the frame forwarded by the attacker is the same as the original frame generated by the Transmitter at the bit level, but different at the symbol level.

Moreover, with replay attack detection mechanism such as sequence numbers, if the Verifier can receive the original frames sent by the Transmitter, it can easily identify frames forwarded by the attacker as duplicates and discard them. Thus, the unresolved threats are from the following two cases: (1) when the attacker can jam and replay the Transmitter's frames (jam-and-replay attack [36]), and (2) when the Transmitter and the Verifier are out of communication range, but the jammer forwards frames from the Transmitter to the Verifier.

In our study, we focus on the unresolved threats, assuming existing mechanisms such as cryptographic authentication and sequence numbers can be used. In the following, we clarify the attacker's capabilities in forwarding frames.

We assume the attacker may launch *frame repeater attacks*. That is, the attacker may receive a frame sent by the Transmitter and then forward it to the Verifier. Such frame repeaters are widely available commercially (e.g., 802.11 repeaters).

The attacker may also launch physical layer *symbol repeater attacks*. That is, the attacker can observe the transmission of each physical layer symbol, which may represent one or multiple bits in the frame, and then forward the symbol to the Verifier directly. Such repeaters can be developed using noise canceling techniques and proper positioning of antennas [22]. Compared with frame repeater attacks, symbol repeater attacks are much harder to defend against.

Link signatures are specific to wireless communication channels, and usually require a training phase. The attacker may target at either the *training phase* to mislead the Transmitter and the Verifier about their link signature, or the *operational phase* when the link signature is used for physical layer authentication. Thus, a secure link signature has to protect both the training and the operational phases.

5.3.2 Design Strategy

The fundamental reason for the mimicry attack is that the attacker can establish a set of equations based on (1) the knowledge of the training sequence and (2) the Transmitter's signal (i.e., physical layer symbols) at the Verifier's location. These allow the attacker to manipulate the transmitted physical layer

symbols so that a forged frame has a valid link signature.

Initial Idea: To defend against this attack, our strategy is to deprive the attacker at least one of these two pieces of information. It is in general very difficult to prevent a passive attacker from receiving signals (and then extracting valid link signatures). However, it is possible to prevent the attacker from knowing the training sequences. Thus, our initial idea is to use *unpredictable*, *dynamic*, and *authenticated* training sequences for extracting link signatures from wireless packets (frames).

Detecting Frames Forwarded by Attackers: It is not hard to realize that simply using unpredictable, dynamic, and authenticated training sequences is still insufficient. The attacker can receive and analyze the Transmitter's signal to learn the training sequence, and forge link signatures by manipulating and forwarding frames received from the Transmitter.

To handle this threat, we propose to bring "time" into the scheme. We assume the Transmitter and the Verifier have synchronized clocks. (Our scheme will include a time synchronize component to meet this assumption.) The Transmitter may include a timestamp in the transmitted frame, which indicates the time when a particular bit or byte called the *anchor* (e.g., the Start of Frame Delimiter (SFD) field [45]) is transmitted over the air. We assume that the Transmitter can use authenticated timestamping techniques (e.g., [94]) to ensure that the timestamp precisely represents the point in time when the anchor is transmitted. Upon receiving a frame, the Verifier can use this timestamp and the frame receiving time to estimate the frame traverse time. An overly long time indicates that the frame has been forwarded by an intermediate attacker.

Defending against Physical Layer Symbol Repeater Attacks: A physical layer symbol repeater attack is much harder to detect than frame repeater attacks. If the attacker knows where the training sequence is located in the frame, she can start repeating the physical layer symbols right after receiving the symbols for the training sequence. This reduces the delay that the attacker has to tolerate to the transmission time of the training sequence, which could be much shorter than the transmission time of the entire frame.

To defend against such physical layer symbol repeater attacks, we propose to integrate a third idea into the scheme, that is, to make the location of the training sequence *unpredictable until the end of the frame transmission*. Specifically, we insert the training sequence at a *random* location in the payload, and place this location, which can be represented as the offset from the start of the frame header, at the end of the frame. In order for a physical layer symbol repeater to mimic the link signature of the Transmitter, she has to manipulate the physical layer symbols corresponding to the training sequence in a frame. If the location of the training sequence is not revealed until the end of the frame, the attacker will have to wait until the end of the transmission to learn it. This forces a physical layer symbol repeater attack to degenerate into a frame repeater attack.

Minimum Frame Length: If a frame is too short, the Verifier may have difficulty seeing the delay caused by a frame repeater. One solution is to pad extra bits into the frame if the frame length is less than a minimum frame length.

The minimum frame length can be determined based on the errors of the time synchronization and time measurement. Assume the maximum errors in clock discrepancy and transmission time are e_δ and e_τ, respectively, and the maximum time measurement errors in the Transmitter and the Verifier are e_T and e_V, respectively. Thus, the maximum error that the Verifier has to tolerate is $e_{all} = e_\delta + e_\tau + e_T + e_V$. Assume that the data rate of the wireless communication is R. It is easy to see that when the frame length is greater than the minimum frame length $L_{min} = R \cdot e_{all}$, the Verifier is guaranteed to detect frames forwarded by frame repeaters.

It has been demonstrated in an implementation of Radio Frequency (RF) distance bounding protocol [73] that nano-second processing delay is feasible to achieve. The time-synched link signature requires much less precision in time synchronization. For example, even assuming e_{all} is between $1\mu s$ and $10\mu s$, in a 54 Mbps 802.11g wireless network, L_{min} will range between 7 bytes and 68 bytes.

Overall Design: Figure 5.4 illustrates how these ideas can be integrated into a physical layer protocol. A physical layer frame typically consists of a series of preamble symbols, the frame header, and the payload. To detect frames forwarded by attackers, we include in each frame a timestamp t_s, which indicates the transmission time of the frame. To defend against physical layer repeater attacks, we include the randomly generated offset P of the training sequence in each frame at the end of the frame (to force the attacker to wait until the end of frame transmission).

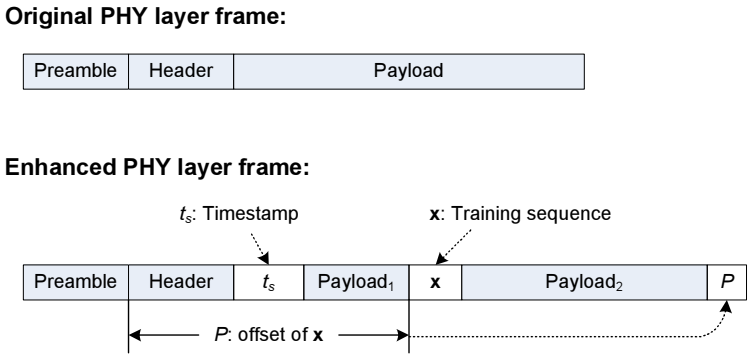

Figure 5.4: PHY layer frame: Dynamic training sequence with random offset

Assume the Transmitter and the Verifier share a secret key K. We piggyback the authentication of the frame with the generation of the unpredictable, dynamic, and authenticated training sequence. Specifically, we propose to use the MIC of the entire frame as the training sequence $\mathbf{x}$. In situations where there is a mismatch between the MIC and the training sequence (e.g., when a longer training sequence is needed), we can simply generate the training sequence as $\mathbf{x} = F(K, t_s)$, where F is a pseudo-random generator, and compute the frame MIC separately. The use of K and t_s makes $\mathbf{x}$ dy-

namic and unpredictable, and the frame MIC allows $\mathbf{x}$ to be authenticated.

In the following, we present the details of the training and the operational phase in time-synched link signature.

5.3.3 Training Phase

The training phase is intended for the Verifier to collect enough information from the Transmitter so that the Verifier can verify the link signatures of the future frames from the Transmitter. The Verifier should obtain the valid link signature from the Transmitter whenever the link signature may change. This can be accomplished by executing the training phase protocol periodically or whenever one of them moves.

In the training phase, the Verifier needs to synchronize its clock with the Transmitter, and obtain the link signature for the current communication channel. Moreover, it needs to confirm that there is no successful attack during the training phase.

We use the classic time synchronization technique (e.g., [59]) to estimate the clock discrepancy between the Transmitter and the Verifier as well as the frame traverse time. We refer to the point in time when the anchor (e.g., the SFD field) in a frame is transmitted or received as the *transmission time* or the *receiving time* of this frame. Specifically, the Verifier sends a *request frame* to the Transmitter, and at the same time records the frame transmission time t_1 in the Verifier's local clock. When the Transmitter receives the request frame, it records the receiving time t_2 of this frame, and then sends a *reply frame* to the Verifier, in which t_2 and the transmission time t_3 of the reply frame (in the Transmitter's clock) are included. Finally, the Verifier receives the reply frame and records the receiving time t_4 in its clock. The clock discrepancy δ between the Verifier and the Transmitter and the one-way frame traverse time τ can then be estimated as $\delta = \frac{(t_2-t_1)-(t_4-t3)}{2}$ and $\tau = \frac{(t_2-t_1)+(t_4-t_3)}{2}$ [59].

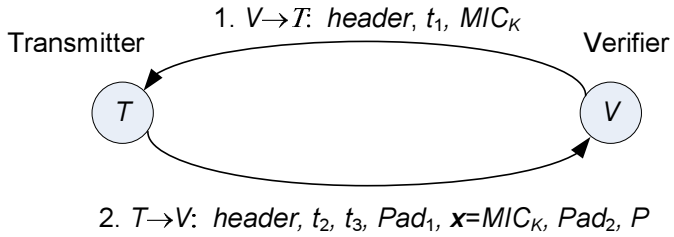

Figure 5.5: Training phase protocol

Figure 5.5 shows the training phase protocol between the Transmitter and the Verifier.

Training Request: The Verifier sends the first training request frame to the Transmitter, which includes the frame header, the transmission time t_1 of this frame, and the frame MIC that covers the entire frame (excluding the preambles). Upon receiving of the request frame, the Transmitter immediately

records the receiving time t_2 of the frame, and authenticates the request frame by verifying the MIC.

Training Reply: Upon verifying a training request frame, the Transmitter should send back a training reply frame. The Transmitter should include time t_2 and the actual transmission time t_3 of the reply frame in the frame. The Transmitter also pads the frame payload to at least the minimum frame length L_{min} and randomly selects an offset P to place the training sequence as discussed earlier. The Transmitter then leaves a placeholder (e.g., all 0's) in place of the training sequence and computes the frame MIC using the shared key K. Finally, the Transmitter places the frame MIC as the training sequence $\mathbf{x}$ in the reply frame and sends it over the air.

Once the Verifier receives the training reply frame, the Verifier computes the clock discrepancy δ and the one-way transmission time τ. If τ is greater than a threshold τ_{max}, which is the maximum possible direct transmission time, the Verifier should consider the reply frame as possibly forwarded by the attacker and discard it. Otherwise, the Verifier locates the frame MIC by following the offset P at the end of the frame, authenticates the frame MIC using the shared key K, and uses the frame MIC (i.e., the training sequence $\mathbf{x}$) to extract the link signature. The Verifier may run the training phase several times to get a better quality link signature.

5.3.4 Operational Phase

Once the Verifier obtains the clock discrepancy and the valid link signature from the Transmitter, they can start the operational phase, during which the Verifier uses this link signature to verify frames that require physical layer authentication.

Transmitter: To defend against the threats discussed in Section 5.3.1, the Transmitter follows the design shown in Figure 5.4. Specifically, the Transmitter randomly selects an offset in the frame payload to include the field for the training sequence. places the offset P at the end of the frame, and computes the frame MIC using the shared secrete key K, with a placeholder (e.g., all 0's) for the training sequence. The Transmitter then uses the frame MIC as the training sequence $\mathbf{x}$, puts it in the frame, and sends the frame over the air. Similar to the training phase, the Transmitter estimates the frame transmission t_s based on the current time and the estimated duration for the deterministic MIC computation.

Verifier: When the Verifier receives the frame, it immediately records the receiving time t_r. The Verifier then retrieves the frame transmission time t_s from the received frame and estimates the frame traverse time $\tau = t_s - t_r - \delta$, where δ is the clock discrepancy between the Verifier and the Transmitter learned in the training phase. If τ is greater than the threshold τ_{max}, the maximum possible direct transmission time, the Verifier should consider the frame possibly forwarded by the attacker and discard it. Otherwise, the Verifier locates the frame MIC by using the offset P at the end of the frame, verifies the frame MIC using the shared key K, and then uses the frame MIC as the training sequence to extract the link signature. Finally, the Verifier compares this link signature with the one derived during the training phase. The frame is accepted if this link signature does not deviate from the valid one learned

in the training phase. Otherwise, the frame is considered forged and discarded.

5.3.5 Security Analysis

The time-synched link signature uses a training sequence authenticated with a shared secret key only known to the Transmitter and the Verifier, and the training sequence changes from frame to frame due to the involvement of the timestamp when computing the training sequence. Thus, the training sequence is authenticated, dynamic, and unpredictable. This prevents the attacker from forging frames with training sequences of its choice. The only choice left for the attacker is to reuse and manipulate valid frames from the Transmitter.

The use of random offset for the training sequence in the frame payload forces the attacker to wait for the end of the frame transmission to understand where the training sequence is located in the frame. As a result, the attacker cannot launch physical layer symbol repeater attacks and at the same time manipulate the training sequence correctly to bypass link signature verification. The attacker may still perform the frame repeater attack. However, due to the enforcement of the minimum frame length, a frame forwarded by a frame repeater will introduce at least the amount of delay caused by the receiving of the frame, which is detectable by the Verifier.

The attacker may launch a probabilistic mimicry attack by randomly guessing the location of the training sequence and forging the frame symbols accordingly. The attacker may also overestimate the length of the training sequence and perform the forgery. If the assumed training sequence $\mathbf{y}'_t$ is a superset of the actual one $\mathbf{y}_t$ (i.e., $\mathbf{y}_t$ is a subsequence of $\mathbf{y}'_t$), due to the linear property of Equation (5.5), the forged symbols $\hat{\mathbf{x}}'_a$ will include $\hat{\mathbf{x}}_a$ as a subsequence. This will allow the attacker's symbols to be accepted by the receiver. However, the attacker cannot delay the transmission of a frame for L_{min} or more; otherwise, its interference will be detected. This means that the probability for the attacker to succeed is at most $p = \frac{L_{min}-|x|+1}{F-|x|+1}$ when L_{min} is greater than or equal $|x|$, where $|x|$ and F are the lengths of the training sequence and the frame payload, respectively. When L_{min} is less than $|x|$, the probability of a successful mimicry attack degrades to 0. Nevertheless, the probabilistic mimicry attack does increase the requirement for time synchronization. In other words, the Transmitter and the Verifier need to obtain fine-grained time synchronization so that the success probability of a probabilistic mimicry attack becomes negligible.

5.4 Experimental Evaluation

We have implemented the link signature scheme in [65], the basic mimicry attack, and the newly proposed time-synched link signature. We have also implemented the frame repeater attack, which can be used along with the mimicry attack. Our prototype uses USRP2 [57], which are equipped with AD and DA converters as the RF front ends, and XCVR2400 daughter boards operating in the 2.4 GHZ range

as transceivers. The software toolkit is GNURadio [1].

USRP2s are capable of processing signals up to 100MHz wide. Such a high bandwidth enables the capture of multipath effects and measurement of link signatures. GNURadio configuration requires setting the values of interpolation (decimation) rate at the transmitter (receiver) and the number of samples per symbol. If these parameters are set too high, the bandwidth will be significantly reduced. To guarantee the capture of multipath effect, we set those parameters at the minimum values allowed by GNURadio (i.e., 5 for the interpolation and decimation rate, and 2 for the number of samples per symbol).

5.4.1 Evaluation Methodology

Evaluation Scenarios: Our prototype system consists of a transmitter, a receiver, a symbol sensor, and an attacker. The receiver is 15 meters from the transmitter, and the symbol sensor is 0.5 meters from the receiver. Each node is a USRP2 connected to a commodity PC. The receiver estimates the received link signatures and compares them with the transmitter's link signature. We consider three scenarios in our evaluation: (1) *normal scenario*, (2) *forgery scenario*, and (3) *defense scenario*. In a normal scenario, the attacker simply sends original symbols to the receiver. In both the forgery and the defense scenarios, the attacker launches the mimicry attack, during which it transmits manipulated symbols to the receiver. However, the forgery scenario uses the previous link signature scheme in [65], while the defense scenario uses the newly proposed time-synched link signature scheme. The symbol sensor estimates the link signature of the attacker and provides this link signature and the received symbols sent by the transmitter to the attacker.

Evaluation Metrics: Intuitively, the attacker wants to reduce the difference between its own link signature and the transmitter's link signature, whereas the defense method aims to increase this difference to alert the receiver. Thus, the link difference between both the attacker's and the transmitter's link signatures can visually reveal the impact of mimicry attacks and the effectiveness of the defense method.

Link signature authentication serves as a detector that decides if a received signal is from the desired source. Thus, besides link difference, we also use detection rate P_D (i.e., the rate that an attacker's link signature is successfully detected by the receiver) and false alarm rate P_{FA} (i.e., a transmitter's link signature is incorrectly identified as the attacker's link signature) as two additional metrics. Finally, we measure the time delay introduced by the transmitter and the attacker to assess how well the frame repeaters can be detected.

5.4.2 Evaluation Results

We now show how mimicry attacks affect the link difference, false alarm rate, detection rate, and the tradeoff between the detection and the false alarm rates in the normal, forgery, and defense scenarios.

Link Difference

According to [65], given a set $\mathcal{H}$ of link signature obtained in the training phase and a link signature $\mathbf{h}$ under consideration, the link difference $d_{\mathbf{h},\mathcal{H}}$ is calculated using $\frac{1}{\sigma}\min_{\mathbf{g}\in\mathcal{H}}\|\mathbf{g}-\mathbf{h}\|$, where σ is the *historical average difference* between link signatures in $\mathcal{H}$, given by $\sigma = \frac{1}{N(N-1)}\sum_{\mathbf{g}\in\mathcal{H}}\sum_{\mathbf{q}\in\mathcal{H}-\{\mathbf{g}\}}\|\mathbf{q}-\mathbf{g}\|$.

In each evaluation scenario, the receiver first measures a set $\mathcal{H}$ of $N = 50$ link signatures of the transmitter in the training phase. It then collects 450 link signatures of the attacker and calculates the link difference $d_{a,\mathcal{H}}$ for each. Moreover, the receiver collects another 450 link signatures of the transmitter, and calculates the link difference $d_{t,\mathcal{H}}$ for each of them.

Figures 5.6, 5.7, and 5.8 show the link differences for the attacker $d_{a,\mathcal{H}}$ and the transmitter $d_{t,\mathcal{H}}$ in the normal, forgery, and defense scenarios, respectively. Figure 5.6 shows that in the normal scenario $d_{a,\mathcal{H}}$ is generally larger than $d_{t,\mathcal{H}}$. Moreover, Figure 5.9 shows the histograms of $d_{a,\mathcal{H}}$ and $d_{t,\mathcal{H}}$. Most of the transmitter's link difference is less than 0.15, whereas most of the attacker's link difference is larger than 0.15. Thus, based on the link difference, the receiver can achieve a high accuracy in distinguishing between the transmitter and the attacker.

In the forgery scenario, the attacker launches mimicry attacks to make its own link signatures similar to the transmitter's link signatures. Figure 5.7 shows that $d_{a,\mathcal{H}}$ decreases to the same level as $d_{t,\mathcal{H}}$, and $d_{a,\mathcal{H}}$ and $d_{t,\mathcal{H}}$ substantially overlap with each other. The histogram of $d_{a,\mathcal{H}}$ (i.e., the top graph in Figure 5.10) shows that the link difference distribution of the attacker is very close to that of the transmitter. The mimicry attack reduces the link difference between the attacker and the transmitter, leading to high false negative rate at the receiver.

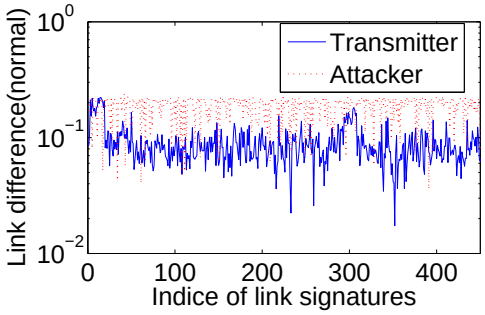

Figure 5.6: Normal

In the defense scenario, as indicated in Figure 5.8, the use of time-synched link signature increases the link difference $d_{a,\mathcal{H}}$ for the attacker. In particular, the mean value of $d_{a,\mathcal{H}}$ under the defense and forgery scenarios are 0.2847 and 0.1170, respectively. The histogram of $d_{a,\mathcal{H}}$ in the defense scenario

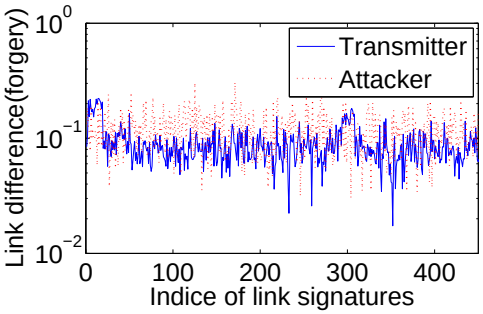

Figure 5.7: Forgery

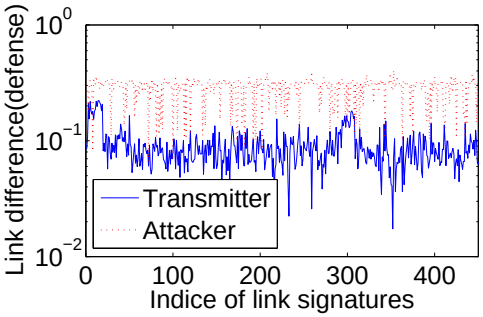

Figure 5.8: Defense

(i.e., the bottom graph in Figure 5.10) shows that the link difference computed from a majority of forged signatures is larger than 0.15. Thus, the receiver can again distinguish between the transmitter and the attacker with low error rate.

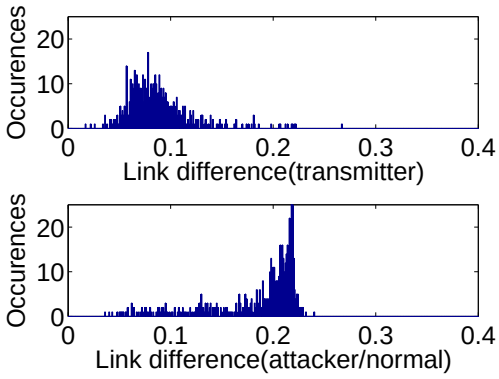

Figure 5.9: Histograms of link difference for the transmitter's and the attacker's link signatures in normal scenario

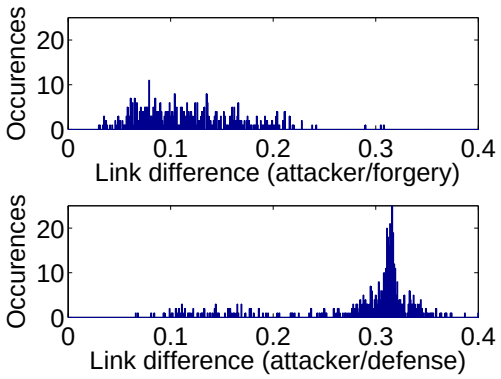

Figure 5.10: Histograms of link difference for the attacker's link signatures in forgery and defense scenarios

Detection and False Alarm Rates

As mentioned earlier, a history of N transmitter's link signatures is measured and stored at the receiver, and the receiver computes the link difference d between a newly measured link signature and history link signatures. In our experiment, we follow the same detection rule as in [65]: If d is smaller than a certain threshold r, the receiver concludes that this link signature is from the transmitter; otherwise, it is from the attacker.

Let N_{FA} denote the number of link signatures that are from the transmitter but incorrectly identified as from the attacker, and N_D denote the number of attack link signatures that are correctly detected by the receiver. The false alarm rate P_{FA} is calculated as the ratio of N_{FA} to the total number of false link signatures claimed by the receiver, and the detection rate P_D is computed as the ratio of N_D to the total number of the attacker's link signatures. Figure 5.11 shows P_{FA} and P_D as a function of the threshold r. A large threshold can reduce false alarm rate P_{FA}, whereas a small threshold can increase detection rate P_D. A common decision is to pick the *operational threshold* as the point where the distance between P_{FA} and P_D is the largest (i.e., $P_D - P_{FA}$ is the largest).

The operational thresholds of the normal, defense, and forgery scenarios are 0.1262, 0.1099, and 0.1410, respectively, in our experiments. For the normal scenario, the corresponding P_{FA} and P_D achieved by the operational threshold are $P_{FA} = 0.0893$ and $P_D = 0.8710$. The defense scenario outperforms the normal scenario in terms of reducing P_{FA} and increasing P_D with the operational threshold, leading to $P_{FA} = 0.0583$ and $P_D = 0.9242$. The forgery scenario has the worst performance. With the operational threshold, $P_{FA} = 0.1608$ and $P_D = 0.5142$. Note that in our experiment a link signature is either from the transmitter or from the attacker, and thus the probability that a blind guess hits the true source of this link signature is 0.5. In the forgery scenario, the operational threshold

achieves a detection rate that is just slightly better than a blind guess.

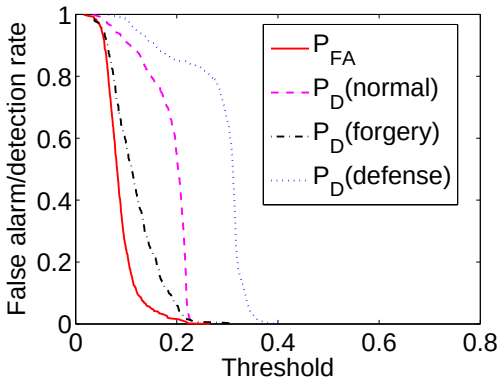

Figure 5.11: False alarm rate P_{FA} and detection rate P_D as a function of threshold

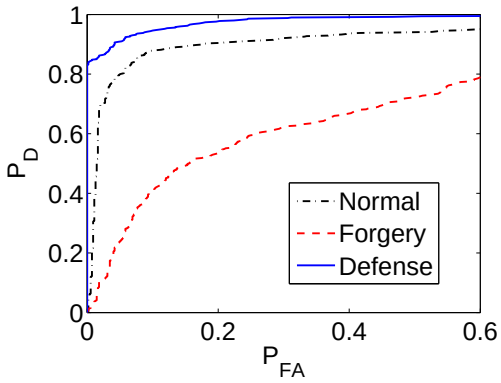

Figure 5.12: Tradeoff between false alarm and detection rate for normal, forge, and defense scenarios

Figure 5.12 shows the receiver operating characteristic (ROC) curves for the three scenarios, in which the P_{FA} and P_D are the x-axis and y-axis, respectively. The curve representing the defense scenario is on the top-left corner of the figure, indicating good performance of the time-synched link signature.

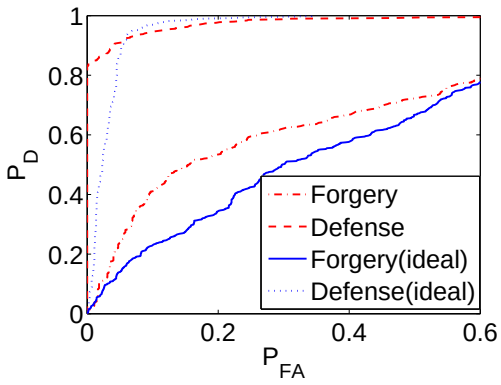

Figure 5.13: Tradeoff between false alarm and detection rate for an ideal mimicry attacker

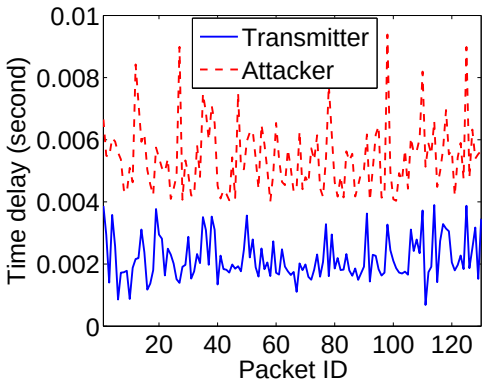

Figure 5.14: Delay of original and forwarded frames

Ideal Mimicry Attack

We examined the mimicry attack in a realistic setting, where a symbol sensor is used to obtain an approximation of the transmitter's symbols at the receiver. To have a conservative assessment of our approach, we also examine the case where there is an ideal mimicry attacker, who knows the exact form of the transmitter's symbols at the receiver. We collect the link difference caused by an ideal mimicry attacker, and compare the corresponding ROC curves. As shown in Figure 5.13, in the forgery scenario, the ideal attacker results in an even lower detection rate than the previous attacker. In the defense scenario, both attackers can be detected with high confidence. In particular, for a 0.05 false alarm rate, the receiver can achieve a detection rate larger than 0.9 in the presence of both attackers.

Frame Time Delay

The proposed time synched-link signature uses estimated frame traverse to filter out frames forwarded by the attacker. In this experiment, we measure the time delay of frames from the transmitter and the attacker, respectively, to examine this approach. In our experiment, the frame length is 190 bits and the transmission rate is set to 500Kbps. The transmitter sends 130 frames, and the attacker forwards all of them. Thus, the receiver receives 260 frames in total. Figure 5.14 shows that the time delays of frames forwarded by the attacker significantly exceed those of the frames directly by the transmitter. Our further analysis indicates that the ratio of attacker's delay to the transmitter's delay ranges between 2.2 and 2.6, indicating that the forwarding by the attacker approximately doubles the time delay.

Chapter 6

Related Work

Previous work that is related to the research in the dissertation fall into three areas, which are: *anti-jamming defense design*, *primary user detection in cogitative radio networks*, and *wireless transmitter authentication*. In the following, I will summarize those related work.

6.1 Anti-jamming Defense Design

The jamming problem in wireless communication has been widely studied during the past few decades (e.g., [30, 67, 68, 76, 86, 98, 99]), and spread spectrum such as DSSS and FHSS are traditional anti-jamming techniques [86, 97]. However, those techniques require that senders and receivers pre-share secret keys. There have been a few recent attempts to enable the establishment of pre-shared secret keys [87, 92, 93]. Unfortunately, all those works cannot be used for broadcast communication where there exist malicious receivers that may also act as jammers. An insider jammer who knows the shared key can use the key to prevent other receivers from receiving the messages.

To address this problem, some researchers recently investigated how to enable jamming-resistant broadcast communication without shared keys [8, 69]. As discussed earlier, the decoding process of [8] cannot be extended to DSSS or FHSS. UDSSS proposed in [69] is the most relevant to ours. However, the broadcast communication provided by UDSSS is still vulnerable if an insider jammer with sufficient computational power launches reactive jamming attacks as indicated in [69]. Inspired by UDSSS, RD-DSSS tolerates reactive jamming attacks by using multiple code sequences and permutation to protect each message.

FHSS and DSSS are dominantly used for the purpose of anti-jamming. Besides the requirement of shared secret, they also assume that the jammer has limited capabilities. For example, FHSS assumes that the jammer cannot jam all channels simultaneously, and DSSS assumes that the jammer cannot overwhelm the spreading gain. However, both FHSS and DSSS will fail when there are high power, broadband jammers. A recent paper considers threats from broadband jammers, and proposes to use

timing-based covert channels to address the threats from broadband jammers [110]. The basic idea is to map the inter-arrival times of a sender's corrupted packets into information bits [50]. However, this method also fails when attacked by a high power jammer, because packets transmitted by the sender are overwhelmed by the jammer and a receiver cannot collect corrupted packets to decode the original message.

Our work is also related to the detection of reactive jammer [91]. The jamming detector proposed in [91] aims to identify the cause of bit errors for individual packets, whereas the jamming detector presented in this dissertation aims to distinguish unjammed bits from jammed bits rather than determining the cause of bit errors. Hence, the work done by [91] is complementary to ours.

Besides the literature discussed above, there exist other related works. For example, a code tree based technique that enables the system to identify insider jammers was proposed in [19, 20]. Xu et al. proposed to employ consistency checking for detecting jamming attacks [111]. The problems such as how to mitigate jamming on control channels [50, 95] and sensor networks [52, 109] were also studied by previous researchers. These approaches are complementary to ours.

6.2 Primary User Detection in Cogitative Radio Networks

Primary user detection has been intensively studied in the past few years (e.g., [38, 46, 70, 78, 84, 85, 96, 107]). Traditional detection techniques in general can be categorized into energy detection (e.g., [85]) and feature detection (e.g., [38, 70, 78, 84, 107]). In energy detection, any captured signal whose energy exceeds a threshold is identified as a primary user's signal. In feature detection, signal features (e.g., pilot, synchronization words, and cyclostationarity) are extracted and used to detect the presence of a primary user's signal. However, those traditional techniques will fail in hostile environments, where an attacker transmits with large power or mimics a primary user's signal features to gain unfair share of the bandwidth.

A recent attempt considered the security aspects of primary detection and proposed to utilize RSS-based location distinction for detecting primary users' signals in the presence of attackers [17]. Specifically, a secondary user verifies whether a received signal is from a primary user or not by estimating the location of the signal source. If the estimation result deviates from the known location of the primary user, it is highly possible that the signal is sent by an attacker. However, as indicated in [65], RSS-based location distinction approach, which is used in [17], can be easily disrupted if the attacker is equipped with array antennas. Moreover, such an approach requires multi-node collaboration, which is expensive in terms of bandwidth and energy.

In this work, I designed a primary user detection scheme by exploiting link signatures, which do not have the weakness of RSS based location distinction and achieve a higher accuracy [65, 112]. I integrate cryptographic and wireless link signatures to authenticate a primary user's signal, and use two levels of detections (i.e., detection at a helper and at a secondary user) to address the technical challenges caused

by adopting link signatures in CRNs.

There are other related works, including cooperative feature detection or energy detection [37,60,82, 105], secure data fusion in the presence of false information for distributed spectrum sensing [18, 100], performance evaluation of primary user detection in IEEE 802.22 [24], trade off between a secondary user's data transmission and the detection of a primary user's signal [44], and IEEE 802.22 standard for CRNs [25, 27]. These works are complementary to our work.

6.3 Wireless Transmitter Authentication

Existing non-cryptographic techniques for authenticating wireless transmitters can be divided into three categories [13]: software fingerprinting (e.g., [35, 47, 61]), location distinction (e.g., [17, 53, 65, 112]), and radiometric identification (e.g., [13, 29, 71])

In software fingerprinting approaches, discrepancies in software configuration are used as fingerprints to distinguish between wireless nodes [13]. For example, Franklin et al. [35] proposed to use the implementation dependent differences among device drivers to identify 802.11 nodes. Kohno et al. [47] proposed to use clock skews in TCP and ICMP timestamps to fingerprint networked devices.

In location distinction based authentication, a signal is authenticated by verifying whether it originates from the expected location of the transmitter. RSS (e.g., [17]) and link signatures have been used to enable such location distinction [65]. The RSS based methods directly estimate the location of a signal origin using the RSS values. However, such methods can be defeated with an array antenna, which can fake arbitrary source locations [65]. The link signature based approaches authenticate the channel characteristics between the transmitter and the receiver [53, 65, 112]. In this paper, we showed that all these link signature schemes are vulnerable to mimicry attacks. Our newly proposed time-synched link signature is developed to fill this gap.

In radiometric identification approaches, the distinctive physical layer characteristics exhibited by wireless devices are utilized to distinguish between them. Transient based techniques (e.g., [29, 71]) identify a wireless device by looking at the unique features "during the transient phase when the radio is turned on" [28]. Modulation based techniques (e.g., [13]) measure differentiating artifacts of individual wireless frames in the modulation domain to identify the device.

Recently, it was demonstrated in [28] and [33] that radiometric identification techniques were vulnerable to impersonation attacks. The results in [28] revealed that both transient and modulation based techniques are vulnerable to impersonation attacks, though transient-based techniques are harder to reproduce. Edman et al. [33] showed that an attacker can significantly reduce the accuracy of such techniques by simply using a commodity RF hardware platform. These works are complementary to ours in this paper.

Chapter 7

Future Work

7.1 Motivation

As a key enabling technique for cognitive networks, spectrum sensing has been a hot topic among researchers in recent years. An overview of implementation issues and challenges in spectrum sensing for cognitive networks are provided by [14]. In particular, advantages and disadvantages of three traditional signal detection techniques are discussed, including matched filter, energy detector, and cyclostationary feature detector. Researchers from Berkeley have verified the feasibility of energy detector on a wireless testbed as well and proposed collaborative sensing to improve sensing accuracy [16].

Although spectrum sensing techniques have attracted more and more attentions as an indispensable tool that allows a cognitive network to function well, the security aspects of the spectrum sensing have received relatively less attention [102]. There exist two main security threats to spectrum sensing, which are Incumbent Emulation (IE) attacks and the Spectrum Sensing Data Falsification (SSDF) attacks [102].

In IE attacks, an attacker mimics specific features of a primary user's signal and transmits fake signals to the secondary users to bypass the existing primary user detection methods [17]. Consequently, secondary users may incorrectly identify the attacker's signal as a primary user's signal and do not use relevant channels. In our previous work, we proposed countermeasures against IE attacks. We developed a new primary user detection method that integrates cryptographic signatures with wireless link signatures to deal with IE attacks. The proposed method distinguishes a primary user's signal from an attacker's signal and conforms to the FCC's requirement of not modifying primary users. We evaluate the effectiveness of our method through both theoretical analysis and experiments using real-world link signatures obtained from the CRAWDAD data set [64]. Moreover, we demonstrate the feasibility of our proposed method by a prototype implementation on a software-defined radio platform.

Another type of attacks are SSDF attacks, which targets distributed spectrum sensing (DSS). In DSS, each secondary user executes spectrum sensing on its own and sends the local spectrum sensing

result data to the fusion center that processes the data and makes a final spectrum-sensing decision. In SSDF attacks, the attacker impersonates a secondary user and sends fake local sensing results to the fusion center, thus causing the fusion center to make a wrong spectrum sensing decision. In our future work, we would like to study SSDF attacks and develop defenses against SSDF attacks

7.2 Possible Approaches

The attackers sends false local spectrum sensing results to mislead the decision process at the fusion center. To maintain a desired level of accuracy in the presence of SSDF attacks, we need a robust security mechanisms that can tolerate the false local spectrum sensing results reported by adversaries. Possible solutions are listed below.

- Cryptographic based authentication: Symmetric and asymmetric key based authentication algorithms may be employed to protect the distributed spectrum sensing. In symmetric key based authentication, the fusion center shares a secret key with a secondary user. The secondary user and the fusion center signs outgoing messages and verifies incoming messages using their shared key, respectively. As long as the adversary does not know the shared key, it cannot impersonate a secondary user. A drawback of this approach is that the initial secret key establishment may become the target of the adversaries. In asymmetric key based authentication, the secondary users sign messages using their private key, and release their public keys to enable the fusion center to verify received messages. Public keys can be exposed to the public, and thus asymmetric key based approach does not need initial secret key establishment processes. However, asymmetric key based approach leads to a longer time delay in signing and key generation.
- Consistency check: A powerful adversary may take advantage of high-performance computers to compromise the symmetric/asymmetric keys used in cryptographic authentication. Therefore, besides cryptographic authentication, we may further enhance the security of distributed spectrum sensing with consistency check, in which we exploit spatial correlations between secondary users to identify and filter out compromised sensing results. Intuitively, secondary users that are geographically close to each other are likely to report similar sensing results, since it is high possible that they hear signals from the same primary user and observe similar signal patterns. Therefore, if a secondary user reports a sensing result that is far away from those reported by its neighbors, it is probably a “bad” node.

7.3 Evaluation Plan

Evaluation Methodology: We will use both theoretical analysis and simulation to evaluate the effectiveness of defense mechanisms. For consistency check algorithms, our evaluation objective is mainly

two-fold. First, we would like to see how well the algorithm performs. Second, we want to see the computational efforts required for identifying malicious sensing report.

Evaluation Metrics: False negative errors and false positive errors are two typical errors that occur during the consistency check. With a false negative, a malicious sensing report bypasses the consistency check. With a false positive, a benign report triggers alarms from the consistency check. To see how well a consistency check algorithm works, we will use probability of false negative and positive as the evaluation metrics. To see the computational efforts, we will use the execution time required by the consistency check algorithm to conclude whether a received sensing result is malicious or not. We expect to identify more metrics as our research proceeds.

Parameters in Evaluation: We will perform the evaluation under different parameter configurations to gain understanding of the SSDF attacks and defense mechanisms in different situations. Possible parameters to be used include the fraction of secondary users under the adversarys control and configurations of the cognitive radio networks.

Chapter 8

Conclusion

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

In the second work, we developed BitTrickle to enable wireless communication when a broadband and high power reactive jammer is present. BitTrickle delivers information by taking advantage of the reaction time of a reactive jammer. Two technical challenges were addressed in the development of BitTrickle. First, we designed a jamming detector that utilizes modulation properties to distinguish jammed bits from unjammed ones. In addition, we proposed BitTrickle encoding/decoding techniques that can successfully recover a transmitted message from message fragments that partially survive reactive jamming. Unlike FHSS and DSSS, BitTrickle does not assume a reactive jammer with limited spectrum coverage and transmit power, and thus can be used in scenarios where traditional approaches fail. We also implemented a prototype of BitTrickle on GNU Radio. Our experimental results showed that the BitTrickle prototype achieved a reasonable throughput that enabled message exchanges between victim wireless devices when 802.11 DSSS and the GNURadio benchmark were completely disabled by a reactive jammer.

In the third work, we developed a novel approach for authenticating primary users' signals in CRNs, which conforms to FCC's requirement. The proposed approach integrates cryptographic signatures and wireless link signatures to enable primary user detection in the presence of attackers. Essential to the proposed approach is a *helper node* placed physically close to each primary user, which serves

as a “bridge” to enable a secondary user to verify the cryptographic signature carried by the helper node’s signals, and then obtain the helper node’s authentic link signatures to verify the primary user’s signals. A key contribution in this work is a novel physical layer authentication technique that enables the helper node to authenticate signals from its associated primary user. Unlike previous techniques for link signatures, the proposed approach explores the geographical proximity of the helper node to the primary user, and thus does not require any training process. We have examined the proposed approach through theoretical analysis, experimental evaluation using the CRAWDAD data set [64], and a prototype implementation on USRPs based on GNUradio [2]. The obtained results indicate that the proposed approach is a promising solution for authenticating primary users’ signals in CRNs.

In the last work, we identified a vulnerability of existing link signature schemes by introducing the mimicry attack. Our results indicated that some assumptions that form the foundation of link signatures should be revisited. In particular, we demonstrated that an attacker can forge a transmitter’s link signature if she knows *approximately* the legitimate symbols at the receiver. To defend against the mimicry attack, we proposed the time-synched link signature scheme by integrating cryptographic protection and time factor into wireless features. Our experimental results demonstrated both the feasibility of mimicry attacks and the effectiveness of the proposed method.

REFERENCES

[1] GNU Radio - The GNU Software Radio. http://www.gnu.org/software/gnuradio/.

[2] Gnu radio software. http://gnuradio.org/trac.

[3] Marcum q-function. http://mathworld.wolfram.com/MarcumQ-Function.html.

[4] Omni antenna vs. directional antenna. https://www.cisco.com/en/US/tech/tk722/tk809/technologies_tech_note09186a00807f34d3.shtml#topic4.

[5] Reactive jamming technologies. http://www.ece.gatech.edu/academic/courses/ece4007/08fall/ece4007l02/lm5/jammer.doc.

[6] U.S. frequency allocation chart. http://www.ntia.doc.gov/osmhome/allochrt.pdf.

[7] White-paper: 3g infrastructure sharing. *Siemens*, 2001.

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

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

[10] A. J. Berni and W. D. Greeg. On the utility of chirp modulation for digital signaling. *IEEE Transaction on Communications*, 21:748–751, 1973.

[11] M. Biguesh and A. B. Gershman. Training-based mimo channel estimation: A study of estimator tradeoffs and optimal training signals. *IEEE Transaction on Signal Processing*, 54(3):884–893, March 2006.

[12] V. Brik, S. Banerjee, M. Gruteser, and S. Oh. Wireless device identification with radiometric signatures. In *MobiCom '08: Proceedings of the 14th ACM international conference on Mobile computing and networking*, pages 116–127, New York, NY, USA, 2008. ACM.

[13] V. Brik, S. Banerjee, M. Gruteser, and S. Oh. Wireless device identification with radiometric signatures. In *Proceedings of the 14th ACM International Conference on Mobile Computing and Networking (MobiCom '08)*, pages 116–127, 2008.

[14] D. Cabric, S. M. Mishra, and R. W. Brodersen. Implementation issues in spectrum sensing for cognitive radios. In *The 38th Asilomar Conference on Signals, Systems and Computers*, volume 1, pages 772–776, 2005.

[15] D. Cabric, A. Tkachenko, and R. W. Brodersen. Experimental study of spectrum sensing based on energy detection and network cooperation. In *TAPAS '06: Proceedings of the First International Workshop on Technology and Policy for Accessing Spectrum*, page 12, New York, NY, USA, 2006. ACM.

[16] D. Cabric, A. Tkachenko, and R. W. Brodersen. Experimental study of spectrum sensing based on energy detection and network cooperation. In *Proceedings of the First International Workshop on Technology and Policy for Accessing Spectrum*, 2006.

[17] R. Chen, J. Park, and J. H. Reed. Defense against primary user emulation attacks in cognitive radio networks. *IEEE Journal on Selected Areas in Communications*, 26(1):25–37, 2008.

[18] R. Chen, J. M. Park, and K. Bian. Robust distributed spectrum sensing in cognitive radio networks. In *Proceedings of IEEE INFOCOM 2008 mini-conference*, April 2009.

[19] J. Chiang and Y. Hu. Extended abstract: Cross-layer jamming detection and mitigation in wireless broadcast networks. In *Proceedings of ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc '07)*, 2007.

[20] J. Chiang and Y. Hu. Dynamic jamming mitigation for wireless boradcast networks. In *Proceedings of IEEE International Conference on Computer Communications (INFOCOM '08)*, 2008.

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

[22] J.I. Choi, M. Jain, K. Srinivasan, P. Levis, and S. Katti. Achieving single channel, full duplex wireless communication. In *Proceedings of the 16th ACM Mobicom (Mobicom '10)*, September 2010.

[23] Federal Communications Commission. Facilitating opportunities for flexible, efficient, and reliable spectrum use employing spectrum agile radio technologies. *ET Docket*, (03-108), Dec. 2003.

[24] C. Cordeiro, K. Challapali, and M. Ghosh. Cognitive phy and mac layers for dynamic spectrum access and sharing of tv bands. In *TAPAS '06: Proceedings of the first international workshop on Technology and policy for accessing spectrum*, page 3, New York, NY, USA, 2006. ACM.

[25] C. M. Cordeiro, K. Challapali, and D. Birru. Ieee 802.22: An introduction to the first wireless standard based on cognitive radios. *Journal of communications*, 1, April 2006.

[26] L. H. A. Correia, D. F. Macedo, D. A. C. Silva, A. L. dos Santos, A. A. F. Loureiro, and J. M. S. Nogueira. Transmission power control in mac protocols for wireless sensor networks. In *MSWiM '05: Proceedings of the 8th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems*, pages 282–289, New York, NY, USA, 2005. ACM.

[27] C.Stevenson, G. Chouinard, Z. Lei, W. Hu, S. Shellhammer, and W. Caldwell. Ieee 802.22: The first cognitive radio wireless regional area network standard. *Communications Magazine, IEEE*, 47, January 2009.

[28] B. Danev, H. Luecken, S. Čapkun, and K. El Defrawy. Attacks on physical-layer identification. In *In Proceedings of the 3rd ACM Conference on Wireless Network Security (WiSec '10)*, pages 89–98, March 2010.

[29] B. Danev and S. Čapkun. Transient-based identification of wireless sensor nodes. In *In Proceedings of ACM/IEEE Conference on Information Processing in Sensor Networks (IPSN'09)*, 2009.

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

[31] G. D. Durgin. *Space-Time Wireless Channels*. Prentice Hall PTR, 2002.

[32] O. Edfors, M. Sandell, J. J. van de Beek, S. K. Wilson, and P. O. Börjesson. Ofdm channel estimation by singular value decomposition. *IEEE Transaction on Communications*, 46(7):931–938, July 1998.

[33] M. Edman and B. Yener. Active attacks against modulation-based radiometric identification. Technical Report TR 09-02, Rensselaer Polytechnic Institute, 2009.

[34] ETTUS. Usrp-universal software radio peripheral. `http://www.ettus.com`.

[35] J. Franklin, D. McCoy, P. Tabriz, V. Neagoe, J. V. Randwyk, and D. Sicker. Passive data link layer 802.11 wireless device driver fingerprinting. In *Usenix Security Symposium*, 2006.

[36] S. Ganeriwal, S. Capkun, C. Han, and M. B. Srivastava. Secure time synchronization service for sensor networks. In *Proceedings of 2005 ACM Workshop on Wireless Security (WiSe 2005)*, pages 97–106, September 2005.

[37] G. Ganesan and Y. Li. Cooperative spectrum sensing in cognitive radio networks. In *Proceedings of IEEE DySPAN*, pages 137–143, November 2005.

[38] L. P. Goh, Z. Lei, and F. Chin. Dvb detector for cognitive radio. In *ICC'07: Proceedings of the International Conference on Communications 2007*, pages 6460–6465, 2007.

[39] A. Goldsmith. *Wireless Communications*. Cambridge University Press, August 2005.

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

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

[42] L. J. Greenstein, V. Erceg, Y. S. Yeh, and M. V. Clark. A new path-gain/delay-spread propagation model for digital cellularchannels. *IEEE Transactions on Vehicular Technology*, 46:477–485, 1997.

[43] S. Hengstler, D. P. Kasilingam, and A. H. Costa. A novel chirp modulation spread spectrum technique for multiple access. In *Proceedings of the IEEE International Symposium on Spread Spectrum Techniques and Applications*, pages 73–77, September 2002.

[44] A.T. Hoang and Y.-C. Liang. Adaptive scheduling of spectrum sensing periods in cognitive radio networks. In *Proceedings of the the IEEE GLOBECOM 2007*, pages 3128–3132, November 2007.

[45] IEEE Std 802.15.4-2003. IEEE standard for information technology – telecommunications and information exchange between systems – local and metropolitan area networks – specific requirements – part 15.4: Wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (LR-WPANs).

[46] H. Kim and K. G. Shin. In-band spectrum sensing in cognitive radio networks: energy detection or feature detection? In *MobiCom '08: Proceedings of the 14th ACM international conference on Mobile computing and networking*, pages 14–25, 2008.

[47] T. Kohno, A. Broido, and K. C. Claffy. Remote physical device fingerprinting. *Dependable and Secure Computing*, 2(2):93–108, 2005.

[48] S. Koskie and Z. Gajic. A nash game algorithm for sir-based power control in 3g wireless cdma networks. *IEEE/ACM Transaction on networking*, 13(5):1017–1026, 2005.

[49] L. B. Kuechle. Selecting receiving antennas for radio tracking. `http://www.atstrack.com/PDFFiles/receiverantrev6.pdf`.

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

[51] T. Leibner. Network and infrastructure sharing in 2g networks. *Siemens*, 2004.

[52] M. Li, I. Koutsopoulos, and R. Poovendran. Optimal jamming attacks and network defense policies in wireless sensor networks. In *Proceedings of IEEE International Conference on Computer Communications (INFOCOM '07)*, 2007.

[53] Z. Li, W. Xu, R. Miller, and W. Trappe. Securing wireless systems via lower layer enforcements. In *In Proceedings of ACM Workshop on Wireless Security (WiSe'06)*, 2006.

[54] D. Liu, P. Ning, and W.K. Du. Detecting malicious beacon nodes for secure location discovery in wireless sensor networks. In *Proceedings of the 25th International Conference on Distributed Computing Systems (ICDCS '05)*, pages 609–619, June 2005.

[55] Y. Liu and P. Ning. Mimicry attacks against wireless link signature and defense using time-synched link signature. Technical Report TR-2011-17, NC State University, Computer Science Department, July 2011.

[56] Y. Liu, P. Ning, H. Dai, and A. Liu. Randomized differential dsss: Jamming-resistant wireless broadcast communication. In *Proceedings of the 2010 IEEE INFOCOM*, 2010.

[57] Ettus Research LLC. The USRP product family products and daughter boards. `http://www.ettus.com/products`. Accessed in August 2010.

[58] S. Mathur, W. Trappe, N. Mandayam, C. Ye, and A. Reznik. Radio-telepathy: extracting a secret key from an unauthenticated wireless channel. In *Proceedings of the 14th ACM International Conference on Mobile Computing and Networking (MobiCom '08)*, 2008.

[59] D.L. Mills. Internet time synchronization: The network time protocol. *IEEE Transactions on Communications*, 39(10):1482–1493, 1991.

[60] S. M. Mishra, A. Sahai, and R. Brodersen. Cooperative sensing among cognitive radios. In *ICC'06: Proceedings of the International Conference on Communications 2006*, volume 4, pages 1658–1663, 2006.

[61] J. Pang, B. Greenstein, R. Gummadi, S. Seshan, and D. Wetherall. 802.11 user fingerprinting. In *In Proceedings of the 13th annual ACM international conference on Mobile Computing and Networking (MobiCom'07)*, 2007.

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

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

[64] N. Patwari and S. K. Kasera. CRAWDAD utah CIR measurements. `http://crawdad.cs.dartmouth.edu/meta.php?name=utah/CIR`.

[65] N. Patwari and S. K. Kasera. Robust location distinction using temporal link signatures. In *MobiCom '07: Proceedings of the 13th annual ACM international conference on Mobile computing and networking*, pages 111–122, New York, NY, USA, 2007. ACM.

[66] A. Perrig, R. Canetti, D. Song, and D. Tygar. Efficient authentication and signing of multicast streams over lossy channels. In *Proceedings of the IEEE Symposium on Security and Privacy*, May 2000.

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

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

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

[70] Y. Qi, T. Peng, W. Wang, and R. Qian. Cyclostationarity-based spectrum sensing for wideband cognitive radio. In *CMC '09: Proceedings of the 2009 WRI International Conference on Communications and Mobile Computing*, pages 107–111, Washington, DC, USA, 2009. IEEE Computer Society.

[71] K. B. Rasmussen and S. Čapkun. Implications of radio fingerprinting on the security of sensor networks. In *In Proceedings of International ICST Conference on Security and Privacy in Communication Networks (SecureComm'07)*, 2009.

[72] K.B. Rasmussen, C. Castelluccia, T.S. Heydt-Benjamin, and S. Čapkun. Proximity-based access control for implantable medical devices. In *Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS '09)*, 2009.

[73] K.B. Rasmussen and S. Čapkun. Realization of rf distance bounding. In *Proceedings of the USENIX Security Symposium*, 2010.

[74] M. Rasti, A. Sharafat, and B. Seyfe. Pareto-efficient and goal-driven power control in wireless networks: a game-theoretic approach with a novel pricing scheme. *IEEE/ACM Transaction on networking*, 17(2):556–569, 2009.

[75] S. O. Rice. Mathematical analysis of random noise. *Bell System Technical Journal*, 24:46–156, 1945.

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

[77] R. Safaya. A multipath channel estimation algorithm using a kalman filter. `http://www.ittc.ku.edu/research/thesis/documents/rupul_safaya_thesis.pdf`.

[78] A. Sahai and D. Cabric. Cyclostationary feature detection. *Tutorial presented at the IEEE DySPAN 2005 (Part II)*, November 2005.

[79] R. A. Scholtz. Multiple access with time hopping impulse modulation. In *Proceedings of the 2008 IEEE MILCOM conference*, pages 447–450, 1993.

[80] Robert A. Scholtz. *Spread Spectrum Communications Handbook*. McGraw-Hill, 2001.

[81] Media Security and Reliability Council. Communications infrastructurevsecurity, access, and restoration working group, final report. `http://hraunfoss.fcc.gov/edocs_public/attachmatch/DOC-244430A1.pdf`, Feb. 2004.

[82] S. Shankar, C. Cordeiro, and K. Challapali. Spectrum agile radios: utilization and sensing architectures. In *Proceedings of IEEE DySPAN*, pages 160–169, November 2005.

[83] K. S. Shanmugan and A. M. Breipohl. *Random signals: detection, estimation, and data analysis*. Wiley, May 1988.

[84] S. Shellhammer. An atsc detector using peak combining. *IEEE 802.22-06/0243r0*, November 2006.

[85] S. Shellhammer, S. Shankar N., R. Tandra, and J. Tomcik. Performance of power detector sensors of dtv signals in ieee 802.22 wrans. In *TAPAS '06: Proceedings of the first international workshop on Technology and policy for accessing spectrum*, New York, NY, USA, 2006. ACM.

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

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

[88] D. Song, D. Zuckerman, and J.D. Tygar. Expander graphs for digital stream authentication and robust overlay networks. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, 2002.

[89] SPAN. Measured channel impulse response data set. `http://span.ece.utah.edu/pmwiki/pmwiki.php?n=Main.MeasuredCIRDataSet`.

[90] A. Springer, W. Gugler, M. Huemer, L. Reindl, C. C. W. Ruppel, and R. Weigel. Spread spectrum communications using chirp signals. In *Proceedings of the IEEE/AFCEA EUROCOMM Conference*, pages 166–170, May 2000.

[91] M. Strasser, B. Danve, and S. Capkun. Detection of reactive jamming in sensor networks. *ACM Transaction on Sensor Networks*, 7, Aug. 2010.

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

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

[94] K. Sun, P. Ning, C. Wang, A. Liu, and Y. Zhou. TinySeRSync: Secure and resilient time synchronization in wireless sensor networks. In *Proceedings of 13th ACM Conference on Computer and Communications Security (CCS '06)*, pages 264–277, October/November 2006.

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

[96] V. Tawil. Dtv signal captures. *IEEE 802.22-06/0038r0*, March 2005.

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

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

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

[100] W. Wang, H. Li, Y. L. Sun, and Z. Han. Attack-Proof collaborative spectrum sensing in cognitive radio networks. In *IEEE 43rd Annual Conference on Information Sciences and Systems*, 2009.

[101] X. Wang, M. Fei, and X. Li. Performance of chirp spread spectrum in wireless communication systems. In *Proceedings of the 11th IEEE Singapore International Conference on Communication Systems*, pages 466–469, Novemeber 2008.

[102] J. Wei and X. Zhang. Two-tier optimal-cooperation based secure distributed spectrum sensing for wireless cognitive radio networks. In *INFOCOM IEEE Conference on Computer Communications Workshops , 2010*, pages 1 –6, 2010.

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

[104] Wikipedia. Channel state information, 2011. [Online; accessed 14-April-2011].

[105] B. Wild and K. Ramchandran. Detecting primary receivers for cognitive radio applications. In *Proceedings of IEEE DySPAN*, pages 124–130, November 2005.

[106] M. Win and R. Scholtz. Impulse radio: How it works. *IEEE Communications Letters*, 2(2):36–38, February 1998.

[107] W. Xia, S. Wang, W. Liu, and W. Cheng. Correlation-based spectrum sensing in cognitive radio. In *CoRoNet '09: Proceedings of the 2009 ACM workshop on Cognitive radio networks*, pages 67–72, New York, NY, USA, 2009. ACM.

[108] W. Xu, K. Ma, W. Trappe, and Y. Zhang. Jamming sensor networks: Attack and defense strategies. *IEEE Network*, pages 41–47, 2006.

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

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

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

[112] J. Zhang, M. H. Firooz, N. Patwari, and S. K. Kasera. Advancing wireless link signatures for location distinction. In *MobiCom '08: Proceedings of the 14th ACM international conference on Mobile computing and networking*, New York, NY, USA, 2008. ACM.