# ABSTRACT

OH, YOUNG HYUN. Channel Detecting Jamming Attacks on Rendezvous Algorithms for Cognitive Radio Networks. (Under the direction of Dr. David J. Thuente.)

Traditional spread spectrum techniques such as Frequency Hopping Spread Spectrum (FHSS) and Direct Sequence Spread Spectrum (DSSS) have been used for anti-jamming solutions in wireless broadcast communication. However, these systems generally have the fundamental limitation of a prior key sharing between a sender and receiver. If a jammer is a compromised receiver, then it uses the secret key to jam the entire wireless communication. To address this problem, new enhancements allow a sender and receiver to independently generate random channel hopping (CH) or frequency hopping (FH) sequences so that it is unfeasible for a jammer to compute the same sequences. These schemes can provide fast rendezvous or key exchange methods for Cognitive Radio Networks (CRNs) called blind rendezvous algorithms in which a sender and receiver have no prior knowledge of a shared key, time synchronization information, or common control channels (CCCs).

However, we present new *channel detecting jamming attacks* (CDJAs) against these enhancements for CRNs. For most rendezvous algorithms, our channel detecting jammer can compute the same sequences as the sender's by utilizing the properties of blind rendezvous schemes. We investigated the state-of-the-art blind rendezvous algorithms for CRNs to demonstrate the effectiveness of our CDJAs. Through simulations, we show that CDJAs can significantly reduce their rendezvous probability for both the symmetric and asymmetric rendezvous systems. Thus, our CDJAs are a major security problem for most blind rendezvous algorithms since any secondary user or even group of users in CRNs can easily be denied access to the network with high probability. To mitigate this problem, we revisit the Random rendezvous scheme to increase the rendezvous probability against CDJAs. Overall, the Random scheme vastly outperforms these representative blind rendezvous algorithms for both the symmetric and asymmetric models when there are security concerns about a channel detecting jammer. A new partially random algorithm is shown to outperform all others for the asymmetric system.

# Channel Detecting Jamming Attacks on Rendezvous Algorithms for Cognitive Radio Networks

by
Young Hyun Oh

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

2014

APPROVED BY:

Dr. Harry G. Perros

Dr. Mihail L. Sichitiu

Dr. Khaled Harfoush

Dr. David J. Thuente
Chair of Advisory Committee

## DEDICATION

***To my parents, my lovely wife Hyemi, my brave son Brandon, and my sweet daughter Hannah.***

## BIOGRAPHY

Young-Hyun Oh was born in Geosan, South Korea. He received a B.S. degree in computer science from Kyung Hee University, Korea, in 2000 and an M.S. degree in computer science from Pennsylvania State University, USA, in 2005. He continued to pursue his Ph.D program in computer science at North Carolina State University under the supervision of Dr. David J. Thuente. His research interests were the network and system security. In particular, he focused on jamming attacks and their countermeasures to rendezvous algorithms for cognitive radio networks. During his Ph.D. studies, he also worked for IBM as a graduate CO-OP for three years and involved in several open cloud computing projects. He is an Apache Software Foundation (ASF) committer for the Apache VCL project since December, 2013. He is currently working for IBM as an advisory software engineer since December, 2013.

## ACKNOWLEDGEMENTS

First, I would like to express deepest thanks to my advisor, Dr. David J. Thuente, who devotes continuous and significant efforts to guide me for improving my research skills during the entire my doctoral program. I would also like to thank my committee members, alphabetically, Dr. Khaled Harfoush, Dr. Harry Perros, and Dr. Mihail Sichitiu for their valuable suggestions and advice to improve my dissertation. I would like to extend my appreciation to Dr. Peng Ning and Dr. Andy Rindos for their continuing encouragement and support. I am also grateful to my friends and colleagues, Dr. Attila Yavuz, Dr. Wu Zhou, Dr. Yao Liu, Dr. An Liu, Donghoon Kim, Dr. Yoonki Song, Dr. Jungki So, Dr. Younghee Park's family, Dr. Soon Chung's family, Dr. Byungwon On, and Dr. Dongkook Park's family for their supports. Finally, I would like to express my deepest gratefulness to my family and in-laws family. Words cannot describe my thanks for the unfailing faith, love, support, and encouragement they have provided with throughout time in my doctoral program.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

Chapter 1

# Introduction

Recently, wireless technology has been widely used and promoted by new wireless devices such as wireless sensors and smart phones. The emergence of these newer technologies in wireless networks has generated a serious problem of spectrum availability due to the lack of the freely available spectrum resources for wireless data communication. Moreover, managing the free spectrum resources (i.e., unlicensed spectrum) is challenging due to the limited resources compared to the growing demand. For example, since many researchers have intensively studied and developed wireless applications using the 2.4GHz spectrum, the band is now saturated by industry, scientific and medical (ISM) users. Thus, it can be difficult to implement wireless applications on this spectrum band without having interference with other users.

On the other hand, the licensed spectrum is assigned by the Federal Communications Commission (FCC) to the long-term licensed users (i.e., *primary users*) but the majority of the spectrum is under-utilized most of the time. (e.g., the occupancy of the licensed spectrum is less than $6\%$ [1]). This inefficient usage of the licensed spectrum has motivated the FCC [8] to allow the unlicensed users (i.e., *secondary users*) to temporarily access the unused spectrum to alleviate the problem of the limited unlicensed spectrum left to assign. That is, the secondary users (SUs) are able to opportunistically access the unused licensed spectrum using Dynamic Spectrum Access (DSA) techniques without interfering with the primary users (PUs) [8]. Cognitive Radios (CRs) are the key devices for the DSA techniques to dynamically and autonomously adjust their parameters to access the unused channels based on the spectrum sensing results.

In Cognitive Radio Networks (CRNs), however, a problem occurs when SUs try to exchange data and information with each other. Since the SUs are required to find a common channel from a large number of sensed open channels to establish the common control channel, the *rendezvous problem* occurs between SUs. The rendezvous problem occurs when two or more radios (i.e., secondary users) try to opportunistically find one another under the presence of the multiple PUs in a DSA network [6].

To solve this rendezvous problem in CRNs, many rendezvous algorithms have been proposed that can be categorized based on the presence of an assisted controller; aided rendezvous or unaided rendezvous [6]. Under the aided rendezvous, a centralized server controls the SUs' access to set up a common transmission link [5]. The SUs can also use dedicated common control channels to establish the transmission links [9, 20, 32, 39]. The advantages of these algorithms are the simple implementation and management of establishing the transmission links for the SUs. However, these algorithms have limitations such as reliability issues due to bottlenecks and single point failure on the control channel, and low flexibility and scalability issues. Moreover, security attacks on the control channel such as jamming and denial of service attacks can hinder the entire wireless communication. For example, suppose that traditional spread spectrum techniques, such as Frequency Hopping Spread Spectrum (FHSS) and Direct Sequence Spread Spectrum (DSSS) [18], are used for wireless communication to decrease jamming attacks. Spread spectrum techniques require a shared secret key between a sender and a receiver before the communication starts. Thus, it is unfeasible for a jammer to compute the same sequences or frequencies as the sender's without knowledge of the key. However, a compromised receiver can be a jammer in reactive jamming attacks where it uses the secret key to generate the same code sequences or frequencies to jam the wireless communication.

To avoid jamming attacks and other limitations on control channel(s), CRNs can use blind rendezvous algorithms which allow a sender and receiver to independently generate random channel hopping (CH) or frequency hopping (FH) sequences without the knowledge of time synchronization or CCCs. Thus, it is difficult for a jammer to compute the same code sequences or frequencies of the sender due to the randomness of channel selection. Many blind rendezvous channel hopping (CH) or frequency hopping (FH) algorithms have been proposed but we investigated the state-of-the-art for blind rendezvous schemes for CRNs [4, 6, 10, 12–14, 25, 26, 43].

Yang D. et al. proposed two rendezvous algorithms called the Deterministic Rendezvous Sequence (DRSEQ) [12] and the Channel Rendezvous Sequence (CRSEQ) [43]. In DRSEQ, each user generates the same CH sequence of $M$ available channels and it is guaranteed to rendezvous in at most $2M + 1$ time slots. This is called the maximum time to rendezvous (MTTR). When each user has the same set of $M$ available channels, we say this is the symmetric model. When the set of available channels is not the same for each user, we say we have an asymmetric model. The DRSEQ is only applicable to a symmetric model. In CRSEQ, each user constructs the CH sequences by using the properties of triangle numbers and Chinese Remainder Theorem (CRT). The CRSEQ provides the MTTR of the symmetric case at least $(P - 1)(3P - 1)$. However, both DRSEQ and CRSEQ sequences are completely deterministic and, we will see, can be easily jammed.

In [13], Luiz A. DaSilva et al. proposed a sequence-based CH rendezvous scheme in which all radios can generate the same CH sequences using the pre-defined sequence generators. The scheme supported the asynchronous systems (i.e., no time synchronization) and was referred to as Generated Orthogonal Sequence (GOS) algorithm in [6]. One fundamental limitation of GOS is that it is applicable only to the symmetric scenario.

In [6], Nick C. Theis et al. proposed the Modular-based Clock (MC) and Modified Modular-based Clock (MMC) CH rendezvous schemes to provide blind rendezvous for both symmetric and asymmetric scenarios respectively. Under the assumption that a distinct forward-hop is chosen by each user, MC and MMC can guarantee upper bounds on rendezvous time. The MC schemes do not require time synchronization between SUs. Using the MC algorithm, the sender and receiver can independently generate CH sequences among the same or different number of available channels. Then the sender and the receiver, with no time synchronization, can rendezvous in bounded time only if their forward-hops are different.

In [25], Lin Zhiyong et al. proposed the Jump-Stay (JS) based CH rendezvous schemes to provide guaranteed rendezvous for both symmetric and asymmetric scenarios. The JS rendezvous scheme solved limitations of other blind rendezvous schemes such as unbounded time for rendezvous, time synchronization requirements, not being applicable for multi-users, and working only for symmetric models. The JS schemes do not require time synchronization between the SUs. That is, in the JS system, the sender and receiver can rendezvous any time even though they independently generate the CH sequences among the same or different number of available channels and have hop sequences with different starting times. In [26], Lin Zhiyong proposed the Enhanced JS (EJS) rendezvous algorithms [25] by redesigning the hoping sequences. The EJS can reduce the expected time to rendezvous (ETTR) for the asymmetric model from $O(P^3)$ to $O(P^2)$ where $P$ is the smallest prime number greater than the total number of available channels.

In addition to the above rendezvous schemes for CRNs, we also exploited the Frequency Quorum-based Rendezvous (FQR) scheme for wireless networks [17] because it provides the lowest maximum time to rendezvous compared to other schemes. Moreover, it can be applicable to CRNs without difficulty [22]. In [17], Eun-Kyu Lee et al. proposed the FQR algorithm to provide fast rendezvous and secure key establishment under the various jamming attacks. The FQR scheme utilizes the non-empty property of cyclic quorum sets [19, 44]. Thus, the sender and receiver can select independent random FH sequences but they are guaranteed to rendezvous within a bounded time.

Moreover, we also investigated a general random spread spectrum technique called a delayed random seed disclosure DSSS (DSD-DSSS) scheme for broadcast wireless communications [28] because it is one of the most secure DSSS algorithms due to its unpredictable random spreading codes. The DSD-DSSS scheme removes the pre-shared key dependency of traditional DSSS techniques and allows a sender to randomly generate spread code sequences for each message using random seeds. Then the sender spreads each message using different code sequences and discloses the seeds at the end of each message. Therefore, without the knowledge of the seeds, the jammer cannot generate the same code sequences before the receiver receives the entire message.

However, we found that these blind rendezvous schemes are vulnerable to a new type of jamming attacks named *channel detecting jamming attacks (CDJAs)* in which a jammer can compute the same CH or FH sequences by utilizing the properties of rendezvous schemes. We exploit the effectiveness of CDJAs for representative symmetric rendezvous algorithms that are derived from Modular-based channel hopping algorithms such as Modular Clock (MC), Jump-Stay (JS), Enhanced Jump-Stay (EJS) channel hopping rendezvous algorithms [6, 25], and quorum-based frequency hopping algorithms named FQR algorithms [17], respectively. In addition, we will expand our CDJA to the remaining state-of-the-art blind rendezvous algorithms such as Generated Orthogonal Sequence (GOS) [6,13], Deterministic Rendezvous Sequence (DRSEQ) [12], and Channel Rendezvous Sequence (CRSEQ) [43]. In our CDJAs, the channel detecting jammer can compute the channel hopping rates using one or two listening channel. Using this information, the jammer can compute the sender's entire hopping sequences within a short period and then jam the remaining channel or frequency sequences. For the asymmetric models, we focus on the asymmetric EJS scheme because many of the rendezvous algorithms are not applicable for the asymmetric models and the asymmetric EJS scheme provides a fast guaranteed rendezvous compared to others. Our simulation results show that the rendezvous probability of these blind rendezvous schemes for both symmetric and asymmetric models will be dramatically decreased under the CDJA.

The contributions of this thesis are as follows:

1. We introduced a novel *channel detecting jamming attacks* (CDJAs) against the blind rendezvous algorithms for CRNs in which a jammer, with capabilities similar to a normal user, can find the sender's channel hopping sequences by utilizing the properties of their rendezvous algorithms.

2. We investigated the state-of-the-art blind rendezvous algorithms [6, 13, 24–26] to demonstrate the effectiveness of our CDJAs. The channel detecting jammer uses one or two listening channels to find the sender's CH sequence within a short period and computes the senders' CH sequences. Then it jams the remailing sequences so that the rendezvous probability of the state-of-the-art blind rendezvous algorithms will be significantly decreased under our CDJAs. Our simulation results demonstrate that the state-of-the-art blind rendezvous schemes are extremely vulnerable to the CDJAs.

3. We revisited the Random CH schemes to evaluate the rendezvous probability against CDJAs. This

completely defeats any predictability for the channel detecting jammer. In this thesis, we presented the theoretical expected time to rendezvous (ETTR) for the asymmetric Random system based on the number of common channels $G$ between a sender and receiver. The ETTR for the asymmetric Random is $\frac{|m_1|\cdot|m_2|}{|G|}$ where $|m_1|$ and $|m_2|$ is the number of available channels out of the total number of available channels $M$ for the sender and receiver, respectively. Through simulations and theoretical analysis, we demonstrate that the symmetric and asymmetric Random schemes for CRNs can be an effective, efficient and robust rendezvous scheme against CDJAs.

4. We investigated a Frequency Quorum-based Rendezvous (FQR) algorithm [24] and then we presented a *sophisticated jamming attack* in which a jammer can find the sender's quorum set within the second frame (i.e., $2k$ time slots where $k$ is the number of elements from a minimal difference set) [36]. Then the jammer can completely jam the sender's frequency hopping sequence after $2k$ time slots. Our simulation results show that the rendezvous probability of the FQR system under the sophisticated jamming attack dramatically decreased as the number of available channel increases.

5. We investigated a delayed random seed disclosure DSSS (DSD-DSSS) scheme [28] and then we presented a new type of jamming attack called a *seed jamming attack* in which an attacker particularly focuses on jamming the random seed(s) in fixed-size messages Thus, the receiver cannot despread received messages because it cannot find the seed and hence cannot regenerate the correct spread code sequences. To mitigate this jamming attack, we propose an advanced random seed DSSS (ARS-DSSS) scheme which strengthens the previous algorithm called DSD-DSSS by using an additional location seed. Our security analysis and implementation results demonstrate how to defeat the seed jamming attacks and how to reduce the computation overhead of the DSD-DSSS scheme.

The rest of the thesis is organized as follows. In Chapter 2, we presented Channel Detecting Jamming Attacks (CDJAs) against modular-based channel hopping (CH) rendezvous algorithms for the symmetric CRNs. In Chapter 3, we presented CDJAs against Jump-Stay based CH rendezvous algorithms for the symmetric CRNs. in Chapter 4, we presented CDJAs on symmetric blind rendezvous algorithms for CRNs. In Chapter 5, we presented CDJAs on Enhanced Jump-Stay (EJS) and countermeasures. In Chapter 6, we presented limitations of quorum-based rendezvous and key establishment schemes against sophisticated jamming attacks. In Chapter 7, we presented enhanced security of random seed DSSS algorithms against seed jamming attacks. Finally, in Chapter 8, we conclude this thesis.

# Chapter 2

# Channel Detecting Jamming Attacks against Modular-Based Channel Hopping Rendezvous Algorithms for Cognitive Ratio Networks

Efficient utilization of wireless bandwidth is a critical, if not the key, component of the wireless network architectures that balance availability and access. Cognitive Radio Networks (CRNs) are an important part of the solution to this problem. Common control channels (CCCs) for rendezvous in CRNs have limitations such as single point of failure, low scalability, and susceptibility to jamming attacks. Several rendezvous algorithms have recently been proposed that remove the need for CCCs. In particular, the Modular Clock (MC) channel hopping rendezvous algorithms provide extremely efficient rendezvous times for CRNs without using time synchronization and CCCs (i.e., blind rendezvous). In this chapter, however, we present new Channel Detecting Jamming Attacks (CDJAs) in which the jammer can estimate the channel hopping sequences within the first-half period of the MC algorithm. Using a single jammer and two listening channels, we show how to compute the entire MC Channel Hopping (CH) sequence and thus reduce the rendezvous success rate from around $95\%$ to around $15\%$ for the basic period. We show this is a major security problem for CRNs utilizing the MC algorithm since any secondary user or even group of users can easily be denied access to the network with high probability. We also compare these results to the Random rendezvous algorithm and show it vastly outperforms the MC algorithm when there are security concerns about a channel detecting jammer.

## 2.1 Introduction

In cognitive radio networks (CRNs), the rendezvous problem is difficult because of the dynamically changing of available channels at any given time. To address this problem, many different rendezvous algorithms have been proposed and categorized based on the presence of the assisted controller, aided rendezvous, and unaided rendezvous [6]. Under the aided rendezvous, a centralized server can control the access from the secondary users to set up a common transmission link [5]. The secondary users can also use dedicated common control channels to establish the transmission links [9, 20, 32, 39]. The advantages of these algorithms are the simple implementation and management of establishing the transmission links for the secondary users. However, these algorithms have limitations such as reliability issues due to bottleneck and single point of failure on the control channel, low flexibility and scalability issues, and security attacks on the control channel (e.g., jamming and denial of service attacks).

For the unaided rendezvous, secondary users can establish one or multiple common control channels by finding each other on one of the available channels. The secondary users can also *blind rendezvous* without having any centralized controller or dedicated CCCs. To achieve blind rendezvous, several channel hopping (CH) algorithms [4, 6, 10, 14] have been proposed but they have limitations such as unbounded time for rendezvous, time synchronization (TS) requirements, not being applicable for multi-users, and working only for symmetric models. To address these limitations, the Modular based clock CH rendezvous schemes [6] have been proposed to provide guaranteed rendezvous for both symmetric and asymmetric scenarios provided each node uses a distinct forward-hop. The MC schemes do not require time synchronization between the secondary users. That is, in the MC system, the sender and receiver can rendezvous any time even though they independently generate the CH sequences among the same or different number of available channels and have hop sequences with different starting times.

However, the MC systems are still vulnerable to the CDJAs where a jammer has the capability of listening on one or two channels and jamming one channel at a time (here the jammer's capabilities are close to normal secondary users). In this chapter, we focus on the symmetric MC scheme in which two secondary users have the same number of available channels since it more clearly illustrates the characteristics of our jamming attack. The asymmetric MC scheme is more complicated but the techniques we develop can be effectively be used on it as well. Under the CDJA, the jammer can take advantage of the MC scheme to determine the sender's CH sequences within $P$ time slots (here $P$ is the smallest prime number greater than or equal to the number of available channels $M$) by using two listening channels. Then the jammer jams the remaining time slots after an average of $\lfloor\frac{(P+1)}{2}\rfloor$ time slots. Therefore, the rendezvous probability of the MC scheme will be decreased dramatically because the sender and receiver must rendezvous within the first-half of one P period or less on average out of the MC period of 2P.

In this chapter, we present the CDJA and evaluate its effectiveness particularly on the MC schemes [6]. Moreover, we revisit the Random rendezvous scheme [6] where the sender and receiver generate their CH sequences by randomly selecting from the $M$ available channels for each time slot. Since the sender's CH sequences are totally random, it is unfeasible for the channel detecting jammer to estimate the sender's CH sequences.

The contributions of this chapter are: first, we introduce a novel CDJA model which can detect the sender's CH sequence within the $P$ time slots while using two listening channels. This can dramatically decrease the rendezvous probability of the MC schemes [6]. Second, we revisit the Random CH scheme to evaluate the rendezvous probability against CDJAs. This completely defeats any predictability for the channel detecting jammer. Our simulation results demonstrate that the MC scheme is extremely vulnerable to the CDJAs. On the other hand, the Random scheme for CRNs can be an effective, efficient and robust rendezvous scheme against CDJAs.

The rest of the chapter is organized as follows. Section 2.2 presents the MC scheme and the CDJA. Section 2.3 provides the simulations of our jamming attacks for both MC and Random schemes and discusses their results. Finally, Section 2.4 concludes the chapter.

## 2.2 Proposed Schemes

We propose a novel CDJA in which the jammer can dramatically decrease the probability of rendezvous for the MC scheme [6] by utilizing the characteristics of the modular-based CH algorithm in the MC scheme. In [6], the authors proposed both the symmetric and asymmetric scheme named Modified MC (MMC) but we only consider the symmetric model in this chapter. The expansion of the CDJA to the MMC scheme will be shown in a later chapter. In our CDJA, the jammer has capabilities close to the normal user with one transmitting channel but two listening channels. However, as we will show, the jammer can find the sender's CH sequences within $P$ time slots using two listening channels. Then it can jam the sender's remaining CH sequences after an average $\lfloor \frac{(P+1)}{2} \rfloor$ time slots. Therefore, the probability of rendezvous for the MC scheme will be dramatically decreased under the CDJA.

We first present the MC algorithm and show how the channel detecting jammer can find the sender's CH sequences in the MC scheme within such a short period of time. Then we analyze its effectiveness on the MC scheme in detail. We also revisit the Random CH rendezvous scheme to address jamming attacks. Finally, we compare the effectiveness of the CDJA on MC and Random schemes.

### 2.2.1 Modular Clock Algorithm

Theis et al. [6] proposed the MC schemes for both symmetric and asymmetric models (MMC) to provide fast rendezvous in CRNs. However, we present only the MC scheme because our jamming scheme is most clearly presented for the symmetric case.

In the MC scheme, each secondary user runs the *Modular Clock (MC)* CH algorithm to generate its CH sequences for the next $2P$ times slots. For example, consider the number of available channels, $M$ which is the same for all secondary users. Then each secondary user (e.g., unique user identification $i$) finds the smallest prime number $P \geq M$ and selects random channel $j_i^0$ and increment $r_i^0$ (i.e., $0 \leq j_i^0 < M$ and $0 \leq r_i^0 < P$) for the initial channel and the forward-hop rate of the user $i$, respectively.

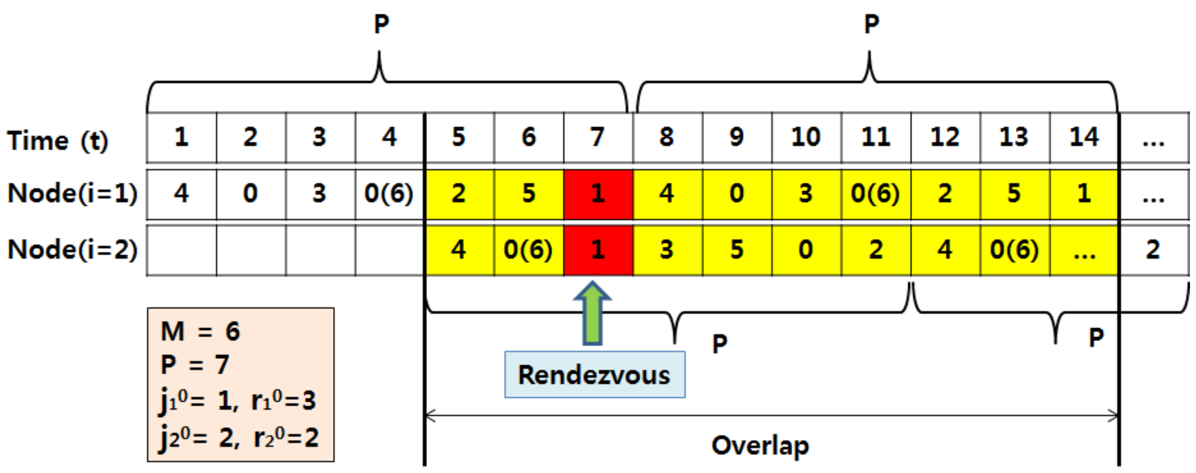


Figure 2.1: A rendezvous example of the MC scheme

Then the channel in the MC scheme [6] is determined by

$$c_i^t = (j_i^0 + t \times r_i^0)\ mod\ P \tag{2.1}$$

where $c_i^t$ is the channel of user $i$ at $t$-th time slot. Furthermore, the modular clock algorithm constantly remaps the channels in $[M, P)$ into channels $[0, M)$. Using this MC algorithm, each user generates CH sequences for a round of $2P$ time slots. Theis et al. [6] have shown that if two users run the CH sequences, without synchronization, from the MC scheme, the time to rendezvous (TTR) is guaranteed in $2P$ time slots only if the two forward-hops are different.

Figure 2.1 illustrates a rendezvous example of the MC scheme where $M = 6$ and $P = 7$. Since there is no time synchronization, users can start at any time slots within the $2P$ time slots of another user but the overlap between two CH sequences is not less than $P$ time slots. In this example, one user (e.g., $i = 1$) starts with the channel 4 at $t = 1$ when it randomly selects the initial channel $j_1^0 = 1$ and it jumps with the forward-hop $r_1^0 = 3$. The other user (e.g., $i = 2$) starts with the channel 4 at $t = 5$ when it randomly selects the initial channel $j_2^0 = 2$ and it hops with the forward-hop $r_2^0 = 2$. Then the two user rendezvous on channel 1 at time slot $t = 7$.

### 2.2.2 The channel detecting Jamming Attacks

In this subsection, we present a novel CDJA against the MC scheme. Here we consider one sender and one receiver in the CRNs trying to rendezvous under the CDJA. In the MC scheme, each secondary user uses the MC algorithm to generate its CH sequences. This guarantees that the sender and receiver can rendezvous within the maximum $2P$ time slots when there are no jamming attacks and their forward-hops are different. However, our channel detecting jammer takes advantage of the modular-based CH properties to find the sender's CH sequences. With two listening channels, it takes an average $\lfloor\frac{(P+1)}{2}\rfloor$ times slots for the jammer to find the sender's CH sequences and begins jamming in the first period. Moreover, it takes an average $\lfloor\frac{(P+1)}{4}\rfloor$ time slots for the next and all subsequent periods. Consequently, the jamming attack can dramatically decrease the rendezvous probability of the MC system.

**Assumptions:** We assume that the channel detecting jammer resides in the network before the beginning of the sender's communication and waits for the sender's signals with two listening channels. The sender is a secondary user who starts the communication first. Without loss of generality, we assume that the jammer can use spectrum sensing techniques to determine whether channels are used by other secondary users or not. Since we consider one sender and one receiver in this scenario, the jammer can observe the time when one user occupies one of the many available channels at a given time. Traditionally, the spectrum sensing techniques can determine whether a spectrum is used by primary users or not [11, 15] We assume that these techniques can be extended to spectrum sensing for the secondary users by sensing the pilots or energy of the secondary users.

For ease of presentation, we also assume the channel detecting jammer can detect the beginning of the sender's communication time. This assumption can be removed with only a very slight decrease in jamming effectiveness and no change in the first $2P$ jamming time slots for CDJA. The essential step to removing this assumption is to determine the end of the sender's first $2P$ time slots. A key part of the algorithm is the fact that $j_i^0$ is the last slot in the first $2P$ slots and the base for the first slot in the second $2P$ slots. Using the two increments of $r_i^0$, which we determine below, from the first and the second $2P$ slots allow us to determine $j_i^0$. The algorithm is more complex and we do not have space to develop it here.

**Finding the forward-hop:** Now we describe the algorithm the jammer uses to find the sender's CH sequences in the MC scheme [6]. In the CDJA, the fundamental goal for the jammer is to find the sender's CH sequences before the sender can rendezvous. Then it is easy to jam all of the sender's subsequent channel hops. Thus, it is critical for the jammer to find the sender's forward-hop $r_i^0$ quickly, since the forward-hop is the key value to generate the CH sequences in the MC scheme. If the channel detecting jammer can find the forward-hop of the sender, then the jammer can exactly generate the subsequent CH sequences of the sender so that it can jam the remaining channels in the CH sequences with a single jammer. We first give an example of the sender and receiver rendezvousing and show how the jammer determines the sender's CH sequences by hearing two different sender's signals using two listening channels. Then we analyze how the channel detecting jammer uses this information to interrupt the rendezvous in the MC system.

As we described the above, in the MC scheme [6], the channel $c_i^t$ is determined by Equation (2.1). Using the property of the modular clock in the MC algorithm, the channel detecting jammer can find the forward-hop $r_i^0$ by detecting the sender's signals from two different channels. Indeed, suppose that the jammer selects two distinct listening channels and receives the sender's signals on $c_i^{t_1}$ at time $t_1$ and on $c_i^{t_2}$ at time $t_2$. If the jammer is not detecting the beginning of the sender's CH sequences, then the time $t_1$ and $t_2$ are the jammer's time. We see below the important part is $(c_i^{t_2} - c_i^{t_1})$. In either case, the jammer can calculate the sender's forward-hop $r_i$ by:

$$c_i^{t_1} = (j_i^0 + t_1 \times r_i^0) \; mod \; P \tag{2.2}$$

$$c_i^{t_2} = (j_i^0 + t_2 \times r_i^0) \; mod \; P \tag{2.3}$$

and then Equation (2.3) $-$ (2.2) gives,

$$((t_2 - t_1) \times r_i^0) \; mod \; P = (c_i^{t_2} - c_i^{t_1}).$$

Thus, from properties of the mod function, the forward-hop $r_i^0$ is

$$r_i^0 = \lceil \frac{(P \times k + (c_i^{t_2} - c_i^{t_1}))}{(t_2 - t_1)} \rceil \tag{2.4}$$

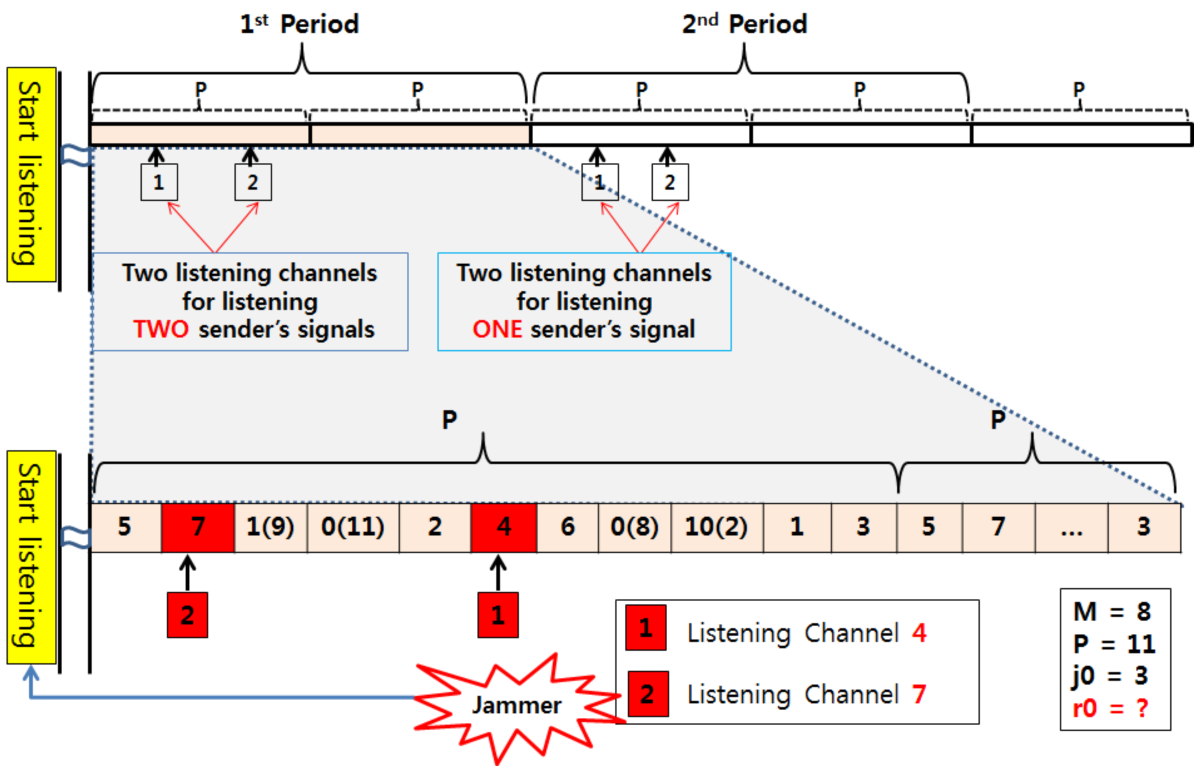


Figure 2.2: An example of finding the forward-hop in the MC scheme

where $i$ is the node's identification, $c_i^t$ is the channel at $t$-th time slot, $j_i^0$ is the initial channel for the node $i$, $r_i^0$ is the forward-hop rate for the node $i$, and the $k$ is the constant that satisfies the property of $0 \leq r_i^0 < P$. Once the jammer computes the forward-hop $r_i^0$, then it computes the initial channel $j_i^0$ from Equation (2.2) and exactly generates the sender's CH sequences from Equation (2.1). If the jammer does not know the beginning of the sender's communication, it can still compute the sender's next hop by just adding $r_i^0$ to the last hop and hence get every forward-hop. It is not necessary to get $j_i^0$ in this case.

Figure 2.2 describes how the channel detecting jammer finds the sender's forward-hop in the MC scheme using two listening channels for the first period. The channel detecting jammer selects the random listening channels to hear the sender's signals before the sender's communication starts. In this example, the partial CH sequence of the sender (e.g., $i = 1$) for the first $P$ time slots is

$$5, 7, 1(9), 0(11), 2, 4, 6, 0(8), 2(10), 1, 3$$

where $M = 8$, $r_1^0 = 2$, $j_1^0 = 3$, and $P = 11$. The CH sequences of the first $P$ time slots repeat for the next $P$ time slots. For the first period, the jammer needs to randomly selects two distinct listening channels to find the sender's forward-hop. In Figure 2.2, the jammer randomly selects distinct channels 4 and 7 and receives the signals on the channel 7 ($c_1^{t_1}$) at time $t_1 = 2$ and channel 4 ($c_2^{t_2}$) at time $t_2 = 6$ for the first and second time listening times. From the listening channels and time slot information, the jammer can compute the $r_1^0$ from Equation (5.4),

$$\begin{aligned} r_1^0 &= \lceil \frac{(P \times k + (c_2^{t_2} - c_1^{t_1}))}{(t_2 - t_1)} \rceil \\ &= \lceil \frac{(11 \times k + (4 - 7))}{(6 - 2)} \rceil = \lceil \frac{(11 \times k - 3)}{4} \rceil \\ & \quad 0 \leq \lceil \frac{(11 \times k - 3)}{4} \rceil < 11, \ \because 0 \leq r_1^0 < P \end{aligned}$$

then the constant $k$ is,

$$3 \leq \lceil 11k \rceil < 47, \ 0.27 \lesssim \lceil k \rceil \lesssim 4.27.$$

where $c_1^{t_1} = 7$ at $t_1 = 2$ and $c_2^{t_2} = 4$ at $t_2 = 6$. The possible values of $k$ are 1, 2, 3, and 4. Now the forward-hop $r_1^0$ is clearly constant so only one value of $k$ can be chosen. The proof of Lemma 2 in [25] actually shows that the $k$ value is unique when $c_1^{t_1}$ and $c_2^{t_2}$ are not duplicate channels. We can easily see that only $k = 1$ and the corresponding value of $r_1^0 = 2$ satisfies the CH sequences. From Equation (2.2), we easily see $j_1^0 = 3$.

Using two listening channels, the jammer can find the sender's CH sequences within a maximum $P$ time slots and an average $\lfloor \frac{(P+1)}{2} \rfloor$ time slots. To find the sender's forward-hop within $P$ time slots, it is necessary for the jammer to use two listening channels for the first time period. However, the jammer can find the forward-hop after hearing only one signal channel for the next and subsequent periods. Since the jammer learned $j_i^0$ from the first period, it can compute the new forward-hop $r_i^{0}$' when it detects one sender's signals at time $t$. That is, the jammer can find the new $r_i^{0}$' from known $t$, $c_i^t$, $P$, and

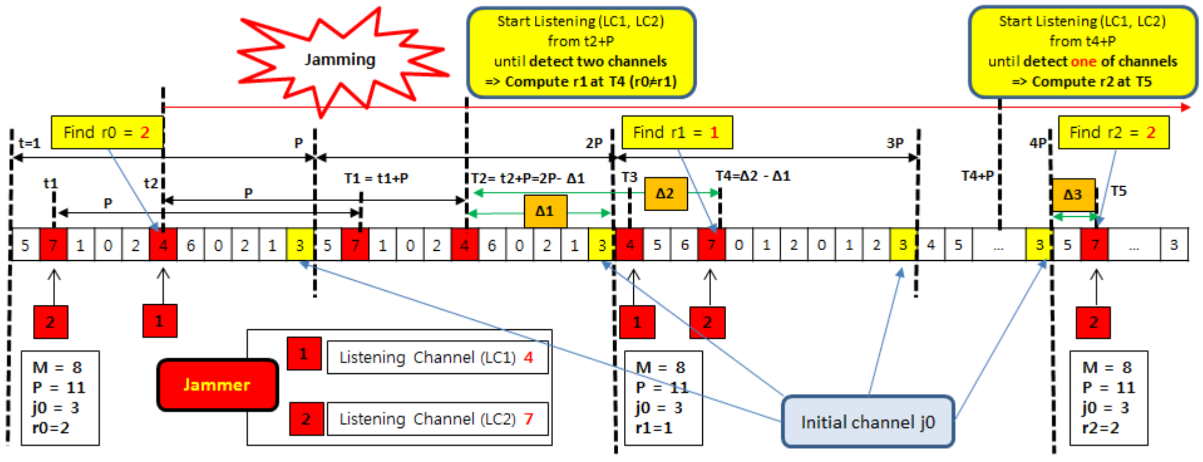


Figure 2.3: An example of finding the initial channel

$j_i^0$ using Equation (2.2). Therefore, the jammer can find the sender's CH sequence within an average $\lfloor\frac{(P+1)}{4}\rfloor$ time slots for the second and subsequent periods.

**Finding the beginning of the next period:** In the CDJA, the jammer has no information about the beginning of the sender's communication. Thus, it is important for the jammer to find the beginning of the next period because the sender randomly chooses a new forward-hop for every $2P$ period. The jammer needs to know the beginning of the next $2P$ period to determine when to find and use the new forward-hop using channel listening. We should stop using the current sequence to jam at $2P$. In other words, the jammer has heard the sender's signals from two listening channels but does not know their positions in the sender's sequence without the knowledge of the initial channel information.

However, our channel detecting jammer can find the exact positions of the detected channels in the sender's CH sequence when the forward-hops $r_i^0$ and $r_i^1$ change for the first $2P$ period and the next $2P$ period. Figure 2.3 describes how the jammer can find the both $r_i^0$ and $r_i^1$ using two listening channels (LCs) when the two forward-hops are different (i.e., $r_i^0 \neq r_i^1$). Using this detected information, the jammer can find the initial channel $j_0$ which is the last time slot of each period. In this example, the jammer selects two random channels $LC_1 = 4$ and $LC_2 = 7$ and listens to signals before the communication starts. Then the jammer receives two signals from $c_i^{t_1} = 7$ at time $t_1$ and $c_i^{t_2} = 4$ at time $t_2$ in the first $P$ period. Since the CH sequence repeats for the second $P$ period, the $c_i^{t_1}$ and $c_i^{t_2}$

will appear again at time $T_1 = t_1 + P$ and time $T_2 = t_2 + P$, respectively. When the jammer detects $c_i^{t_2} = 4$ at time $t_2$, it can compute both the forward-hop $r_i^0$ and the sender's CH sequence for the next $2P - max\{t_1, t_2\}$ period. Then the jammer starts to jam the channels in the order of the sequence from the time $t_2$. The jammer also starts to listen on the same LCs after the time $T_2$ to find the new forward-hop $r_i^1$ for the next $2P$ period. The jammer will detect two channels $c_i^{T_3} = 4$ at time $T_3$ and $c_i^{T_4} = 7$ at time $T_4$ in the $3P$ period as illustrated in Figure 2.3. The time slots will reset from 1 to $2P$ for every $2P + 1$ time slots. $\Delta_1$ is the gap between the time $T_2$ to the end of the second $P$ period and $\Delta_2$ is the gap between the time $T_2$ and the time $T_4$. Then the jammer can compute the new forward-hop $r_i^1$ at time $T_4$. All those time slots in this example are the sender's time slots. The jammer does not know their corresponding positions in the sender's sequence before it finds the initial channel $j_0$. But the jammer knows the gap $\Delta_2 = |T_4 - T_2|$. Also, $(T_2 + \Delta_1)$ is the time slot for the initial channel $j_0$. Now the jammer knows all the information except the gap $\Delta_1$. Once it computes the $\Delta_1$, it can find the initial channel $j_0$. The $\Delta_1$ can be computed by using the two detected channel's information $c_i^{T_2}$ and $c_i^{T_4}$. From Equation (2.1), $c_i^{T_2}$ at time $T_2 = 2P - \Delta_1$ and $c_i^{T_4}$ at time $T_4 = \Delta_2 - \Delta_1$ are

$$\begin{aligned} c_i^{T_2} &= (j_i^0 + T_2 \times r_i^0)\ mod\ P \\ &= (j_i^0 + (t_2 + P) \times r_i^0)\ mod\ P \\ &= (j_i^0 + (2P - \Delta_1) \times r_i^0)\ mod\ P \end{aligned} \tag{2.5}$$

$$\begin{aligned} c_i^{T_4} &= (j_i^0 + T_4 \times r_i^1)\ mod\ P \\ &= (j_i^0 + (\Delta_2 - \Delta_1) \times r_i^1)\ mod\ P \end{aligned} \tag{2.6}$$

where $j_0$ is the initial channel, $P$ is the smallest prime number greater than or equal to the number of available channels $M$, $r_i^0$ is the forward-hop for the first $2P$ period, and $r_i^1$ is the forward-hop for the next $2P$ period, $\Delta_1$ is the gap between the $T_2$ and the last time slot of the second $P$ period, and $\Delta_2$ is the time slots between the $c_i^{T_2}$ and the $c_i^{T_4}$. Thus, if $r_i^0 \neq r_i^1$, the $\Delta_1$ can be derived from Equation (2.5)

– Equation (2.6),

$$\begin{aligned} c_i^{T_2} - c_i^{T_4} &= ((j_i^0 + (2P - \Delta_1) \times r_i^0) - (j_i^0 + (\Delta_2 - \Delta_1) \times r_i^1))\ mod\ P \\ &= (2P \times r_i^0 + \Delta_1(r_i^1 - r_i^0) - \Delta_2 \times r_i^1)\ mod\ P \\ \Delta_1 &= \lceil \frac{(c_i^{T_2} - c_i^{T_4}) + P \times k + \Delta_2 \times r_i^1 - 2P \times r_i^0)}{r_i^1 - r_i^0} \rceil, for\ 0 \leq \Delta_1 \leq P - 2 \end{aligned} \quad (2.7)$$

where $r_i^0 \neq r_i^1$ and $k$ is the constant that satisfies the property for $0 \leq \Delta_1 \leq P - 2$. From the example, we can compute the $\Delta_1$ using Equation (2.7),

$$\begin{aligned} \Delta_1 &= \lceil \frac{(c_i^{T_2} - c_i^{T_4}) + P \times k + \Delta_2 \times r_i^1 - 2P \times r_i^0}{r_i^1 - r_i^0} \rceil, for\ 0 \leq \Delta_1 \leq P - 2 \\ &= \lceil \frac{(4 - 7) + 11 \times k + 9 \times 1 - 2 \times 11 \times 2}{1 - 2} \rceil, \\ & 0 \leq \lceil 38 - 11 \times k \rceil \leq 9, \because\ 0 \leq \Delta_1 \leq 9 \end{aligned}$$

then the const $k$ is,

$$27 < \lceil 11 \times k \rceil < 38$$

Thus, only the constant $k = 3$ satisfies the property and the corresponding value of $\Delta_1$ is 5. Therefore, the jammer can compute the initial channel $j_0 = 3$ that is the channel of $T_2 + \Delta_1$ time slot in the second $P$ period. The beginning of the next period is $T_2 + \Delta_1 + 1$. However, if the forward-hop $r_i^0$ and $r_i^1$ are the same, the jammer cannot find the initial channel from Equation (2.7). The jammer needs to repeat the same procedure for the next $2P$ period and to wait until it finds a different forward-hop (e.g., $r_i^1 \neq r_i^2$) to compute the initial channel $j_0$.

For both cases of the same forward-hop or different forward-hop, the jammer can keep jamming from the time $t_2$. The jammer avoids jamming on the two LCs for the next $2P$ period because it needs to compute the new forward-hop $r_i^1$. If the jammer detects the $c_i^{T_1}$ at the $T_1$ and $T_1 + P$ time slots and $c_i^{T_2}$ at the $T_2$ and $T_2 + P$ time slots, the jammer knows $r_i^0$ and $r_i^1$ are the same. The jammer cannot find the initial channel $j_0$ in this case but it can keep jamming the channels. The jammer repeats this process of finding the $j_0$ until it finds a different forward-hop. If two forward-hops are different, the jammer can compute the initial channel $j_0$ and then knows the beginning of the next $2P$ period. Once the jammer knows the initial channel $j_0$, the jammer only needs to know one channel information (i.e., either $LC_1$ or $LC_2$) to compute the new forward-hop for every $2P$ period after the $4P$ time slot. For example, as illustrated in Figure 2.3, the jammer heard the channel $c_i^{T_5}$ at time $T_5$ which is the $\Delta_3$ time slots after the time slot of the $j_0$ in the $4P$ period. $\Delta_3$ is the gap between the time slot of $j_0$ in the $4P$ period and the time $T_5$. Since $c_i^{T_5}$ is mapped onto only once by Equation (2.1), there exists a unique $r_i^2$ such that

$$c_i^{\Delta_3} = c_i^{T_5} = (j_i^0 + \Delta_3 \times r_i^2) \; mod \; P.$$

where $c_i^{\Delta_3}$ is the $\Delta_3$-th time slot of the sender's CH sequence and the $r_i^2$ is in $1 \leq r_i^2 < P$. From the above the example,

$$c_i^{\Delta_3} = 7 = (3 + 2 \times r_i^2) \; mod \; 11.$$

where $c_i^{T_5} = 7$, $j_i^0 = 3$, $\Delta_3 = 2$, and $P = 11$. Then the unique value of $r_i^2$ is 2 in $1 \leq r_i^2 < P$. There are possible rendezvous on an average of $\lceil \frac{P}{2} \rceil$ time slots between the $2P + 1$ and $4P$ time slots. But the rendezvous probability is on an average $\lceil \frac{P}{4} \rceil$ for every $2P$ period after the $4P$ time slot once $j_0$ is found.

**Listening channel selection:** The channel detecting jammer can randomly choose two listening channels from all the available channels. Obviously, the jammer must select distinct listening channels for the first period. It is expedient (makes the algorithm and its analysis simpler) but not necessary for the jammer to avoid selecting channels from $[0, P - M)$ as these are duplicate channels in the MC algorithm [6] that are mapped onto by channels $[M, P)$. If the jammer selects the channels from $[0, P - M)$, then it is possible that the forward-hop computation allows multiple or no integer value for $k$ since the MC algorithm maps $[M, P)$ onto them. For example, suppose that the jammer selects 0 and 1 for listening channels of the sender (e.g., $i = 1$) from the above example. Then, the jammer can detect the signals on the channel $c_1^{t_1} = 1$ (9) at time $t_1 = 3$ and the channel $c_1^{t_2} = 0$ at time $t_2 = 4$. The forward-hop $r_1^0$ from Equation (5.4) is

$$\begin{aligned} r_1^0 &= \lceil \frac{(P \times k + (c_1^{t_2} - c_1^{t_1}))}{(t_2 - t_1)} \rceil \\ &= \lceil \frac{(11 \times k + (0 - 1))}{(4 - 3)} \rceil = \lceil 11 \times k - 1 \rceil \\ & \quad 0 \leq \lceil 11 \times k - 1 \rceil < 11, \ \because 0 \leq r_1^0 < P \end{aligned}$$

then the constant $k$ is,

$$1 \leq \lceil 11k \rceil < 12, \ \ 0.09 \lessapprox \lceil k \rceil \lessapprox 1.09$$

We can easily see that only $k = 1$ and the corresponding value of $r_1^0 = 10$. However, the $r_1^0$ does not satisfied the channel sequence in this case.

To solve this problem, the jammer computes the forward-hop $r_1^0$ using $c_1^{t_1} = 9$ instead of $c_1^{t_1} = 1$. Then the jammer can correctly compute the forward-hop $r_1^0 = 2$. However, the jammer cannot immediately determine whether the $c_1^{t_1}$ is the channel 1 or the channel 9 $(1)$. Thus the jammer needs to compute the forward-hop for both $c_1^{t_1} = 1$ and $c_1^{t_1} = 9$ and generates the sender's next CH for both forward-hops. Then the jammer listens on one channel and jams the other for one slot. This clearly determines which CH sequence is correct. To remove this ambiguity, we assume that the jammer avoids selecting duplicated channels. The number of duplicated channels is also relatively small compared to $M$ (e.g., the maximum number of duplicated channels is 7 for $M = 89$, where $3 \leq M \leq 100$). We can, in fact, improve our channel finding algorithm by choosing one or two channels in $[0, P - M)$ since it doubles the number of possible "hits" and it will further degrade the MC scheme Our simulation results listen for channels not in $[0, P - M)$.

### 2.2.3 Random Channel Hopping Rendezvous

The rendezvous probability of the MC scheme can be dramatically decreased under the CDJA. To mitigate this jamming attack, we consider Random CH rendezvous algorithms. There are no guaranteed rendezvous algorithms under any jamming attacks. The Random CH rendezvous algorithm is a robust CH algorithm against jamming attacks. In the Random system, a sender and receiver randomly select one channel out of the $M$ available channels for each time slot. The formula $\sum(1/M)(1-1/M)^{k-1} * k = M$ shows the expected TTR for the Random scheme is $M$ time slots when there are no jamming attacks. However, it is unfeasible for any jammer to estimate when the Random sender and receiver might rendezvous. That is, the expected TTR for the Random scheme would be almost the same as $M$ time slots under the CDJAs. Hence, the Random scheme for CRNs can be an effective, efficient and secure rendezvous scheme against CDJAs.

## 2.3 Simulation results

We implemented the CDJA on Matlab 2010b to evaluate its effectiveness against the MC scheme. We implement the natural scenario where the jammer resides in the network and is listening on two distinct channels before the communication starts. The jammer has one jamming channel for the MC scheme. We also implemented the Random rendezvous scheme to demonstrate that it is more robust against the CDJA compared to the MC scheme. The jamming attack for the Random scheme jams two random channels for the entire rendezvous time.

First we compare the probability of rendezvous for both MC and Random schemes when there are no jamming attacks. The rendezvous time is always measured from the start time of the receiver. Figure 2.4 depicts the rendezvous probability for both schemes within $P$ time slots where the number of available channels $M$ varies from 3 to 100. We ran 1000 simulations for each available channel $M$ and calculated the average probability of rendezvous for both schemes. In this case, the sender and receiver start to rendezvous at the same time so we can compare these results with the theoretical probability of MC rendezvous described in [6]. Our simulation results are close to the theoretical values. This simultaneous start of sender and receiver is the most optimistic, but very unrealistic, scenario for the MC algorithm. The MC scheme does not required time synchronization (TS) between the sender and receiver. Figure 2.5 depicts the probability of rendezvous for both MC and Random without TS. That is, the receiver can start at any time slot after the sender starts. The probabilities of rendezvous for the MC

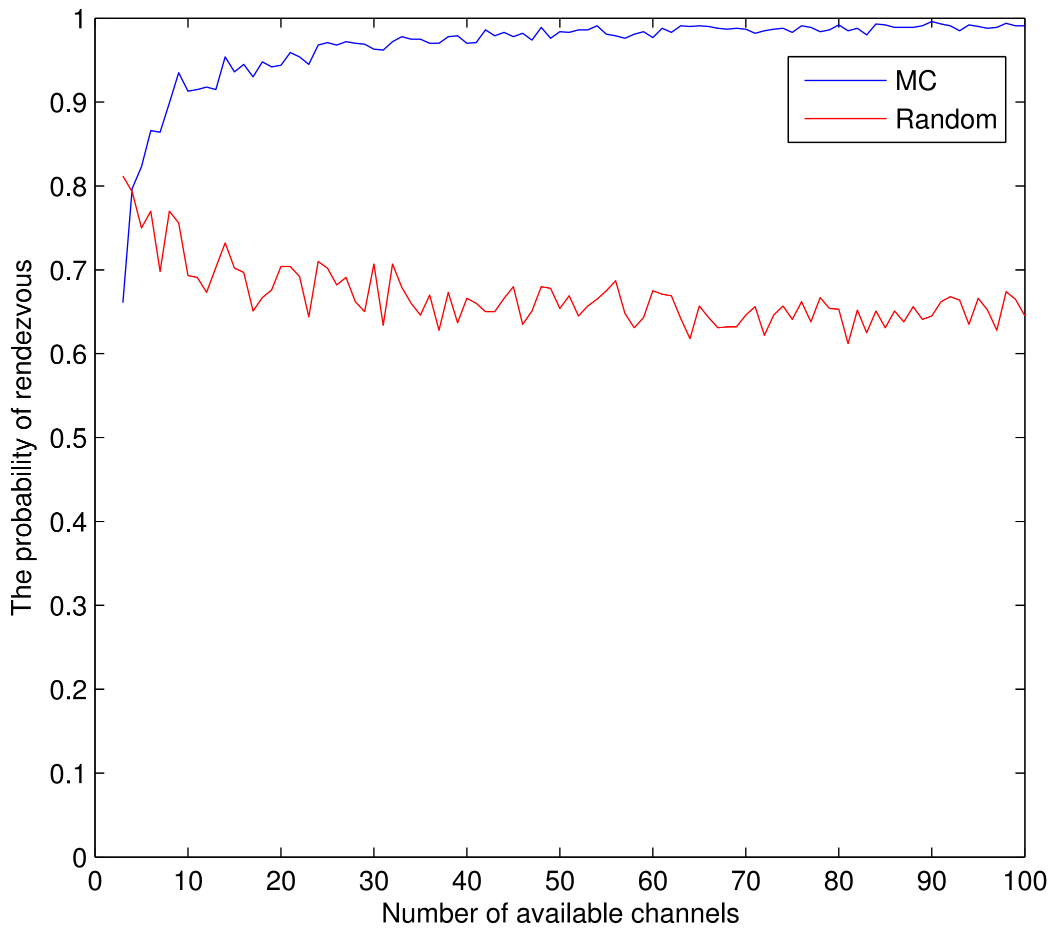


Figure 2.4: The rendezvous probability for the MC and Random schemes with time synchronization (TS) for $\leq P$ time slots

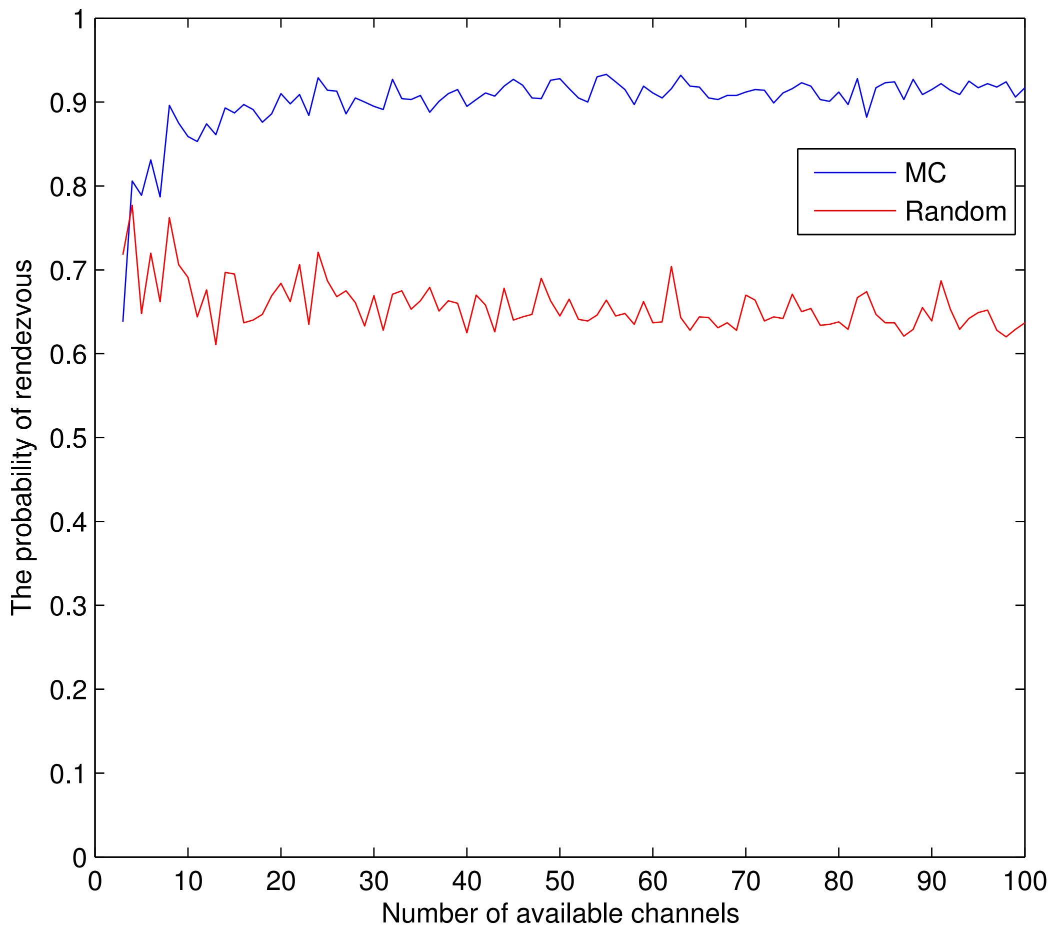


Figure 2.5: The rendezvous probability for the MC and Random schemes without time synchronization (TS) for $\leq P$ time slots

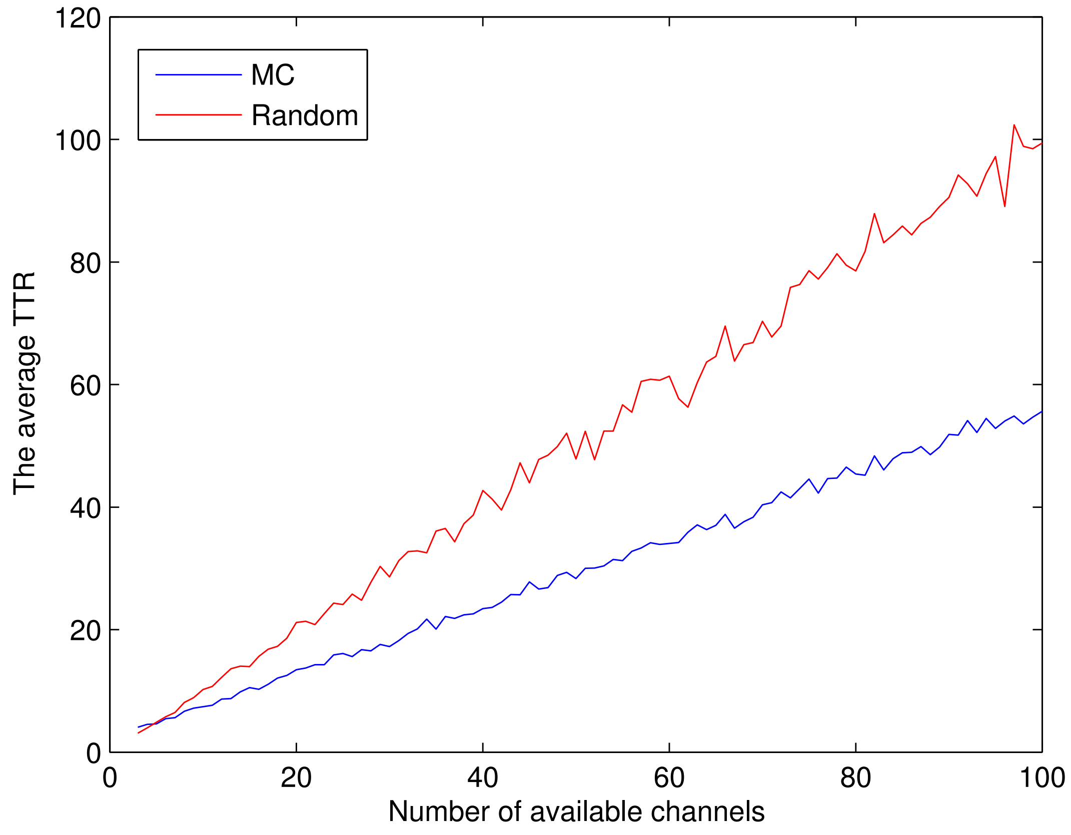


Figure 2.6: The average time to rendezvous (TTRs) for the MC and Random without time synchronization (TS) (100% success rate)

scheme decrease around $5\%$ for no start synchronization case but the probability of rendezvous for the Random scheme is essentially unchanged.

Next we compare the average TTRs for both MC and Random schemes without jamming attacks. Since both MC and Random schemes cannot guarantee the rendezvous, the sender and receiver must continue their algorithms until rendezvous occurs. Thus the rendezvous success rates is $100\%$ for both schemes. Figure 2.6 gives the average TTRs for MC and Random schemes with no TS between the sender and receiver. This figure shows that the average TTRs for both schemes increase steadily as the number of available channels increases. For the MC scheme, the maximum expected TTR is bound by $\frac{2P^2}{P-1}$; see [6]. The average TTR from our implementation is slightly over half the available channels $M$

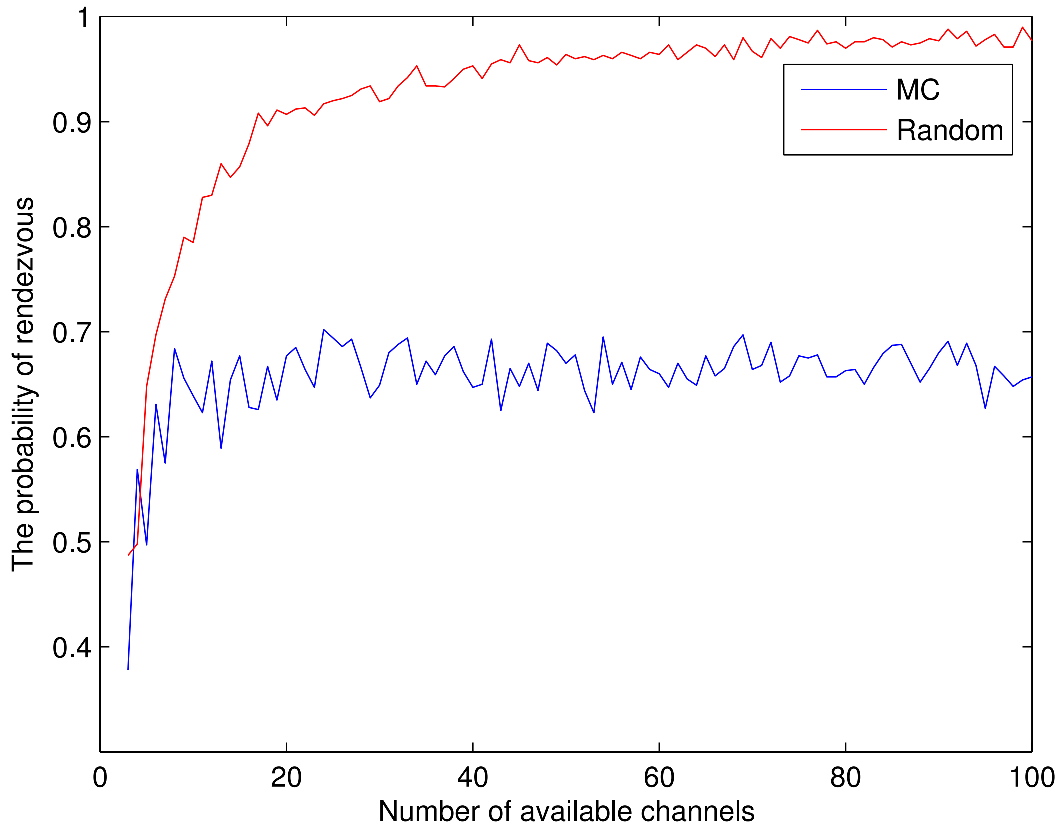


Figure 2.7: The probability of rendezvous for the MC and Random with TS under the CDJA ($\leq 2P$ time slots)

which is an extension of but consistent with the results from [6]. The sender and receiver frequently rendezvous during the first overlap of $\frac{P}{2}$ time slots. The average TTR for the Random scheme is close to $M$ time slots.

Now we compare the probability of a rendezvous for both MC and Random schemes under the CDJA. We implemented both TS and no TS cases between the sender and receiver for both schemes. Since the channel detecting jammer always resides in the network before the communication, the jammer can detect the sender's CH sequence in an average $\lfloor\frac{(P+1)}{2}\rfloor$ time slots with two listening channels. Then the jammer can jam the remaining CH sequence with a single jammer. Figure 2.7 shows the probability of rendezvous for both systems with TS until the end of $2P$ time slots. The probability

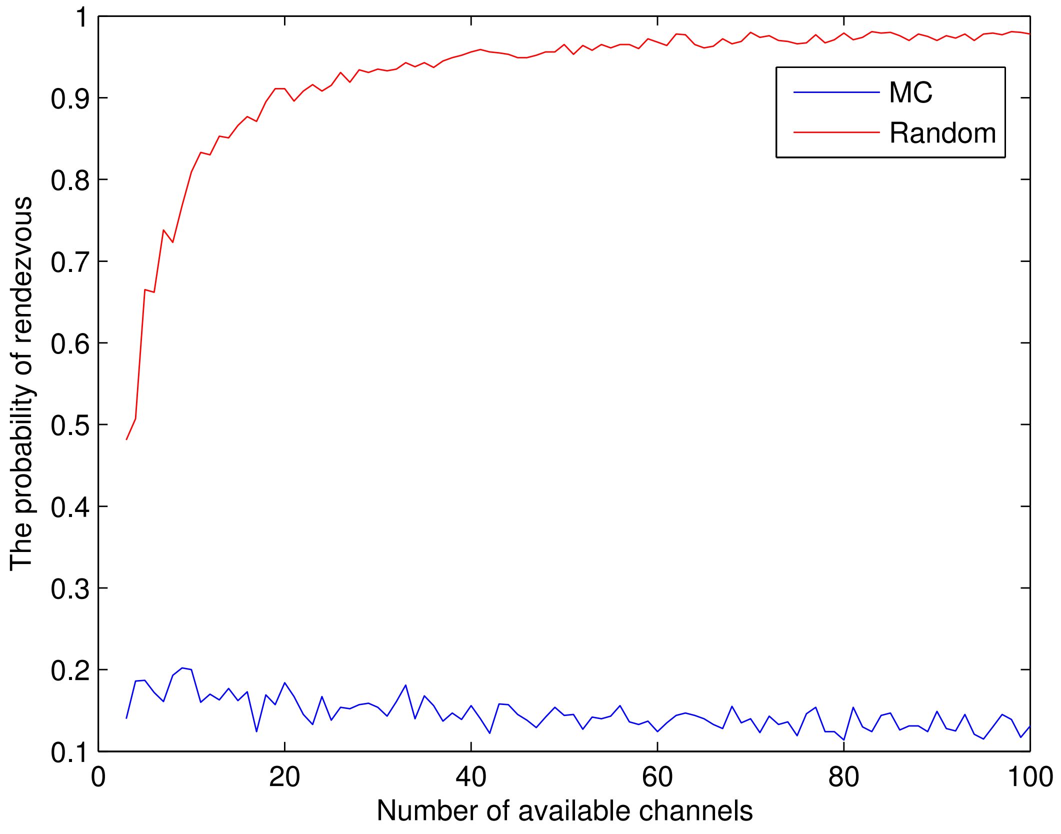


Figure 2.8: The probability of rendezvous for the MC and Random without TS under the CDJA ($\leq 2P$ time slots)

of rendezvous for the MC scheme decreases from around $95\%$ to around $65\%$. Figure 2.8 shows the probability of rendezvous for both systems without sender/receiver start time synchronization (normal situation). In this case, the probability of rendezvous for the MC scheme is dramatically decreased under the CDJA. The rendezvous probability is less than $15\%$ for almost all available channels numbers. However, the Random scheme rendezvous probabilities for both cases are almost the same as when there are no jamming attacks. It is over $90\%$ for most $M$.

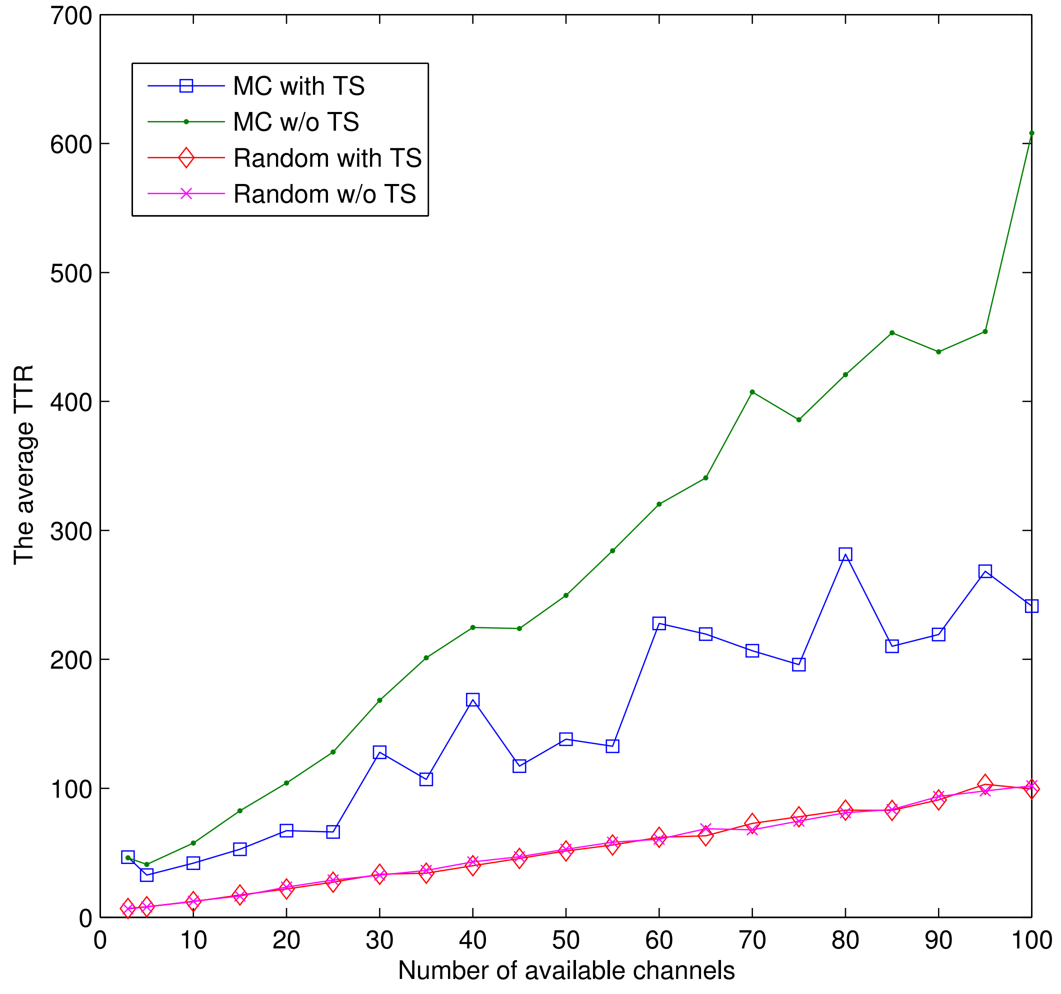


Figure 2.9: The average TTRs for the MC and Random under CDJAs (100% success rate)

Finally, we compute the average TTRs for MC and Random schemes under the CDJAs. We calculated the average TTRs by running the algorithms until rendezvous occurred (i.e., success rendezvous rates are $100\%$ for both schemes with TS and without TS cases). Figure 2.9 gives the average TTRs for MC and Random schemes. This figure shows that the average TTRs for the Random scheme are approximately $M$ time slots for both TS and no TS cases which are close to the TTR for the no jamming attack case. However, for most $M$, the expected TTRs for the MC scheme are close to $3P$ time slots and $5P$ time slots for the TS and no TS respectively. Therefore, the CDJA with a single jammer is extremely effective for the MC scheme but is minimally so for the Random scheme. The MC algorithm itself can be very modestly improved against CDJA by selecting a new channel $j_i^0$ every $2P$ time slots.

## 2.4 Conclusion

In this chapter, we presented a novel jamming attack called *CDJA* and demonstrated its effectiveness against the MC scheme [6]. The channel detecting jammer is able to take advantage of the MC algorithm (its method of generating its CH sequence) to find the sender's CH sequence within the first $P$ time slots using two listening channels. Then the jammer can completely jam the sender after an average $\lfloor \frac{(P+1)}{2} \rfloor$ time slots using just a single channel jammer. To remedy this jamming problem, we revisited the Random CH rendezvous scheme. Our simulation results demonstrate that the rendezvous probability of the MC system under the CDJA is dramatically decreased for all available channels $M$ (e.g., to around $15\%$ for most $M$). On the other hand, the rendezvous probability of the Random scheme is almost steady for all $M$ (nearly $90\%$). Therefore, the Random scheme can be an effective, efficient and robust rendezvous scheme against CDJAs. Since MC does not guarantee TTR and it is very susceptible to jamming attacks and the Random scheme provides shorter expected TTR, it appears that the Random scheme provides more security when timely rendezvous for CNRs are required.

Chapter 3

# Channel Detecting Jamming Attacks against Jump-Stay Based Channel Hopping Rendezvous Algorithms for Cognitive Radio Networks

In this chapter, we exploited the Jump-Stay (JS) based channel hopping rendezvous algorithms because it can provide guaranteed rendezvous for CRNs with no time synchronization or CCCs (i.e., blind rendezvous). However, the JS algorithms are still vulnerable to Channel Detecting Jamming Attacks (CDJAs) in which the jammer can estimate the channel hopping sequences within the first jump-pattern. The jammer can compute the entire JS channel hopping sequence and thus reduce the rendezvous success rate from $100\%$ to less than $20\%$ and $10\%$ using one and two listening channels respectively. To mitigate this problem, we revisit both the Random rendezvous scheme and the Role-based Channel Rendezvous (RCR) scheme extended from role-based rendezvous algorithms to increase the probability of the rendezvous against the CDJAs. We also compare the JS algorithm to both the Random and RCR algorithms and show the Random and RCR vastly outperform the JS algorithm when there are security concerns about a channel detecting jammer. Especially, the effectiveness of CDJA is negligible for the Random and RCR schemes but their expected time to rendezvous (TTR) is close to the JS's expected TTR.

## 3.1 Introduction

In the unaided rendezvous [6], the secondary users (SUs) can establish one or multiple common control channels by finding each other on one of the available channels. The SUs can also *blind rendezvous* without having any centralized controller or dedicated CCCs. To achieve blind rendezvous, several channel hopping (CH) algorithms [4, 6, 10, 14] have been proposed but they have the limitations such as unbounded time for rendezvous, time synchronization requirements, not being applicable for multi-users, and working only for symmetric models. To address these limitations, the Jump-Stay (JS) based channel hopping rendezvous schemes [25] have been proposed to provide guaranteed rendezvous for both symmetric and asymmetric scenarios. The JS schemes does not require time synchronization between the SUs. That is, in the JS system, the sender and receiver can rendezvous any time even though they independently generate the CH sequences among the same or different number of available channels and have hop sequences with different starting times.

However, the JS algorithms are still vulnerable to channel detecting jamming attacks (CDJAs) where a jammer has the capability of listening on one or two channels and jamming one channel at a time (e.g., the jammer's capabilities are close to normal SUs). We first focus on the symmetric JS scheme in which two SUs have the same number of available channels because it is easier to present the CDJA in this setting. The asymmetric JS scheme is solved after the basic techniques are set. Under the CDJAs, the jammer can take advantage of the JS scheme so that it can determine the sender's channel hopping sequences within $2P$ time slots or $P$ time slots (here $P$ is the smallest prime number greater than the available channels $M$) using one listening or two listening channels respectively. Then the jammer jams the remaining time slots after an average $P$ time slots for one listening channel and an average $\lfloor \frac{(P+1)}{2} \rfloor$ time slots for two listening channels. Therefore, the rendezvous probability of the JS scheme will be decreased dramatically because the sender and receiver must rendezvous within the first period or less on average.

In this chapter, we present the CDJA and evaluate its effectiveness on the JS schemes. Moreover, we revisit both the Random rendezvous scheme and the Role-based Channel Rendezvous (RCR) scheme extended from the role-based rendezvous algorithm [16]. For example, the sender and receiver in the Random scheme randomly select one of the available channels $M$ for each time slot. However, in the RCR scheme, the sender generates its channel hopping sequences by permuting the $M$ slots for each period and the receiver selects one random channel from $M$ and waits there until the end of $2P$ slots. The main difference between the Random and RCR schemes is that the RCR scheme can guarantee the rendezvous when there are no jamming attacks but requires the roles of sender and receiver. Since the sender's channel hopping sequences are random for both schemes, it is unfeasible for the channel detecting jammer to estimate the sender's sequences.

The primary contribution of this chapter is that we show how to mount a debilitating jamming attack against the JS rendezvous scheme. We introduce a novel CDJA model which can detect the sender's channel hopping sequence within the $2P$ time slots or $P$ time slots while using one or two listening channels respectively. This can dramatically decrease the rendezvous probability of the JS schemes. As an alternative to the JS scheme, we revisit the Random and RCR schemes to increase the rendezvous probability against the CDJAs. They effectively decrease predictability for the jammer to find the sender's CH sequence in CRNs. Our simulation results demonstrate that the JS scheme is extremely vulnerable to the CDJAs. On the other hand, the Random and RCR schemes for CRNs can be an effective, efficient and robust rendezvous scheme against CDJAs. In a benign environment, the expected TTR of the RCR and JS schemes are almost the same.

The rest of the chapter is organized as follows. Section 3.2 describes relevant background for the JS scheme. Section 3.3 presents the CDJAs and Random and RCR schemes. CDJA extensions follow in Section 3.4. Section 3.5 provides the simulations of our jamming attacks on the JS, Random, and RCR schemes and discusses their results. Finally, Section 3.6 concludes the chapter.

## 3.2 Preliminary

Lin Zhiyong et al. [25] proposed the JS schemes for both symmetric and asymmetric models to provide guaranteed rendezvous in CRNs [25]. We present the JS scheme for the symmetric model in this section since it clearly illustrates the characteristics of our jamming attack. Moreover, our attack is extended in Section 3.4 to the asymmetric JS scheme.

In the JS scheme, each SU runs the *JSHopping* algorithm to generate its CH sequences that consists of a jump-pattern and a stay-pattern. Each SU first performs the jump-pattern twice to jump on available channels and then performs the stay-pattern to stay on a specific channel. Let the number of available channel to be $M$ and that is the same for all the SUs in the symmetric JS scheme. Then each SU finds the smallest prime number $P \geq M$ and selects random channels $i_0$ and $r_0$ (i.e., $1 \leq \{i_0, r_0\} \leq M$) for the starting channel and the step-length, respectively. Using this channel information, the user generates CH sequences for a round of $3P$ time slots that consists of two jump-patterns for $2P$ time slots and one stay-pattern for $P$ time slots. Then Lin Zhiyong et al. [25] have shown that if two users runs the JS CH sequences, rendezvous is guaranteed in $3P$ time slots.

In this section, we briefly introduce the properties of the JS algorithm. We borrow the terminologies as defined in [25].

**Lemma 1** Given a positive integer $P$, if $r \in [1, P)$ is relatively prime to $P$, (i.e., the only common factor between them is 1), then for any $x \in [0, P)$ the sequence $S =< x\%P + 1, (x + r)\%P + 1, \cdots, (x + (P - 1)r)\%P + 1 >$ is a permutation of $< 1, 2, \cdots, P >$.

**Lemma 2** Given a prime number $P$, if $r_1$ and $r_2$ are two different numbers in $(0, P)$, then for any $x_1 \in [0, P)$ and $x_2 \in [0, P)$, there must be an integer $k \in [0, P)$ such that $(x_1 + kr_1)\%P = (x_2 + kr_2)\%P$.

**Theorem 1** Under the symmetric model, any two users performing $JS\_2\_SM$ (where $JS\_2\_SM$ is as described above) achieve rendezvous in at most $3P$ time slots, where $P$ is the smallest prime number greater than $M$.

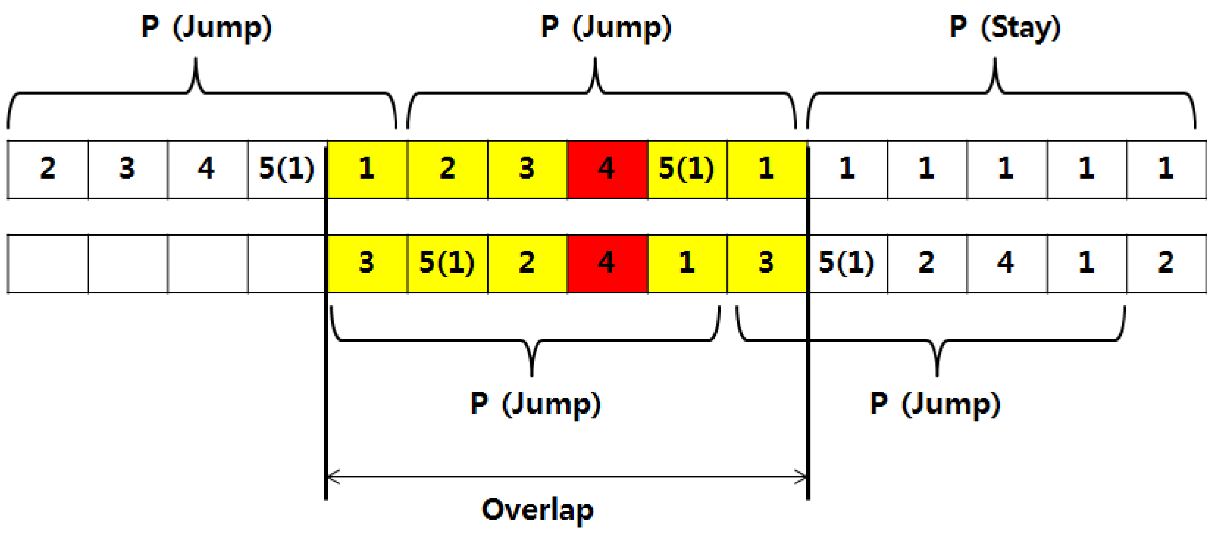


Figure 3.1: An example of the symmetric JS scheme

The two lemmas guarantee not only that the *JSHopping* algorithm visits all available the channels in any consecutive $P$ time slots of the jump-pattern but also two users will rendezvous if the overlap of two jump-patterns is not less than $P$ time slots with different step-lengths. Figure 3.1 illustrates a rendezvous example of the JS scheme where $M = 4$ and $P = 5$. Since there is no time synchronization, users can start at any time slots within the $3P$ time slots of another user but the overlap between two jump-patterns is not less than $P$ time slots. For example, one user starts with $i_1 = 2$ at $t = 1$ and jumps with the step-length $r_1 = 1$. The other user starts with $i_2 = 3$ at $t = 5$ and jumps with the step-length $r_2 = 2$. Then the two users rendezvous on channel $4$ at time slot $t = 8$ which then satisfies the Lemmas. In the JS scheme, the authors also proved that the maximum time to rendezvous (MTTR) is at most $3P$ time slots as described in the Theorem 1.

## 3.3 Proposed Schemes

We propose a novel CDJA in which the jammer can dramatically decrease the probability of rendezvous for the JS scheme [25] by utilizing the characteristics of the jump and stay-patterns in the JS scheme. In our CDJA, the jammer has capabilities close to the normal user such as listening on one or two channels at the same time but jamming one channel at a time. However, as we will show, the jammer can find the sender's CH sequences within $2P$ or $P$ time slots using one or two listening channel respectively. Then it can jam the sender's remaining CH sequences after an average $P$ or $\lfloor\frac{(P+1)}{2}\rfloor$ time slots for one or two listening channels. Therefore, the probability of rendezvous for the JS scheme will be dramatically decreased under the CDJAs.

We first present algorithms showing how the channel detecting jammer can find the sender's CH sequences in the JS scheme [25] within such a short period of time. Then we analyze its effectiveness on the JS scheme. We also revisit the Random rendezvous algorithm and the RCR scheme to address jamming attacks.

### 3.3.1 The Channel Detecting Jamming Attacks

In this subsection, we present a CDJA. Here we consider one sender and one receiver in the CRNs trying to rendezvous under the CDJA. In the JS scheme, each SU uses the jump-stay channel hopping algorithm to generate its CH sequences described in the preliminary section. This guarantees that the sender and receiver can rendezvous within the maximum $3P$ time slots when there are no jamming attacks. However, our channel detecting jammer takes the advantage of the jump-stay channel hopping properties to find the sender's CH sequences.

Now we describe how the channel detecting jammer finds the sender's CH sequences in the JS scheme. In CDJA, the fundamental goal for the jammer is to find the sender's CH sequences before the sender can rendezvous. The step-length is the key value to generating the JS CH sequences so that it is critical for the jammer to find the sender's step-length $r_0$ as fast as possible. We also need the initial channel selection $i_0$ but that is easy to determine once the step-length is found. Using the step-length, the jammer can exactly generate the entire CH sequences of the sender. Therefore, the jammer can jam the remaining channels in the CH sequences with a single jammer. We first give an example of the sender and receiver rendezvousing and how the jammer determines the sender's CH sequences by hearing the sender's signals with one or two listening channels. Then we analyze how the channel detecting jammer uses this information to defeat the rendezvous in the JS system.

We consider two different scenarios for the CDJA on the JS scheme depending on whether the jammer has knowledge of the sender's communication time or not. The first scenario is that the jammer can determine the beginning time of the sender's communication while the second scenario assumes the jammer cannot find this beginning time. Without loss of generality, we assume in the first scenario that the jammer can use spectrum sensing techniques to determine whether channels are used by other SUs or not. Since we consider one sender and one receiver in this scenario, the jammer can observe the time when one user occupies one of the many available channels at any given time. This is not a specific channel though. The jammer keeps spectrum sensing to check whether any new channels are occupied by other SUs or not. Initially we implement the first scenario since CDJA is clearer there and we solve the second scenario in Section 3.4.

For both scenarios, the jammer's capabilities are close to the legitimate SUs. We consider one or two listening channels. First, we describe the first scenario of the CDJA on the JS scheme when the jammer uses one listening channel. In this case, the jammer randomly selects one of the available channels and waits there until it receives signals from the sender. When the jammer receives signals, it then randomly selects another different channel and wait there until it receives the sender's signals on that channel. Since the jammer knows the beginning of the sender's communication, the jammer can receive two signals from different channels within the first or second jump-pattern of the JS scheme. We show the jammer can calculate the step-length of the sender's CH sequences from two channel hits and then it can exactly generate the sender's CH sequences. In the JS scheme, the channel $c$ in the jump-pattern is determined by

$$c_i = (i_0 + t_i \times r_0 - 1) \,\%\, P + 1$$

where $c_i$ is the channel at $i$th time slot and remapped $c_i = c_i \,\%\, M + 1$ for $c_i > M$, $i_0$ is the initial channel, $r_0$ is the step-length, and $P$ is the smallest prime number greater than the number of available channels. Suppose that the jammer receives the sender's signals on $c_1$ at time $t_1$ and on $c_2$ at time $t_2$. Assume we have chosen $c_1$ and $c_2$ as discussed in Section 3.4-B so that they are not mapped onto by the

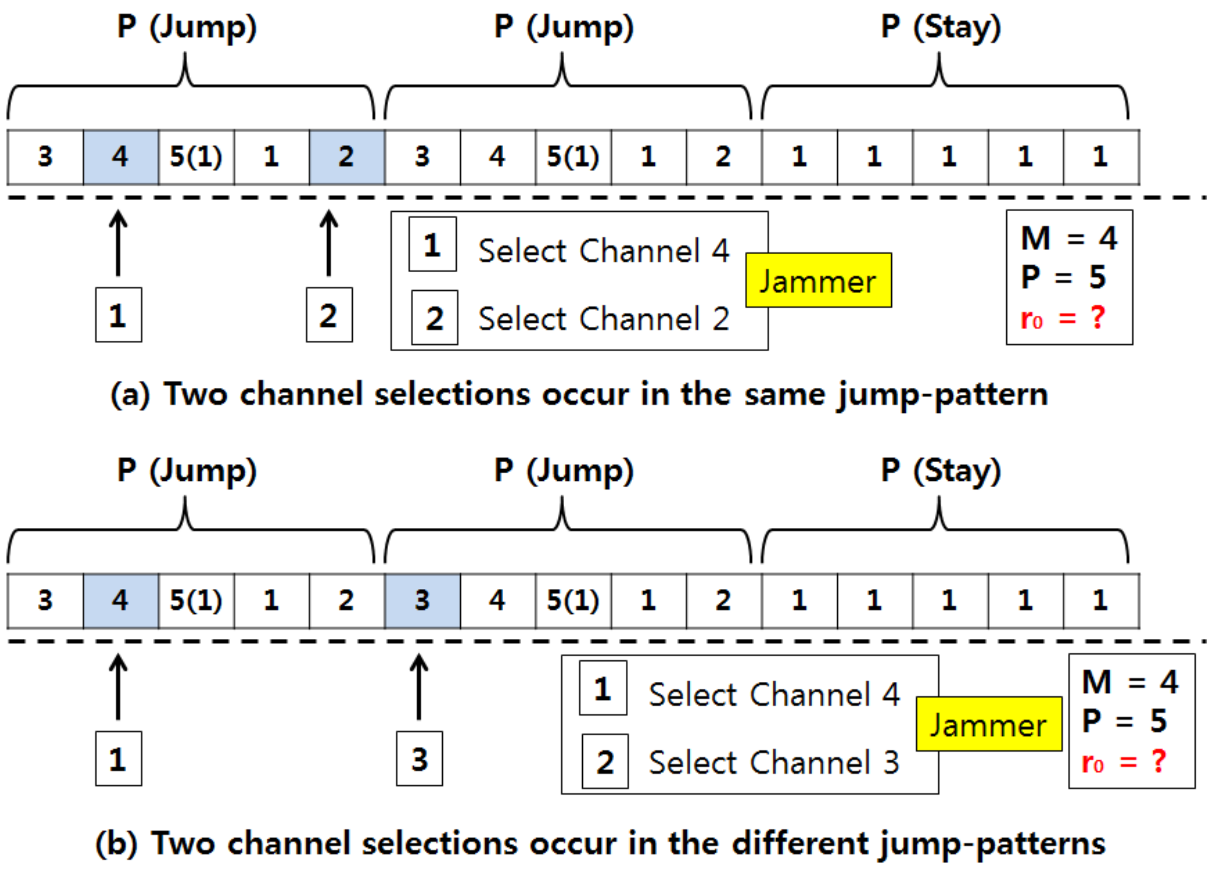


Figure 3.2: An example of finding the step-length with one listening channel

$M$ remapping. Then the jammer can calculate the sender's step-length $r_0$ using

$$c_1 = (i_0 + t_1 \times r_0 - 1)\%P + 1 \tag{3.1}$$

$$c_2 = (i_0 + t_2 \times r_0 - 1)\%P + 1 \tag{3.2}$$

and then equation (5.3) − (5.2) gives,

$$((t_2 - t_1) \times r_0)\%P = (c_2 - c_1).$$

Thus, the step-length $r_0$ is

$$r_0 = \lceil \frac{(P \times k + (c_2 - c_1))}{(t_2 - t_1)} \rceil \tag{3.3}$$

where the $k$ is the constant that satisfies the property of $1 \leq r_0 \leq M$ and matches the jump-pattern. Once the jammer can compute the step-length $r_0$, then the jammer computes the initial channel $i_0$ from equation (1) and exactly generates the sender's CH sequences.

Figure 3.2 describes how the channel detecting jammer finds the sender's step-length in the JS scheme using one listening channel. The channel detecting jammer selects the random listening channels to hear the sender's signals where $M = 4$, $r_0 = 1$, $i_0 = 3$, and $P = 5$. In this example, the complete channel hopping sequences of the sender is

$$3, 4, 5(1), 1, 2, 3, 4, 5(1), 1, 2, 1, 1, 1, 1, 1$$

where the total $3P$ time slots consists of the first $2P$ time slots for the jump-patterns and the last $P$ time slots for the stay-pattern. The stay-pattern is just the step size $r_0$. To detect two sender's signals, the jammer randomly selects channel 4 for the first listening channel and channel 2 for the second listening channel. Then the jammer receives the signals on the channel 4 ($c_1$) at time $t_1 = 2$ and channel 2 ($c_2$) at time $t_2 = 5$. From the listening channels and time slot information, the jammer can compute the $r_0$ from the equation (5.4),

$$\begin{aligned} r_0 &= \lceil \frac{(P \times k + (c_2 - c_1))}{(t_2 - t_1)} \rceil \\ &= \lceil \frac{(5 \times k + (2 - 4))}{(5 - 2)} \rceil = \lceil \frac{(5 \times k - 2)}{3} \rceil \\ & \quad 1 \leq \lceil \frac{(5 \times k - 2)}{3} \rceil \leq 4, \ \because 1 \leq r_0 \leq M \end{aligned}$$

then the constant $k$ is,

$$5 \leq \lceil 5k \rceil \leq 14, \ \ 1 \leq \lceil k \rceil \leq 2.8$$

where $c_1 = 4$ at $t_1 = 2$ and $c_2 = 2$ at $t_2 = 5$. The possible values of $k$ are 1 or 2 . Now the step-length $r_0$ is clearly constant so one value of $k$ can be chosen. The proof of Lemma 2 in [25] actually shows that the $k$ value is unique. We can easily see that $k = 1$ and the corresponding value of $r_0$ satisfies the

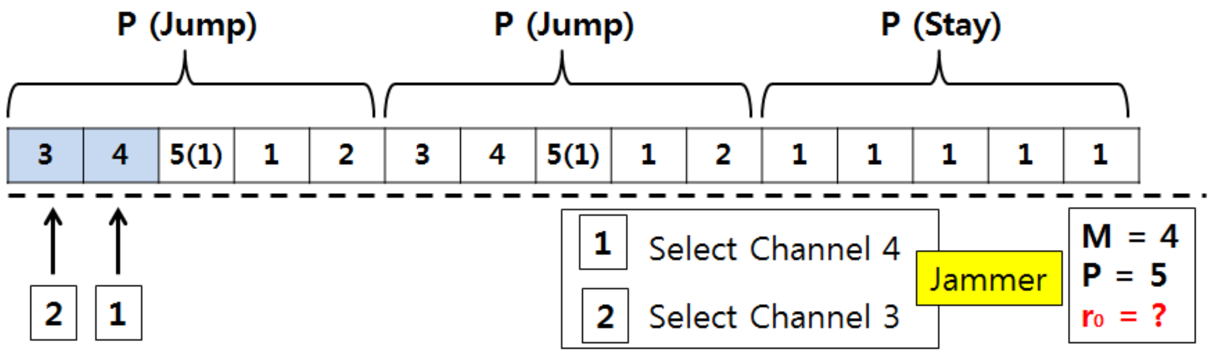


Figure 3.3: An example of finding the step-length with two listening channels

jump-pattern. Hence we choose $k = 1$ and reject $k = 2$. Thus, the step-length $r_0$ should be 1 for $k = 1$.

With one listening channel, the jammer might not receive the sender's signals within the first jump-pattern. Since the jammer uses one listening channel, the time to detect two sender's signals depends on the order of listening channels described in Figure 3.2(a) and (b). Thus, the channel detecting jammer can detect the sender's signals from two different channels within two jump patterns ($2P$) and an average of $P$ time slots. However, for two listening channels, the jammer can always detect the sender's two signals within the first jump-pattern ($P$ time slots). In Figure 3.2 (b), the jammer with one listening channel cannot detect channels 4 then 3 within the first jump-pattern (i.e., $P$ time slots). But Figure 3.3 shows that the jammer with two listening channels can detect both channel 3 and 4 within the first jump-pattern because there is no order of listening channels. Thus, the jammer can find the sender's CH sequences within a maximum $P$ time slots and an average $\lfloor \frac{(P+1)}{2} \rfloor$ time slots. Therefore, the jammer in the first scenario can jam the remaining channels for an average $2P$ time slots and an average $2P + \lfloor \frac{(P+1)}{2} \rfloor$ time slots for one and two listening channels respectively. The probability of rendezvous will be dramatically decreased under the CDJA.

### 3.3.2 Role-based Channel Rendezvous

The rendezvous probability of the JS scheme can be dramatically decreased under the CDJA. To mitigate this jamming attack, we consider Anderson and Weber's algorithm [16] called role-based rendezvous

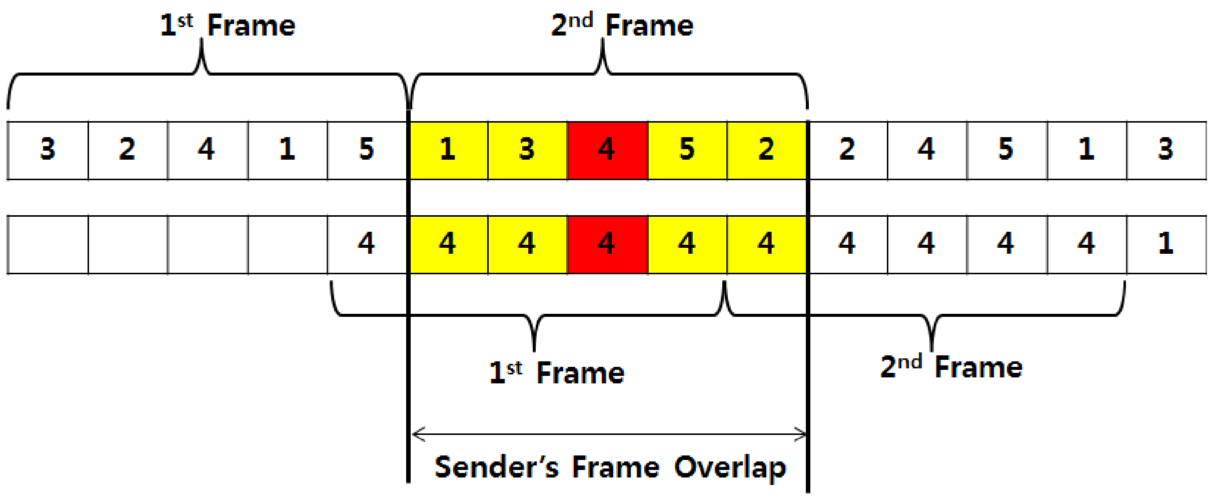


Figure 3.4: An example of rendezvous in the RCR Symmetric scheme

scheme [6] and extend it to the Role-based Channel Rendezvous (RCR) scheme as a countermeasure against the CDJAs. In the RCR system, a sender generates a randomized permutation of the $M$ available channels for every frame. The receiver randomly selects one channel out of $M$ channels stays there until the end of two frames (i.e., $2M$ time slots). Thus it is unfeasible for the channel detecting jammer to estimate when the sender and receiver might rendezvous. Figure 3.4 illustrates the example of RCR scheme where $M = 5$. One user generates the CH sequences by permuting $\{1, ..., M\}$ for every frame. The other user randomly selects channel $4$ and waits on the channel until rendezvous. The two users rendezvous on the channel $4$ at time $t = 8$. In the RCR scheme, if there is no time synchronization between users, then there must be an overlap between the two users of $2M$ time slots to rendezvous. Thus the maximum time to rendezvous (TTR) of the RCR scheme is $2M$ time slots but its expected TTR is between $\lfloor\frac{(M+1)}{2}\rfloor$ and $M$ time slots.

However, the RCR scheme has limitations such as the requirement of predefined roles for users and an impersonated attack to find the receiver's channel. When a user acts as a sender, it generates the permutated sequence of $M$ channels. When it plays a receiver's role, it randomly selects one random channel out of $M$ and stays that channel for $2M$ time slots. These roles are predefined before the system starts. Moreover, a jammer in the RCR system can impersonate the sender and send signals using a random permutation sequence. If the receiver receives the jammer's signals before the sender's, then the receiver will first response to the jammer to rendezvous. Therefore, the jammer can find the receiver's random channel and jam the channel in the remaining time slots to prevent the possible rendezvous between the sender and receiver. One possible solution to this attack has the receiver immediately choosing another random channel when it fails the authentication of the jammer or receives no response from it. For example, suppose that a receiver randomly selects a channel from $M$ available channel for every $2M$ time slots. The jammer sends a random $M$ permutation sequence to the receiver. When the receiver and the jammer rendezvous, the receiver will respond to the jammer to exchange information or data. When the jammer receives the response, the jammer knows the receiver's random channel so that it can jam the remaining channels of the sequence. However, it is difficult for the jammer to establish a connection to the receiver without passing authentication process. Thus, the receiver immediately choose a another random channel for $2M$ time slots when it knows there is a failure. The jammer never receives more channel information.

### 3.3.3 Random Channel Rendezvous

We also revisit the Random CH rendezvous algorithm [6] to mitigate the CDJAs. In fact, there are no guaranteed rendezvous algorithms under any jamming attacks so that the Random CH rendezvous algorithm could be the most robust CH algorithm against jamming attacks. In the Random system, a sender and receiver randomly select one channel out of the $M$ available channels for each time slot. The formula $\sum(1/M)(1-1/M)^{k-1} * k = M$ shows the expected TTR for the Random scheme is $M$ time slots when there are no jamming attacks. However, it is unfeasible for any jammer to estimate when the Random sender and receiver might rendezvous in the Random scheme Thus the expected TTR for the Random scheme would be almost the same as $M$ under the CDJAs. Hence, the Random scheme for CRNs can be an effective, efficient and secure rendezvous scheme against CDJAs.

## 3.4 Extensions of CDJA

In this section, we describe how to remove time synchronization restriction for the second scenario, the efficiencies in selecting listening channels, and the effectiveness of CDJAs on the asymmetric JS scheme.

### 3.4.1 No Time Synchronization

In the second CDJA scenario for the JS scheme, the channel detecting jammer cannot estimate the beginning of the sender's communication by using spectrum sensing techniques. In this scenario, we assume only that the jammer resides in the network before the communication starts. Moreover, we only consider the case of a jammer with two listening channels because it simplifies the presentation and is significantly more efficient. However, this can be done with one listening channel. The cost of a second listening channel is modest.

The jammer can always hear two sender's signals within the first pattern (i.e., $P$ time slots) of the JS scheme. When the jammer detects two sender's signals, the jammer can find the step-length $r_0$ from equation (3), generate the sender's CH sequences from the time of the second signal, and jam all remaining time slots using $c_{next} = (c_{last} + r_0)\%P + 1$.

However, the jammer cannot immediately compute the initial channel $i_0$ because it has no knowledge about the beginning of the sender's communication. The initial channel information is important for the jammer to determine the beginning of the stay-pattern and the next period so that the jammer can jam all the $r_0$ channels. For example, the first channel of the second period in the JS scheme is

$$c_i = (i_0 + t_i \times (r_0^{'}) - 1)\ \%\ P + 1$$

where the new step-length is $r_0^{'} = r_0 + 1$. Thus the jammer can generate the sender's sequences for the consecutive periods from known $r_0$ and $i_0$ information and jam all slots continuously with a single jammer.

To find the initial channel $i_0$, the jammer needs to add an additional step to find the beginning of the stay-pattern. Figure 3.5 describes how to find the step-length $r_0$ and the initial channel $i_0$ using

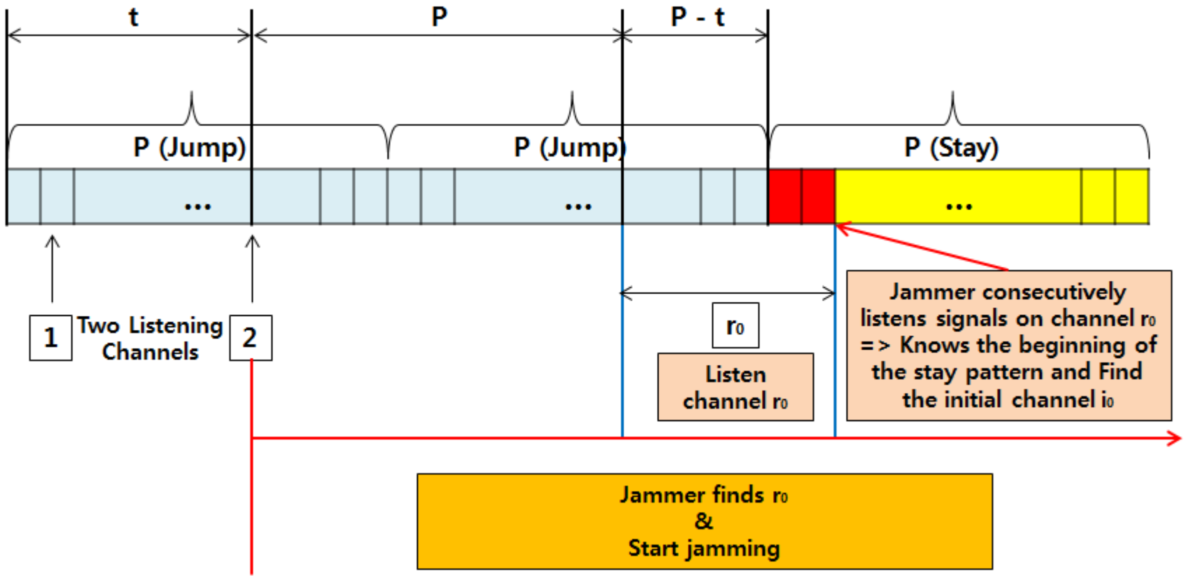

t
P
P - t
P (Jump)
P (Jump)
P (Stay)
...
1
Two Listening Channels
2
r0
Listen channel r0
Jammer consecutively listens signals on channel r0 => Knows the beginning of the stay pattern and Find the initial channel i0
Jammer finds r0 & Start jamming

Figure 3.5: Cases for detecting the sender's channels in the second scenario

two listening channels. When the jammer detects the sender's second signal on a channel at time $t$, the jammer can compute $r_0$ from equation (3) and jam the remaining time slots using $c_{next} = c_{last} + r_0$. Note that $t$ is in the first jump-pattern. After $P + t$ time slots, the jammer listens on the channel $r_0$ to find the beginning of the stay-pattern. That is, if the jammer hears two consecutively signals on the channel $r_0$ between $P + t$ and $2P + 2$ time slots, then the jammer knows that the first time slot is the beginning of the stay pattern. There is another possibility that the jammer detects a signal on $r_0$ between $P + t$ and $2P$ time slots because the channel $r_0$ is a channel in the sender's sequence of the second jump-pattern. In this case, the jammer waits to detecting another signal on the channel $r_0$. That time slot is the beginning of the stay pattern. In either case, the jammer does not jam $r_0$ for at most two time slots because the jammer cannot listen and jam a channel at the same time. When the jammer finds the beginning of the stay pattern, it can compute the initial channel $i_0$ from the channel information of the last time slot in the second jump-pattern using equation (1). Therefore, the jammer can still effectively jam the JS scheme when there is no time synchronization. Simulation results will confirm this.

### 3.4.2 Listening Channel Selection

Jammer selection of the listening channels in CDJA can improve performance. Most of time the jammer can randomly select listening channels but the jammer should avoid selecting channels from $(M, P]$ because the JS algorithm maps these channels back onto $[1, M]$ channels which are then duplicated. That is, if the jammer selects the channels that are mapped onto by $(M, P]$, then it is possible that the step-length can be ambiguous because the channels from $(M, P]$ are duplicated with other channels in the JS algorithm.

From the above example, suppose that the jammer selects 1 and 3 for listening channels and detects signals from the following CH sequence

$$3, 4, 5(1), 1, 2, 3, 4, 5(1), 1, 2, 1, 1, 1, 1, 1.$$

Then the jammer can detect the signals on the channel $c_1 = 1\ (5)$ at time $t_1 = 3$ and the channel $c_2 = 3$ at time $t_2 = 6$ for the first time and second time, respectively. The step-length $r_0$ from the equation (5.4) is

$$\begin{aligned} r_0 &= \lceil \frac{(P \times k + (c_2 - c_1))}{(t_2 - t_1)} \rceil \\ &= \lceil \frac{(5 \times k + (3 - 1))}{(6 - 3)} \rceil = \lceil \frac{(5 \times k + 2)}{3} \rceil \\ & \quad 1 \leq \lceil \frac{(5 \times k + 2)}{3} \rceil \leq 4, \ \because 1 \leq r_0 \leq M \end{aligned}$$

then the constant $k$ is,

$$1 \leq \lceil 5k \rceil \leq 10, \ \ 0.2 \leq \lceil k \rceil \leq 2$$

Therefore, the step-length $r_0 = 2, 4$ for the constant $k = 1, 2$ respectively. Neither of which satisfies the jump-pattern.

On the other hand, if the jammer computes the step-length $r_0$ using $c_1 = 5$ instead of $c_1 = 1$, then it can correctly compute the step-length $r_0 = 1$. However, the jammer cannot immediately determine whether the $c_1$ is the channel $1$ or the channel $5$. Thus, in this case, the jammer needs to compute the step-lengths for both $c_1 = 1$ and $c_1 = 5$ cases and generates all possible the sender's CH sequences for all the step-lengths. Then the jammer listens on the next channel for one of the possible CH sequences to determine whether the CH sequence is correct or not. In this chapter, we assume that the jammer avoid selecting the duplicated channels because this removes the ambiguity of the step-lengths. The number of duplicated channels is relatively small compared to the number of available channels (e.g., the maximum number of duplicated channels for $4 \leq M \leq 100$ is 7 for $M = 89$). Hence it is expedient to avoid, for the channel selection, those channels that are mapped onto by $(M, P]$ channel selection but it is not required.

### 3.4.3 CDJA for the Asymmetric JS Scheme

The asymmetric model for CH rendezvous assumes two users have different available channels sets and they are not known to each other. The basic idea of JS for the asymmetric model is to expand the set of possible channels to the union of the available channels and let $G$ denote the intersection of the available channels. As in [25], we let $M$ denote the set of all channels under possible consideration and we make exactly the same assumptions here as in [25]. Their basic algorithm for asymmetric CH, which we denote AJS, does not guarantee rendezvous in $3P$ time slots but does so in $6MP(P - G)$ time slots. AJS is the same as JS except it increments the step-length every $3P$ times slots and the starting-index every $6MP$ time slots.

Here we assume CDJA does not have time synchronization with the sender but uses two listening channels. Our CDJA then applies as above with the modification that we need to increment the $i_0$ as needed. Hence CDJA is effective against the AJS. In [25], the authors subsequently integrate a random-replace operation into AJS that replaces unavailable channels in their CH sequence with random available channels. The guaranteed rendezvous was not proven for this case. However we can modify our CDJA so that it applies effectively here as well.

Suppose we hear $c_1$ at $t_1$ and $c_2$ at $t_2$ as before. We compute $r_0$ as before but $c_1$ or $c_2$ may be a random channels and hence $r_0$ is not certain. We jam on $c_{next} = c_{last} + r_0$ as before but we continue to listen on two other channels for up to $P$ slots beyond $max\{t_1, t_2\}$. When we hear $c_3$ at $t_3$ (guaranteed to happen) we compute two new $r_0$s based on each pair of the three data points. If two of the $r_0$s match we use it; otherwise we use the $r_0$ based on the most recent channels heard. We continue to listen and compute all possible $r_0$s and use the majority $r_0$ to compute the CH sequence on which to jam. It has the highest probability of being correct. Once we get beyond $max\{t_1, t_2\} + P$, then we listen on the two most probable $r_0$s for the stay-pattern as was done above since time is not synchronized here either. Then we know with high probability both $r_0$ and $i_0$ and can jam every subsequent channel (except of course the random-replace channels) until we complete $6MP$ time slots. Since $r_0$ and $i_0$ are just incremented, we can trivially compute all subsequent CH sequences and jam them as well.

Both Random and RCR extend easily to the asymmetric case by just choosing randomly or random permutations from their own set of available channels. Future work will examine their performance for various percentages of common channels between users and also under various jamming scenarios.

## 3.5 Evaluation

We implemented the CDJA on Matlab 2010b to evaluate its effectiveness against the JS scheme. We first implemented the jamming attacks for the first scenario in which the jammer knows the beginning of the sender's communication time. This implementation includes the cases that the jammer uses both one and two listening channels. We also implemented the CDJA for the second scenario where the jammer has no knowledge of the beginning of the sender's communication time. In this scenario, the jammer uses two listening channels because it simplifies the presentation and is significantly more efficient without increasing cost much. We then implemented both the Random and the RCR schemes to demonstrate that they are more robust against the CDJA compared to the JS scheme. The jamming attack for both the Random and RCR schemes are random channel jamming that continues for the entire rendezvous time. Because of the permutation in the RCR scheme, a very modest increase in jamming effectiveness can be gained by looking at the end of each period which can limit the possible channels to jam.

First we compare the expected TTR for the JS, Random, and RCR schemes when there are no jamming attacks. Figure 3.6 gives the average TTRs for the JS, Random, and RCR schemes where

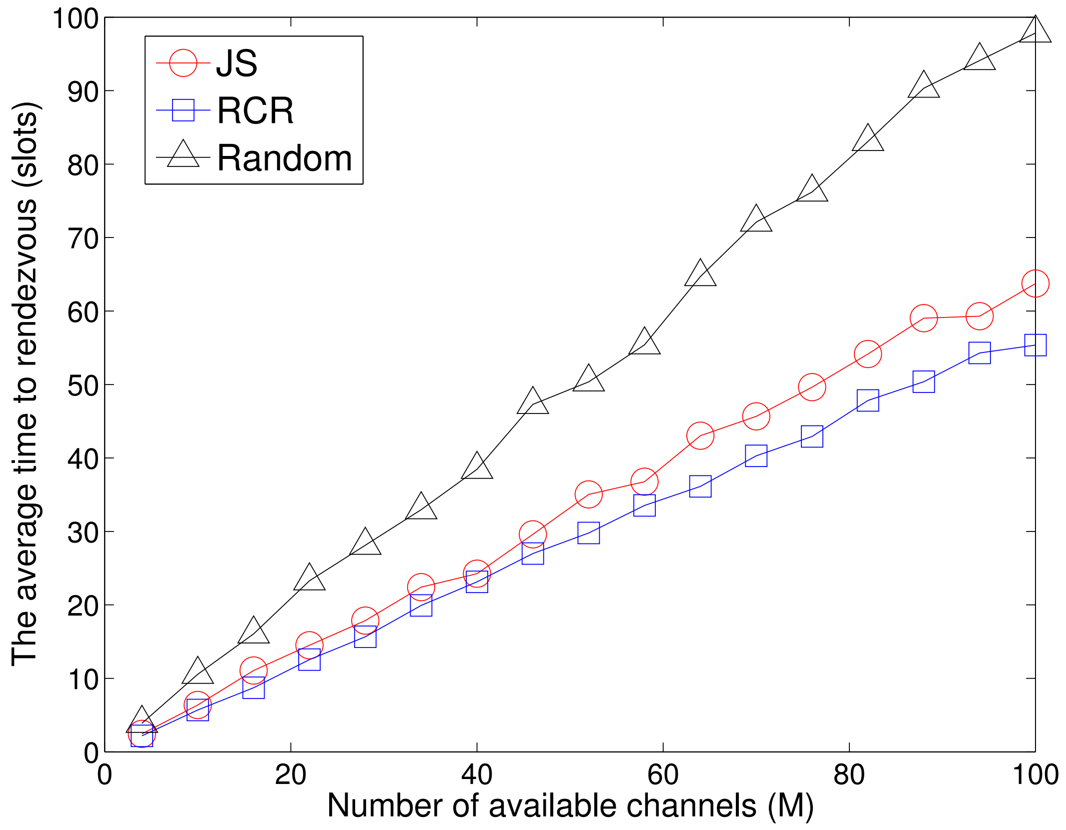


Figure 3.6: The average time to rendezvous (TTRs) for the JS, Random, and RCR schemes

the number of available channels $M$ varies from $4$ to $100$. We ran $1000$ simulations for each available channel and calculated the average TTR for them. This figure shows that the average TTRs for all schemes increase steadily as the number of available channel increases. For the Random scheme, the expected TTR is $M$. However, the expected TTR for the RCR scheme is between $\lfloor\frac{(M+1)}{2}\rfloor$ and $M$ and the average TTR from the implementation is slightly over half the available channels $M$. The main reason for better performance is that the sender and receiver usually rendezvous during the first overlap of $M$ time slots. Overall, the average TTR for the Random scheme is almost twice that of the other schemes. The average TTR for the RCR scheme is slightly better than the JS scheme.

Next we compare the expected time for finding the step-length in the JS scheme with the expected

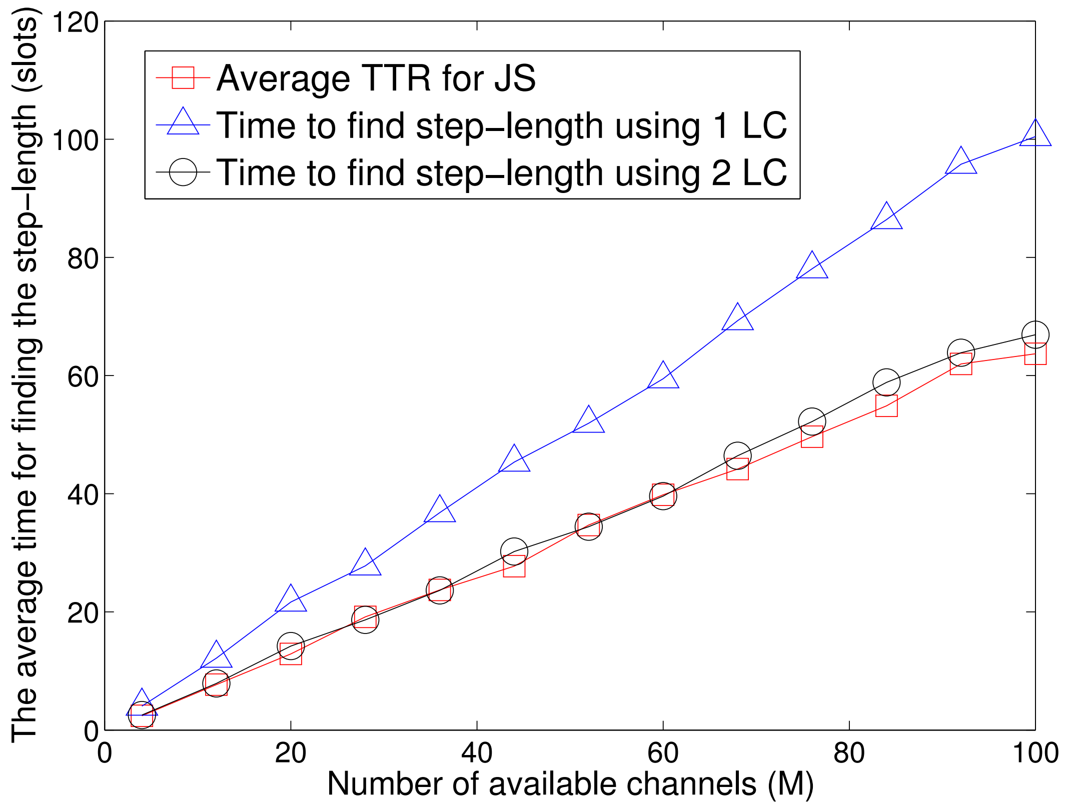


Figure 3.7: The expected time for finding the step-length in the JS scheme

TTR of the JS scheme. We implemented finding the sender's step-length algorithms using both one and two listening channels (LCs). If the jammer uses one listening channel, the upper bound of finding the sender's step-length is $2P$ time slots and the average time is approximately $P$ time slots. However, using two listening channels, the jammer can find the sender's step-length within the first $P$ time slots and it can find the step-length in an average of $\lfloor\frac{(P+1)}{2}\rfloor$ time slots. Figure 3.7 shows the expected time for finding the sender's step-length using both one and two listening channels. The expected time for two listening channels is slightly higher than $\lfloor\frac{(P+1)}{2}\rfloor$ time slots because of the duplicated channels from $(M, P]$ channels. The average time for finding the step-length with two listening channels is close to the expected TTR for the JS scheme. In the CDJA, if the jammer finds the sender's step-length, then the jammer can exactly generate the sender's CH sequences to jam. This means that the sender and receiver must rendezvous in an average $\lfloor\frac{(P+1)}{2}\rfloor$ time slots under the CDJA or they will fail to rendezvous. But [25] shows the expected TTR for the JS is $\frac{5P}{3} + \frac{11}{3} + \frac{1}{M-1}$. Hence theory says CDJA will severely limit the JS chances to rendezvous.

Now we compare the rendezvous probability for the JS, Random, and RCR schemes under the CDJA. In this experiment, the channel detecting jammer uses one or two listening channels but it uses a single jammer. Since the jammer can determine the sender's CH sequences from our finding step-length algorithm, a single jammer jams the remaining channels. Figure 3.8 shows the probability of rendezvous for all systems until the end of $3P$ time slots. The probability of rendezvous for the JS scheme is dramatically decreased under the CDJA. The rendezvous probability is less than $20\%$ for the almost all available channels numbers when the jammer uses one listening channel. Moreover, the probability of rendezvous decreases to generally under $10\%$ when the jammer uses two listening channels. However, the rendezvous probability for both the Random and the RCR schemes are almost steady and more than $90\%$ and close to $100\%$ for most numbers of available channels $M$, respectively.

Next we implement the cumulative rendezvous probability for all schemes where the channel detecting jammer uses a single jammer. Figure 3.9 displays the cumulative probability of rendezvous for the three systems until the end of $3P$ time slots. We selected two $M$ as the number of available channels ($M = \{25, 75\}$). For the JS scheme and CDJA with one listening channel, the cumulative rendezvous

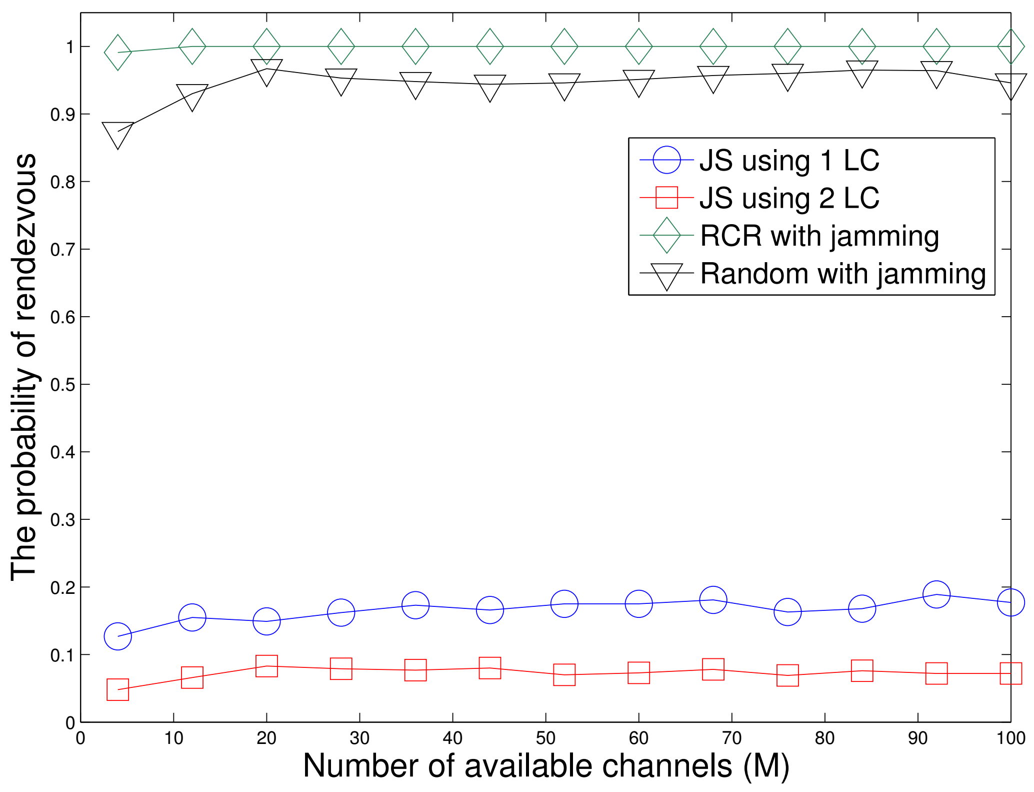


Figure 3.8: The probability of rendezvous for the JS, Random, and RCR schemes under the CDJAs with one or two listening channels (LCs)

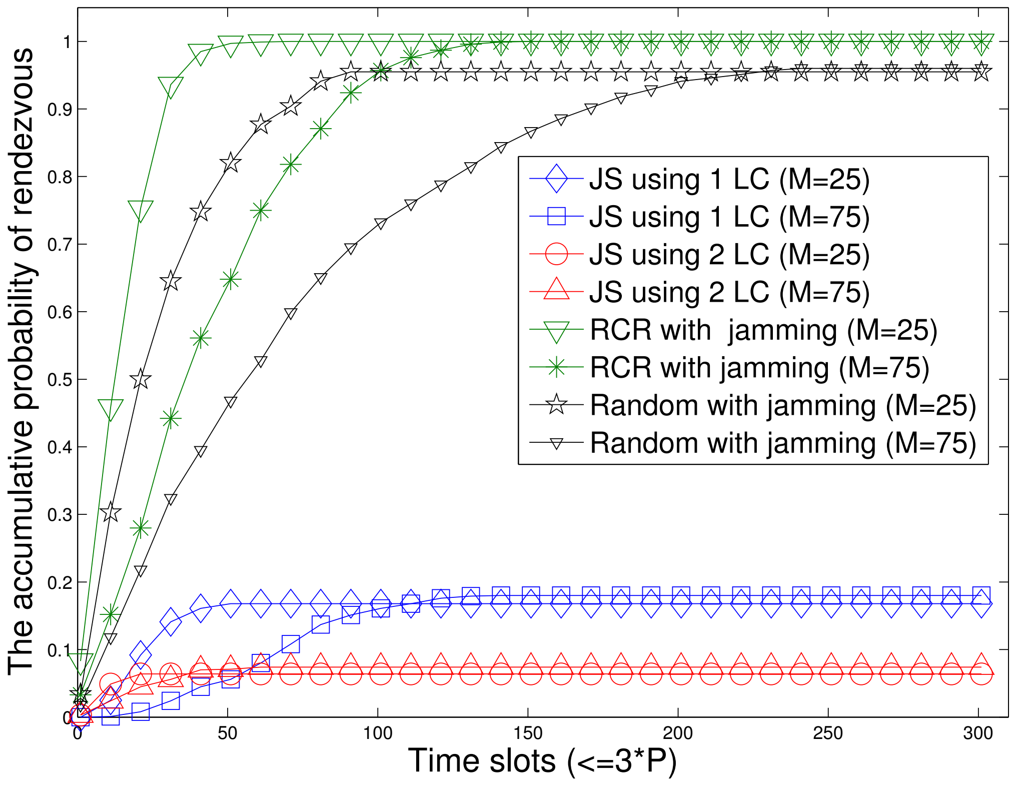


Figure 3.9: The cumulative probability of Rendezvous for the JS, Random, and RCR schemes under the CDJAs (M={25,75})

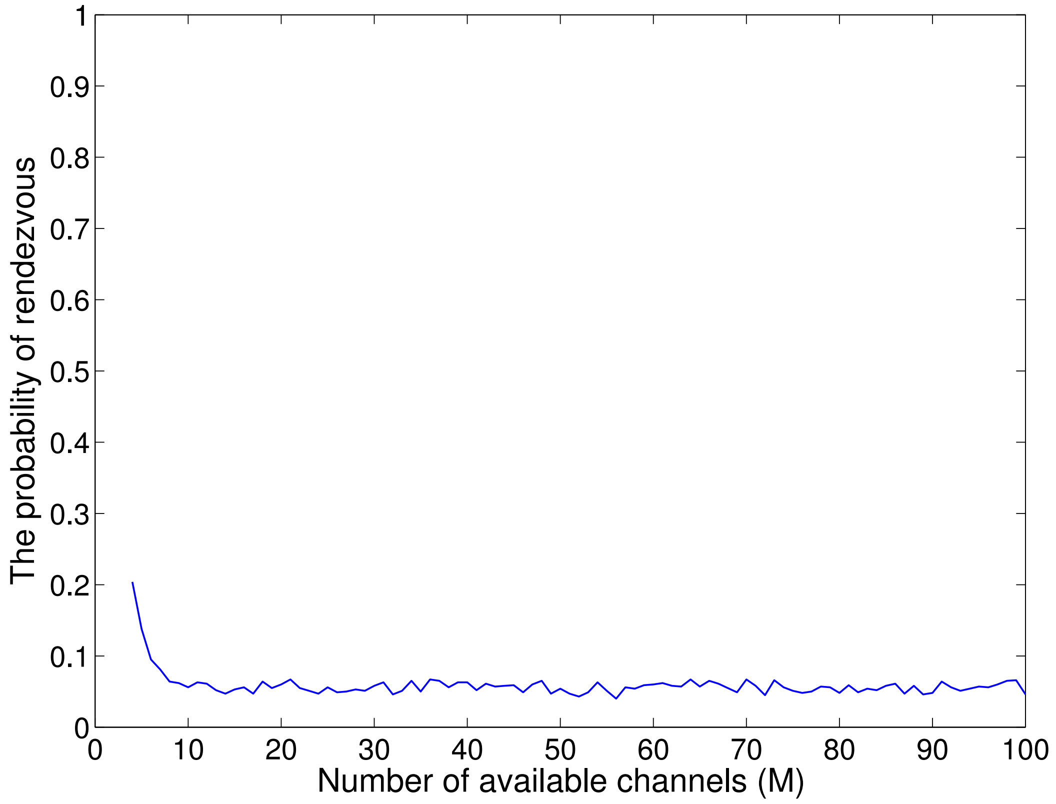


Figure 3.10: The probability of JS rendezvous for the second scenario under the CDJA (the jammer with two listening channels and no time synchronization)

probabilities increased to approximately $20\%$ until $2P$ time slots and holds steady for the remaining time slots. For the JS scheme with two listening channels, the cumulative rendezvous probabilities increased to approximately $10\%$ until $2P$ time slots and holds steady for the remaining time slots. On the other hand, the cumulative rendezvous probabilities for both the Random and the RCR scheme increases linearly up to the its original period ($M$ time slots) and slowly increased from $M$ to $3P$ time slots. In both the Random and the RCR system, we extend the period from $2M$ to $3P$ time slots to more accurately compare with the JS scheme. The sender in the RCR system permutates the CH sequences again at $2M + 1$ time slots. This increases the probability of rendezvous to nearly $100\%$ for most available channels $M$. Therefore, the effectiveness of CDJA is minimal for both the Random and RCR schemes but it is extremely strong for the JS scheme.

Finally, we implemented the second scenario model under the CDJA. In this experiment, the channel detecting jammer uses two listening channels and does not have the knowledge of the beginning of the sender's communication. Since the jammer resides in the network before the sender's communication, the jammer can hear two sender's signals within the first-pattern. Then the jammer computes the step-length $r_0$ and jam all remaining time slots. As we discussed in 3.4.1, for the second scenario, there are at most two possible slots after $r_0$ is determined where CDJA does not jam the JS CH sequence. Hence the JS rendezvous probability could be slightly higher than the probability of the first scenario particularly for small $M$. Figure 3.10 shows the probability rendezvous for the second scenario under CDJA. The JS rendezvous probability is slightly higher than the first scenario's one when the number of channels $M$ is small. However, the rendezvous probability is less than $10\%$ for $M \geq 6$ and that is almost the same as the first scenario's. Therefore, the CDJA is still extremely strong for the JS scheme even if the jammer does not have any information about the sender's communication.

## 3.6 Conclusion

In this chapter, we presented a novel jamming attack called *CDJA* and demonstrated its effectiveness against the JS scheme [25]. The channel detecting jammer is able to take advantage of JS algorithm (its method of generating its CH sequence) to find the sender's CH sequence within the first $2P$ or $P$ time slots for using one or two listening channels respectively. Then the jammer can completely jam the sender after an average $P$ time slots for one listening channel and an average $\lfloor \frac{(P+1)}{2} \rfloor$ time slots for two listening channels using just a single channel jammer. To remedy this jamming problem, we revisited both the Random scheme and the RCR scheme extended from role-based rendezvous algorithm. Our simulation results demonstrate that the rendezvous probability of the JS system under the CDJA is dramatically decreased for all available channels $M$ (e.g., around $20\%$ for one listening channel and $10\%$ for two listening channels for all $M$). On the other hand, the rendezvous probability of both Random and the RCR scheme is almost steady for all available channels $M$ (e.g., more than $90\%$ for the Random and nearly $100\%$ for the RCR). Therefore, the Random and RCR schemes can be an effective, efficient and robust rendezvous scheme against CDJAs.

Chapter 4

# Channel Detecting Jamming Attacks on Symmetric Blind Rendezvous Algorithms for Cognitive Radio Networks

We have demonstrated in Chapter 2 and 3 that CDJAs, with capabilities similar to normal users, can significantly reduce rendezvous success rates for the Modular Clock and Jump Stay symmetric blind rendezvous algorithms. In this chapter, we extend our CDJAs to the Generated Orthogonal Sequence (GOS) [6] algorithms. Our CDJAs, with one/two listening channels, quickly determine the channel hopping sequence for the GOS algorithm. Corresponding simulation results show the rendezvous success rates of GOS and two other efficient blind rendezvous algorithms, DRSEQ [12] and CRSEQ [43], are dramatically decreased under CDJAs. We compare these results to the Random rendezvous algorithm and show Random vastly outperforms five efficient blind algorithms under CDJAs and in other measures as well. Our CDJA is a major security concern for the state-of-the-art symmetric blind rendezvous algorithms for cognitive radio networks.

## 4.1 Introduction

We have previously investigated state-of-the-art symmetric blind rendezvous algorithms for CRNs and found them susceptible to new types of jamming attacks. In Chapter 2 and 3, we presented new jamming attacks named *channel detecting jamming attacks (CDJAs)* for symmetric blind rendezvous algorithms for CRNs that are derived from Modular-based channel hopping algorithms [6, 25]. The channel detecting jammer, with capabilities similar to normal users, takes advantage of rendezvous algorithm properties to compute their channel hopping steps. Using these steps, the jammer computes and jams the remaining channel hopping (CH) sequences. Our previous work demonstrated that CDJA dramatically decreases the rendezvous success.

In this chapter, we substantially modify our CDJA for the remaining representative CRN blind rendezvous algorithms and demonstrate its effectiveness against them. The remaining state-of-the-art symmetric blind rendezvous algorithms not previously considered are the Generated Orthogonal Sequence (GOS) [6] and the deterministic rendezvous algorithms (DRSEQ) [12] and (CRSEQ) [43]. Since both DRSEQ and CRSEQ are deterministic, they are trivial to jam using one listening and one jamming channels and are included here only for completeness.

The primary work here shows how to execute a debilitating jamming attack against the GOS rendezvous scheme. Using our previous results (see Chapter 2.3 and Chapter 3.4) on CDJAs on Modular Clock (MC) and Jump Stay (JS), Figure 4.1 shows the probability of rendezvous for the MC and JS schemes under our CDJAs within $3P$ time slots. $P$ is the smallest prime number greater than or equal to the number of available channels $M$. $3P$ is the maximum time to rendezvous (MTTR) for JS. MC does not guarantee a rendezvous but it does have a TTR of $2P$ if it does indeed rendezvous. The MTTR for GOS is $M(1 + M)$ so we see MC and JS are generally more efficient than GOS. However, from Figure 4.1, we see GOS has a much better probability of rendezvousing within 3P slots than MC or JS under CDJAs. If we go beyond $3P$ slots, GOS totally dominates both MC and JS under CDJA attacks. If we cannot jam the random permutation algorithm of GOS, then it would appear to be the best blind rendezvous algorithm against CDJAs. Hence we need to determine if our modified CDJA does or does not work against the GOS scheme. If not, GOS would be a recommended rendezvous algorithm when

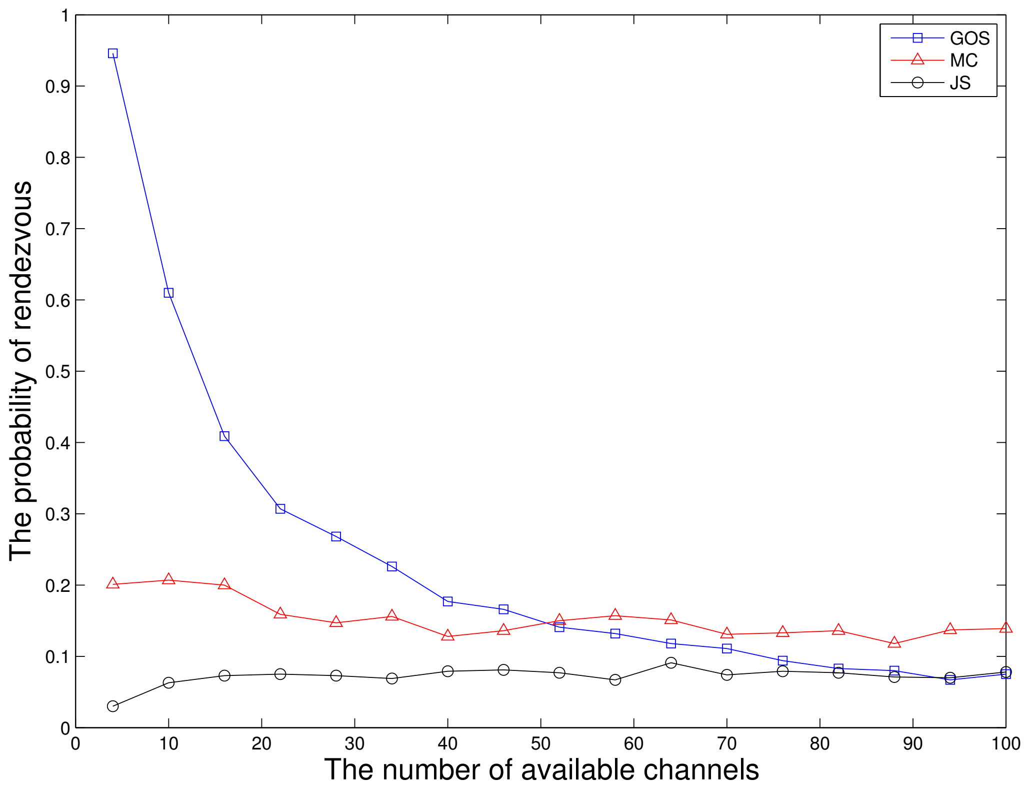


Figure 4.1: The probability rendezvous for the GOS scheme without jamming attacks and for the MC and JS schemes under CDJAs ($\leq 3P$ time slots)

our CDJAs cripple the MC and JS schemes.

We will show our modified CDJA, using two listening channels, can detect the GOS sender's channel hopping sequence within $(\frac{M}{2}+1)\times(1+M)$ time slots out of the total $M\times(1+M)$ time slots. Critically, we also jam channels as soon as they are determined and we jam only known sender channels. This maximizes the jamming effectiveness and minimizes the jamming footprint. As an alternative to the GOS scheme, we revisit the Random scheme where the sender and receiver generate their CH sequences by randomly selecting from the $M$ available channels for each time slot. Obviously, the Random scheme keeps the jammer from finding the sender's channel hopping sequence. Our simulation results demonstrate that the GOS scheme is extremely vulnerable to the modified CDJA but the Random scheme is an effective, efficient and robust rendezvous scheme against CDJAs. The primary contribution of this chapter is the CDJA on GOS and hence our CDJA poses a major security problem for the state-of-the-art symmetric blind rendezvous algorithms in CRNs.

The rest of the chapter is organized as follows. Section 4.2 presents the GOS scheme and its CDJA. Section 4.3 provides the simulations of our jamming attacks for both GOS and Random schemes, compares the results with our previous work on jamming MC and JS schemes, and provides analysis. Finally, Section 4.4 concludes the chapter.

## 4.2 Proposed Schemes

In this section, we present the Generated Orthogonal Sequence (GOS) scheme [6] and our CDJA on it which uses the characteristics of the GOS channel hopping algorithm to dramatically decreases the probability of rendezvous.We also present two other state-of-the-art blind CRN rendezvous algorithms: Deterministic Rendezvous Sequence (DRSEQ), Channel Rendezvous Sequence(CRSEQ) and their jamming attacks. We will show the jammer can find the GOS sender's CH sequences within $(\frac{M}{2}+1)\times(1+M)$ time slots using two listening channels. Simultaneously, the jammer jams the detected channels at most $(M+1)$ time slots after first detected. We also revisit the Random CH rendezvous scheme to address jamming attacks. Finally, we compare the effectiveness of the CDJA on GOS and Random schemes.

### 4.2.1 Generated Orthogonal Sequence (GOS)

L. DaSilva et. al. proposed the GOS technique for blind rendezvous in which all radios use the same random pre-defined sequences [6]. The radios use pre-defined sequence generators to create the CH sequences. Any two radios generate the same sequences and follow the same order to rendezvous. The radios can have different start times but they will eventually occupy the same channel and rendezvous. However, most sequences will not work for the GOS scheme. For example, consider two radios that follow the ascending order of channels from 0 to $M-1$, where $M$ is the number of available channels.

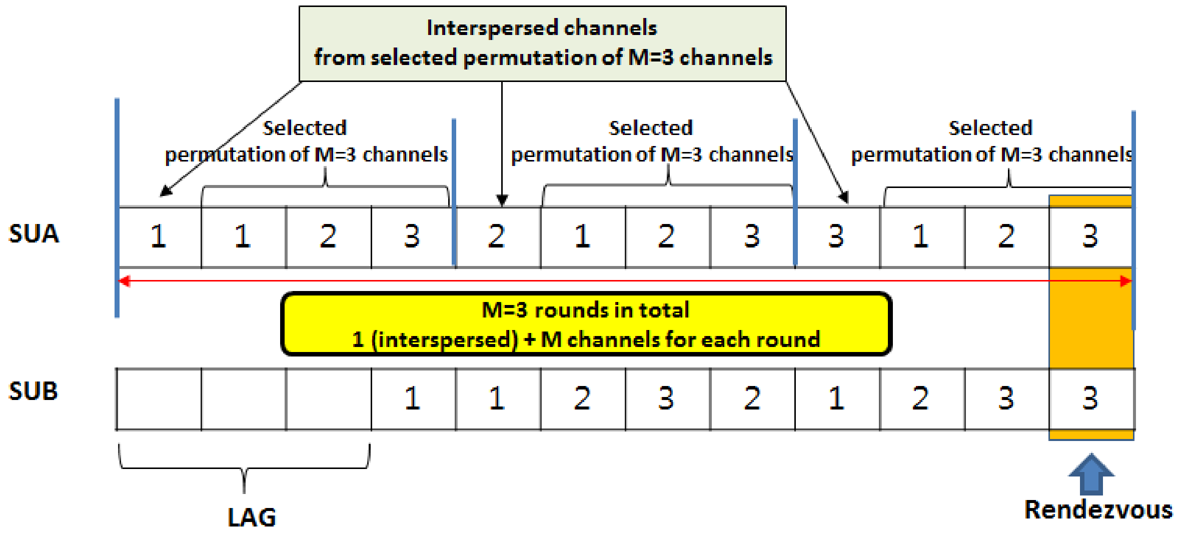

Interspersed channels
from selected permutation of M=3 channels
Selected permutation of M=3 channels
Selected permutation of M=3 channels
Selected permutation of M=3 channels
SUA
1 1 2 3 2 1 2 3 3 1 2 3
M=3 rounds in total
1 (interspersed) + M channels for each round
SUB
1 1 2 3 2 1 2 3 3
LAG
Rendezvous

Figure 4.2: An example of the GOS Rendezvous [13]

If two radios are not synchronized and start at different times, then the two radios would never meet. Thus, selecting an appropriate sequence is critical for the GOS scheme. Moreover, the sequence cannot be deterministic or it is easy to jam (see section 4.2.4).

Figure 4.2 illustrates how the GOS scheme generates a sequence with $M = 3$ available channels. The radios first select a permutation of the $M$ channels from $M!$ permutations. Then it generates a sequence in which the selected permutation appears contiguously $M$ times and once interspersed with the other $M$ permutations. That is, the GOS sequence consists of $M$ rounds and each round is composed of one interspersed slot followed by the $M$ permuted slots for $1 + M$ channels for each round. Thus, the total number slots in the GOS sequence is $M \times (1 + M)$. In Figure 4.2, the secondary users (SUs) $A$ and $B$ use the same GOS sequence,

$$\underline{1}, 1, 2, 3, \underline{2}, 1, 2, 3, \underline{3}, 1, 2, 3.$$

The underlined channels are the channels for interspersed time slots, and $B$ starts 3 time slots after $A$ starts. Then $A$ and $B$ rendezvous at channel 3.

### 4.2.2 CDJA on GOS scheme

We now demonstrate that GOS is vulnerable to CDJAs where the jammer uses two listening channels and jams one channel at a time. These CDJAs can still be effective with one listening channel but we do not have the space to pursue it here. We assume the normal situation where the jammer resides in the network before the communication starts. Here we consider one sender and one receiver in the CRNs trying to rendezvous under the CDJA. With GOS scheme, each secondary user uses the same pre-defined sequence generator to build its channel hopping sequences. This guarantees that the sender and receiver can rendezvous in at most $M \times (1 + M)$ time slots [6] when there are no jamming attacks. However, our channel detecting jammer takes the advantage of the GOS properties to find the sender's sequences.

Now we describe the GOS jamming attack. We use an example, see Figure 4.3, of the sender and receiver rendezvousing to show how the jammer determines the sender's CH sequences but provide the theoretical basis as well. We consider the secondary user $A$ with $M = 5$ available channels and initial permutation $\{3, 2, 5, 1, 4\}$. The GOS sequence is

$$\underline{3}, 3, 2, 5, 1, 4, \underline{2}, 3, 2, 5, 1, 4, ..., \underline{4}, 3, 2, 5, 1, 4.$$

The jammer randomly selects two distinct listening channels and detects at least two distinct signals in the first round, $1 + M$ time slots. As soon as the jammer hears a signal on one listening channel, it chooses another channel (not previously tried) for that listening channel. Thus, the jammer hears at least two distinct channels and, on average, expects to hear four of the random channels in each round. Hence, the jammer expects to hear, on average, all $M$ channels in $\lceil \frac{M}{4} \rceil$ rounds.

We may need additional time to determine the location of the interspersed time slot before finding the entire GOS channel hopping sequence because the channel for the interspersed time slot is changed for each round in the GOS scheme. There are two cases for detected channel information: (1) the detected channel is on a non-interspersed time slot or (2) it is on an interspersed time slot. For both cases, the jammer needs to listen for the detected channel after $(1 + M)$ time slots to verify whether the detected time slot is an interspersed time slot or not. If the jammer hears the detected channel again after $(1 + M)$ time slots, then the jammer knows the detected time slot is a non-interspersed time slot. Then

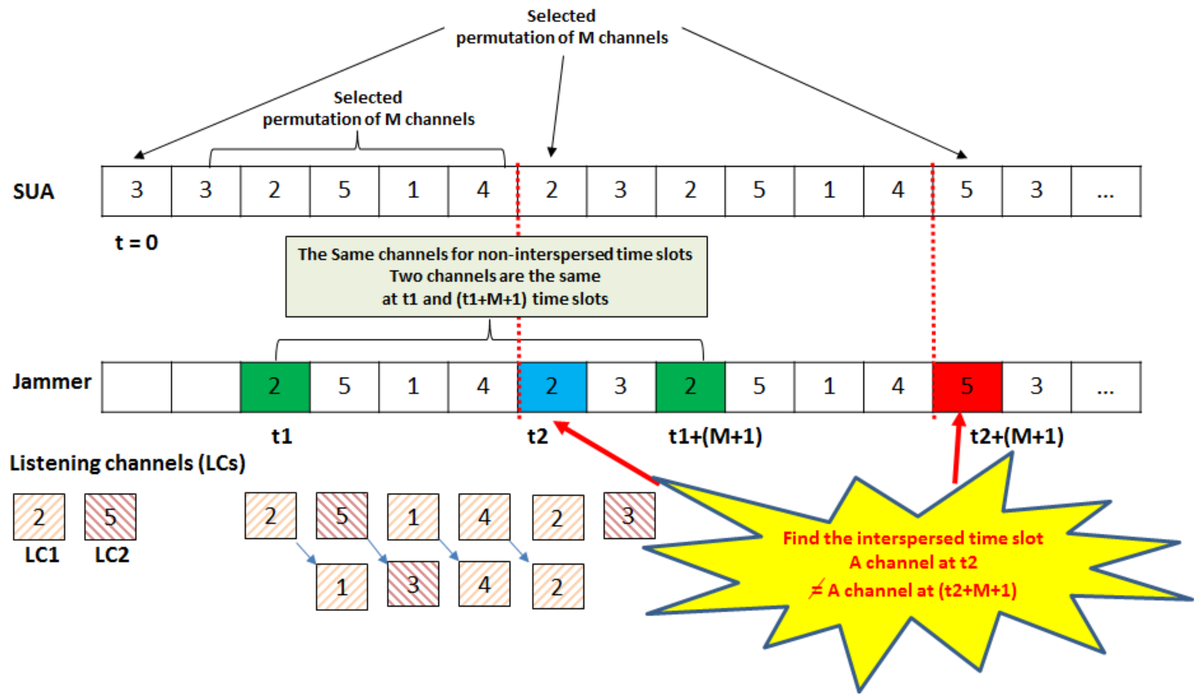

Selected permutation of M channels
Selected permutation of M channels
SUA
3 3 2 5 1 4 2 3 2 5 1 4 5 3 ...
t = 0
The Same channels for non-interspersed time slots
Two channels are the same
at t1 and (t1+M+1) time slots
Jammer
2 5 1 4 2 3 2 5 1 4 5 3 ...
t1
t2
t1+(M+1)
t2+(M+1)
Listening channels (LCs)
LC1
LC2
Find the interspersed time slot
A channel at t2
≠ A channel at (t2+M+1)

Figure 4.3: An example of CDJA for the GOS scheme

the detected channel on the non-interspersed time slot will appear in the same time slot or position for the remaining rounds in the GOS channel hopping sequence. Thus, every $(1+M)$ time slot the jammer jams the detected channel after determining it is a non-interspersed time slot. On the other hand, if the jammer does not hear the detected channel after $(1+M)$ time slots, the jammer knows the time slot is an interspersed time slot. Once the jammer determines one interspersed slot, then it knows all interspersed slots and it can jam every channel immediately and forever after it has been detected. The jammer can expect to determine the interspersed channel position before the $\lceil\frac{M}{4}\rceil$ round with probability $\frac{(M-4)}{M}$. However, there is one exception that the jammer might not determine the interspersed slot until the $\lceil\frac{M}{4}\rceil$ round. That is, it hears all channels in $\lceil\frac{M}{4}\rceil$ rounds but needs to listen one more round to determine the interspersed slot. Therefore, the expected number of time slots to find the entire GOS sequence is $\lceil(\frac{M}{4}+1)\rceil\times(1+M)$ time slots. As discussed in the next paragraph, we actually implemented a slightly more efficient algorithm but it does not have a computationally trackable performance analysis.

We can find the interspersed slot more quickly as follows. Suppose $c_1$ is the first channel detected in round 1. Then we listens again on $c_1$ for the next time slot. If the $c_1$ is heard again, then the first $c_1$ was the first interspersed slot and the jammer knows the position of every interspersed slot and every $c_1$. Then the jammer knows all other detected channels are non-interspersed time slots so that it can immediately jam them for the remaining rounds in the GOS sequence. If the jammer does not hear $c_1$ again then it resumes the listening pattern described below.

In Figure 4.3, the jammer resides in the network before the communication starts. It selects two random channels, 2 and 5, for its listening channels and listens. The jammer changes its listening channels as soon as it detects channel 2 and channel 5 as illustrated in Figure 4.3. When the jammer hears channel 2 at time $t_1$, it needs to listen on channel 2 again at time $t_1 + (1 + M)$ to determine whether the $t_1$ time slot is an interspersed time slot or not. In this example, the jammer knows the $t_1$ time slot is a non-interspersed slot because it detects channel 2 again at time $t_1 + (1 + M)$. Thus, the jammer knows all channel 2 positions in the GOS sequence (e.g., $t_1$, $t_1 + (1 + M)$, $t_1 + 2(1 + M)$, and so on) and jams them. However, if the jammer hears channel 2 at $t_2$, it would fail to hear channel 2 again at time $t_2 + (1 + M)$ because the $t_2$ time slot is an interspersed time slot. Once it determines the interspersed time slot, the jammer stops verifying (listening $(1+M)$ slots later) for all detected channels on all other time slots. Since all other time slots are non-interspersed time slots, the jammer immediately knows the detected channel positions in the GOS sequence. For all detected channels on non-interspersed time slots, the jammer immediately and exactly jams them for the rest of the GOS channel hopping sequence. Moreover, the jammer can compute the entire GOS sequence in a maximum $(\frac{M}{2}+1)(1+M)$ time slots and, on average, $\lceil(\frac{M}{4}+1)\rceil \times (1+M)$ time slots and then it can completely jam the remaining channels with a single jammer. Therefore, our CDJAs should decrease the rendezvous probability of the GOS scheme.

### 4.2.3 Random Channel Rendezvous

We now revisit the Random CH rendezvous algorithm [6] to mitigate the CDJAs. In fact, there are no guaranteed rendezvous algorithms under any jamming attacks so that the Random CH rendezvous algorithm could be the most robust CH algorithm against jamming attacks. In the Random system, a sender and receiver randomly select one channel out of the $M$ available channels for each time slot. We will see it is the "best" against the CDJAs. The formula $\sum(1/M)(1-1/M)^{k-1} * k = M$ shows the expected TTR for the Random scheme is $M$ time slots when there are no jamming attacks. However, it is unfeasible for any jammer to estimate the CH sequence for either the sender or receiver or where the Random sender and receiver might rendezvous in the Random scheme. Thus, the expected TTR for the Random scheme would be almost the same as $M$ under the CDJAs. Hence, the Random scheme for CRNs can be an effective, efficient and secure rendezvous scheme against jamming attacks.

### 4.2.4 Jamming Attacks on DRSEQ and CRSEQ

DRSEQ and CRSEQ are included here just to complete the jamming attacks on the blind rendezvous algorithms given in [29]. Deterministic Rendezvous Sequence (DRSEQ) [12] generates a rendezvous sequence for a multichannel access network that is k-shift-invariant for all $k$. A rendezvous sequence for $M$ available channels can be generated as follows:

$$a_i = \begin{cases} i+1 & \text{for } 0 \le i \le M-1 \\ e & \text{for } i = M \\ 2M - i + 1 & \text{for } M+1 \le i \le 2M \end{cases}$$

where $e$ denotes empty slot and the number of elements is $2M+1$. For example, the two nodes, $A$ and $B$ with $M = 5$ available channels generate the DRSEQ sequence

$$1, 2, 3, 4, 5, e, 5, 4, 3, 2, 1, 1, 2, 3, 4, 5, e, 5, 4, 3, 2, 1, ...$$

Node $A$ starts at slot $0$ and $B$ start at any non-negative integer slot $k$. Then the nodes, $A$ and $B$ rendezvous within $2M+1$ time slots after node $B$ starts.

Jongmin Shin et. al. proposed Channel Rendezvous Sequence (CRSEQ) algorithm [43] to generate a rendezvous sequence for CRNs. This algorithm has the k-shift-rendezvous property for all $k$ as described in [43]. The CH algorithm generates the rendezvous sequence based on properties of triangular numbers and the Chinese Remainder Theorem (CRT). Triangular numbers are given by $T_n = \frac{n(n+1)}{2}$ where $n$ is

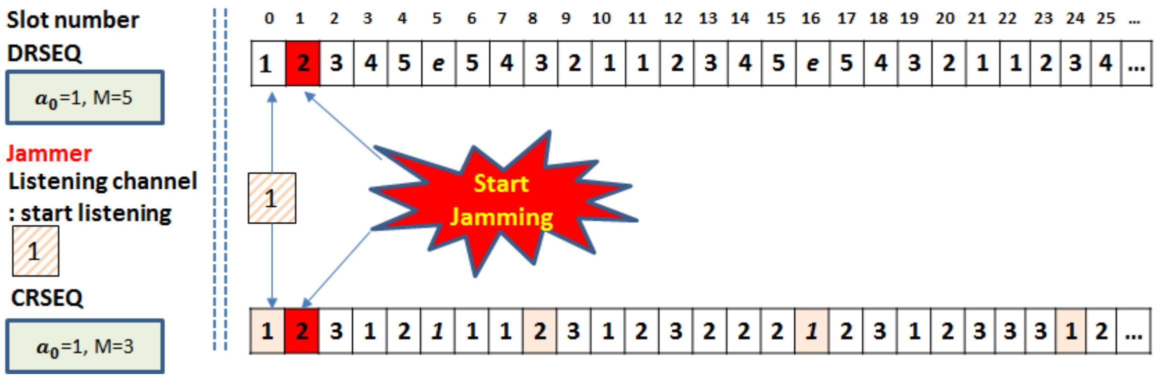


Figure 4.4: A jamming example on both DRSEQ and CRSEQ

an integer. Then the rendezvous sequence consists of $M$ subsequences as seen in Figure 4.4. The CH algorithm for all $M \geq 2$ channels is given by the following:

$$a_i = \begin{cases} z\ MOD\ M + 1 & \text{for } 0 \leq y < 2P - 1 \\ x\ MOD\ M + 1 & \text{for } 2P - 1 \leq y < 3P - 1 \end{cases}$$

where $(z = \frac{x(x+1)}{2} + y)\ mod\ P$, $x = \lfloor \frac{i}{3P-1} \rfloor$, $y = i\ mod\ (3P-1)$, $0 \leq i < P(3P-1)$, and $P$ is the smallest prime number greater than or equal to $M$. Using the CRSEQ CH algorithm, two nodes can rendezvous in $S = P(3P-1)$ slots and $S$ is the number of elements of the sequence.

Both DRSEQ and CRSEQ schemes provide the shortest MTTR for symmetric and asymmetric cases, respectively. However, both are deterministic and hence easily jammed with one listening channel and one jamming channel as illustrated in Figure 4.4. We assume that the jammer resides in the network before the communication starts. Then jammer listens on channel $a_0$, the first channel for both schemes, and jams the remaining channels after hearing $a_0$. Therefore, DRSEQ and CRSEQ are excellent in a benign environment but are not practical under jamming attacks due to their completely deterministic sequences.

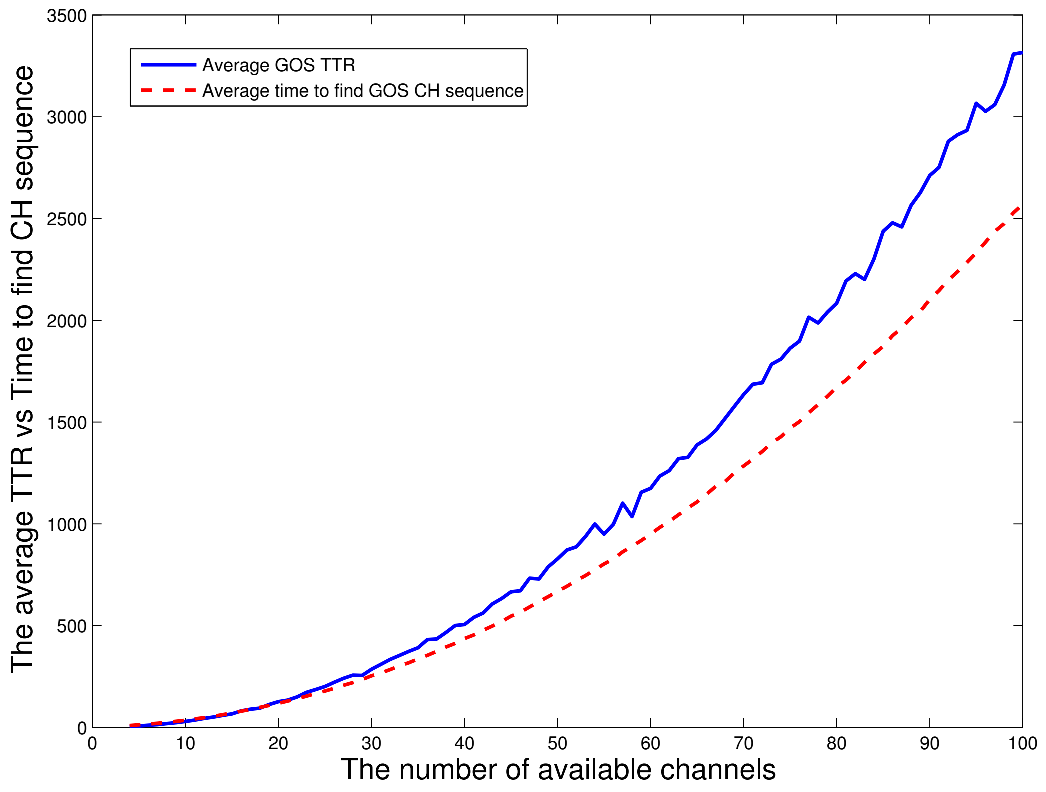


Figure 4.5: The average time of finding the GOS sequence using two listening channels

## 4.3 Evaluation

We implemented the CDJA on Matlab 2010b to evaluate its effectiveness against the GOS scheme. We implemented the natural scenario where the jammer resides in the network and is listening on two distinct channels before the communication starts. The jammer has one jamming channel. We also implemented the Random rendezvous scheme to demonstrate that it is more robust against the CDJA compared to the GOS scheme. The jamming attack for the Random scheme jams two random channels, different for each slot, for the entire rendezvous time.

First, we compare the expected time to find the entire GOS sequence with the expected TTR of the GOS scheme where the number of available channels $M$ varies from $4$ to $100$. In this implementation, we used the more efficient sequence location algorithm given in section 4.2.2. Figure 4.5 shows the expected TTR of the GOS scheme and the expected time for finding the GOS channel hopping sequence. We ran $1000$ simulations for each number of available channels and calculated the average TTR of the GOS scheme and expected time for finding the GOS sequence. Our simulation results show that the average time for finding the GOS sequence is close to $\lceil \frac{M}{4} \rceil$ rounds (i.e.,$\lceil \frac{M}{4}(1+M) \rceil$ time slots). Since we used the more efficient channel location algorithm, this time should be slightly less than the bound $\lceil (\frac{M}{4}+1) \rceil \times (1+M)$ and it is. In the CDJA, if the jammer detects the interspersed time slot or verifies the non-interspersed time slot, the jammer immediately starts to jam the detected channels for the remaining rounds of the GOS sequence. Moreover, the jammer can compute the entire GOS sequence in the maximum $\lceil \frac{M}{2} \rceil$ rounds and, on average, less than $\lceil \frac{M}{4} \rceil$ rounds. This means that the sender and receiver cannot rendezvous after, on average, at most $\lceil \frac{M}{4}(1+M) \rceil$ time slots under the CDJA. But the jammer also jams the verified channels, which are detected as non-interspersed time slots, from the second to the $\lceil \frac{M}{4} \rceil$ rounds. From [6], the overall expected TTR for the GOS is $\frac{M^4+2M^2+6M-3}{3M(M+1)}$ time slots. The expected time to determine the GOS sequence, $\lceil (\frac{M}{4}+1) \rceil \times (1+M)$, is less than the expected time to rendezvous for all M (actually close for small values of M) as, indeed, seen in Figure 4.5. But we have jammed earlier slots as soon as they are determined to further decrease the probability of rendezvous. Hence, the theoretical results on the expected TTR and the expected time to jam shows CDJA will severely limit the GOS chances to rendezvous. This is clearly seen in Figure 4.6 and the rendezvous probability is higher for small values of M as predicted.

Next we compare the rendezvous probability for the MC, JS, GOS, and Random schemes under the CDJA. We make the usual assumption that there is no start time synchronization between sender and receiver and the jammer resides in the network before the sender starts. Figure 4.6 shows the probability of rendezvous for all systems under the CDJA. In [37, 38], we have shown theoretically that a channel detecting jammer for MC and JS schemes can compute their step-lengths and generate their CH sequences early in their attempted rendezvous period. Thus, we consider only the number of slots equal to the MTTR for this implementation. Since the MTTR of the MC and JS schemes are $2P$ and $3P$ time slots respectively (see details in [37, 38]), the simulations in Figure 4.6 are run for $3P$ time

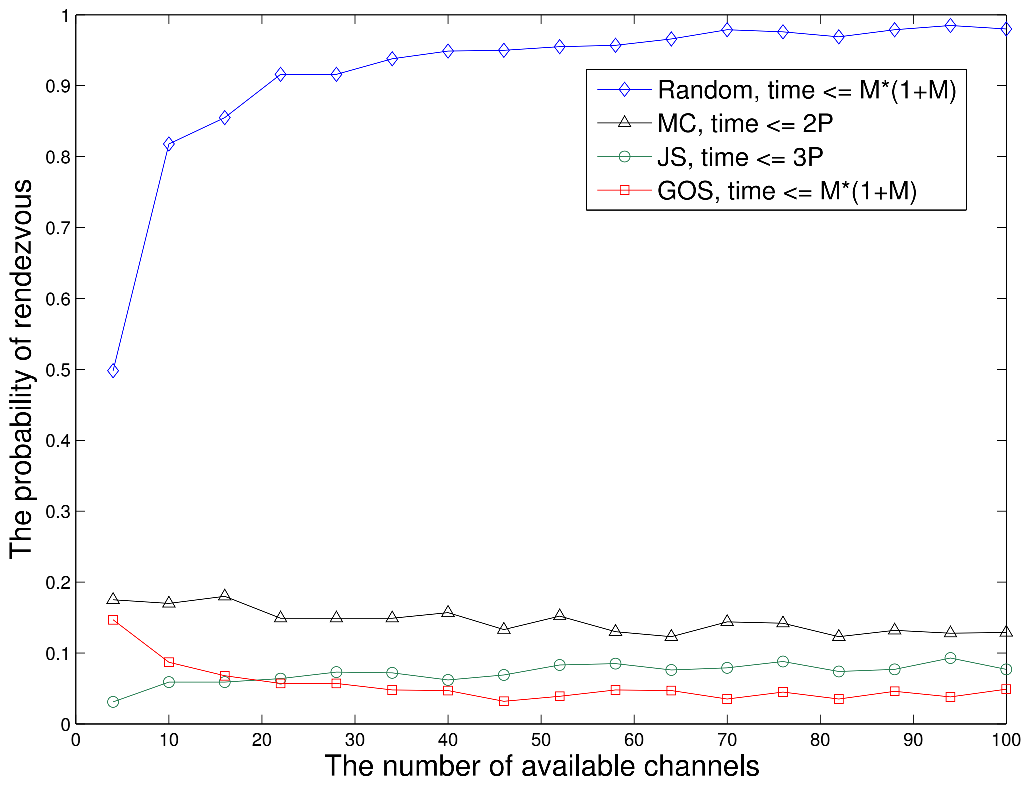


Figure 4.6: The probability of rendezvous for the GOS, MC, JS, and Random schemes under jamming attacks

slots. The probabilities of rendezvous for the MC, JS, and GOS schemes are all dramatically decreased under CDJAs. The rendezvous probability of the MC scheme is less than $20\%$ for any $M$ available channels. For the JS and GOS scheme, the probability of rendezvous decreases to under $10\%$ for almost all $M$ channels. However, the rendezvous probability for the Random is almost steady and more than $90\%$ for most $M$ channels. Therefore, CDJA is extremely effective for the MC, JS, GOS schemes but is minimally so for the Random scheme.

## 4.4 Conclusion

In this chapter, we presented our modified CDJA and demonstrated its effectiveness against the GOS scheme [6, 13]. The channel detecting jammer is able to take advantage of the GOS algorithm to find the sender's CH sequence within the maximum time slots of $(\frac{M}{2} + 1) \times (1 + M)$ and an upper bound expected time slots of $(\frac{M}{4} + 1) \times (1 + M)$. Significantly, the jammer can immediately jam the detected channels and can completely jam the sender after the maximum of $(\frac{M}{2} + 1) \times (1 + M)$ time slots. To remedy this, we revisited the Random CH rendezvous scheme. Our simulation results demonstrate that the rendezvous probability of the GOS, JS, and MC systems under the CDJA is dramatically decreased to less than $10\%$, $10\%$ and $20\%$ respectively for most available channels $M$.

In addition, since upper bounds of the expected time to rendezvous for GOS, JS, MC, DRSEQ, CRSEQ and Random are $\frac{M^4+2M^2+6M-3}{3M(M+1)}$, $5P/3+3$, $2P^2/(P-1)$, unknown, unknown, and $M$ respectively (see [6, 29] and results in this chapter), Random could be the most effective, efficient and robust rendezvous but it does sacrifice the guaranteed rendezvous time some others have. We should note that DRSEQ and CRSEQ have MTTR of $2M + 1$ and $P(3P - 1)$ respectively but that does not change that Random could have the best ETTR.

We have shown that MC, JS, GOS, DRSEQ, and CRSEQ are all very susceptible to debilitating jamming attacks and the Random scheme is almost impervious to such attacks. Hence, it appears that the Random scheme provides more security when timely rendezvous for CNRs are required.

The results in this chapter complete the jamming of the symmetric state-of-the-art blind rendezvous algorithms as presented in [29].

Chapter 5

# Channel Detecting Jamming Attacks on Enhanced Jump Stay and Countermeasures

In our previous work (see Chapter 3.3), we demonstrated that our Channel Detecting Jamming Attacks (CDJAs) can dramatically decrease the probability of rendezvous for the symmetric JS algorithms. In this chapter, we expand our CDJA to the Enhanced Jump Stay (EJS) algorithm [26] to demonstrate its effectiveness for both the symmetric and asymmetric EJS models. Our simulation results show that a channel detecting jammer can compute the entire symmetric EJS channel hopping sequence and thus reduce the rendezvous success rate from $100\%$ to less than $10\%$ using two listening channels. In addition, our CDJA can significantly reduce the rendezvous probability of the asymmetric EJS system (e.g., less than $15\%$ for ratios of $\frac{|m_1|}{|M|} = 0.5$ where $m_1$ is the number of sender's available channels and the total number of available channel is $M = 40$). To mitigate this problem, we revisit the Random rendezvous scheme for the symmetric model and extend it to the asymmetric model to increase the probability of the rendezvous against the CDJAs. Overall, the Random scheme vastly outperforms the EJS algorithm for both the symmetric and asymmetric cases when there are security concerns about a channel detecting jammer. Finally, we show for asymmetric rendezvous, the Random scheme is superior to the Modified

Modular Clock (MMC) [6].

## 5.1 Introduction

In previous chapters, we demonstrated that the CDJA can dramatically decrease the rendezvous success probabilities of the state-of-the-art symmetric blind rendezvous algorithms. In this chapter, we investigate on the Enhanced Jump Stay (EJS) scheme [26] that enhances the previous JS scheme [25] to decrease the expected time to rendezvous for the asymmetric model from $O(P^3)$ to $O(P^2)$ where $P$ is the smallest prime number greater than the number of available channels $M$. It appears the EJS algorithm is one of the best blind rendezvous algorithms for CRNs based on the non-deterministic CH sequence and the guaranteed rendezvous with reasonable upper bounds for both symmetric and asymmetric models. Thus, this chapter expands our CDJA to the Enhanced Jump Stay (EJS) [26] blind rendezvous algorithm for CRNs to demonstrate its effectiveness for both symmetric and asymmetric models of the EJS scheme. We first present our novel CDJA model against the symmetric EJS using two listening channels which can detect the EJS sender's CH sequence within the first $(\frac{P}{2})$ time slots out of the total $4P$ time slots. Then it jams the remaining channels so that it can dramatically decrease the rendezvous probability of the symmetric EJS. In addition, we show how to mount a debilitating jamming attack against the asymmetric EJS rendezvous scheme. Since the number of available channels might be different for each participant, the effectiveness of our CDJA can vary based on the number of common channels between them.

As an alternative to the EJS scheme, we revisit the Random scheme where the sender and receiver generate their CH sequences by randomly selecting from the $M$ available channels for each time slot. The Random scheme keeps the jammer from finding the sender's CH sequence in CRNs. Our simulation results demonstrate that the symmetric EJS scheme is extremely vulnerable to the CDJAs and even more so than the CDJAs on JS (see Chapter 3.3). On the other hand, the Random scheme for CRNs can be an effective, efficient and robust rendezvous scheme against CDJAs. Moreover, we present the theoretical expected time to rendezvous (ETTR) for the asymmetric Random system based on the number of common channels $G$ between a sender and receiver. The ETTR for the asymmetric Random is $\frac{|m_1| \cdot |m_2|}{|G|}$ where $|m_1|$ and $|m_2|$ are the number of available channels for the sender and receiver, respectively out of the total number of available channels $M$. The ETTR of the asymmetric EJS is $4P(P+1-G) - [4PG(P-G) + G/2]/(|m_1| \cdot |m_2|)$. Since there are no guaranteed rendezvous algorithms under any jamming attacks, the asymmetric Random outperforms the asymmetric EJS in the ETTR even when there are no jamming attacks.

The primary contributions of this chapter are to first demonstrate that our CDJA poses a major security problem for the remaining state-of-the-art symmetric and asymmetric blind rendezvous algorithms, EJS and MMC, in CRNs since any secondary user or even group of users can be denied access to the network with high probability. Second, our simulation results and theoretical analysis demonstrate that the symmetric and asymmetric Random CH rendezvous algorithms are the most robust CH algorithm against jamming attacks. Third, the Random CH rendezvous algorithm has the best ETTR for any of the asymmetric algorithms.

The rest of the chapter is organized as follows. Section 5.2 describes relevant background for the EJS scheme for both symmetric and asymmetric models. Section 5.3 presents the CDJAs on EJS and Random schemes. Section 5.4 provides the simulations of our jamming attacks for both symmetric and asymmetric EJS and Random schemes. Section 5.5 describes the related work, and finally Section 5.6 concludes the chapter.

## 5.2 Preliminary

Zhiyong Lin et. al. [26] proposes the enhanced jump-stay rendezvous (EJS) algorithm for CRNs that reduces the maximum time to rendezvous (MTTR) and the expected time to rendezvous (ETTR) from $O(P^3)$ to $O(P^2)$ for the asymmetric model with the same order of rendezvous times for the symmetric model, where the number of available channels is $M$ and the $P$ is the smallest prime number greater than $M$ [25]. Both the EJS and JS [25] schemes use the jump and stay patterns but the EJS generates a new sequence structure and a new hopping sequence. First, the EJS scheme increases the length of each round from $3P$ in the JS scheme to $4P$. Second, the EJS scheme re-designs the CH sequence by using a new sequence structure. It uses the fixed step-length $r$ for all rounds while the JS scheme changes the

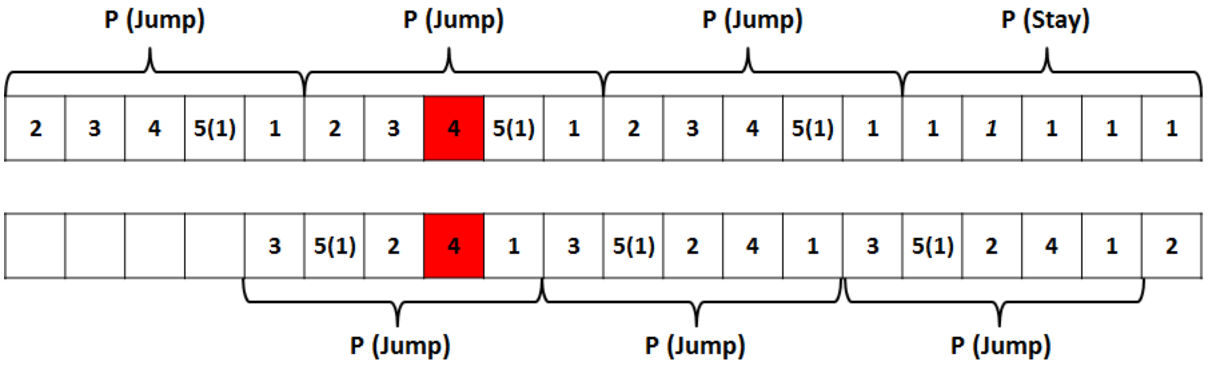


Figure 5.1: An example of the symmetric EJS Rendezvous

step-length $r$ for each round. The EJS scheme uses only the index of available channels for the step-length $r$ rather than using the index of available or unavailable channels for the step-length $r$ in the JS scheme. In addition, the EJS scheme changes the starting index $i$ every $4P$ time slots instead of using $3MP$ time slots in the JS scheme.

In the EJS system, each user uses the EJS CH algorithm to generate a CH sequence in rounds that consists of one jump pattern and one stay pattern. The total time slots of the sequence is $4P$ time slots composed of the first $3P$ time slots for the jump pattern and the subsequent $P$ time slots for the stay pattern. In jump pattern, the user starts from the initial channel $i$ and jumps with the step-length $r$ for $1 \leq i \leq P$ and $1 \leq r \leq M$. Then the user stays on the channel $r$ for the entire stay pattern. In the EJS scheme, the channel $c$ in the jump-pattern is determined by

$$c = ((i + t \times r - 1) \; mod \; P) + 1, \tag{5.1}$$

where $c$ is the channel at time $t = t \; mod \; 4P$. If $c$ is greater than $M$, then it is re-mapped into $[1, M]$ (i.e., $c = ((c - 1) \; mod \; M) + 1$ for $c > M$).

The EJS system has slightly different channel hopping algorithms for the symmetric model and the asymmetric model. In the symmetric model, the number of available channels is the same for all participants. For example, Figure 5.1 describes a rendezvous of the symmetric EJS scheme where $M = 4$ and $P = 5$. Since there is no time synchronization, users can start at any time slot within the $4P$ time

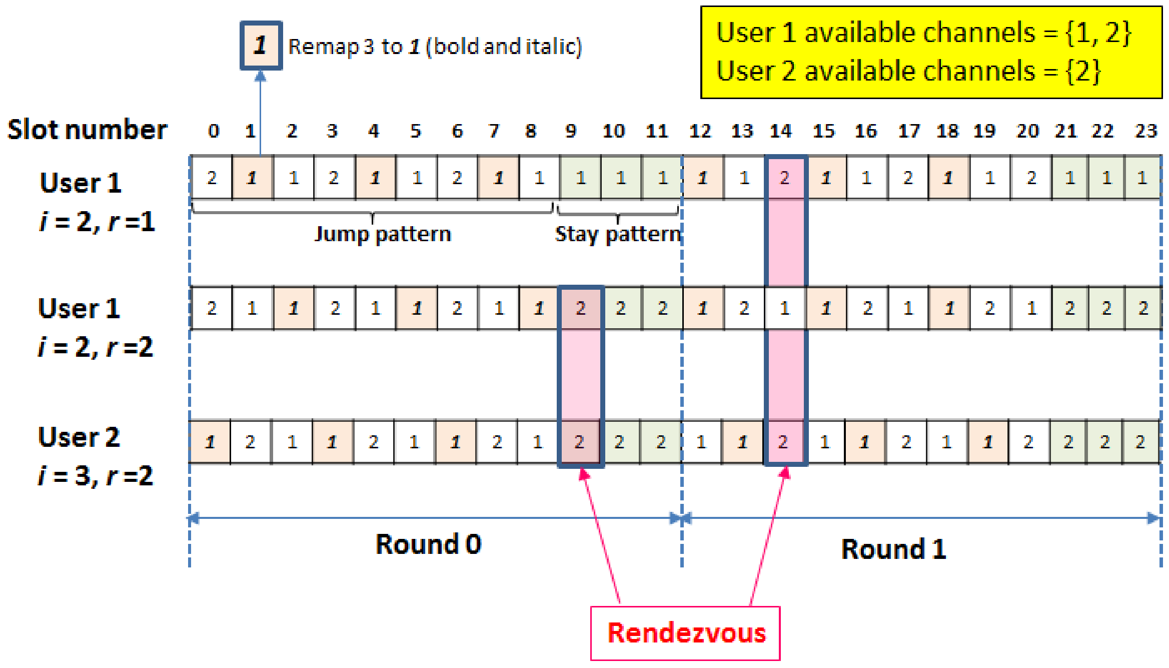


Figure 5.2: An example of the asymmetric EJS Rendezvous

slots of another user but the overlap between two jump-patterns is not less than $P$ time slots. Here we have one user starting with $i_1 = 2$ at $t = 0$ and jumps with the step-length $r_1 = 1$. The other user starts with $i_2 = 3$ at $t = 4$ and jumps with the step-length $r_2 = 2$. Then the two users rendezvous on channel 4 at time slot $t = 7$. In the symmetric EJS scheme, the maximum time to rendezvous (MTTR) is at most $4P$ time slots.

In the asymmetric model, the number of available channels might be different for each participant but there must exist at least one common channel on which to rendezvous. Figure 5.2 illustrates the CH sequences of two users performing asymmetric EJS scheme where $M = 2$ and $P = 3$. Two channels $\{c_1, c_2\}$ are available for the user 1 but only the channel $\{c_2\}$ is available for the user 2. User 1 can select different step-lengths with the same initial channels but user 1 and user 2 rendezvous for both cases within the maximum $4P(P + 1 - G)$ time slots.

## 5.3 Proposed Schemes

Previously (see Chapter 3.3), we proposed a novel CDJA in which the jammer dramatically decreases the probability of rendezvous for the symmetric JS scheme [25] by utilizing the characteristics of the JS jump and stay-patterns. In this section, we extend our CDJA to Enhanced Jump Stay (EJS) algorithm for both symmetric and asymmetric models [26]. We assume that a jammer has capabilities close to a normal user but can listen on two channels at the same time. We first present algorithms showing how a channel detecting jammer with two listening channels can find the sender's CH sequences for the symmetric EJS scheme within $P$ time slots. Then it can jam the sender's remaining CH sequences after an average $\lfloor\frac{(P+1)}{2}\rfloor$ time slots using one jamming channel. For the asymmetric EJS scheme, we assume that a jammer can determine the available channels of the sender. For example, if a sender has available channels $m_1$ out of $M$, then the jammer knows $m_1$ and selects two random listening channels from $m_1$ instead of selecting them from $M$. Using this information, the jammer can find the sender's CH sequence for the asymmetric EJS scheme with high probability. The probability of finding the sender's CH sequence before the sender and receiver rendezvous depends on the number of common channels between the sender and receiver. To mitigate these jamming attacks, we revisit the Random rendezvous algorithm for the symmetric blind rendezvous model and extend the Random to the asymmetric model. We also present the theoretical expected time to rendezvous (ETTR) for the asymmetric Random system, which vastly outperforms the ETTR of the asymmetric EJS system even when there is no jamming. Finally, we compare the effectiveness of the CDJAs on EJS and Random schemes for both symmetric and asymmetric models.

### 5.3.1 CDJA on Symmetric EJS

We consider one sender and one receiver in the CRNs trying to rendezvous under the CDJA. In the symmetric EJS scheme, each SU uses the jump-stay channel hopping algorithm to generate its CH sequences described in Section 5.2. This guarantees that the sender and receiver can rendezvous within the maximum $4P$ time slots when there are no jamming attacks. However, our channel detecting jammer takes the advantage of the jump-stay channel hopping properties to find the sender's CH sequences.

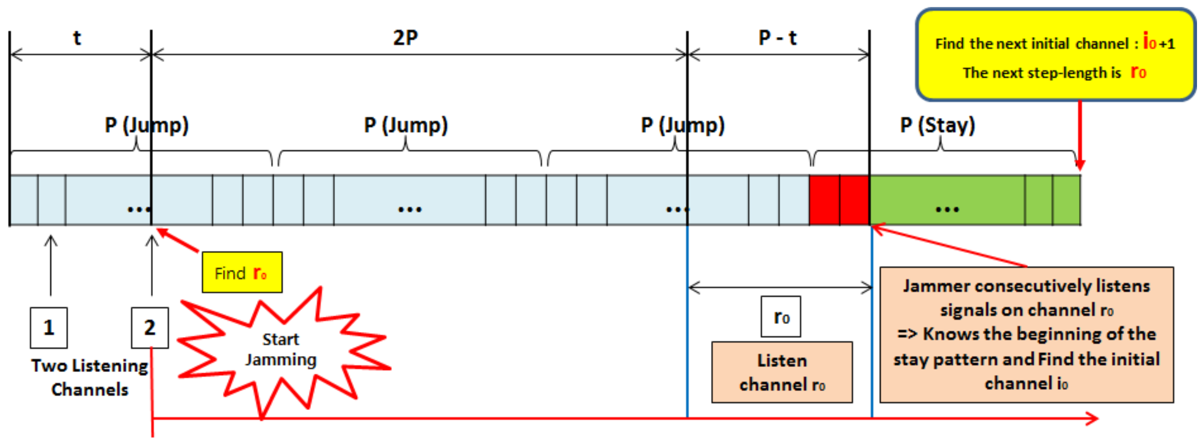

t
2P
P - t
Find the next initial channel : $i_0+1$
The next step-length is $r_0$
P (Jump)
P (Jump)
P (Jump)
P (Stay)
Find $r_0$
1
2
Two Listening Channels
Start Jamming
$r_0$
Listen channel $r_0$
Jammer consecutively listens signals on channel $r_0$ => Knows the beginning of the stay pattern and Find the initial channel $i_0$

Figure 5.3: The CDJA example on the symmetric EJS

In our CDJA, we assume the normal situation where the jammer resides in the network before the communication starts. Then the jammer waits for the sender's signals with two listening channels (LCs). The sender is the secondary user who starts the communication first. Using two listening channels, the jammer is guaranteed to find the sender's CH sequence within $P$ time slots and within $\lfloor\frac{(P+1)}{2}\rfloor$ time slots on average. Then it can jam the sender's remaining CH sequences after an average $\lfloor\frac{(P+1)}{2}\rfloor$ time slots using one jamming channel. In the most optimistic rendezvous case, the sender and receiver are synchronized and even then they must rendezvous within an average $\lfloor\frac{(P+1)}{2}\rfloor$ time slots or the rendezvous will be jammed. Moreover, the probability of rendezvous will be significantly less because the sender and receiver are generally not synchronized. Thus, the probability of rendezvous for the symmetric EJS will be decreased dramatically under the CDJA since EJS guarantees rendezvous in $4P$ time slots but CDJA jams EJS after only $\lfloor\frac{(P+1)}{2}\rfloor$ time slots on average.

Now we describe how the channel detecting jammer finds the sender's CH sequences in the symmetric EJS scheme. As we illustrated in our previous work (see Chapter 3.3), it is important for the jammer to find the step-length of the sender as fast as possible because the step-length is the key value to generating the EJS CH sequence (see Equation 5.1). Using the step-length, the jammer can exactly generate the entire CH sequences of the sender. Therefore, the jammer can jam the remaining channels in the CH sequences with a single jammer. Figure 5.3 describes how the channel detecting jammer finds the sender's step-length in the symmetric EJS scheme using two listening channels. In our previous work (see Chapter 3.3), we described how the jammer can calculate the step-length of the sender's CH sequences for the JS scheme from two channel hits and then it can exactly generate the sender's CH sequences. For example, if the jammer receives the sender's signals on a channel $c_1$ at time $t_1$ and on the another channel $c_2$ at time $t_2$, then the jammer can calculate the sender's step-length $r_0$ using

$$c_1 = (i_0 + t_1 \times r_0 - 1)\%P + 1 \tag{5.2}$$

$$c_2 = (i_0 + t_2 \times r_0 - 1)\%P + 1 \tag{5.3}$$

and then equation (5.3) $-$ (5.2) gives,

$$((t_2 - t_1) \times r_0)\%P = (c_2 - c_1).$$

Thus, the step-length $r_0$ is

$$r_0 = \lceil \frac{(P \times k + (c_2 - c_1))}{(t_2 - t_1)} \rceil \tag{5.4}$$

where the $i_0$ is the initial channel, the $P$ is the smallest prime number greater than the number of available channels $M$, and the $k$ is the constant that satisfies the property of $1 \leq r_0 \leq M$ and matches the jump-pattern. Most of time, the jammer can randomly select listening channels but the jammer should avoid selecting channels from the $[1, P - M]$ because the EJS algorithm maps the $(M, P]$ channels back onto the $[1, P - M]$ channels which are then duplicated. That is, if the jammer selects the channels that are mapped onto by $(M, P]$, then it is possible that the step-length can be ambiguous because the channels from $[1, P - M]$ are duplicated with other channels in the EJS algorithm. Once the jammer computes the step-length $r_0$ using Equation 5.4, it can compute the sender's CH sequences from the time of the second detected signal and jam the remaining time slots until the stay pattern using $c_{next} = ((c_{last} + r_0 - 1)\%P) + 1$.

However, the jammer cannot immediately compute the initial channel $i_0$ because it has no knowledge about the beginning of the sender's communication. To find the initial channel $i_0$, the jammer needs to add an additional step to find the beginning of the stay-pattern. In Figure 5.3, the jammer detects the second channels at time $t$ is in the first jump-pattern. After $2P + t$ time slots, the jammer listens on the channel $r_0$ to find the beginning of the stay-pattern. This clearly must occur in the first jump-pattern. That is, if the jammer hears two consecutively signals on the channel $r_0$ between $2P + t$ and $3P + 2$ time slots, then the jammer knows that the first time slot is the beginning of the stay pattern. When the jammer finds the beginning of the stay pattern, it can compute the initial channel $i_0$ and find the beginning of the communication. Thus, the jammer can generate the sender's sequences for the consecutive periods from known $r_0$ and $i_0$ information and jam all slots continuously with a single jammer. Therefore, the probability of rendezvous for the symmetric EJS system will be dramatically decreased under the CDJA.

### 5.3.2 CDJA on Asymmetric EJS

The asymmetric model for the EJS CH rendezvous algorithm assumes that a sender and receiver might have different available channels but they must have at least one common channel between them for rendezvous [26]. The basic model of the asymmetric EJS system is to expand the set of possible channels to the union of the available channels (e.g., $m_1$ for a sender and $m_2$ for a receiver) and let $G$ denote the intersection of the available channels (i.e., $G = m_1 \cap m_2$). We let $M$ denote the set of all channels under possible consideration and we make exactly the same assumptions here as in [26]. Then the asymmetric EJS system guarantees rendezvous in $4P(1 + P - G)$ time slots.

Here we assume CDJA does not have time synchronization with the sender but uses two listening channels (LCs). In addition, we assume that our channel detecting jammer resides in the sender's network before the communication starts. Thus, the jammer can detect or sense the same available channels as the sender's. This is important for the jammer to select two random LCs from $m_1$ in order to guarantee that it can hear two channels within $P$ time slots. The probability of hearing two channels early in the first $P$ time slots increases as the number of unavailable channels (i.e., $|M - m_1|$) increases due to the re-mapping of EJS. However, if the jammer uses the total available channels $M$ from which to select two LCs, it may never hear those channels. This means that the jammer might not find the correct step-length $r_0$ within $P$ time slots. Thus, the jammer must choose its listening channels from $m_1$.

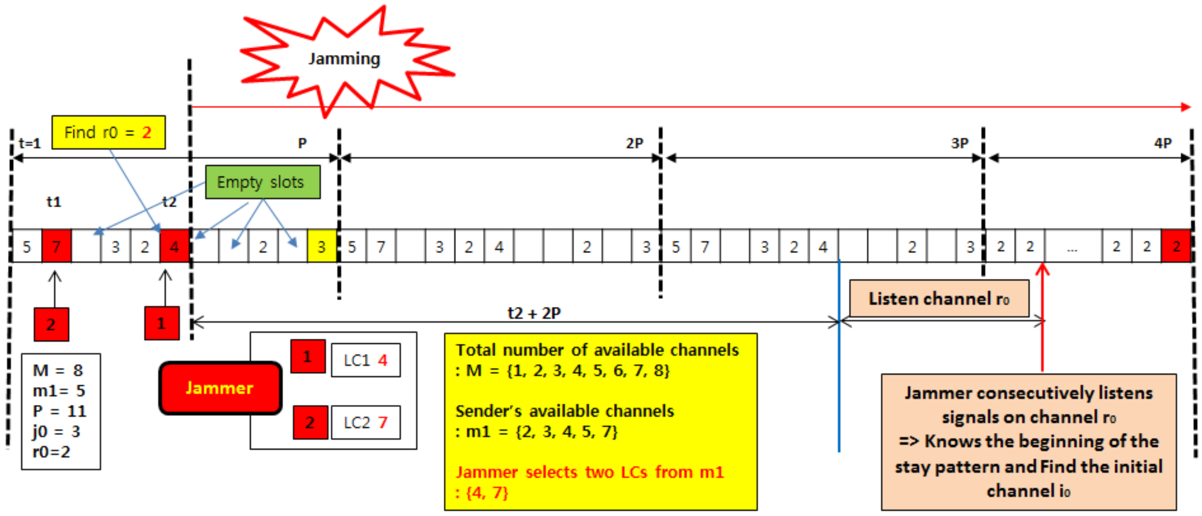
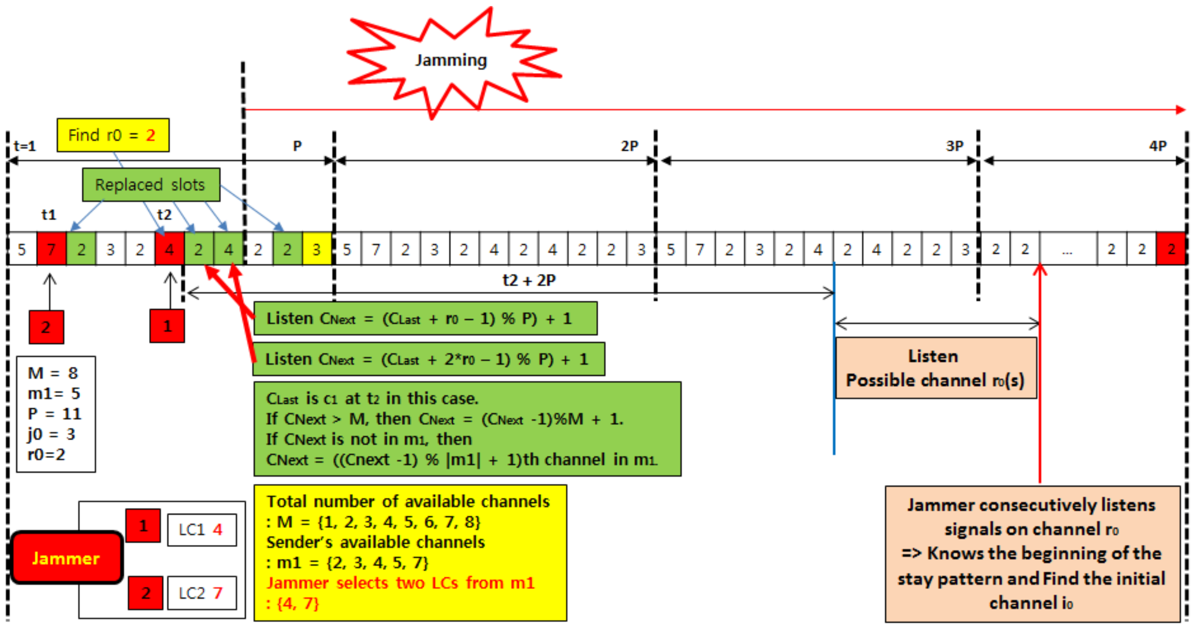

Jamming
Find r0 = 2
t=1
P
2P
3P
4P
Empty slots
t1
t2
5 7 3 2 4 2 3 5 7 3 2 4 2 3 5 7 3 2 4 2 3 2 2 ... 2 2 2
t2 + 2P
Listen channel r0
2
1
M = 8
m1= 5
P = 11
j0 = 3
r0=2
Jammer
LC1 4
LC2 7
Total number of available channels : M = {1, 2, 3, 4, 5, 6, 7, 8}
Sender's available channels : m1 = {2, 3, 4, 5, 7}
Jammer selects two LCs from m1 : {4, 7}
Jammer consecutively listens signals on channel r0 => Knows the beginning of the stay pattern and Find the initial channel i0

Figure 5.4: The CDJA on the asymmetric EJS without replacement

To compute the sender's CH sequences, we consider two cases: the asymmetric EJS without replacement algorithms and the asymmetric EJS with replacement algorithms for unavailable channels. We include the asymmetric EJS without replacement algorithms because Lin et. al. in [26] used it as a basis to compare EJS with other algorithms. In our CDJA, the jammer computes the sender's CH sequence from detected channel information. However, if a detected channel is a replaced channel, then the jammer might not find the correct $r_0$ so that it needs additional steps to verify whether the step-length is correct or not. This is addressed in the next paragraph. But our CDJA in the asymmetric EJS without replacement algorithm works similar to the CDJA on the symmetric EJS. Figure 5.4 shows how the jammer selects two LCs for the asymmetric EJS without replacement channels for unavailable channels (i.e., in $c \in M - m_1$). Suppose that the total number of available channels $M$ is 8 and the number of available channel for the sender $m_1$ is 5. The sender randomly selects the step-length $r_0$ to be 2 from $m_1 = \{2, 3, 4, 5, 7\}$ and the initial channel $i_0$ to be 3. Since the jammer resides in the sender's network, the jammer knows $m_1$ and it can select two random LCs $\{4, 7\}$ from $m_1$. Thus, the jammer hears two channels within the $P$ time slots and computes the correct step-length $r_0$. Therefore, our CDJA determines the sender's $r_0$ and $i_0$ for the asymmetric EJS without replacement algorithms in exactly the same method as for the symmetric EJS. Hence, its asymmetric rendezvous probability decreases dramatically



Jamming
Find r0 = 2
t=1
P
2P
3P
4P
Replaced slots
t1
t2
5 7 2 3 2 4 2 4 2 2 3 5 7 2 3 2 4 2 4 2 2 3 5 7 2 3 2 4 2 4 2 2 3 2 2 ... 2 2 2
t2 + 2P
2
1
Listen CNext = (CLast + r0 – 1) % P) + 1
Listen CNext = (CLast + 2*r0 – 1) % P) + 1
CLast is c1 at t2 in this case.
If CNext > M, then CNext = (CNext -1)%M + 1.
If CNext is not in m1, then
CNext = ((Cnext -1) % |m1| + 1)th channel in m1
M = 8
m1= 5
P = 11
j0 = 3
r0=2
Jammer
1 LC1 4
2 LC2 7
Total number of available channels
: M = {1, 2, 3, 4, 5, 6, 7, 8}
Sender's available channels
: m1 = {2, 3, 4, 5, 7}
Jammer selects two LCs from m1
: {4, 7}
Listen
Possible channel r0(s)
Jammer consecutively listens signals on channel r0
=> Knows the beginning of the stay pattern and Find the initial channel i0

Figure 5.5: The CDJA on the asymmetric EJS with replacement

under the CDJA.

However, Lin et. al. subsequently integrates a replacement operation into the asymmetric EJS that replaces unavailable channels in their CH sequence with available channels in $m_1$ by

$$c_j = ((j-1)\ mod\ |m_1|) + 1)^{th}\ channel\ in\ m_1. \tag{5.5}$$

Thus, we need to modify our CDJA so that it applies effectively here as well. Figure 5.5 shows how the jammer finds the correct $r_0$ with high probability for the asymmetric EJS with replacement channels. Suppose we hear $c_1$ at $t_1$ and $c_2$ at $t_2$ as before. We compute the step-length $r_0$ as before but $c_1$ or $c_2$ may be a replaced channel and hence $r_0$ is not certain. Without loss of generality assume $t_1 < t_2$ and let $c_{last} = c_2$ and $c_{last}$ be the index of $c_2$ in the $[1, M]$ channels. We let $c_{last}$ and $c_{next}$ be both the channel in $[1, M]$ and the index of the channel in $[1, M]$ depending on whether it is a channel or an integer. To verify the step-length $r_0$, we need to listen on the next channel

$$c_{next} = ((c_{last} + r_0 - 1)\%P) + 1$$

If $c_{next} \notin [1, M]$, then $c_{next} = ((c_{next} - 1)\%M) + 1$. If $c_{next} \notin m_1$, $c_{next} = ((c_{next} - 1)\%|m_1| + 1)^{th}$ channel in $m_1$ from Equation 5.5. For example, suppose that $M = 10$, $P = 11$, $m_1 = \{1, 3, 7, 8, 9\}$, $|m_1| = 5$, $i_0 = 7$, and $r_0 = 1$. Then the CH sequence for the first $P$ time slots is

$$8, 9, 9\{R\}, 1, 1, 3\{R\}, 3, 8\{R\}, 9\{R\}, 1\{R\}, 7$$

where $\{R\}$ means a replaced channel for an unavailable channel. If the two listening channels $L_1 = 8$ and $L_2 = 9$, then the jammer can compute $r_0 = 1$ at time $t = 2$. So, $c_{last} = L_2 = 9$ and $c_{last}$ is a non-replaced channel in this case. Thus, $c_{next}$ is

$$c_{next} = (c_{last} + r_0 - 1)\%P + 1 = (9 + 1 - 1)\%11 + 1 = 10.$$

However, $c_{next} = 10$ is not in $m_1$. Thus, the $c_{next}$ is

$$c_{next} = ((c_{next} - 1)\%|m_1| + 1)^{th} = ((10 - 1)\%5 + 1)^{th} = 5^{th} \text{ channel in } m_1,$$

which is $c_{next} = 9$. The $c_{next}$ is the same as the channel $C$ at $t = 3$ from the EJS CH algorithm [26]. That is, the channel $C$ at $t = 3$ is

$$C = (i_0 + t * r_0 - 1)\%P + 1 = (7 + 3 * 1 - 1)\%11 + 1 = 10.$$

Since the channel $C$ is not in $m_1$, the EJS scheme replaces $C$ with $5^{th}$ channel in $m_1$ that is the same as $c_{next} = 9$. And the consequent channel $c_{next+1}$ is

$$c_{next+1} = (c_{last} + 2 * r_0 - 1)\%P + 1 = (9 + 2 - 1)\%11 + 1 = 11.$$

Since $c_{next+1} = 11 > M$, $c_{next+1}$ is

$$c_{next+1} = (c_{next+1} - 1)\%M + 1 = (11 - 1)\%10 + 1 = 1.$$

The $c_{next+1}$ is the same as the channel $C$ at $t = 4$ from the EJS CH algorithm. That is, the channel $C$ at $t = 4$ is

$$C = (i_0 + t * r_0 - 1)\%P + 1 = (7 + 4 * 1 - 1)\%11 + 1 = 11.$$

Since $C > M$, the channel $C$ at $t = 4$ is re-mapped into $[1, M]$ that is the same as $c_{next+1} = 1$. Hence we see that the next two actual channels match our detecting algorithm and we start to jam. This means that if the $c_{last}$ is a non-replace channel, $c_{next}$ is the same channel as the channel computed by using the index $j$, $t$, $r_0$, and $i_0$. But if the $c_{last}$ is a replaced channel, we might not find the correct $r_0$ and also $c_{next}$ may be wrong even when we use the correct $r_0$. If we detect the channel $c_{next}$, then, with high probability, the $r_0$ might be the correct step-length. However, if the $c_{last}$ is a replaced channel, then the $c_{next}$ is based on a replaced channel $c_{last}$ and a probably wrong $r_0$. This $c_{last}$ could be the same channel as the $c_{next}$ from a non-replaced channel $c_{last}$ and the correct $r_0$. Thus, we need to listen on one more channel on $c_{next+1} = ((c_{last} + 2 * r_0 - 1)\%P) + 1$ to increase the probability of finding the correct $r_0$. That is, we can determine that the step-length $r_0$ would be correct with higher probability when we detect two consecutive channels $c_{next}$ and $c_{next+1}$ that are correct immediately following the $t_2$ time slot. Then, we start to jam on $c_{next+2} = (((c_{last} + 3 * r_0 - 1)\%P) + 1$ as before. We could continue to listen on $c_1$ here and if $r_0$ was not confirmed, use ($c_2$, $t_2$) and the new ($c_1$, $t_3$) to repeat the procedure. We did not do this as the simpler $r_0$ finding algorithm worked well. If we do not detect the next two consecutive channels with the possible $r_0$, then we continue to listen on the same channels $c_1$ and $c_2$ for up to $P$ slots beyond $min\{t_1, t_2\}$. When we hear $c_1$ or $c_2$ at $t_3$, we compute two new $r_0$s based on each pair of the three data points. The new $r_0$ for the old pair of same data points would be incorrect. Since we avoid selecting listening channels from re-mapped channels $(M, P]$, each channel of $m_1$ exists only one time as a non-replaced channel within $P$ time slots. We do not repeat the computation with the original two data points $c_1$ at $t_1$ and $c_2$ at $t_2$. Thus, with high probability, we can compute all possible $r_0$s including the correct step-length within $P$ + $min\{t_1, t_2\}$ time slots. However, we might not select the correct $r_0$ from all possible $r_0$s on proper time slot because we only have two listening channels to verify all the possible $r_0$. To increase the probability of selecting correct $r_0$, we first use the majority $r_0$ from all possible $r_0$s because it has the highest probability of being correct. Then we revisits all possible $r_0$s to find the correct step-length up to $max\{t_1, t_2\} + 2P$ time slots.

Alteratively, we can switching channels whenever the jammer hears the signals on its listening channels instead of staying on the same channels. Suppose we hear $c_1$ at $t_1$ and $c_2$ at $t_2$ as before. We compute

$r_0$ and generate the next channel $c_{next}$ to verify whether it is the correct $r_0$ or not. If the possible $r_0$ is not the correct one, we continue to listen immediately after $t_2$ on two other channels. When we hear $c_3$ at $t_3$, we compute two new $r_0$s based on each pair of the three data points. We do not repeat the computation with the original two data points $c_1$ at $t_1$ and $c_2$ at $t_2$. Then we use only new $r_0$s to determine whether it is the correct $r_0$ or not.

For both cases, we continue to listen and compute all possible $r_0$s up to $t_2 + 2P$ time slots. If we find the correct $r_0$, we compute the CH sequence on which to jam. Once we get beyond $max\{t_1, t_2\} + 2P$, then we listen on the two most frequent $r_0$s to find the beginning of the stay-pattern as was discussed in 5.3.1. Through these steps, we know with high probability both $r_0$ and $i_0$ and can jam every subsequent channel until we complete $4 * P * (1 + P - G)$ time slots. Since $i_0$ is just incremented for the consecutive rounds and $r_0$ is unchanged, we can trivially compute all subsequent CH sequences and jam them as well. We do not claim that the above jamming attack is optimal but we will show it is very effective.

### 5.3.3 Jamming Attacks on Random CH Rendezvous

We revisit the Random CH rendezvous algorithm used for the symmetric model to mitigate the CDJAs and extend it to the asymmetric model. In fact, there are no guaranteed rendezvous algorithms under any jamming attacks and our CDJA can cripple the symmetric and asymmetric EJS systems so that the symmetric and asymmetric Random CH rendezvous algorithms could be the most robust CH algorithms against jamming attacks.

In the symmetric Random system, a sender and receiver randomly select one channel for each time slot out of the $M$ available channels. As we discussed in Chapter 3.3, the symmetric Random scheme is the ”best” against the CDJAs. The formula $\sum(1/M)(1 - 1/M)^{k-1} * k = M$ shows the expected time to rendezvous (ETTR) for the symmetric Random scheme is $M$ time slots when there are no jamming attacks. However, it is unfeasible for any jammer to estimate the CH sequence for either the sender or receiver or where the symmetric Random sender and receiver might rendezvous in the Random scheme. Thus, the ETTR for the symmetric Random scheme would be almost the same as $M$ under the CDJAs.

In the asymmetric Random system, a sender and receiver randomly select one channel out of the number of available channels $m_1$ and $m_2$ for each time slot, respectively. However, the ETTR for the asymmetric Random system is not $M$ time slots because it depends on the number of common channels $G$ between them. Suppose we have the number of available channels $m_1$ for the sender and $m_2$ for the receiver out of the total number of available channels $M$. Define random variables $X$ as sample drawn from $m_1$ and $Y$ as sample drawn from $m_2$. We assume that the samples are drawn completely randomly from each set, and they are independent of each other. Let $G$ be the number of common channels of $m_1$ and $m_2$, i.e., $G = m_1 \cap m_2$. Then the probability of selecting same channel $\mathbb{P}\{X = Y\}$ is

$$\begin{aligned}
\mathbb{P}\{X = Y\} &= \sum_{x \in m_1} \mathbb{P}\{Y = x | X = x\} \cdot \mathbb{P}\{X = x\} \\
&= \sum_{x \in m_1} \mathbb{P}\{X = x\} \cdot \mathbb{P}\{Y = x\} \\
&= \sum_{x \in m_1 \backslash G} \mathbb{P}\{X = x\} \cdot \mathbb{P}\{Y = x\} + \sum_{x \in G} \mathbb{P}\{X = x\} \cdot \mathbb{P}\{Y = x\} \\
&= \sum_{x \in G} \mathbb{P}\{X = x\} \cdot \mathbb{P}\{Y = x\} \\
&= \sum_{x \in G} \frac{1}{(|m_1| \cdot |m_2|)} \\
&= \frac{|G|}{|m_1| \cdot |m_2|}
\end{aligned}$$

where the first equality is from the law of total probability, the second equality is from the independence of $X$ and $Y$, the third equality is due to $\mathbb{P}\{Y = x\} = 0$ if $x \in m_1 \backslash G$. Since $X$ and $Y$ are uniformly distributed, $\mathbb{P}\{Y = x\} = \frac{1}{|m_2|}$, and $\mathbb{P}\{X = x\} = \frac{1}{|m_1|}$ for any $x \in G$. Hence the fourth and the last equality follow. Now let $R$ be the number of trials until the first rendezvous. Then the probability of rendezvous $\mathbb{P}\{R = k\}$ is

$$\mathbb{P}\{R = k\} = (1 - p)^{k-1} p$$

where $k$ is the $k$-th independent trial and the probability of success $p$ is $\mathbb{P}\{X = Y\} = \frac{|G|}{|m_1| \cdot |m_2|}$. Thus, the expected value $\mathbb{E}\{R\}$ is

$$\begin{aligned}\mathbb{E}\{R\} &= 1*p+2(1-p)p+3(1-p)^2p+\cdots \\ &= \sum_{k=1}^{\infty} k(1-p)^{k-1}p = \frac{1}{p} \\ &= \frac{|m_1|\cdot|m_2|}{|G|}\end{aligned}$$

Therefore, the ETTR for the asymmetric Random system is $\mathbb{E}\{R\} = \frac{|m_1|\cdot|m_2|}{|G|}$. Since it is also unfeasible for any jammer to estimate the CH sequence of either the sender or receiver for the asymmetric Random system, the ETTR would be the almost same as $\mathbb{E}\{R\}$ under CDJAs. Hence, both the symmetric and asymmetric Random schemes for CRNs can be an effective, efficient and secure rendezvous scheme against jamming attacks.

## 5.4 Evaluation

We implemented the CDJA on Matlab 2010b to evaluate its effectiveness against the EJS scheme. We first implemented the jamming attacks for both the symmetric and asymmetric EJS systems using two listening channels. We then implemented Random schemes to demonstrate that they are more robust against the CDJA compared to EJS schemes. The jamming attack for the Random scheme is jamming two random channels for each time period and that continues for the entire rendezvous time.

First, we compare the expected time to rendezvous (ETTR) for the *symmetric* JS, EJS, and Random schemes when there are no jamming attacks. Figure 5.6 gives the average TTRs for the JS, EJS and Random schemes where the number of available channels $M$ varies from $4$ to 100. We ran $1000$ simulations for each available channel and calculated the average TTR for them. This figure shows that the average TTRs for all schemes increase steadily as the number of available channel increases. Since there is no time synchronization, the receiver can start at any time between $1$ to $4P$ time slots for both Random and EJS schemes and at any time between $1$ to $3P$ time slots for the JS scheme. The expected TTR for the Random is close to $M$ as expected. However, the average TTR of the EJS scheme from

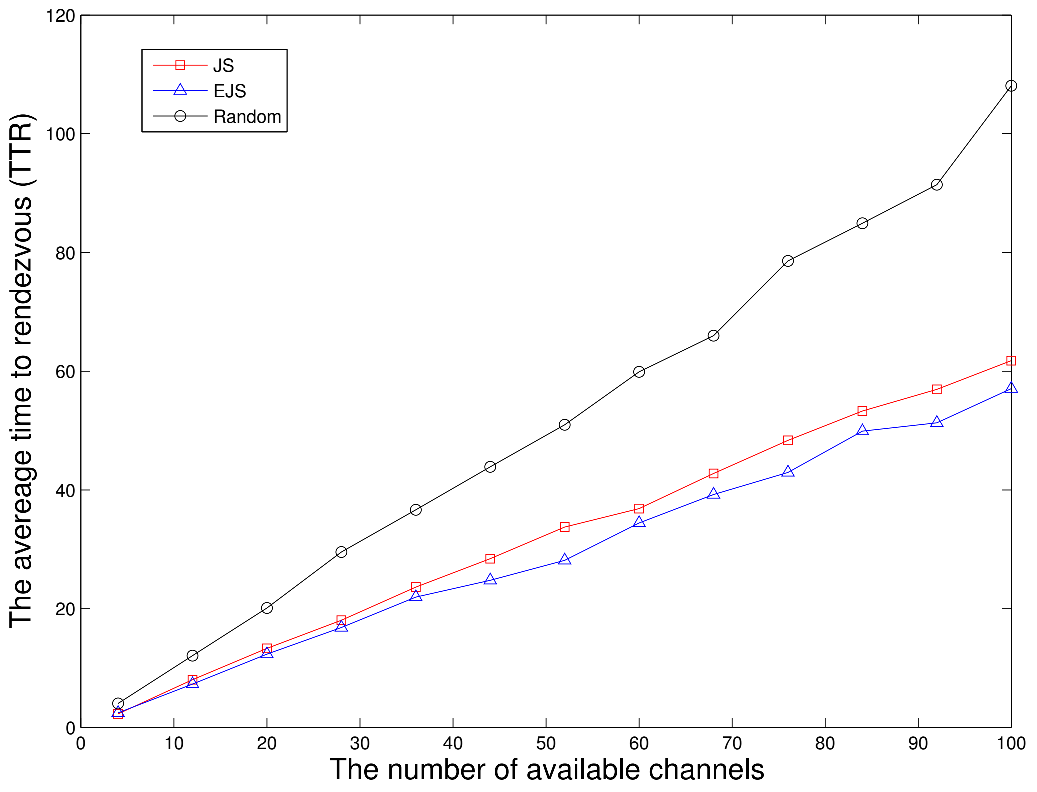


Figure 5.6: The average time to rendezvous (TTR) for the symmetric JS, EJS, and Random schemes without time synchronization

the implementation is around half the available channels $M$ and is slightly better than the average TTR of the JS scheme. Overall, the average TTR for the Random scheme is almost twice that of the other schemes. These results are consistent with the expected time of $M$ for Random and the upper-bound on ETTR of $5P/3 + 3$ and $3P/2 + 3$ for JS and EJS respectively [25, 26].

Next, we compare the rendezvous probability for the *symmetric* JS, EJS and Random schemes under the CDJA. In this experiment, the channel detecting jammer uses two listening channels but it uses a single jammer (actually two jammers for Random). Since the jammer can determine the sender's CH sequences completely within at most $P$ time slots for JS and EJS, a single jammer jams the remaining channels. Figure 5.7 shows the probability of rendezvous for all systems until the end of $3P$ time slots

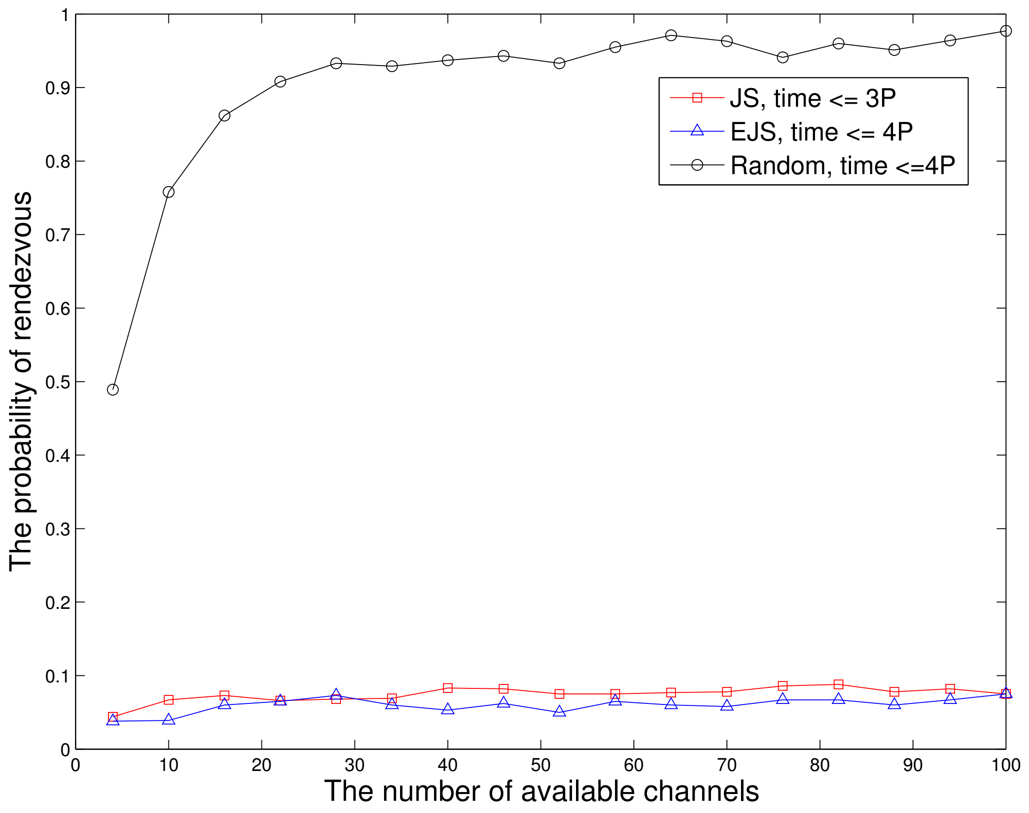


Figure 5.7: The probability of rendezvous for the symmetric JS, EJS, and Random schemes under CDJAs

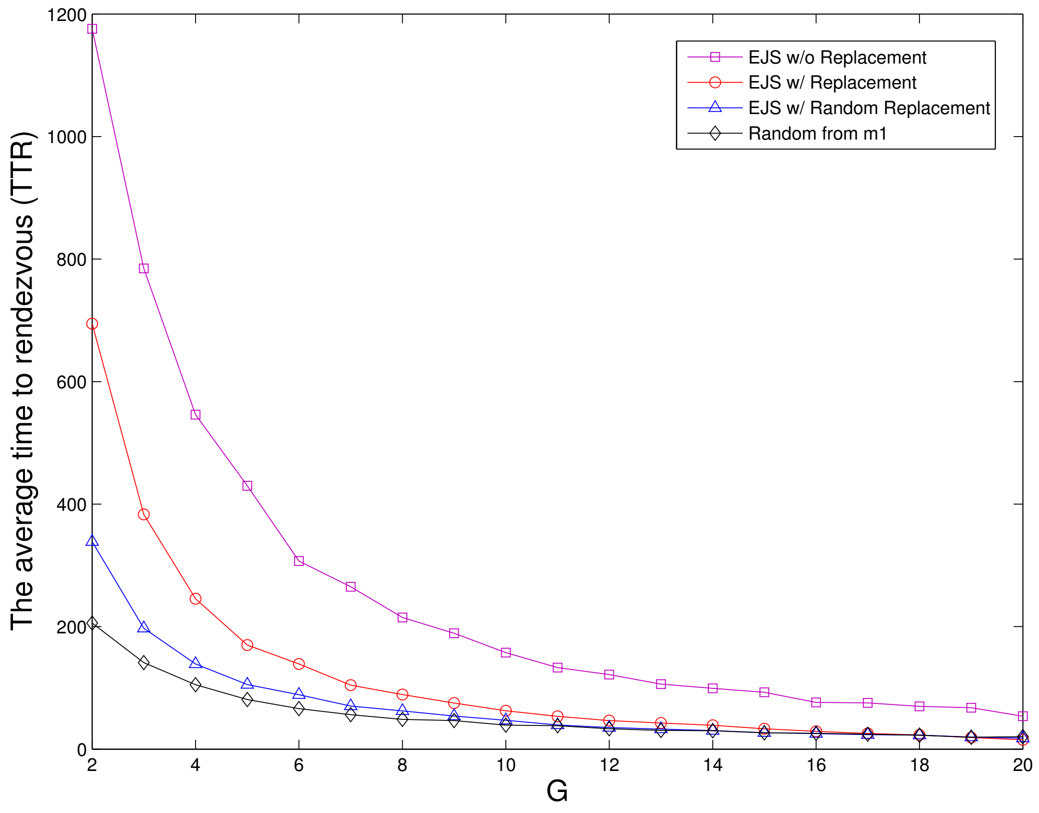


Figure 5.8: The average time to rendezvous (TTR) for the asymmetric EJS, EJS without replacements, EJS with random replacement, and Random schemes for $|m_1| = |m_2| = 20$, M=40, and G = [2-20]

for the JS scheme and $4P$ time slots for the EJS and Random schemes. The probability of rendezvous for both the JS and EJS schemes is dramatically decreased under the CDJA. The rendezvous probability for both JS and EJS schemes decreases to generally under $10\%$ when the jammer uses two listening channels. The probability of rendezvous for the EJS scheme is even slightly less than the JS's rendezvous probability. However, the rendezvous probability for the Random is almost steady and more than $90\%$ and close to $100\%$ for most numbers of available channels $M$.

Next, we compare the expected TTR for the *asymmetric* EJS and Random schemes when there are no jamming attacks. Figure 5.8 gives the average TTRs for the asymmetric EJS and Random schemes where the number of available channels $M$ is 40 but the number of available channels for both the sender and the receiver is 20 ($|m_1| = |m_2| = 20$). The number of common channels $G$ varies from 2 to 20. In the asymmetric EJS [26], authors use a replacement algorithm for unavailable channel to replace $c \notin m_1$ by $((c-1) mod |m_1|)+1)^{th}$ channel in $m_1$ where $|m_1|$ is the number of available channels for the sender. The asymmetric EJS with replacement algorithm can guarantee rendezvous within the maximum time to rendezvous (MTTR) of $4 * P * (1 + P - G)$ time slots. However, the distribution of replacement channel selection for the asymmetric EJS replacement algorithm is not uniformly distributed. Based on our investigation, we conclude that the biased channel selection of the asymmetric EJS replacement algorithm decreases the performance of the asymmetric EJS rendezvous algorithm. To demonstrate this investigation, we implement the EJS with a random replacement algorithm (i.e., uniform distribution of channel selections) for unavailable channels to compare it with the asymmetric EJS replacement algorithm. We also implemented the asymmetric EJS without replacement algorithms for comparison. As shown in Figure 5.8, the average TTR of the asymmetric EJS with random replacement algorithm is approximately half of the asymmetric EJS replacement algorithm and almost three times faster than the symmetric EJS without replacement algorithm. In addition, the average TTRs for both the asymmetric EJS and Random schemes decrease as the number of common channel $G$ increases. For the asymmetric Random scheme, we randomly select a channel from the available channel set $m_1$ for each time slot. Overall, the average TTR for the asymmetric Random scheme is much faster than the asymmetric EJS scheme especially when $G$ is small (e.g., 4 times faster when $G = 2$).

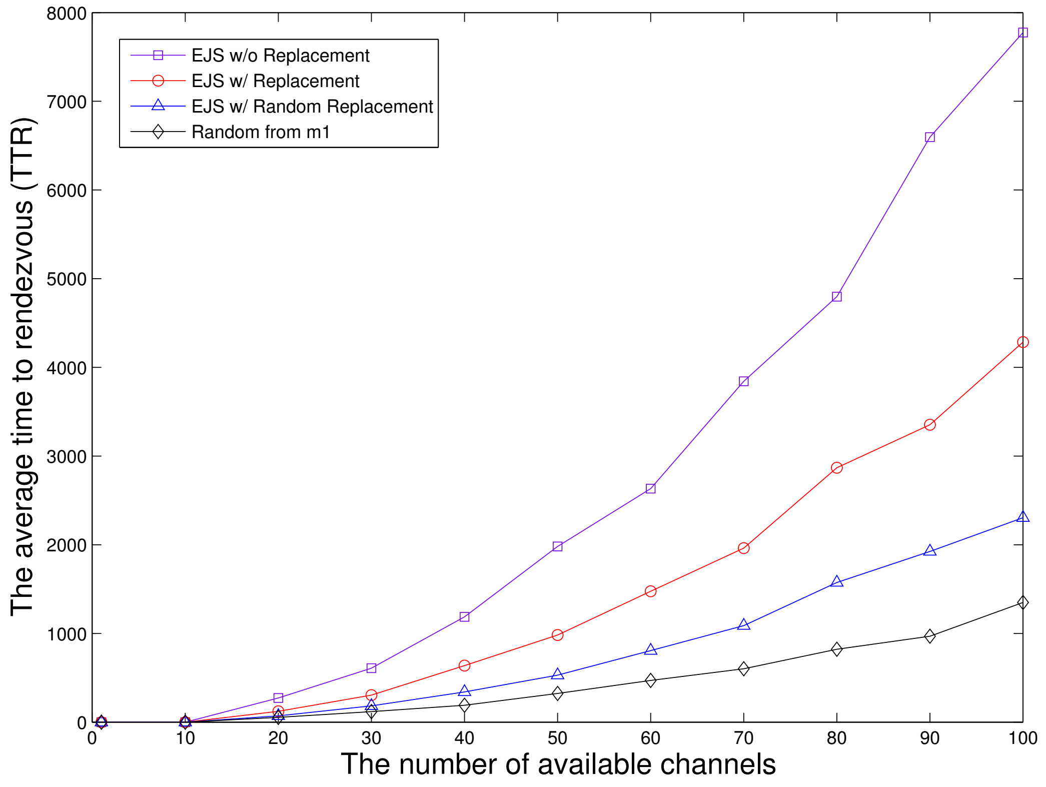


Figure 5.9: The average TTR for the asymmetric EJS, EJS without Replacements, EJS with random replacement, and Random schemes for $|m_1| = |m_2| = 0.5M$, M=[20, 30,.., 100], and G=2

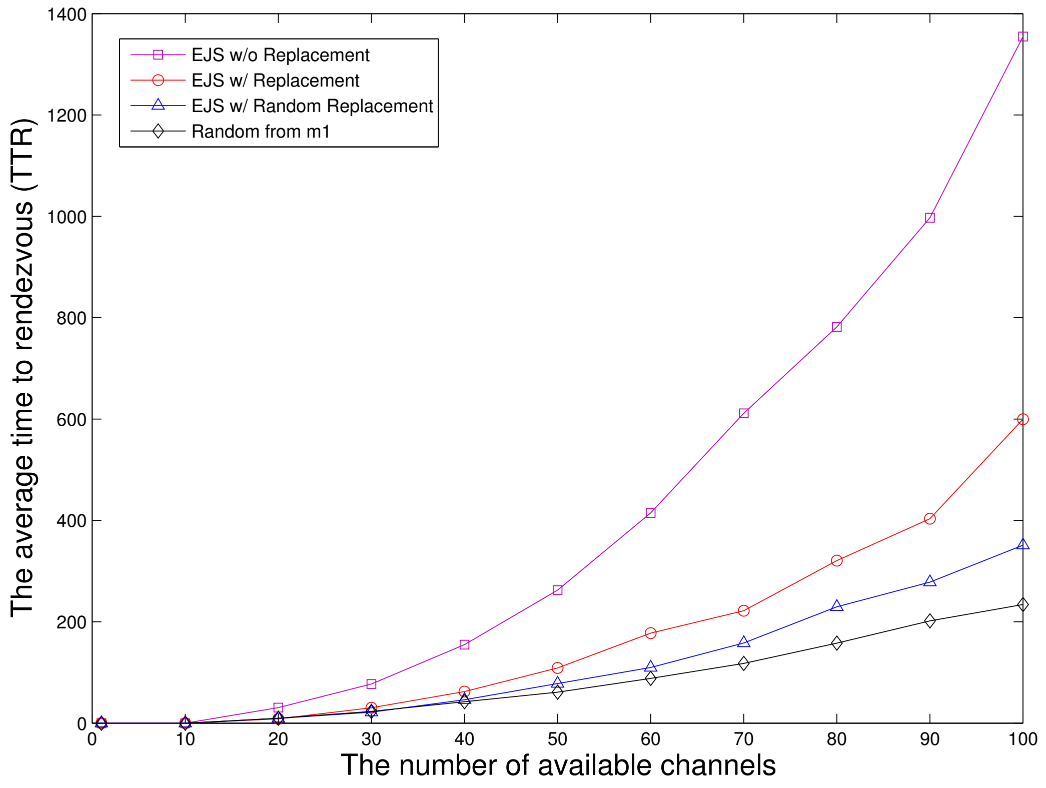


Figure 5.10: The average TTR for the asymmetric EJS, EJS without replacements, EJS with random replacement, and Random schemes for $|m_1| = |m_2| = 0.5M$, M=[20, 30, .., 100], and G=10

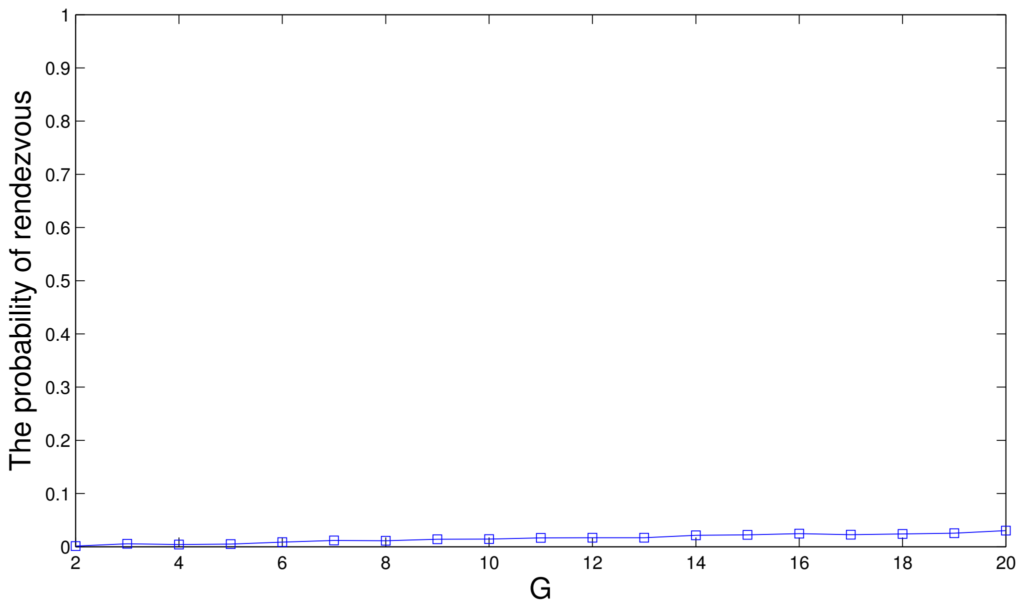


Figure 5.11: The probability of rendezvous for the asymmetric EJS without replacement algorithm under CDJAs for $|m_1| = |m_2| = 20$, M=40, and G = [2 20]

Now, we compare the average time to rendezvous for both the *asymmetric* EJS with replacement and Random algorithms. We consider the number of available channels for the sender $m_1$ to be half of the total number of available channels $M$ (i.e., $|m_1|/|M| = 0.5$). We implement two cases based on the number of common channels $G = \{2\}$ and $G = \{10\}$ for various $M = \{10, 20, \cdots, 90, 100\}$. Figure 5.9 and Figure 5.10 show the average TTR for both the asymmetric EJS replacement algorithms and Random when the number of common channels $G = 2$ and $G = 10$, respectively. For both cases, the average TTRs increase as the number of available channels $M$ increases but decrease rapidly as the number of common channels $G$ increases. The asymmetric Random scheme is the best for the both cases but it does not guarantee rendezvous. However, the asymmetric EJS scheme and, in fact no blind rendezvous scheme, can guarantee rendezvous under jamming attacks or when any of the inputs $m_1$, $m_2$ or $G$ are changing. This last case is probably the most common situation in actual practice.

Next, we compare the rendezvous probability for the *asymmetric* EJS without replacement algorithm under the CDJA. Figure 5.11 shows the probability of rendezvous for $4P$ time slots of the asymmetric

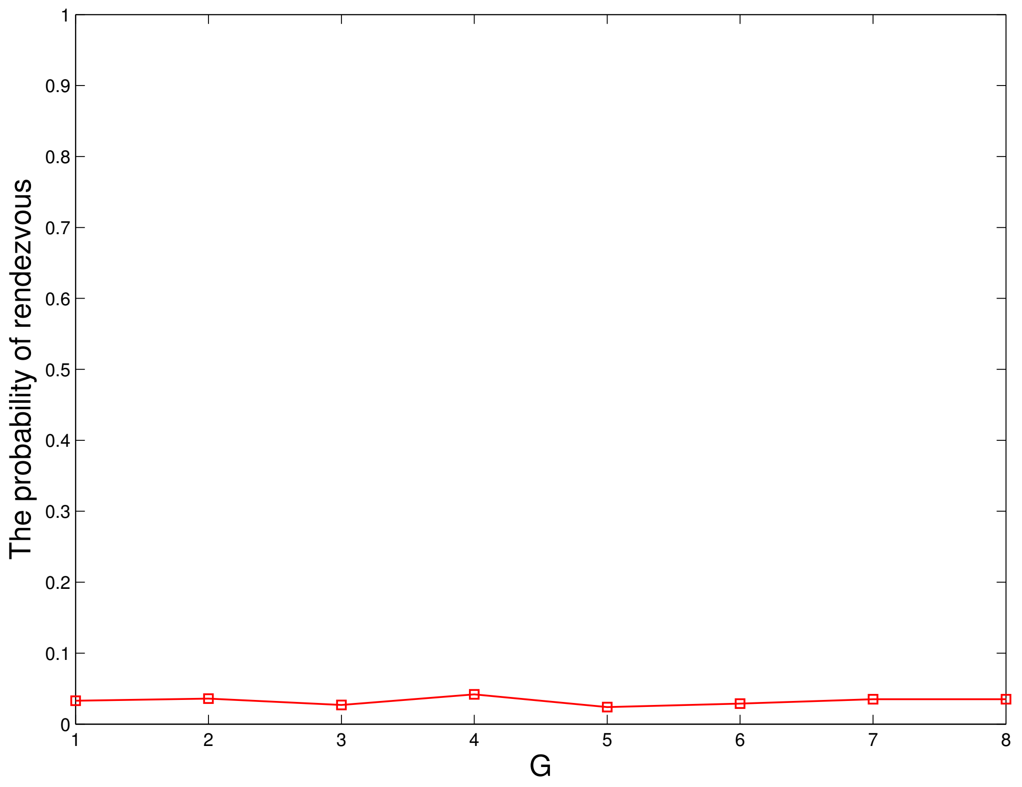


Figure 5.12: The probability of rendezvous for the asymmetric EJS with replacement under CDJAs with $M = 40$, $|m_1| = |m_2| = 0.2M$, and G=[1-8]

EJS scheme. It shows that the probability of rendezvous for the asymmetric EJS scheme is dramatically decreased under the CDJA when there are no replacement algorithms for unavailable channels. The rendezvous probability increases as the number of common channels $G = [2, .., 20]$ increases but it is less than $4\%$ for all $G$ when the jammer uses two listening channels. Moreover, the probability of rendezvous for the EJS scheme is even less than $1\%$ when the number of common channels is less than 7. Therefore, without a replacement algorithm, the asymmetric EJS scheme is extremely vulnerable under our CDJA.

Next, we compare the rendezvous probability for the *asymmetric* EJS with replacement algorithm under the CDJA. Figure [5.12-5.14] show the probability of rendezvous for the asymmetric EJS scheme

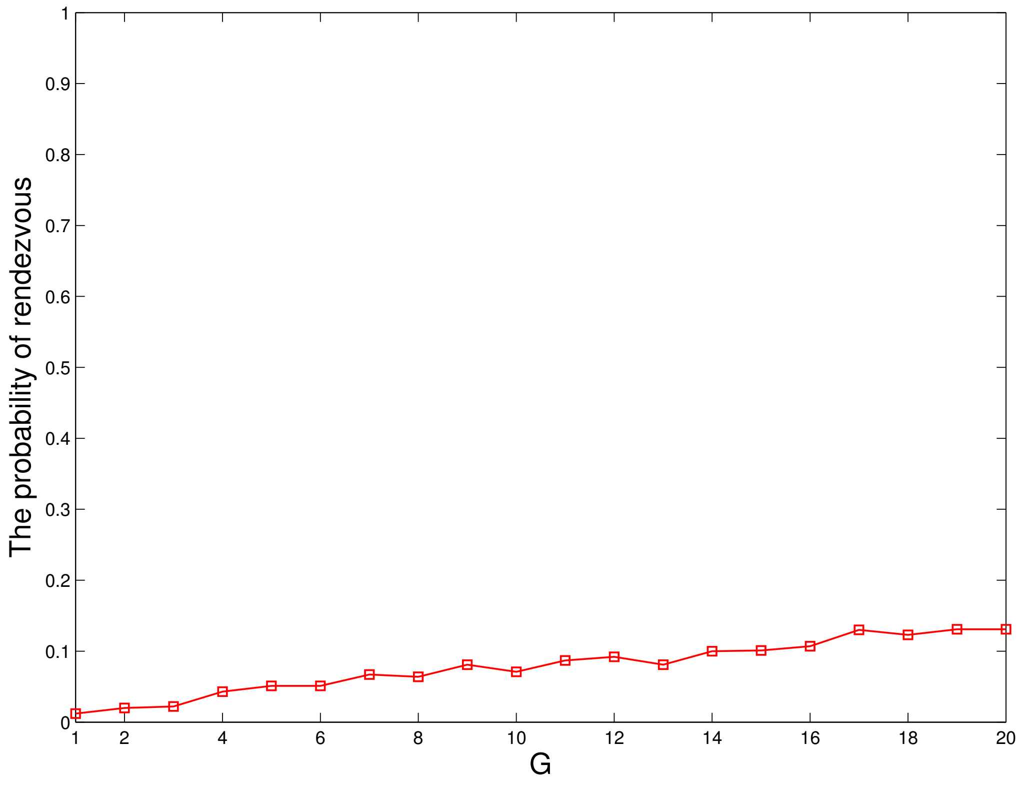


Figure 5.13: The probability of rendezvous for the asymmetric EJS with replacement under CDJAs with $M = 40$, $|m_1| = |m_2| = 0.5M$, and G=[1-20]

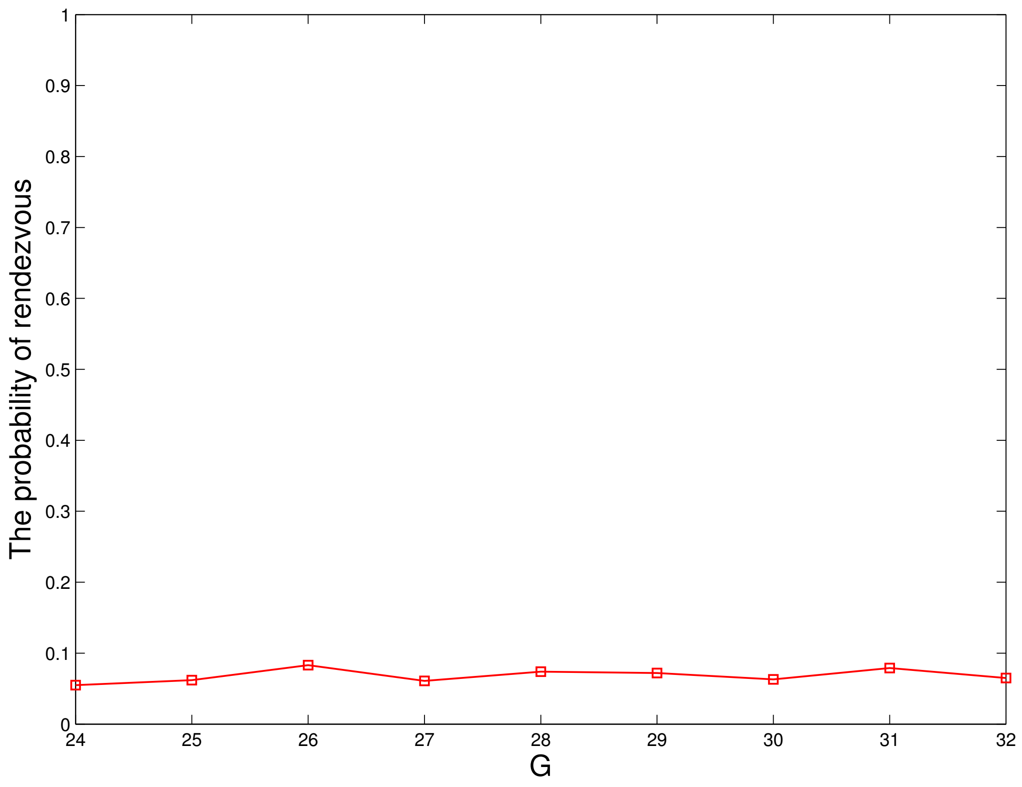


Figure 5.14: The probability of rendezvous for the asymmetric EJS with replacement under CDJAs with $M = 40$, $|m_1| = |m_2| = 0.8M$, and G=[24-32]

for the ratios $|m_1|/|M| = \{0.2, 0.5, 0.8\}$. They show that the probability of rendezvous for the asymmetric EJS scheme with standard replacement is dramatically decreased under the CDJA. The rendezvous probability for all ratios is less than $10\%$ except for the ratio $|m_1|/|M| = 0.5$ when the jammer uses two listening channels. For the ratio $|m_1|/|M| = 0.5$, the probability of rendezvous is the highest compared to other ratios and increases as the number of common channels $G = [1, .., 20]$ increases. As we can see, the rendezvous probability depends on the ratio $|m_1|/|M|$. If the ratio is low, then it is difficult for the jammer to find the correct step-length $r_0$. But the probability of rendezvous for the sender and receiver is also decreased dramatically even though there are no jamming attacks. On the other hand, if the ratio $|m_1|/|M|$ is high, the probability of rendezvous for the sender and receiver will

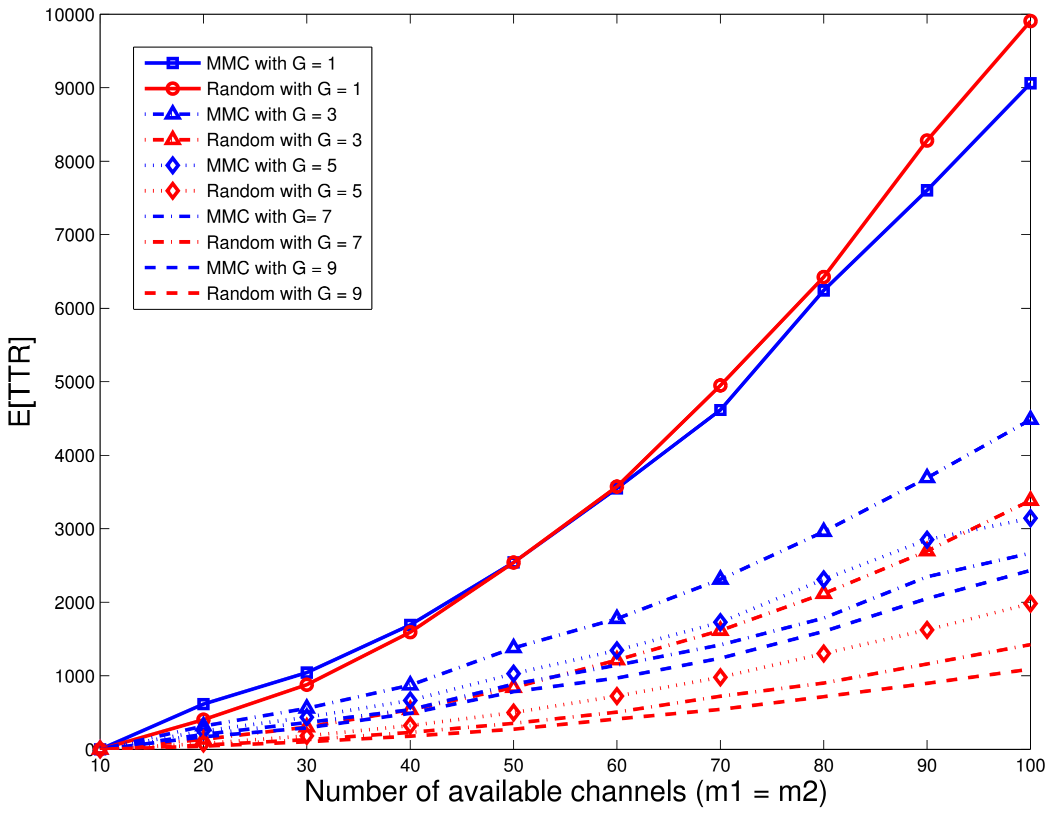


Figure 5.15: The expected time to rendezvous for the MMC and Random schemes with $G = \{1, 3, 5, 7, 9\}$

increase. But the jammer can find the correct step-length $r_0$ in close to $P/2$ times slots as the ratio increases to 1. Therefore, with a replacement algorithm, the asymmetric EJS scheme is also extremely vulnerable under our CDJA.

Finally, we implement the average time to rendezvous for both the Modified Modular Clock (MMC) (i.e., the asymmetric MC) and the *asymmetric* Random algorithms. In the MMC scheme [6], the sender uses both the random replacement in $m_1$ for the channels $(M, P]$ and also uses a random $P$ from $[m_1, 2m_1]$ where $m_1$ is the number of available channel for the sender, $M$ is the total number of available channels, and $P$ is a random prime number between $m_1$ and $2m_1$. The initial channel for the MMC is fixed for the entire rendezvous and the step-length $r_0$ is invariant for $2P^2$ steps. It is possible to effectively jam the MMC even with the random $P$ from $[m_1, 2m_1]$ and it involves using checking for the correct $r_0$ and changing listening channels. This jamming is not as effective for the asymmetric MMC, as it was for the symmetric MC case. There is no need to elaborate on this since Random is more efficient than $MMC$ in the asymmetric scenario as seen in Figure 5.15 and is discussed in the next paragraph. Please see the results in Chapter 2.3 because the asymmetric MMC uses random replacements for the channels $(M, P]$ and $P$ is random as well.

We now show Random is superior to MMC for the asymmetric case. Both Random and MMC systems can not guarantee rendezvous. We have shown in section 5.3.3, that the average time to rendezvous for Random is $\mathbb{E}\{R\} = \frac{|m_1|\cdot|m_2|}{|G|}$ and for MMC is $\mathbb{E}\{\text{MMC}\}= \frac{|p_1|\cdot|p_2|}{|G|}$ [6]. Hence we see that the average time to rendezvous for Random is always less than or equal to that for MMC. Also Random is impossible to jam. It should be noted that MMC is largely a random scheme when $|m_1|/|M|$ is small. We now experimentally compare the average time to rendezvous of asymmetric Random and MMC when the number of common channels is small. Figure 5.15 shows the ETTRs for both the MMC and Random schemes when the number of common channels is $G = \{1, 3, 5, 7, 9\}$ and the number of available channels is same for each participant (i.e., $|m_1| = |m_2|$). The average time to rendezvous for Random is less that for MMC system except for $G = \{1\}$ which is the result of our running only 1000 experiments and the low probability of rendezvousing. For all other $Gs$, the asymmetric Random scheme vastly outperforms the MMC. The average time to rendezvous for the asymmetric Random is close to the theoretical probability of rendezvous $ETTR = \frac{|m_1|*|m_2|}{|G|}$.

## 5.5 Related Work

In [29], Hai Liu et al. reviewed the existing blind rendezvous algorithms for CRNs that are categorized into centralized systems and decentralized systems. Under the centralized rendezvous algorithms [5, 9, 20, 32, 39], the centralized server or the dedicated CCCs control the available channels in CRNs to manage the rendezvous for cognitive radios. For example, the server can set up the common link and schedule transmissions for the SUs. However, these algorithms have limitations discussed earlier.

The decentralized rendezvous algorithms remove the centralized server or the dedicated CCCs but it can still establish one or multiple common control channels from the common spectrum. The *blind rendezvous* occurs where there is no control channel so that the SUs need to find each other blindly. To achieve blind rendezvous, several channel hopping (CH) algorithms [6, 25] have been proposed. Theis et al. [6] proposed several CH algorithms including modular clock algorithm (MC) for the symmetric system and modified MC (MMC) for the asymmetric model. However, these algorithms only guarantee the rendezvous of two users when the users select the different step-lengths for their CH sequences.

Hai Liu et. al. in [25] proposed the Jump-Stay based channel hopping (JS) algorithms to provide guaranteed rendezvous for both symmetric and asymmetric rendezvous in CRNs. The sender and receiver independently generate their own CH sequences through the *JSHopping* algorithms and they can rendezvous within a favorable amount of time compared to other schemes.

Hai Liu et. al. in [26] enhanced the JS algorithm (EJS) to not only provide the same level of rendezvous time for the symmetric model but also reduce the rendezvous time for the asymmetric model from $O(P^3)$ to $O(P^2)$. Again, the EJS scheme is vulnerable to our CDJA in which the jammer takes advantage of the *EJSHopping* algorithms to find the sender's step-length and thereby the CH sequences by using two listening channels but a single jammer.

We revisited the Random rendezvous [6] for the symmetric model to mitigate the CDJA and extend it to the asymmetric model. Due to the random features of the Random scheme, it is unfeasible for the channel detecting jammer to estimate the sender's CH sequences. Thus, the effectiveness of the CDJA is negligible for the symmetric and asymmetric Random schemes.

## 5.6 Conclusion

In this chapter, we presented our modified CDJA and demonstrated its effectiveness against the EJS scheme. The channel detecting jammer is able to take advantage of the EJS algorithm (its method of generating its channel hopping sequence) to find the sender's CH sequence. It can find the sequence of the symmetric EJS within first $P$ time slots using two listening channels. Then the jammer can completely jam the sender's CH sequence after an average $\lfloor \frac{(P+1)}{2} \rfloor$ time slots using just a single channel jammer. For the asymmetric EJS system, the jammer could find the sender's CH sequence with high probability within $P$ time slots and jam the remaining channels. To remedy this jamming problem, we revisited the Random scheme for the symmetric model and extend it to the asymmetric model. Our simulation results demonstrate that the rendezvous probability of the symmetric EJS system under the CDJA is dramatically decreased for all available channels $M$ (e.g., less than $10\%$ for all $M$). Moreover, the rendezvous probability of the asymmetric EJS system under the CDJA is less than $15\%$ for all ratios of $\frac{|m_1|}{|M|}$ with $M = 40$. On the other hand, the rendezvous probability of the symmetric Random is almost steady for all available channels $M$ (e.g., more than $90\%$ for all $M$). Moreover, we present the theoretical expected time to rendezvous (ETTR) for the asymmetric Random system (i.e., $\frac{|m_1|\cdot|m_2|}{|G|}$) and the ETTR would be almost the same under the CDJA because of the random channel selections in the CH sequence. This vastly outperforms the ETTR of the asymmetric EJS system. Therefore, the symmetric and asymmetric Random schemes can be an effective, efficient and robust rendezvous scheme against CDJAs. Finally we have shown that the Random scheme is superior to MMC for the asymmetric model.

Chapter 6

# Limitations of Quorum-based Rendezvous and Key Establishment Schemes against Sophisticated Jamming Attacks

Recently Quorum-based frequency hopping schemes have been studied to increase rendezvous probabilities and to provide fast key establishment techniques in RF communication under jamming attacks. However, these schemes are still vulnerable to sophisticated jamming attacks in which a jammer has the capability of listening and jamming multiple frequencies. In this chapter, we present a sophisticated jamming attack and evaluate its effectiveness on Frequency Quorum-based Rendezvous (FQR) schemes [17]. The sophisticated jammer can find the sender's quorum set within the second frame (i.e., $2k$ time slots) in the FQR system when it has the capability of listening on $k$ frequencies from the minimal $(N, k)$ difference sets. The jammer can completely jam the sender after $2k$ time slots using an average of $\lfloor \frac{k+1}{2} \rfloor$ and a maximum of $k$ frequencies. To remedy this jamming problem, we revisit the role-based rendezvous scheme and extend it to the Role-based Frequency Rendezvous (RFR) scheme. Our simulation results demonstrate that the rendezvous probability of the FQR system under the sophisticated jamming attack dramatically decreased as the number of available channel $N$ increases (e.g., $\leq 35\%$ for $N \geq 30$ in $k^2$ time slots). On the other hand, the rendezvous probability of our RFR scheme is almost steady for all available frequencies $N$ (e.g., $\geq 90\%$ for $N \geq 20$). Therefore, the RFR scheme can be an effective, efficient and robust rendezvous and key establishment scheme against sophisticated jamming attacks.

## 6.1 Introduction

Recently many researchers have proposed different approaches to provide secure key establishment schemes in RF communication under various jamming attacks. Traditional spread spectrum techniques such as Frequency Hopping Spread Spectrum (FHSS) and Direct Sequence Spread Spectrum (DSSS) can be countermeasures to such jamming attacks. However, FHSS and DSSS are required to share a secret key a priori which is the fundamental limitation of these techniques against the reactive jamming attacks [40].

To address this limitation, Quorum-based frequency hopping schemes have been studied to eliminate the dependency on pre-shared keys [17, 22]. In this chapter, we primarily focus on Frequency Quorum-based Rendezvous (FQR) algorithms [17] because these algorithms consider rendezvous and key establishment schemes under various jamming attacks without using common control channels in Cognitive Radio [22]. In [17], the authors utilize the quorum system to avoid a pre-shared frequency hopping sequence. For example, the sender and receiver can select independent random frequency hopping sequences but they can rendezvous within a bounded time because of the non-empty intersection property of cyclic quorum sets [19, 44]. Thus, it can increase the rendezvous probability and decrease the time latency for key establishment under various jamming attacks.

However, FQR systems are still vulnerable to sophisticated jamming attacks in which a jammer has the capability of listening on up to $k$ frequencies and jamming on an average $\lfloor \frac{k+1}{2} \rfloor$ frequencies out of $N$ available frequencies (e.g., $k = 4$ and $k = 12$ for $N = 13$ and $N = 100$, respectively). The sophisticated jammer can find the sender's quorum set within the second frame (i.e., $2k$ out of $k^2$ time slots) and jam the remaining $k - 2$ frames. Thus, the rendezvous probability of the FQR scheme will be decreased dramatically because the sender and the receiver must rendezvous within the first two of $k$ frames.

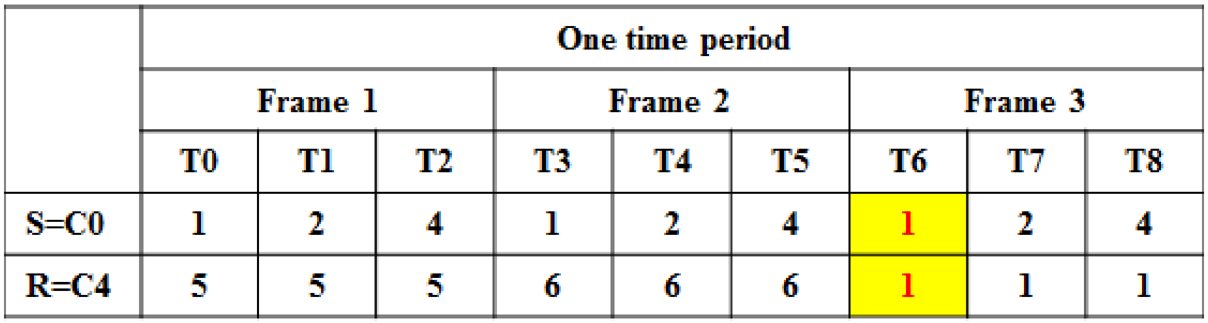
In this chapter, we present the sophisticated jamming attack and particularly evaluate its effectiveness on FQR schemes [17, 24]. Moreover, we revisit the role-based rendezvous scheme [16] and extend it to a Role-based Frequency Rendezvous (RFR) where the sender generates its frequency hopping sequences by permuting $N$ frequencies and the receiver selects one random frequency from $N$ and stays on that frequency until the end of the period ($N$ slots). The sender and receiver generate the frequency hopping sequences every period to defeat replay attacks. Since the sender's frequency sequences are totally random, it is unfeasible for any jammer to estimate the sender's sequences.

The contributions of this chapter are: first, we introduce a novel sophisticated jamming attack model which can detect the sender's frequency quorum set within the second frame. This can dramatically decrease the rendezvous probability of FQR schemes [23, 24]. Second, we propose the RFR scheme to increase the rendezvous probability against sophisticated jamming attacks. This can effectively decrease predictability for the sophisticated jammer to the same level as random frequency hopping selection for the sender's quorum set. Our simulation results demonstrate that FQR schemes are vulnerable to sophisticated jamming attacks. On the other hand, the RFR scheme can be an effective, efficient and robust rendezvous and key establishment scheme against jamming attacks.

The rest of the chapter is organized as follows. Section 6.2 describes relevant background about Quorum systems and FQR schemes. Section 6.3 presents the proposed schemes. Section 6.4 provides the simulations of sophisticated jamming attacks on both FQR and RFR schemes and their results. Section 6.5 describes the related work, and finally Section 6.6 concludes the chapter.

## 6.2 Preliminary

A quorum system is a group of sets that ensure the non-empty intersection property between any two sets. Thus, any pair of quorum sets has at least one common entry in the quorum system. In [17], the authors exploits a cyclic quorum system [44] to generate frequency quorum sequences for FQR systems. In this section, we briefly introduce the properties of the cyclic quorum system and the FQR system. We borrow all the terminologies defined in [17, 19, 34, 44]



| | One time period | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Frame 1 | | | Frame 2 | | | Frame 3 | | |
| | T0 | T1 | T2 | T3 | T4 | T5 | T6 | T7 | T8 |
| S=C0 | 1 | 2 | 4 | 1 | 2 | 4 | 1 | 2 | 4 |
| R=C4 | 5 | 5 | 5 | 6 | 6 | 6 | 1 | 1 | 1 |

Figure 6.1: FQR with $(7, 3)$ difference set example

**Definition 1** Given a finite universal set $U = Z_N = \{0, 1, \ldots, N-1\}$ of $N$ elements, a subset $D = \{a_1, \ldots, a_k\} \subset Z_N$, $a_i \in \{0, \ldots, N-1\}$ and $k \leq N$, is called *a cyclic* $(N, k)$ *difference set* if for every $d \not\equiv 0$ (mod $N$) there exist at least one ordered pair of elements $(a_i, a_j)$ such that $a_i$ - $a_j \equiv d$ (mod $N$).

**Definition 2** Given a $(N, k)$ difference set $D = \{a_1, \ldots, a_k\} \subset Z_N$, a *cyclic quorum system* constructed by $D$ is $Q = \{C_0, \ldots, C_{N-1}\}$, where $C_i = \{a_1 + i, a_2 + i, \ldots, a_k + i\}$ (mod $N$), $i = 0, \ldots, N-1$.

In [17], the authors apply these properties of quorum systems to develop a FQR system. They proposed a FQR system construction algorithm to construct FQR systems by assigning frequencies to time slots. For example, consider that a sender and a receiver use a ($N = 7$, $k = 3$) difference set. Then, the cyclic quorum system is $Q = \{C_0, \ldots, C_6\}$, where $C_0$=\{1, 2, 4\}, $C_1$=\{2, 3, 5\}, $C_2$=\{3, 4, 6\}, $C_3$=\{4, 5, 0\}, $C_4$=\{5, 6, 1\},$C_5$=\{6, 0, 2\} and $C_6$=\{0, 1, 3\}. Thus, the sender and receiver can select a random number and obtain a quorum $S = C_i$ and $R = C_j$ (e.g., $S = C_0 = \{1, 2, 4\}$, $R = C_4 = \{5, 6, 1\}$), respectively. For the sender, it assigns a channel to the time slot $i$ using $C_0 = \{c_0, c_1, c_2\} = \{1, 2, 4\}$: $x_i = (i, c_m)$, where $0 \leq i \leq k^2 - 1$ and $m = i$ mod $k$. Then it obtains $X = \{x_0, x_1, \ldots, x_{k^2-1}\} = \{1, 2, 4, 1, 2, 4, 1, 2, 4\}$. For the receiver, it also assigns a channel to the time slot $i$ using $C_4 = \{c_0, c_1, c_2\} = \{5, 6, 1\}$: $y_i = (i, c_n)$, where $0 \leq i \leq k^2 - 1$ and $n = i$ - ($j$ mod $k$). Then it obtains $Y = \{y_0, y_1, \ldots, y_{k^2-1}\} = \{5, 5, 5, 6, 6, 6, 1, 1, 1\}$.

Figure 6.1 shows how the FQR system assigns the frequencies to time slots. It also shows that the sender and receiver rendezvous on frequency 1 at time $T_6$ and the upper bound of the FQR system is $k^2$

time slots due to the non-empty intersection property of the quorum system.

## 6.3 Proposed Schemes

In this section, we propose sophisticated jamming attacks that are different than intelligent jamming attacks described in FQR schemes [23, 24]. The sophisticated jammer has capabilities similar to the intelligent jammer such as listening and jamming multiple frequencies but it utilizes the characteristics of quorum systems to select listening frequencies instead of selecting random frequencies used in intelligent jamming attacks. Thus, the probability of rendezvousing for FQR schemes can be dramatically decreased using sophisticated jamming attacks.

We validate the effectiveness of sophisticated jamming attacks on both FQR base [17] and FQR advanced schemes [23]. The difference between two schemes is that the advanced FQR scheme scrambles the sender's frequency quorum sequences for every frame. Thus, it is unfeasible for a jammer to estimate the order of sender's frequency quorum sequences. However, for the FQR base scheme, the sophisticated jammer can find the sender's quorum set in the first $\lfloor \frac{k+1}{2} \rfloor$ slots of the first frame on average and jam the remaining frequency quorum sequences for $k^2 - \lfloor \frac{k+1}{2} \rfloor$ time slots using only one jammer. For the FQR advanced scheme, our sophisticated jammer can find the quorum set in the second frame and jam the remaining sequences in $k - 2$ frames with $\lfloor \frac{k+1}{2} \rfloor$ jammers on average. We describe the algorithms for finding the sender's frequency quorum set for both FQR systems. Then we analyze the effectiveness of sophisticated jamming attacks for the two FQR schemes. We also revisit the role-based rendezvous scheme [16] and extend it to the RFR scheme to address these jamming attacks. Finally, we compare the effectiveness of sophisticated jamming attacks on the FQR and RFR schemes.

### 6.3.1 FQR Base Scheme (No Sequence Permutation)

In the FQR base system [17], there are no permutations of frequency quorum sequences for either the sender's or receiver's quorum sets until the end of $k^2$ slots/period. Here we consider one sender and one receiver in RF communication trying to rendezvous under sophisticated jamming attacks. The FQR base scheme uses the minimal $(N, k)$ difference sets to generate the frequency quorum sequences described in [44]. This guarantees that the sender and receiver can rendezvous within $k^2$ time slots. In [23], the authors proposed an advanced scheme for the FQR base system by permuting the sequences for each frame to reduce the jamming effectiveness. The intelligent jammer described in [23] selects random frequencies for listening and jamming. However, our sophisticated jammer takes advantage of quorum system properties to select our frequency sequences. This guarantees that the sophisticated jammer can decrease the rendezvous probability of the FQR base system much more than the intelligent jammer.

Now we describe the algorithm for finding the sender's frequency quorum set and analyze how the sophisticated jammer uses it to attack the rendezvous in the FQR base system. We first give an example of the sender and receiver rendezvousing and how the jammer determines the sender's quorum set. Then the algorithm for finding the sender's frequency quorum set in the FQR base scheme is given.

For our example, suppose that the sender and receiver use the minimal $(N, k)$ difference set when $N = 7$ and $k = 3$. Then all the cyclic quorum sets are

$$Q = \{C_0, \ldots, C_6\}$$

where $C_0$={1, 2, 4}, $C_1$={2, 3, 5}, $C_2$={3, 4, 6}, $C_3$={4, 5, 0}, $C_4$={5, 6, 1},$C_5$={6, 0, 2} and $C_6$={0, 1, 3}. The sender and receiver can independently select a random frequency quorum set from $Q$. If the sender and receiver randomly select quorum sets $S = C_2 = \{3, 4, 6\}$ and $R = C_4 = \{5, 6, 1\}$ respectively, then their frequency quorum sequences on the time slots $T_l$ are

$$(S_i, T_l) = \{(3, T_0), (4, T_1), (6, T_2), (3, T_3), (4, T_4), (6, T_5), (3, T_6), (4, T_7), (6, T_8)\}$$
$$(R_i, T_l) = \{(5, T_0), (5, T_1), (5, T_2), (6, T_3), (6, T_4), (6, T_5), (1, T_6), (1, T_7), (1, T_8)\}$$

where $0 \leq i \leq k - 1, 0 \leq l \leq k^2 - 1$. The sender and receiver rendezvous on the frequency 6 at time $T_5$.

The sophisticated jammer has the same minimal $(N, k)$ difference set and generates the same quorum system $Q' = Q$ as the sender's. Then it also randomly selects one quorum set $J = C_0 = \{1, 2, 4\}$ and listens to the sender's signals on three frequencies for the first frame:

$(S_i, T_l) = \{(3, T_0), (4, T_1), (6, T_2)\}$

$$(J_i, T_l) = \{(\{1,2,4\}, T_0), (\{1,2,4\}, T_1), (\{1,2,4\}, T_2)\}$$

where $0 \leq i, l \leq k-1$. The jammer detects the frequency $4$ at time $T_1$. Once it detects one of sender's frequencies, then it can find the frequency quorum set $C_i$ which contains the sender's frequency $4$ at time $T_1$ or $F_1 = 4$. That is, the $C_i$ is the sender's frequency set and the frequency quorum set $C_3 = \{3, 4, 6\}$ is the unique set ($F_1 = 4$) from $Q'$ in this case. Thus, the sophisticated jammer can exactly generate the same frequency sequences as the sender's so that it can jam the remaining sequences in $k-1$ frames using only one jammer. Therefore, the sender and receiver can not rendezvous.

As illustrated above, the sophisticated jammer can use the algorithm, given below, to find the sender's quorum set in the first frame.

1. Generate the quorum system $Q = \{C_0, \ldots, C_{N-1}\}$ from $(N, k)$ difference sets, where $C_i = \{F_0, \ldots, F_{k-1}\}$ and $i \in U = \{0, \ldots, N-1\}$.

2. The sender selects a random quorum set $S = C_i$ and generates the frequency quorum sequences (e.g., $S' = \{S_0, ..., S_{k-1}\}$, $k$ frames and $k^2$ time slots).

3. The jammer generates $Q' = Q$ and selects a random quorum set $J = C_j$ for its listening frequency quorum set (i.e., $k$ frequencies). $J$ may be the same as $S$.

4. The jammer listens on $k$ frequencies and detects the sender's frequency for the first frame (i.e., $k$ time slots).

5. Once it detects a sender's frequency $F_i$ at time slot $T_l$, where $i$,$l \in \{0, \ldots, k-1\}$, it finds the sender's quorum set $C_j$ which $l$th frequency is $F_i$ (i.e., $C_i(l) = F_i$).

6. The $C_i$ quorum set is the sender's quorum set.

In fact, the average time to find the sender's frequency quorum set is $\lfloor \frac{k+1}{2} \rfloor$ because the expected time to find the the sender's quorum set is $E[X] = \Sigma_{x=1}^{k} x \frac{1}{k} = \lfloor \frac{k+1}{2} \rfloor$. Thus, the jammer can detect the sender's frequency quorum set within $\lfloor \frac{k+1}{2} \rfloor$ time slots in the first frame and jam the remaining quorum sequences in $k^2 - \lfloor \frac{k+1}{2} \rfloor$ time slots. This means that the sender and the receiver must rendezvous within

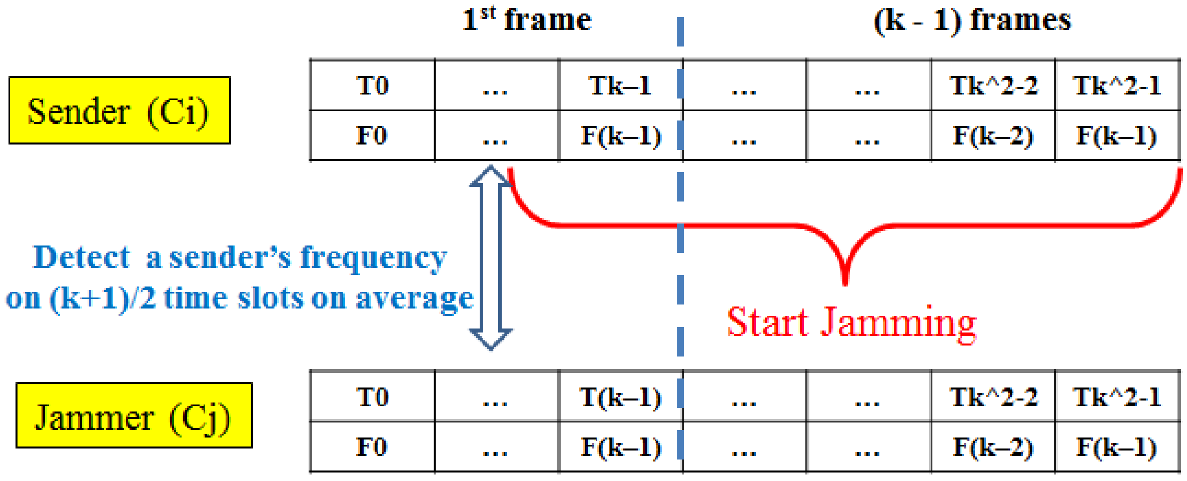


Figure 6.2: The sophisticated jamming attack on FQR base system

the first $\lfloor \frac{k+1}{2} \rfloor$ time slots of frame 1 on average. Figure 6.2 depicts how fast the sophisticated jammer can find the sender's frequency quorum set for the FQR base scheme. Therefore, the sophisticated jamming attacks can dramatically decrease the rendezvous probability of the FQR base system.

### 6.3.2 FQR Advanced Scheme (Sequence Permutation)

The authors proposed the advanced scheme for the FQR base system by scrambling the sequences for each frame to reduce the jamming probability [23, 24]. Due to the random permutation of quorum sequences for each frame, it is unfeasible for the jammer to estimate the entire sequences. However, it is still vulnerable against sophisticated jamming attacks where the jammer has the capability of listening on $k$ frequencies and jamming an average of $\lfloor \frac{k+1}{2} \rfloor$ frequencies.

In this subsection, we briefly describe the algorithm for finding the sender's frequency quorum set in the FQR advanced scheme.

1. Step (1) - (4) are the same as the FQR base system except that the sender permutes the frequency quorum sequences for each frame for step (2).

2. Once the jammer detects the sender's frequency(or frequencies), it excludes the quorum sets from $Q'$ which do not contain this frequency. If more than one jammer frequency is heard, then only

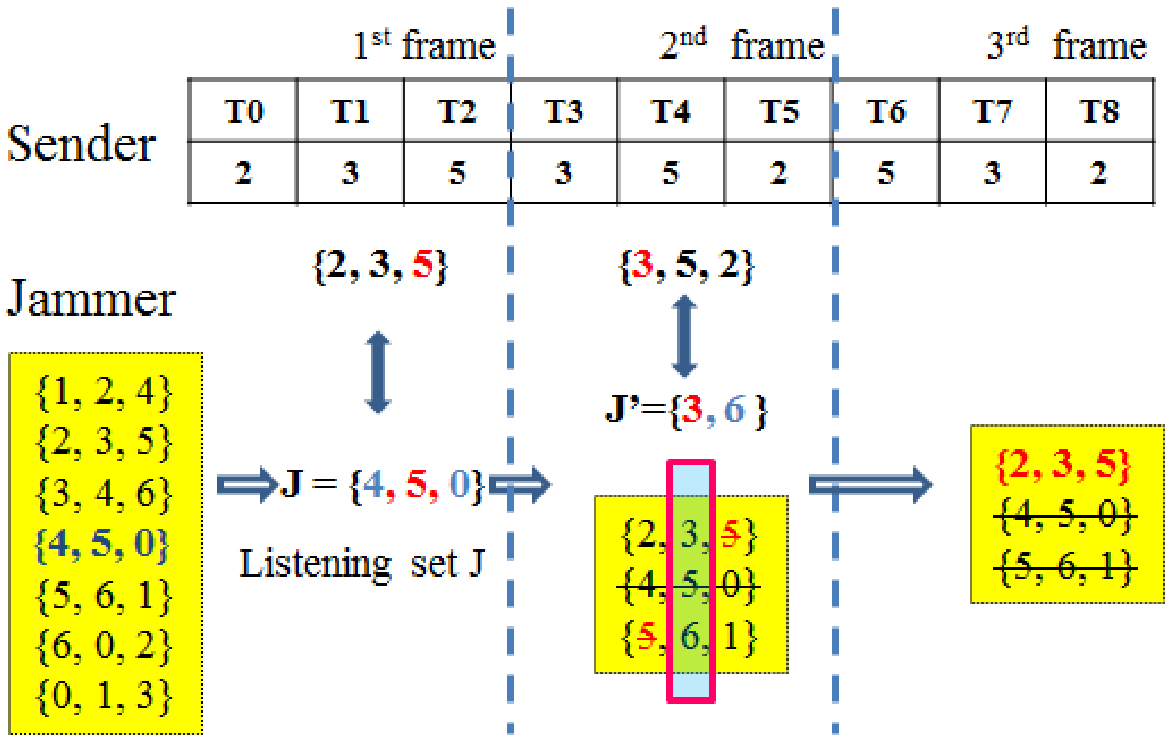


Figure 6.3: Finding the sender's quorum set within the second frame

quorum sets with both or all these frequencies are kept. The number remaining in quorum set $Q''$ is at most $k-1$ and may be unique.

3. Find the unique frequency from each quorum set in $Q''$ and generate new listening set $J'$. It can be easily shown such unique frequencies always exist; see Figure 6.3.

4. Listen on k frequencies in time slots from $k$ to $2k-1$.

5. Once the jammer detects a sender's frequency $F_i$, then it finds the quorum set $C_j$ which contains $F_i$.

6. The $C_j$ quorum set is the sender's quorum set.

As illustrated in the example below, using this algorithm, the sophisticated jammer can find the sender's quorum set in the second frame. Figure 6.3 illustrates finding the sender's quorum set. In this example, we use the same minimal $(N, k)$ difference sets as the FQR base scheme's. The minimal difference set is $(7, 3)$ and the cyclic quorum sets are $Q = \{C_0, \ldots, C_6\}$. The sender and receiver can independently select a random frequency quorum set from $Q$. Then the sender generates the frequency quorum sequences by scrambling sequences for each frame. If the sender and receiver randomly select quorum sets $S = C_1 = \{2, 3, 5\}$ and $R = C_5 = \{6, 0, 2\}$, respectively, then their frequency quorum sequences can be

$$(S_i, T_l) = \{(2, T_0), (3, T_1), (5, T_2), (3, T_3), (5, T_4), (2, T_5), (5, T_6), (3, T_7), (2, T_8)\}$$
$$(R_i, T_l) = \{(6, T_0), (6, T_1), (6, T_2), (0, T_3), (0, T_4), (0, T_5), (2, T_6), (2, T_7), (2, T_8)\}$$

where $0 \leq i \leq k - 1, 0 \leq l \leq k^2 - 1$. The sender and receiver can rendezvous on the frequency 2 at time $T_7$.

The sophisticated jammer also randomly selects one quorum set $J = C_3 = \{4, 5, 0\}$ and listens for the sender's frequencies for the first frame:

$$(S_i, T_l) = \{(2, T_0), (3, T_1), (5, T_2)\}$$

$$(J_i, T_l) = \{(\{4, 5, 0\}, T_0), (\{4, 5, 0\}, T_1), (\{4, 5, 0\}, T_2)\}$$

where $0 \leq i, l \leq k - 1$. The jammer detects the frequency 5 at time $T_2$ but it cannot find the sender's quorum set immediately because the number of quorum sets containing the frequency 5 is more than one set (i.e., $k - 1 = 2$). Thus, the jammer needs to regenerate the new listening set $J' = \{3, 6\}$ from the remaining sets in $Q'' = \{C_1 = \{2, 3, 5\}, C_4 = \{5, 6, 1\}\}$ and then wait another frame to determine the sender's quorum set. The jammer detects the frequency 3 at time $T_4$ and determines that the sender's quorum set to be $C_1 = \{2, 3, 5\}$. That is, the sophisticated jammer can find the sender's frequency quorum set for the second frame. Then it can jam the remaining quorum sequences for $k - 2$ frames using an average of $\lfloor \frac{k+1}{2} \rfloor$ jamming frequencies. The jammer hears a frequency in each slot so it need not jam that frequency in any subsequent slots in that frame and hence the $\lfloor \frac{k+1}{2} \rfloor$. Thus, the sophisticated jamming attack can dramatically decrease the rendezvous probability of the FQR advanced system because it knows the sender's quorum set exactly but it cannot estimate the order of the sender's quorum sequences.

### 6.3.3 Role-based Frequency Rendezvous

The advanced FQR scheme can increase the rendezvous probability under various jamming attacks but the sophisticated jamming attack can dramatically decrease the rendezvous probability. To mitigate this jamming attack, we consider Anderson and Weber's algorithm [16] called role-based rendezvous scheme [6] and extends it to Role-based Frequency Rendezvous (RFR) scheme as a countermeasure against the sophisticated jamming attack. In the RFR system, a sender generates a randomized permutation of $N$ frequencies for every period. The receiver randomly selects one frequency out of $N$ frequencies for every period and stays on the frequency until the end of the period. Hence, it is unfeasible for the sophisticated jammer to estimate when the sender and receiver might rendezvous. For example, when $N = 8$ and $0 \leq i \leq 7$, the sender's and the receiver's frequency sequences can be

$$(S, T_i) = \{(4, T_0), (3, T_1), (0, T_2), (5, T_3), (7, T_4), (2, T_5), (6, T_6), (1, T_7)\}$$
$$(R, T_i) = \{(5, T_0), (5, T_1), (5, T_2), (5, T_3), (5, T_4), (5, T_5), (5, T_6), (5, T_7)\}$$

. The sender and receiver rendezvous on frequency $5$ at $T_3$. In the RFR scheme, the upper bound for a rendezvous is $N$ and the expected Time To Rendezvous (TTR) is $\lfloor \frac{(N+1)}{2} \rfloor$ [16].

## 6.4 Evaluation

We implement the sophisticated jamming attack on the FQR advanced scheme [23] to evaluate its effectiveness. We exclude the implementation of the FQR base scheme because it is trivial for the sophisticated jammer to find the sender's frequency quorum set. For the FQR advanced system, we use the same optimal cyclic quorum sets [44] and generate the frequency quorum sets from the minimal $(N, k)$ difference sets as the FQR system [17]. We also implement the RFR scheme to demonstrate how it is more robust against these jamming attacks compared to the FQR advanced scheme.

First, we compare the expected TTR for both FQR advanced and RFR schemes. Figure 6.4 gives the average TTRs for FQR and RFR schemes where the number of available frequencies $N$ varies from $4$ to $100$ and there are no jamming attacks. We ran $1000$ simulations for each frequency and calculated the average TTR for them. This figure shows that the average TTR for RFR scheme increases steadily as the

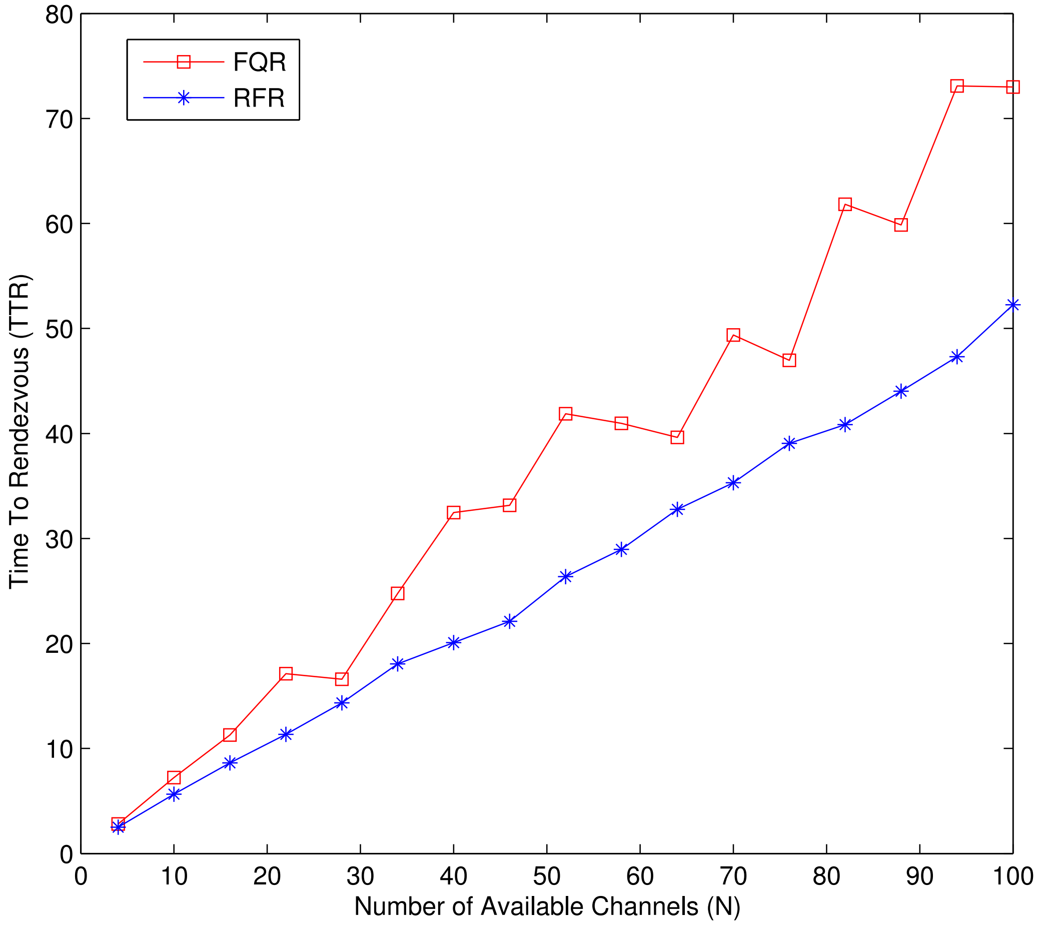


Figure 6.4: TTR for both FQR and RFR without jamming attacks

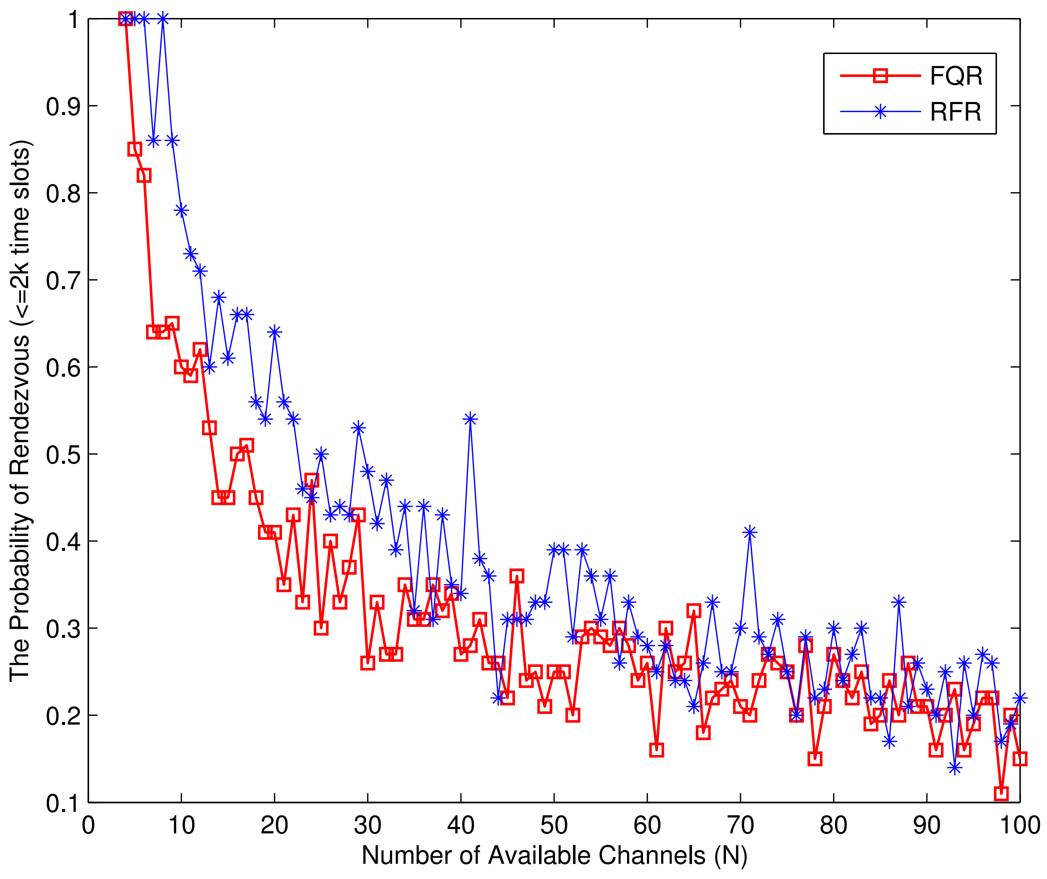


Figure 6.5: The probability of rendezvous within $\leq 2k$ time slots

number of available frequencies increases. Since the expected TTR for the RFR scheme is $\lfloor \frac{(N+1)}{2} \rfloor$, the average TTR is approximately half of the available channels time slots. The average TTR for the FQR system depends on the quorum sets. One quorum set can take less time than another with a larger number of available frequencies. Overall, the average TTR for the RFR system is lower than FQR system's TTR for all $N$.

Now we compare the probability of a rendezvous by $2k$ time slots for the FQR advanced and RFR schemes. This is important for the FQR advanced system because, using our sophisticated jammer, the sender and receiver can only rendezvous by the end of the second frame ($\leq 2k$). The sophisticated jammer can find the sender's quorum set within the second frame and jam the remaining frequency quorum sequences for $(k-2)$ frames. Figure 6.5 provides the probability distribution of a rendezvous for FQR and RFR schemes. This figure shows that the rendezvous success rates for both schemes are close each other and it dramatically decreases as the number of available frequencies increases. Overall, the rendezvous probability of the RFR scheme is slightly higher than that of FQR advanced. For the FQR advanced scheme, the success rate is less than $40\%$ when the number of frequencies is more than 20 and it is around $21\%$ when $N = 100$. Note this is only the probability of a rendezvous in $2k$ time slots and when $N$ is large, $2k$ is much smaller than both $N$ and $k^2$.

Now we compare the probability of a rendezvous for both schemes under the sophisticated jamming attack. In this experiment, the sophisticated jammer uses $\lfloor \frac{k+1}{2} \rfloor$ jamming frequencies on average. Since the sender's frequencies are not repeated in any frame, it is redundant to keep jamming any frequencies detected by the jammer in that frame. Thus the number of jammed frequencies decreases with the time slot and therefore the average number of jamming frequencies is the $\lfloor \frac{k+1}{2} \rfloor$. The jamming attack on FQR begins only after the first $2k$ slots while the random jamming attack of $\lfloor \frac{k+1}{2} \rfloor$ frequencies starts immediately for RFR. Figure 6.6 shows the probability of rendezvous for both systems until the end of $k^2$ time slots. The probability of rendezvous for the FQR advanced scheme dramatically decreased as the number of frequencies increases. When the minimal difference set is $(100, 12)$, the rendezvous probability is less than $20\%$. However, the rendezvous probability for the RFR scheme is almost steady and more than $80\%$ for all the available frequencies $N$. In the RFR system, we extend the period for the RFR scheme from $N$ to $k^2$ time slots to compare with the FQR advanced scheme. Thus, the RFR sender can permutate the frequency sequences again at $N + 1$ and thus increase the rendezvous probability slightly more but this is not critical. When $N$ is 100, the probability for RFR is close to $95\%$ (i.e., 949 rendezvous out of 1000 times). The FQR average TTR is much larger than RFR's as seen from Figure 6.5 and Figure 6.6.

Lastly, we implement the cumulative rendezvous probability for both schemes under $\lfloor \frac{k+1}{2} \rfloor$ jamming frequencies on average. Figure 6.7 describes the cumulative probability of rendezvous for both systems until the end of $k^2$ time slots. We select 5 different $N$ as the number of available frequencies. For the

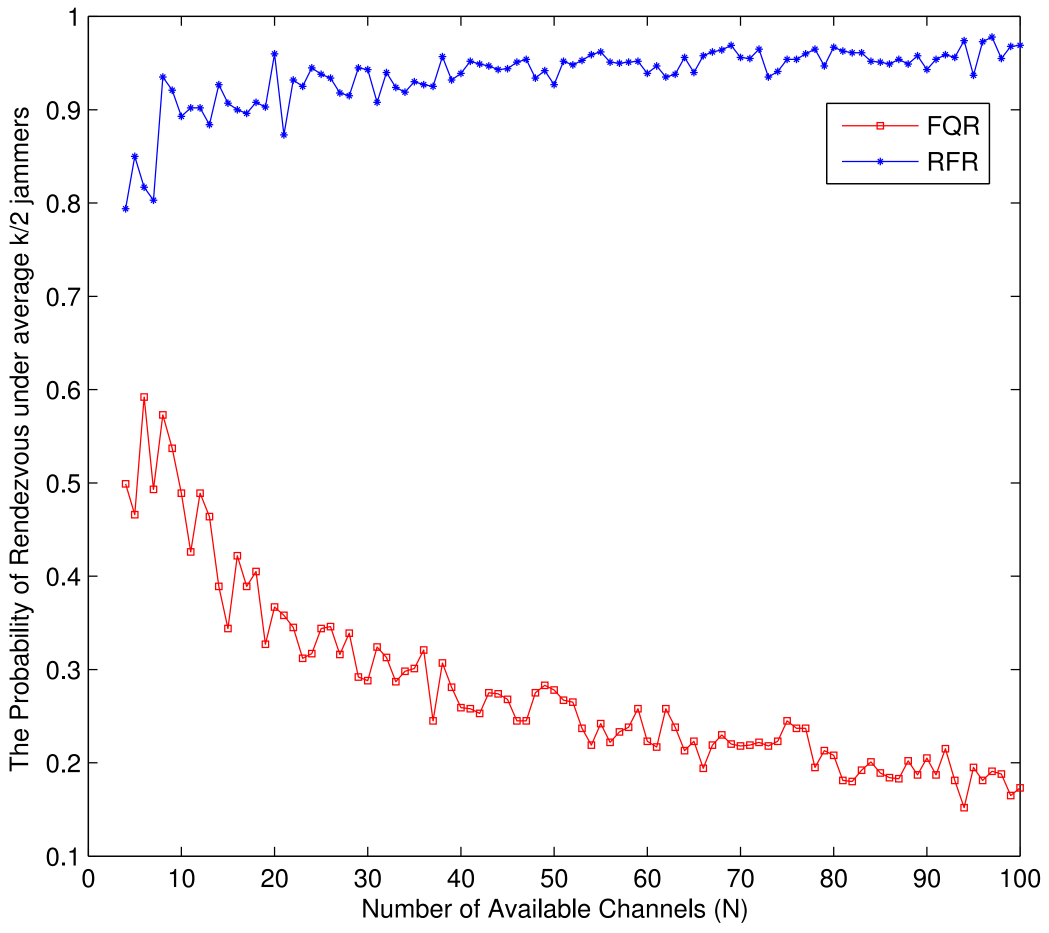


Figure 6.6: The probability of rendezvous under an average $\lfloor \frac{k+1}{2} \rfloor$ jammers in $k^2$ time slots

FQR advanced scheme, the cumulative probability of rendezvous is almost $50\%$ when $N = 10$ but it decreases to $15\%$ for $N = 100$. On the other hand, the cumulative rendezvous probability of the RFR scheme linearly increases up to the its original period ($N$ time slots) and slowly increases from $N$ to $k^2$ time slots. This means that the effectiveness of sophisticated jamming attack is not critical in RFR scheme but it is critical for the FQR advanced scheme.

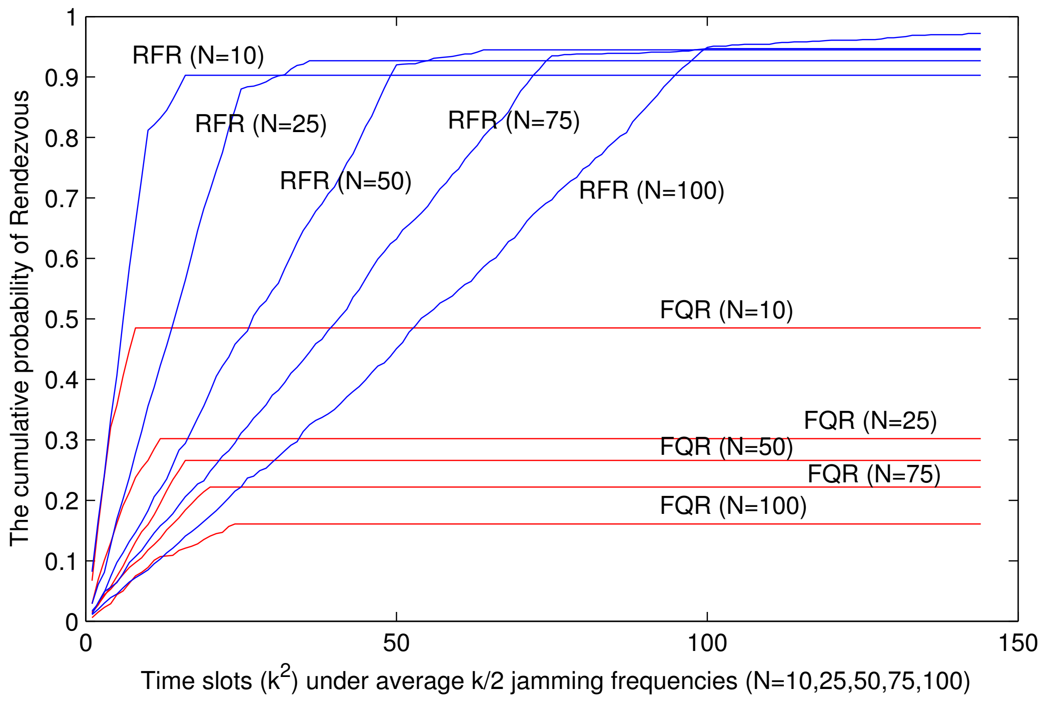


Figure 6.7: The cumulative probability of rendezvous under an average $\lfloor \frac{k+1}{2} \rfloor$ jammers in $k^2$ time slots

## 6.5 Related Work

The traditional spread spectrum techniques are vulnerable to the reactive jamming attacks [21,40] where the jammer can be a compromised receiver and it can learn the pre-shared key between the sender and receivers.

To address the limitation, several approaches have been studied to eliminate the dependency on pre-shared keys and establishing a random shared secret before the communication starts [17, 27, 35]. The Uncoordinated FHSS (UHSS) approach [35] proposes a scheme by using the *Diffie-Hellman* (DH) protocol to establish a shared key between the sender and receivers. The sender sends the same message with a randomly selected frequency from the known pool of frequency sequences. The receiver can decode if any correct message is received. The USD-FH approach [27] also provides an efficient *Diffie-Hellman* (DH) key establishment method under jamming attacks by transmitting a one-time pseudo-random hopping pattern message and disclosing the relevant seed in an uncoordinated manner before the actual message. In [17], the authors proposed the FQR system which elaborates the Quorum-based rendezvous scheme. Since the time latency for rendezvous and key establishment is not trivial for previous schemes [27, 35], the FQR scheme focuses on decreasing time latency by using cyclic quorum sets. Using this advantage, FQR schemes can provide relatively fast and robust key establishment methods under various attacks.

However, the FQR system is still vulnerable to the sophisticated jamming attacks in which the jammer has the capability of listening and jamming multiple frequencies. To address this problem, we revisit the role-based Rendezvous scheme [16] and present the RFR scheme which extends the scheme to key establishment. Since the role-based rendezvous scheme uses the random rendezvous scheme, it is unfeasible for the sophisticated jammer to estimate the sender's quorum set by using the sophisticated jammer channel selection algorithm. Therefore, the RFR scheme is more robust against sophisticated jamming attacks.

## 6.6 Conclusion

In this chapter, we presented an advanced jamming attack called *sophisticated jamming attack* and demonstrate its effectiveness on both FQR base and FQR advanced systems [17, 23]. Since the sophisticated jammer takes the advantage of the quorum system property to select the listening and jamming frequencies, it can find the sender's quorum set within the second frame in the FQR system. Then, the jammer can completely jam the sender after $2k$ time slots using an average of $\lfloor \frac{k+1}{2} \rfloor$ and a maximum of $k$ frequencies. To remedy this jamming problem, we revisit the role-based rendezvous scheme and extend it to the RFR scheme. Our simulation results demonstrate that the rendezvous probability of FQR system under the sophisticated jamming attack is dramatically decreased as the number of available channel $N$ increases (e.g., $\leq 35\%$ for $N \geq 30$ in $k^2$ time slots). On the other hand, the rendezvous probability of our RFR scheme is almost steady for all available frequencies $N$ (e.g., $\geq 90\%$ for $N \geq 20$). Therefore, RFR scheme can be an effective, efficient and robust rendezvous and key establishment scheme against sophisticated jamming attacks.

# Chapter 7

# Enhanced Security of Random Seed DSSS Algorithms against Seed Jamming Attacks

Researchers have recently studied random spread spectrum techniques to protect the wireless broadcast communications from reactive jamming attacks in traditional Direct Sequence Spread Spectrum (DSSS) networks. They proposed mechanisms to eliminate the pre-shared key vulnerability by generating different code sequences for each message using random seeds and disclosing the seeds at the end of the message. In this chapter, we present a new type of jamming attack called a *seed jamming attack* for the fixed message size and these seed disclosure schemes are vulnerable to it. The seed jamming attack focuses on jamming any part of the random seeds of the messages. Their sizes are relatively small and their position in messages is known to the public a priori. Thus, the receivers cannot despread any part of the messages due to the failure of regenerating proper code sequences with the corrupted seed. Jamming the seed precludes the use of any possible FEC since the receiver cannot decode any bits in the message. To overcome the seed attack, we propose an advanced random seed DSSS (ARS-DSSS) scheme which strengthens the previous algorithm called DSD-DSSS [28] by using an additional location seed. The new seed avoids the seed jamming attack by using variable message sizes instead of using known fixed message sizes while incurring almost no additional performance overhead. Our security analysis and implementation results demonstrate how to defeat the seed jamming attacks and how to reduce the computation overhead of DSD-DSSS from the cardinality of seed code set $C_e$ to $1$.

## 7.1 Introduction

Recently, spread spectrum techniques have been emphasized in wireless broadcast communication due to its anti-jamming characteristics. Frequency Hopping Spread Spectrum (FHSS) and Direct Sequence Spread Spectrum (DSSS) are the most common spread spectrum techniques [18]. However, traditional spread spectrum systems have the fundamental limitation of prior sharing of a secret key between the sender and receivers. This sharing creates a vulnerability to insider jamming attacks (e.g., a reactive jamming attack [40]) when the jammer is a compromised receiver since she can use the pre-shared key to jam the wireless communications.

Many researchers have proposed different approaches to eliminate the dependency on pre-shared keys. The Uncoordinated DSSS(UDSSS) approach [7, 35] randomly selects frequency hopping sequences or the spreading code subsequences from a public pool of code sequences for each message instead of using the pre-shared keys for all messages. However, if a reactive jammer has sufficient computation power, she can find the correct code sequence before the communication ends. The RD-DSSS [45] employs the encoding method for each bit (i.e., 0 or 1) of data by using the correlation of unpredictable spreading codes. The RD-DSSS uses multiple spreading code sequences to spread each message to avoid the dependency seen in UDSSS. It is almost infeasible for the reactive jammer to compute the next code sequence in real time. The recently proposed Delayed Seed-Disclosure DSSS(DSD-DSSS) [28] uses random seeds to generate random code sequences for each message. Using the random seed, the sender and receivers can generate the same code sequences. To address the reactive jamming attack, the DSD-DSSS discloses the random seed at the end of each message. Thus, it is infeasible for the jammer to generate the code sequences before all receivers receive the entire message.

However, they are still vulnerable to a new type of jamming attacks called a *seed jamming attack* in which an attacker particularly focuses on jamming the random seed(s) in messages. For example, since the insider jammer learns the beginning of the messages with the sender using a synchronization method, she can also compute the starting chip of the seed in messages. The jammer can simply jam the entire spectrum for a very short duration when the sender sends even one bit of the random seed. This type of jamming attacks is clearly possible under the premise that an attacker is a sophisticated jammer who is one of compromised receivers and has sufficient transmission power. The jammer can broadcast enough noise (e.g., beyond the jamming margin [21]) for a short period time to jam the entire spectrum for only the last bit of the seed in the transmission. Therefore, the receivers cannot despread the entire messages due to the failure of regenerating correct code sequences from the corrupted seeds. Jamming the seed instead of the bits in the message precludes using FEC while jamming the message bits can be reversible using FEC [31].

In this chapter, we propose an advanced random seed DSSS (ARS-DSSS) technique as a countermeasure to the seed jamming attack. In particular, we chose the DSD-DSSS SUBSET scheme [28] because it is arguably the best algorithm except for its efficiency for variable size messages. However, we show it is clearly vulnerable to the seed jamming attacks for the fixed size messages. Moreover, if it uses the variable size messages, the computational overhead to find the random seed increases by the cardinality of the seed spreading code set $C_e$. Our ARS-DSSS aims to reduce the performance overhead for variable size messages and to provide security against the seed jamming attacks. Our ARS-DSSS introduces the new random location seed to help receivers efficiently find the random seed in variable sized messages.

The contributions of this chapter are: first, we introduce a novel seed jamming attack model which can make previous approaches fail to provide guaranteed security against jamming attacks. The seed jamming attack is fundamentally more effective than jamming any other bits in the message. This is because FEC can be used to recover jammed message bits while jammed seed bits preclude reading any of the message and FEC is not applicable. Second, we propose the advanced random seed DSSS scheme to protect the wireless broadcast communication against the seed jamming attacks. Moreover, our implementation results demonstrate that our ARS-DSSS reduces the performance overhead of the DSD-DSSS SUBSET scheme from cardinality of $C_e$ to 1 for the variable message sizes. In DSD-DSSS SUBSET, the size of the code set $C_e$ is generally at least 56 bits.

The rest of the chapter is organized as follows. Section 7.2 discusses the assumptions and context.

Section 7.3 presents the proposed schemes. Section 7.4 provides the implementation of our scheme and its results. Section 7.5 describes related work, and Section 7.6 concludes this chapter.

## 7.2 Assumptions and Context

We consider mission-critical applications that use wireless broadcast communication mediums. In our system model, the sender broadcasts messages to unknown receivers with the existence of jamming attacks. We assume that there is a sophisticated jammer who has both strong computation power and transmission power. Actually, only minimal computational power is required for the seed jamming attacker. We also assume that the jammer can synchronize with the sender and she can immediately compute the beginning position of the random seed in messages during the synchronization process. We assume that the sophisticated jammer has sufficient power to jam the entire spread spectrum for very short duration (e.g., 1 bit) in DSSS systems. However, we assume that the jammer cannot continuously or extensvely jam the entire spectrum.

As discussed, we introduce the new type of jamming attacks called *seed jamming attacks*. Most previous schemes [7], [45], [28] provide a high level of security against the reactive jamming attack but they are still vulnerable to the seed jamming attacks. In this section, we will define the seed jamming attack and describe its effectiveness against the DSD-DSSS SUBSET scheme [28] in detail.

### 7.2.1 Seed Jamming Attacks

In wireless broadcast communication, a reactive jammer can learn the secret key between the sender and receivers if she is a compromised receiver. Most previous research focuses on eliminating the dependency on the pre-shared key by using randomized code sequences or a random seed for generating code sequences at the receivers. For example, in DSD-DSSS SUBSET [28], the random seed is attached at the end of the message and all receivers can receive the entire message before the seed arrives. Using the seed, all receivers can generate the same code sequences the sender used. However, it is infeasible for a reactive jammer to compute the codes sequences in real-time without knowing the correct random

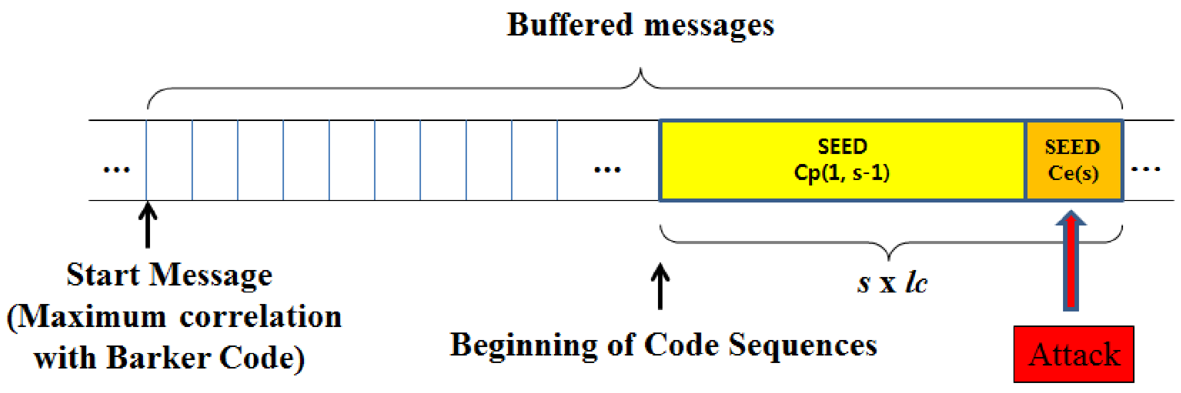


Figure 7.1: A seed jamming attack against DSD-DSSS SUBSET

seed.

Most of these approaches have a premise that it is difficult or impossible for a jammer to continuously jam the entire spread spectrum. However, it is possible, in practice, to jam the entire spread spectrum for a short duration using a commodity system. The continuous jammer would be immediately detected but a single bit jammer much less so. Under this premise, we present the seed jamming attacks in which a sophisticated jammer mainly focuses on jamming part of the random seed in the message. Figure 7.1 describes the seed jamming attack against the DSD-DSSS SUBSET approach. The seed jamming attack is applicable to the scheme only if the message size is fixed. When the message size is fixed, the jammer can compute the starting chip of the random seed in messages during the synchronization (e.g., Barker Code [3]). Then, when the sender sends the random seed, the jammer transmits noises that are beyond the jamming margin [21]. Therefore, the receivers can not despread the received message because they cannot regenerate the same code sequences from the corrupted seed. The following section describes finding the position of the random seed in messages.

### 7.2.2 Finding Random Seeds in Messages

In general, the seed jamming attack is different than other types of jamming attacks such as continuous, periodical, or random jamming attacks because it focuses only on the random seed in the message. Also, as mentioned previously, it is dramatically more effective. It clearly is necessary for the seed jammer to find the position of the random seed during the synchronization process or at least before the seed transmission. This is possible if the message sizes are fixed and known to the public a priori. For example, one of the typical synchronization techniques in DSSS system is the Barker Code [3] that both the sender and receivers know in advance. As we indicate in Figure 7.1, the receiver can find the maximum correlation by comparing the Barker Code and its signals in the received buffer using a sliding window approach. Thus, the sophisticated jammer knows the message begins immediately after the maximum correlation of the signal and can also compute the beginning chip of the random seed in the message. If the message sizes are variable, the jammer is required to continuously jam or estimate the position of the random seed to jam the wireless communications.

**802.11 Family:** We first describe the wireless communication in the 802.11 networks. From 802.11 DSSS PHY specification, the message sizes are fixed and the DSSS PHY contains the LENGTH field in the TXVECTOR [2]. It is in octets and the receiver can convert it to microseconds to calculate the message length. In our system model, once a sophisticated jammer can synchronize the received messages with the sender, she can compute the starting chip of the random seed to jam. Therefore, the wireless communications in the 802.11 protocol family are vulnerable in our system model.

**None-802.11 Family:** Many jamming countermeasures use their own communication protocol in DSSS systems instead of using the 802.11 family. In [28], the authors propose a generic DSSS protocol for the wireless broadcast communication. The protocol does not include any header information which can contain length fields. Thus, we can consider two scenarios for the message sizes. First, the messages sizes are fixed and known to the public a priori. In this scenario, the sophisticated jammer can know the beginning of the seed in the message immediately after synchronization.

Figure 7.2 describes finding the seed in DSD-DSSS SUBSET [28] when the message sizes are fixed. After the synchronization, the jammer can compute the position of the last bit of the random seed in the message. The position is

$$((m \times l_c) + ((s-1) \times l_c))$$

where $m$ is the message size in bits, $s$ is the seed size, and the $l_c$ is the code length.

The other scenario is that the message sizes can be variable so that the receivers are required to

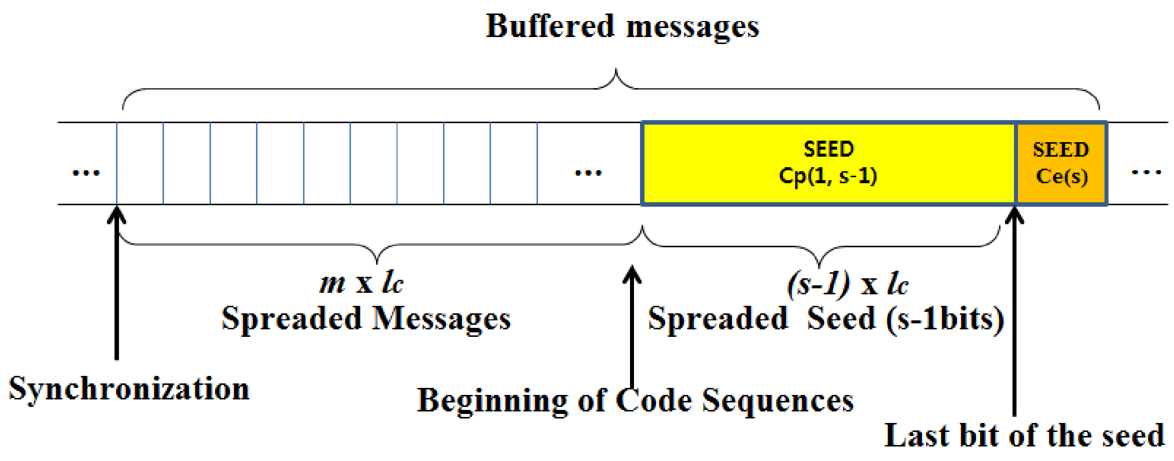


Figure 7.2: Finding the random seed in DSD-DSSS SUBSET

find the maximum correlation of the random seed in messages using sliding window techniques. The receiver buffers all the messages in the circulation queues and moves the sliding windows with the size of chip duration over time. When the receiver finds the highest correlation with the code sequence of the seed, it can then find the entire seed and then the beginning of the messages from that position using backward computation.

As discussed above, if the message sizes are fixed, the receiver can easily compute the beginning and the end of the seed but then it is vulnerable to seed jamming attacks. On the other hand, if the message sizes are variable and change each time, it is difficult or impossible for the sophisticated jammer to find the beginning of the seed in the message until the seed is complete. Moreover, for the receiver, the computation overhead is prohibitive because it must compare the entire message to find the maximum correlation with every element in the seed code set $C_e$. Our proposed solution reduces the computation overhead by a factor of $|C_e|$.

## 7.3 Proposed Schemes

In this section, we propose an Advanced Random Seed DSSS (ARS-DSSS) scheme to improve the DSD-DSSS SUBSET scheme [28]. The DSD-DSSS scheme provides a high level of security against reactive jamming attacks. However, this scheme is vulnerable to seed jamming attacks when the message sizes are fixed. In addition, the performance overhead of finding the seed increases dramatically when the message sizes are variable. To address these shortcomings, we strengthen the DSD-DSSS SUBSET scheme and propose the ARS-DSSS scheme by introducing the new random location seed. We will describe our ARS-DSSS scheme and its security analysis in the following sections.

### 7.3.1 An Advanced Random Seed DSSS (ARS-DSSS)

The DSD-DSSS SUBSET scheme is vulnerable to the seed jamming attacks for the fixed message size. Since the seed size is relatively small compared to the message size and the position of the seed in messages is known to the public, the jammer can corrupt the seed so that the receivers cannot despread entire messages. Moreover, the computation overhead for DSD-DSSS SUBSET of locating the seed in variable sized messages is prohibitively large. In this section, we describe the computation overhead. In DSD-DSSS [28], the computation time of locating the seed in messages is

$$T_s = ((m \times l_c) + (s \times l_c)) \times n$$

where $T_s$ is the computation overhead, $m$ is the length of message, $l_c$ is the length of the chips, $s$ is the length of the seed, and the $n$ is the number of the code sequences in the seed code set $C_e$. The DSD-DSSS SUBSET scheme uses the $C_e$ code set to spread the last bit of the random seed. The $n$ code sequences in $C_e$ are known to the public. But the seed is randomly selected for each message and is not known to the public. The total time for despreading messages is

$$T = T_s + T_g + T_d$$

where $T$ is the total time for despreading received messages, $T_s$ is the time for finding the seed, $T_g$ is the time for generating code sequences from the seed, and $T_d$ is the time for despreading messages using the code sequence.

Thus, the total time $T$ in DSD-DSSS SUBSET scheme mainly depends on the time $T_s$ for finding the seed. If the message sizes are fixed, the receivers can skip the time for finding the seed. On the other hand, if the message sizes are variable, the receiver is required to compare the entire message, one chip

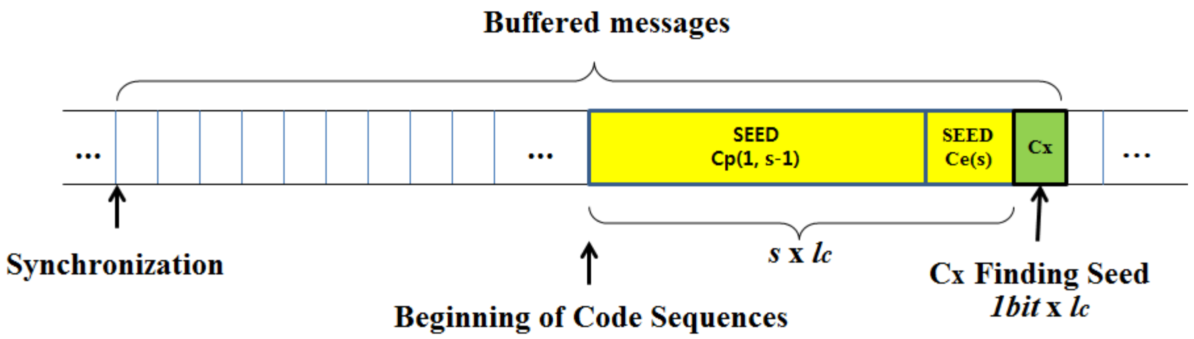


Figure 7.3: Finding the seed in ARS-DSSS

at a time, to the code set $C_e$ to find the maximum correlation with the last seed bit. Please see Figure 2 in [28] for the diagram of the required sliding window scheme. To address this problem, we extend the DSD-DSSS SUBSET scheme and propose ARS-DSSS by introducing the new random location seed. The seed is one bit (e.g., 0 or 1) and it is spreaded

and despreaded by one random code sequence in $C_x$ which replaces the finding the last seed bit algorithm of the DSD-DSSS SUBSET scheme. Figure 7.3 describes the new location seed of our scheme. In ARS-DSSS, the total time for despreading the message is

$$T = T_x + T_g + T_d$$

where $T_x$ is the new time for finding the location seed in messages. The computation time $T_x$ is

$$T_x = ((m \times l_c) + (s \times l_c) + (1 \times l_c)) \times 1.$$

That is, our ARS-DSSS can reduce the time for finding the random seed from the cardinality $|C_e|$ to 1. As the security demands increase, the number of code sequences $(n)$ in $C_e$ must increase. Therefore, our ARS-DSSS is more scalable as the size of $C_e$ increases.

## 7.3.2 Security Analysis

In this chapter, we improve the DSD-DSSS SUBSET scheme [28] so our ARS-DSSS scheme provides the same level of security against the reactive jamming attacks and the same effectiveness against DoS attacks. We now describe the DoS attacks against the new location seed $C_x$ and present a solution.

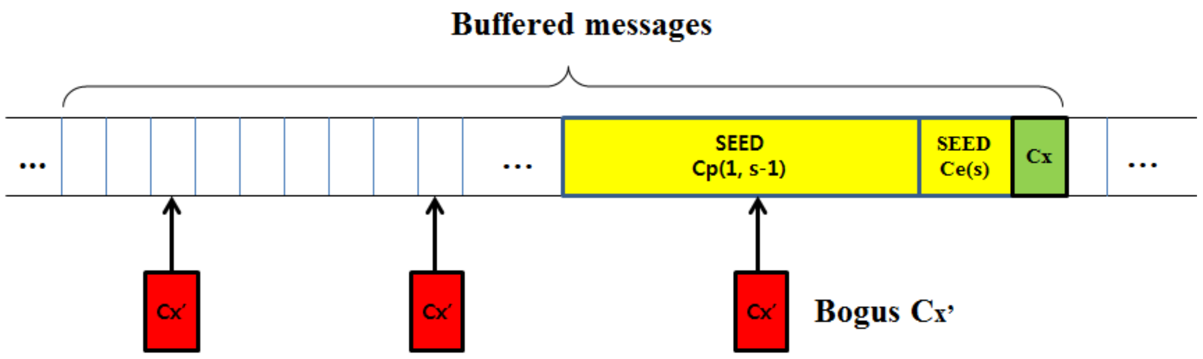


Figure 7.4: The seed jamming attacks in ARS-DSSS

Figure 7.4 indicates how a sophisticated jammer can transmit bogus location seeds $C_x$. As long as these bogus location seeds satisfy for the following condition, the receivers cannot filter them immediately. The condition is that the phase offset between the carriers of the seed jammer and the legitimate sender must be within a fraction $\eta$ of the chip period [33]. Otherwise, the receivers can easily differentiate between the jamming signals and the legitimate sender's signals using DSSS techniques.

When the bogus location seed is within this fraction, the receiver is required to compare the maximum correlation with the $C_x$ code sequence first. Then, the receiver computes the correlation using $C_e$ code set of the previous bit (thinking it is the last bit of the random seed). This is exactly the same as for the DSD-DSSS SUBSET scheme. Our ARS-DSSS scheme provides the same level of the security against the DoS attacks as if the jammer had transmitted bits using $C_e$ instead of $C_x$.

However, there may be a trivial increase in the computation overhead. Let $C_p$ be the code set for the messages and the subset of seeds in the DSD-DSSS SUBSET scheme. It can be shown that the "increased" computation is just the decoding of one bit by $C_e$ for our ARS-DSSS scheme versus decoding of one bit by a subset of $C_p$ for the DSD-DSSS SUBSET scheme. For most cases, $C_e$ is minimally larger than the appropriate subset of $C_p$ but may not be as well. This does occur for each bogus location seed transmitted. Therefore, our scheme is as DoS resistant as the DSD-DSSS SUBSET scheme with little or no increase in computation overhead.

In addition, the DoS attack against ARS-DSSS can be largely mitigated by increasing the size of $C_x$. The fundamental idea here is that each receiver chooses a random member of $C_x$ on which it will synchronize while the sender, after sending the last seed bit of the message, transmits a bit using every member of $C_x$ chosen in random order. The set $C_x$ may be taken as a subset of $C_e$ and $|C_x|$ is given by $m$. Thus the probability that a transmitted DoS bogus position seed will be recognized and have to be handled in the above fashion is now $\frac{1}{m}$. This decreases the cost of the bogus seed attacks by a factor of $m$ over the singleton $C_x$. It can be shown that the bogus code attack from $C_e$ for DSD-DSSS SUBSET scheme for the fixed message size has a cost that is the cardinality of a subset of $C_p$ times $|C_e|$ more costly than ARS-DSSS scheme.

If the code set $C_x$ is a singleton and if the sophisticated jammer is able to jam continuously with the $C_x$ code, then it may be possible that our ARS-DSSS scheme will miss the bit sent using $C_x$ which immediately follows the final seed bit. Fortunately, increasing the size of $C_x$ largely mitigates this attack too. Now the sender transmits $m$ bits using every member of $C_x$ chosen in random order and it repeats this four times. Each receiver chooses a random member of $C_x$ and attempts to synchronize using it for $2m$ bits. Then it chooses another random member of $C_x$ and synchronizes with it for $2m$ bits. It continues to repeat this until it synchronizes with a code from $C_x$. It can be shown that in the absence of a jammer this generates at least two synchronization with members of $C_x$. Since the jammer can jam any code in $C_x$, there cannot be a guarantee of synchronization.

However, the probability for the continuous jammer to successfully jam the codes from $C_x$ or equivalently the location seeds is approximately $\frac{1}{m^2}$ or $1\%$ when $|C_x| = 10$. We could further improve this efficiently if we configure this synchronization with the codes from $C_x$ as a rendezvous problem and apply techniques from that domain. The cost for increasing the size of $C_x$ is that we have to transmits $4m$ additional bits and then use the codes in $C_x$ to back into the last seed bit. In the same fashion as given above, a continuous jammer using codes from $C_e$ can jam the last seed bit for the DSD-DSSS SUBSET scheme. This bit can be jammed with a probability of $\frac{1}{|C_e|}$ which is likely to be a little larger than $\frac{1}{m^2}$ (the jamming probability for the ARS-DSSS scheme). Our approach above may not be optimal but it is still much more efficient and effective than the DSD-DSSS SUBSET scheme.

The actual implementation can keep the size of $C_x$ small (10 or so) but still provide effective and efficient DoS and jamming protection with the cost being the additional bits transmitted. We have not tried to optimize on the size of $C_x$. The results in this chapter stand by themselves and enlarging $C_x$ is just one additional feature. This feature make our ARS-DSSS scheme even more efficient and resilient to DoS and jamming attacks than the DSD-DSSS SUBSET scheme.

## 7.4 Experimental Evaluation

We implemented our ARS-DSSS scheme on GNU Radios [30,42] in the same environment as the DSD-DSSS SUBSET scheme [28]. We used two USRPs with XCVR2450 daughter boards for one sender and one receiver, respectively. Each USRP was connected to a desktop computer (Intel (R) Core (TM) Quad CPU Q6600 @ 2.40 GHz), both through 480 Mbps USB 2.0 links. The operating system is Ubuntu 9.04 and Gnuradio 3.2 softwares for both systems. The code base of the DSD-DSSS SUBSET scheme was used to implement our ARS-DSSS scheme.

In our implementation, we consider two scenarios for the receivers based on fixed and variable message sizes. Since we extend the DSD-DSSS SUBSET, we evaluate DSD-DSSS SUBSET for the computation overhead of despreading messages and compare it with our ARS-DSSS scheme for both scenarios. For both scenarios, we assume there are already synchronization methods between the sender and receivers. That is, the receivers know the beginning of the message during the synchronization even for the variable message size scenario. However, they cannot estimate the end of messages during the synchronization for the variable message size. We will not discuss the synchronization techniques which are not in the scope of this chapter.

For the fixed message size scenario, the receiver takes the advantage of the known position information of the random seed in messages. After the synchronization, the receivers know the position of the last bit of the random seed so that they can directly compare the last bit with the code sequences of the code set $C_e$. Figure 7.5 shows the average time of despreading messages for DSD-DSSS SUBSET and ARS-DSSS schemes when using different message sizes. The average despreading time of the ARS-DSSS scheme is almost the same as the DSD-DSS SUBSET. Since our ARS-DSSS scheme introduces only one code sequence of $C_x$, the additional computation overhead for comparing one code sequence for one bit is trivial.

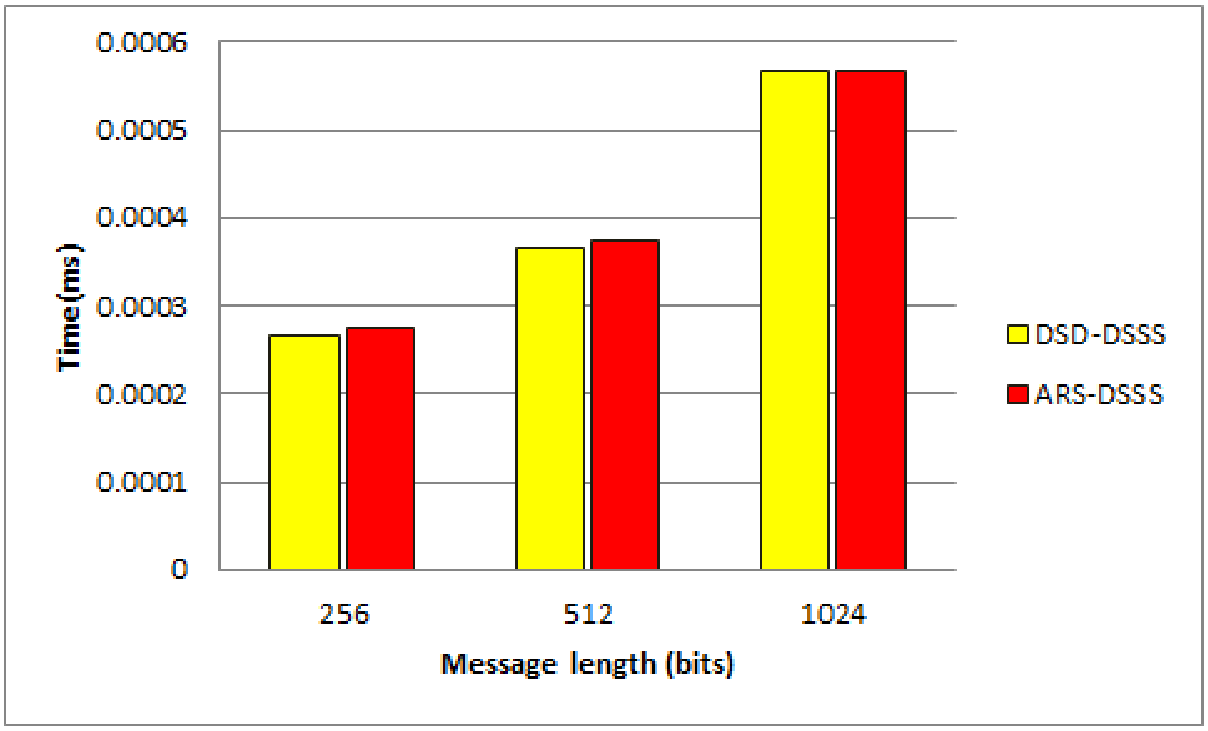


Figure 7.5: Implemented decoding time for the fixed message size scenario

The other scenario is that the receiver has no information about the starting chip of the last bit of the random seeds. Thus, the receiver is required to find the maximum correlation from the beginning bit to the last bit of the message for every code in $C_e$. Recall the sliding window diagram. For DSD-DSSS SUBSET, the computation time for finding the beginning of the last random seed bit in messages ($T_s$) is

$$T_s = ((m \times l_c) + ((s-1) \times l_c) + (1 \times l_c)) \times |C_e|$$

where $m$ is the message size, $s$ is the length of the seed, $|C_e|$ is the number of code sequences in $C_e$, and $l_c$ is the code length (chip length). In our experiments, the number of code sequence $C_e$ is 56 (i.e., $\lceil \frac{(N*N)+N+3}{2} \rceil$, where N=10), the seed size is 64, and the chip length is 32. Thus, the time for finding the seed in the DSD-DSSS SUBSET is $T_s(DSD-DSSS) = (m*32+64*32)*56$ with different message sizes. For ARS-DSSS, the time is $T_s(ARS-DSSS) = (m*32+64*32)*1$, where the cardinality of the code sequence of $C_x$ is 1.

Figure 7.6 shows the average time (implemented) for despreading messages for both the DSD-

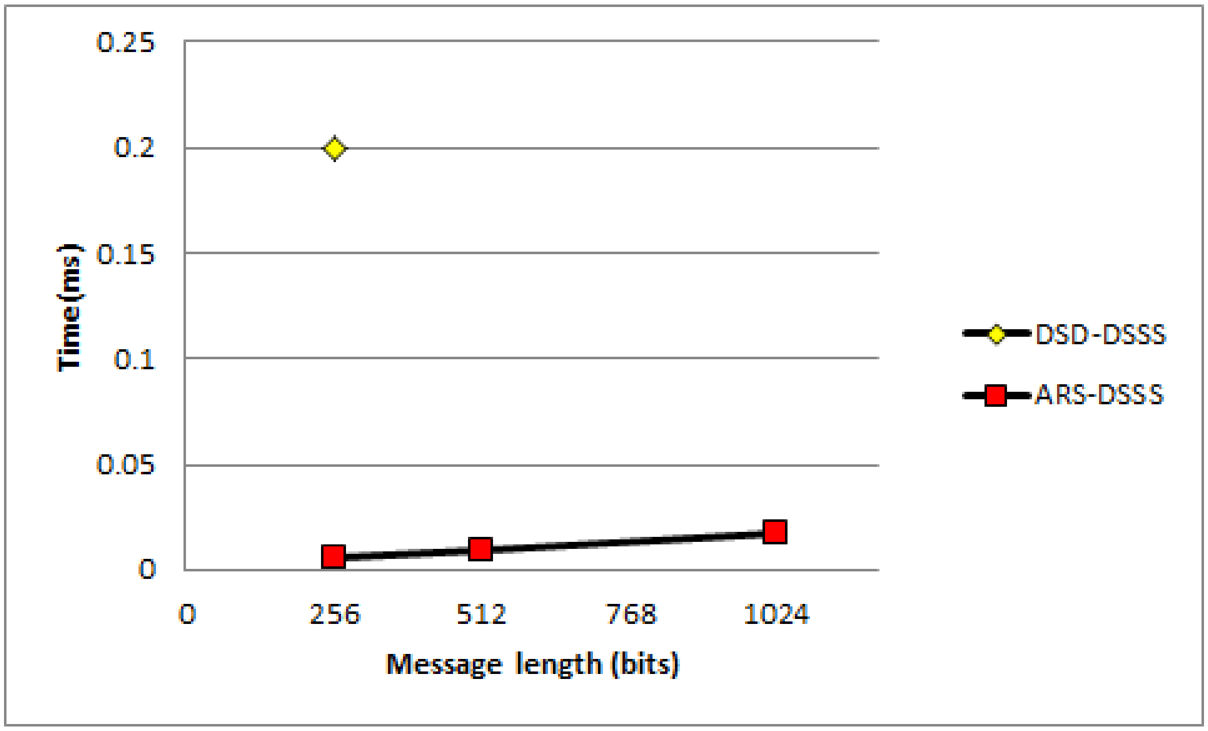


Figure 7.6: Implemented decoding time for the variable message size scenario. The DSD-DSSS receivers could not decode 512 bits or longer messages.

DSSS SUBSET and ARS-DSSS schemes when the receiver has no information about the starting chip of the last bit of the random seed. Without loss of generality, we use fixed message sizes but receivers compare the correlation of the seed from the starting of the messages to the end of the messages. For the DSD-DSSS SUBSET scheme, the receiver can despread an average of only 70 messages out of 790 messages for the 256-bit message size. The average time for despreading a message is around $0.2\ ms$. However, the receiver cannot despread any 512-bit and 1024-bit messages due to the dramatic increase of computation overhead for finding the starting chip of the seed. On the other hand, our ARS-DSSS scheme can despread messages for all three message sizes without a significant computational increase compared to the fixed message size case. The time for despreading a message is around $0.005\ ms$ for 256-bit messages. Therefore, our ARS-DSSS scheme is a practical solution against the seed jamming attack.

## 7.5 Related Work

The traditional DSSS techniques are vulnerable to reactive jamming attacks due to the requirement of pre-shared keys between the sender and receivers in wireless broadcast communications. Many researchers have been proposed schemes to eliminate the pre-shared keys by using random code sequences [7,28,45].

These approaches provide methods for both the sender to randomly generate codes sequences for each message and the receivers to regenerate the same code sequences without sharing the secret keys a priori. In [7], the authors propose the Uncoordinated DSSS scheme (UDSSS) in which the sender and receivers have a set of codes sequences, known to the public, used for spreading and despreading messages, respectively. The key idea is that the communication is the same as the traditional DSSS but the requirement of shared secret keys are randomly released. Then, the sender repeatedly sends the same message with randomly selected codes sequences so that the receivers can decode the message if any unjammed messages arrived. In [45], the sender and receivers also share the spreading code but the authors use the correlation between two codes to find the transmit bit (i.e., high correlation is 1 and low correlation is 0). Moreover, they proposed a set of pre-defined spreading code sequences and its indices to reduce the communication overhead. In [28], the authors proposed the DSD-DSSS SUBSET scheme where they used random seeds for each message to generate code sequences and seeds are disclosed at the end of each message. Thus, it is essentially infeasible for a reactive jammer to generate the same codes sequences before the receivers receive entire messages.

However, all these approaches are vulnerable to the seed jamming attacks. Since all the information is known to the public, a sophisticated jammer can find the position information of seeds or indexes during the synchronization. During the synchronization between the sender and receivers, the sophisticated jammer can also synchronize with them. For example, UDSSS can be optimized by using UDSSS to transmit the spreading key only [41]. RD-DSSS also uses the small number of random code sequences to reduce the communication overhead. The size of the random seeds is small and their positions are known for the fixed message size in DSD-DSSS. Therefore, a sophisticated jammer can take advantage of this information and jam even one bit or small number of the spreading keys, indices, or seeds for these algorithms.

In particular, the point of maximum vulnerability of these techniques is the seed itself since the receivers cannot despread arrived messages correctly from the jammed seeds. Moreover, no error correction techniques can be used to recover the seed or the message in this case. FEC [31] can be used to recover from jamming messages directly. To address these issues, we particularly investigated the DSD-DSSS SUBSET scheme [28] and proposed the ARS-DSS scheme by using an additional location seed. Using this location seed, the sender can hide information to the unknown receivers such as the message size and random seed. Therefore, the sophisticated jammer cannot compute the position of the random seed in messages during the synchronization.

## 7.6 Conclusion

In this chapter, we presented a new type of jamming attack called *seed jamming attack* and particularly chose the DSD-DSSS SUBSET scheme [28] to demonstrate the effectiveness of the attacks. In the scheme, if the message size is fixed, the sophisticated jammer can simply jam the last bit of the random seed in messages to disable the wireless communication. To address these attacks, we proposed the ARS-DSSS scheme by introducing the new random location seed to the DSD-DSSS SUBSET scheme. Using the new location scheme seed, the receivers can find the position of the random seed in messages even for the variable message sizes without incurring a huge performance overhead as required in DSD-DSSS SUBSET. However, it is infeasible for the jammer to find the location seed and then jam the communication in real time. We evaluate the effectiveness of our scheme through our security analysis and its implementation. Our implementation results demonstrate that the ARS-DSSS scheme is effective against the seed jamming attacks and also reduces the computation overhead from $|C_e|$ for the DSD-DSSS SUBSET scheme to $1$ for ARS-DSSS in the variable message size scenario.

# Chapter 8

# Conclusion

In this thesis, we presented three new jamming attacks with primary emphasis on *Channel Detecting Jamming Attacks* (CDJAs) in which a jammer is able to find the sender's channel hopping sequences by taking advantage of properties of blind rendezvous algorithms in CRNs. We investigated their effectiveness on the rendezvous algorithms for wireless networks [24, 28] and the state-of-the-art blind rendezvous algorithms for CRNs as presented in [29].

First, we exploited modular-based symmetric blind rendezvous schemes such as Modular Clock (MC) rendezvous and Jump-Stay (JS) rendezvous algorithms since these algorithms provide fast blind rendezvous methods for CRNs [6, 25]. These schemes utilize prime number modular arithmetic to generate random CH sequence for each user. However, the channel detecting jammer can compute the same CH sequence as the senders' by taking advantage of their modular properties (see Chapter 2.2 and Chapter 3.3) . The jammer is a standard node with possible but not required additional listening capabilities (e.g., two listening channels). It can then find the forward-hop rate within a short period. Using this information, the jammer can generate the CH sequence and jam the remaining CH sequences. Thus, the probability of rendezvous will be dramatically decreased under CDJAs. Our simulation results demonstrate that the rendezvous probability of both the symmetric MC and symmetric JS systems under the CDJA is dramatically decreased for all available channels $M$ (e.g., around $15\%$ for almost all $M$ in the MC system and less than $10\%$ for all $M$ in the JS system).

Second, we expanded our CDJAs to the remaining representative blind rendezvous algorithms for symmetric models; Generated Orthogonal Sequence (GOS) [6], Deterministic Channel Rendezvous Sequence (DRSEQ) [12], and Channel Rendezvous Sequence (CRSEQ) [43]. However, both DRSEQ and CRSEQ schemes are deterministic so they are trivial to jam using one listening and one jamming channel. For the GOS scheme, the channel detecting jammer is able to take advantage of the GOS algorithm to find the sender's CH sequence within the maximum of $(\frac{M}{2} + 1) \times (1 + M)$ time slots and an upper bound of $(\frac{M}{4} + 1) \times (1 + M)$ expected time slots (see Chapter 4.2). Significantly, the jammer can immediately jam the detected channels and can completely jam the sender after the maximum of $(\frac{M}{2} + 1) \times (1 + M)$ time slots. Our simulation results demonstrate that the rendezvous probability of the GOS system under the CDJA is dramatically decreased to less than $10\%$ for most available channels $M$.

Third, we presented our modified CDJA and demonstrated its effectiveness against the EJS scheme for both symmetric and asymmetric models [26]. The channel detecting jammer is able to find the CH sequence of the symmetric EJS within the first $P$ time slots using two listening channels (see Chapter 5.3). Then the jammer can completely jam the sender's CH sequence after an average $\lfloor \frac{(P+1)}{2} \rfloor$ time slots using a single channel jammer. For the asymmetric EJS system, the jammer can find the sender's CH sequence with high probability within $P$ time slots and jam the remaining channels. Our simulation results demonstrate that the rendezvous probability of the symmetric EJS system under the CDJA is dramatically decreased for all available channels $M$ (e.g., less than $10\%$ for all $M$). The rendezvous probability for the asymmetric EJS under the CEJA depends on the ratio $|m_1|/|M|$ where $|m_1|$ is the number of available channels for the sender. For most of ratios, the probability of rendezvous for the asymmetric EJS system under the CDJA is less than $10\%$ except for where the $|m_1|/|M|$ is near $0.5$ (here less than $15\%$) when the jammer uses two listening channels.

Fourth, we investigated a quorum-based frequency hopping scheme in which a sender and receiver can generate their own frequency hopping sequences using the cyclic quorum properties [24]. This algorithm provides a Frequency Quorum-based Rendezvous (FQR) algorithm to a general rendezvous scheme for wireless networks but it can be extended to synchronized CRNs without difficulty. The sender and the receiver randomly select its quorum set from the minimal $(N, k)$ difference sets and generate frequency sequences. However, the sophisticated jammer can find the sender's quorum set within the second frame (i.e., $2k$ time slots) in the FQR system when it has the capability of listening on $k$ frequencies (see Chapter 6.3). Thus, the jammer can completely jam the sender after $2k$ time slots using an average of $\lfloor \frac{k+1}{2} \rfloor$ and a maximum of $k$ frequencies. Our simulation results demonstrate that the rendezvous probability of the FQR system under the sophisticated jamming attack decreased as the number of available channel $N$ increases (e.g., $\leq 35\%$ for $N \geq 30$ in $k^2$ time slots). We must note that jamming here requires $k$ listening channels and $k$ jamming channels.

Fifth and finally, we investigated a more general random spread spectrum technique called a delayed random seed disclosure DSSS (DSD-DSSS) scheme for broadcast wireless communications [28]. The DSD-DSSS scheme removes the pre-shared key dependency and allows a sender to randomly generate spread code sequences for each message using random seeds. Then the sender discloses the seeds at the end of each message so that the jammer cannot generate the same code sequences without the knowledge of the seeds. However, we present a new type of jamming attack called a *seed jamming attack* in which an attacker particularly focuses on jamming the random seed(s) in fixed-size messages (see Chapter 7.3). Thus, the receiver cannot despread received messages because it cannot find the seed and hence cannot regenerate the correct spread code sequences. To mitigate this jamming attack, we propose an advanced

Table 8.1: The upper-bound of MTTR

| Algorithms | Symmetric model | Asymmetric model |
|---|---|---|
| Jump-Stay [25] | $3P$ | $3MP(P-G)+3P$ |
| GOS [14] | $M(M+1)$ | not applicable |
| MC [6] | $2P$ (not gurantteed) | not applicable |
| MMC [6] | unknown | unknown |
| DRSEQ [12] | $2M+1$ | not applicable |
| CRSEQ [43] | $\geq (P-1)(3P-1)$ | $P(3P-1)$ |
| Random | unbounded | unbounded |
| EJS [26] | $4P$ | $4P(P+1-G)$ |

Table 8.2: The upper-bound of ETTR

| Algorithms | Symmetric model | Asymmetric model |
|---|---|---|
| Jump-Stay [25] | $5P/3+3$ | $2MP(P-G)+(\frac{M+5-P-(2G-1)}{M})P$ |
| GOS [14] | $\frac{M^4+2M^2+6M-3}{3(M^2+M)}$ | not applicable |
| MC [6] | $\frac{2P^2}{(P-1)}$ | not applicable |
| MMC [6] | unknown | unknown |
| DRSEQ [12] | unknown | not applicable |
| CRSEQ [43] | unknown | unknown |
| Random[Chapter 5] | $M$ | $\frac{\lvert m_1\rvert \cdot \lvert m_2\rvert}{\lvert G\rvert}$ |
| EJS [26] | $3P/2+3$ | $4P(P+1-G)-\frac{[4PG(P-G)+G/2]}{(\lvert m_1\rvert \cdot \lvert m_2\rvert)}$ |

random seed DSSS (ARS-DSSS) scheme which strengthens the previous algorithm called DSD-DSSS by using an additional location seed. Our security analysis and implementation results demonstrate how to defeat the seed jamming attacks and how to reduce the computation overhead of the DSD-DSSS scheme.

To remedy CDJAs, we revisited the Random CH rendezvous schemes for both symmetric and asymmetric models. Table 8.1 and Table 8.2 show that the maximum time to rendezvous (MTTR) and the expected time to rendezvous (ETTR) for the state-of-the-art blind rendezvous algorithms presented in [6, 26, 29], respectively. Since upper bounds of the expected time to rendezvous for the symmetric GOS, JS, MC, DRSEQ, CRSEQ, EJS, and Random schemes are $\frac{M^4+2M^2+6M-3}{3M(M+1)}$, $\frac{5P}{3}+3$, $\frac{2P^2}{(P-1)}$, unknown, unknown, $\frac{3P}{2}+3$, and $M$ respectively, Random could be the most effective, efficient and robust rendezvous but it does sacrifice the guaranteed rendezvous time of some of these rendezvous schemes. We should note that DRSEQ and CRSEQ have MTTR of $2M+1$ and $P(3P-1)$ respectively but that does not change the fact that Random could have the best ETTR. In addition, we present the theoretical expected time to rendezvous (ETTR) for the asymmetric Random system of $\frac{\lvert m_1\rvert \cdot \lvert m_2\rvert}{\lvert G\rvert}$ and the ETTR would be almost the same under the CDJA because of the random channel selections in the CH sequence. Since upper bounds of the expected time to rendezvous for the asymmetric GOS, JS, MMC, DRSEQ, CRSEQ, EJS, and Random schemes are; not applicable, $2MP(P-G)+(\frac{M+5-P-(2G-1)}{M})P$, unknown, not applicable, unknown, $4P(P+1-G)$, $4P(P+1-G)-\frac{[4PG(P-G)+G/2]}{(\lvert m_1\rvert \cdot \lvert m_2\rvert)}$, and $\frac{\lvert m_1\rvert \cdot \lvert m_2\rvert}{\lvert G\rvert}$

respectively, Random vastly outperforms for ETTR all of the asymmetric blind rendezvous schemes with known ETTR. Therefore, the symmetric and asymmetric Random schemes can be an effective, efficient and robust rendezvous scheme against CDJAs and in most circumstances.

In this thesis, we exploited the state-of-the-art blind rendezvous algorithms for CRNs and then we demonstrated the effectiveness of CDJAs against them. These rendezvous algorithms can be categorized by the existence of jamming attacks in CRNs. If there are no jamming attacks, then DRSEQ [12] and CRSEQ [43] would be the best algorithms based on the MTTR for the symmetric model and the asymmetric model, respectively. On the other hand, if there are jamming attacks, then our CDJAs can significantly decrease the rendezvous probability for most of blind rendezvous algorithms so that Random scheme can be the more reliable rendezvous scheme for the symmetric model. However, for the asymmetric model, the Random scheme can be the best algorithm with occasional jamming attacks when the ratio $\frac{|m_1|}{|M|}$ is less than a certain threshold. If the ratio is greater than the threshold, EJS with random replacement algorithms (see Chapter 5.3.2) would be better than Random scheme. For example, Random scheme is better than EJS with random replacement algorithm for where the $\frac{|m_1|}{|M|}$ ratio is near $0.5$ when there is no jamming attacks. This means that if the ratio $\frac{|m_1|}{|M|}$ is high (e.g., close to 1), then the asymmetric EJS with replacement algorithms would be the best algorithm with occasional jamming attacks. On the other hand, if the ratio $\frac{|m_1|}{|M|}$ is low (e.g., lower than say a 0.5 threshold), then the asymmetric Random scheme is the best algorithm with occasional jamming attacks. However, the threshold would be effective only if the sender and receiver use the same blind rendezvous algorithm. Therefore, it would be worthwhile to investigate on a hybrid approach in which a sender and receiver use different blind rendezvous algorithms based on their ratios $\frac{|m_1|}{|M|}$ or $\frac{|m_2|}{|M|}$. For low ratios, Random scheme will always work well but as one of the ratios approaches 1, EJS and Random schemes may work very well together.

# REFERENCES

[1] I. F. Akyildiz, W.-Y. Lee, M. C. Vuran, and S. Mohanty. Next generation dynamic spectrum access/cognitive radio wireless networks: A survey. *Computer Networks*, 50, 2006.

[2] IEEE Standards Association. LAN/MAN wireless LANS IEEE 802.11 - 2007 IEEE standard for information technology. `http://standards.ieee.org/getieee802/802.11.html`.

[3] R.H. Barker. Group synchronization of binary digital systems. In *Communication Theory*, pages 64–78, 1953.

[4] K. Bian, J.-M. Park, and R. Chen. A quorum-based framework for establishing control channels in dynamic spectrum access networks. In *proceedings of MobiCom'09*, 2009.

[5] M. M. Buddhikot, P. Kolodzy, S. Miller, K. Ryan, and J. Evans. DIMSUMnet: New directions in wireless networking using coordinated dynamic spectrum. In *proceedings of IEEE WoWMoM*, pages 78–85. IEEE, 2005.

[6] Theis Nick C., Thomas Ryan W., and DaSilva Luiz A. Rendezvous for cognitive radios. *IEEE Transactions on Mobile Computing*, 10(2):216–227, 2011.

[7] Pöpper Christina, Strasser Mario, and Čapkun Srdjan. Jamming-resistant broadcast communication without shared keys. In *Proceedings of the USENIX Security Symposium*, 2009.

[8] Federal Communications Commission. FCC Notice of Proposed Rulemaking FCC 04-113.

[9] C. Cordeiro, K. Challapali, D. Birru, and N. Sai Shankar. IEEE 802.22: The first worldwide wireless standard based on cognitive radios. *Journal of Communications*, 1(1):38–47, 2006.

[10] C. Cormio and K. R. Chowdhury. Common control channel design for cognitive radio wireless ad hoc networks using adaptive frequency hopping. *Ad Hoc Networks*, 8:430–438, 2010.

[11] Cabric D., Tkachenko A., and Brodersen R.W. Spectrum sensing measurements of pilot, energy, and collaborative detection. In *proceedings of IEEE Military Communications Conference (MILCOM '06)*, pages 1 –7, 2006.

[12] Yang D., Shin J., and Kim C. Deterministic rendezvous scheme in multichannel access networks. *Electronics Letters*, 46(20):1402–1404, 2010.

[13] L. DaSilva and I. Guerreiro. Sequence-based rendezvous for dynamic spectrum access. In *proceedings of IEEE DySPAN*, pages 1–7. IEEE, 2008.

[14] L.A. DaSilva and I. Guerreiro. Sequence-based rendezvous for dynamic spectrum access. In *New Frontiers in Dynamic Spectrum Access Networks, 2008. DySPAN 2008. 3rd IEEE Symposium on*, pages 1–7, 2008.

[15] Ariananda D.D., Lakshmanan M.K., and Nikookar H. A survey on spectrum sensing techniques for cognitive radio. In *proceedings of the Second International Workshop on Cognitive Radio and Advanced Spectrum Management (CogART '09)*, pages 74 –79, 2009.

[16] Anderson E.J. and Weber R.R. The rendezvous problem on discrete locations. *Journal of Applied Probability*, 28:839–851, 1990.

[17] Lee Eun-Kyu, Oh, Soon Y., and Gerla Mario. Frequency quorum rendezvous for fast and resilient key establishment under jamming attack. *SIGMOBILE Mob. Comput. Commun. Rev.*, 14:1–3, 2010.

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

[19] Jiang Jehn-Ruey, Tseng Yu-Chee, Hsu Chih-Shun, and Lai Ten-Hwang. Quorum-based asynchronous power-saving protocols for IEEE 802.11 ad hoc networks. *Mob. Netw. Appl.*, 10(1-2):169–181, 2005.

[20] Juncheng Jia, Qian Zhang, and Xuemin Shen. HC-MAC: A hardware-constrained cognitive mac for efficient spectrum management. *Selected Areas in Communications, IEEE Journal on*, 26(1):106 –117, 2008.

[21] Simon Marvin K., Omura Jim K., Scholtz Robert A., and Levitt Barry K. *Spread spectrum communications handbook (revised ed.)*. McGraw-Hill, Inc., New York, NY, USA, 1994.

[22] Bian Kaigui, Park Jung-Min, and Chen Ruiliang. A quorum-based framework for establishing control channels in dynamic spectrum access networks. In *Proceedings of the 15th annual international conference on Mobile computing and networking*, MobiCom '09, pages 25–36, New York, NY, USA, 2009. ACM.

[23] E.-K. Lee, S. Y. Oh, and M. Gerla. Fast and resilient key establishment using quorum rendezvous under jamming attack. Technical Report 110005, UCLA, 2011.

[24] Eun-Kyu Lee, Soon Young Oh, and Mario Gerla. Timely and robust key establishment under jamming attack in critical wireless networks. In *Proceedings – IEEE Military Communications Conference MILCOM'11*. IEEE, 2011.

[25] Lin, Zhiyong, Liu, Hai, Chu, Xiaowen, Leung, and Yiu-Wing. Jump-stay based channel-hopping algorithm with guaranteed rendezvous for cognitive radio networks. In *INFOCOM*, pages 2444–2452. IEEE, 2011.

[26] Z. Lin, H. Liu, X. Chu, and Y. Leung. Enhanced jump-stay rendezvous algorithm for cognitive radio networks. *Communications Letters, IEEE*, PP(99):1–4, 2013.

[27] An Liu, Peng Ning, Huaiyu Dai, and Yao Liu. USD-FH: Jamming-resistant wireless communication using frequency hopping with uncoordinated seed disclosure. In *Proceedings of 7th IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS '10)*, 2010.

[28] An Liu, Peng Ning, Huaiyu Dai, Yao Liu, and Cliff Wang. Defending DSSS-based broadcast communication against insider jammers via delayed seed-disclosure. In *Proceedings of 26th Annual Computer Security Applications Conference (ACSAC '10)*, 2010.

[29] Hai Liu, Zhiyong Lin, Xiaowen Chu, and Yiu-Wing Leung. Taxonomy and challenges of rendezvous algorithms in cognitive radio networks. In *Computing, Networking and Communications (ICNC), 2012 International Conference on*, pages 645 –649, 2012.

[30] Ettus Research LLC. The USRP product family products and daughter boards. `https://www.ettus.com/product`, Accessed in March 2012.

[31] Luby M., Vicisano L., Gemmell J., Rizzo L., Handley M., and J. Crowcroft. Forward error correction (FEC) building block. RFC 3452, 2002.

[32] L. Ma, X. Han, and C.-C. Shen. Dynamic open spectrum sharing for wireless ad hoc networks. In *proceedings of IEEE DySPAN*, pages 203–213. IEEE, 2005.

[33] L. Ma and C.-C. Shen. Security-enhanced virtual channel rendezvous algorithm for dynamic spectrum access wireless networks. In *IEEE IEEE DySPAN*. IEEE, 2008.

[34] Maekawa Mamoru. A $N$ algorithm for mutual exclusion in decentralized systems. *ACM Trans. Comput. Syst.*, 3(2):145–159, 1985.

[35] Strasser Mario, Pöpper Christina, and Čapkun Srdjan. Efficient uncoordinated FHSS anti-jamming communication. In *Proceedings of the ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*. ACM, 2009.

[36] Young-Hyun Oh and David J. Thuente. Limitations of quorum-based rendezvous and key establishment schemes against sophisticated jamming attacks. In *Military Communications Conference, MILCOM'12*, pages 1 –6, 2012.

[37] Young-Hyun Oh and David J. Thuente. Channel detecting jamming attacks against jump-stay based channel hopping rendezvous algorithms for cognitive radio networks. In *International Conference on Communications and Networks (ICCCN '13), IEEE*, 2013.

[38] Young-Hyun Oh and David J. Thuente. Sequence sensing jamming attacks against modular-based channel hopping rendezvous algorithms for cognitive ratio networks. In *International Conference on Communications (ICC '13), IEEE*, 2013.

[39] J. Pérez-Romero, O. Sallent, R. Agusti, and L. Giupponi. A novel on-demand cognitive pilot channel enabling dynamic spectrum allocation. In *proceedings of IEEE DySPAN*, pages 46–54. IEEE, 2007.

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

[41] Christina Pöpper, Mario Strasser, and Srdjan Čapkun. Jamming-resistant broadcast communication without shared keys. `http://www.usenix.org/events/sec09/tech/slides/popper.pdf`.

[42] GNU Radio. The GNU software radio. `http://gnuradio.org/redmine/projects/gnuradio/wiki`, Accessed in March 2012.

[43] Jongmin Shin, Dongmin Yang, and Cheeha Kim. A channel rendezvous scheme for cognitive radio networks. *Communications Letters, IEEE*, 14(10):954–956, 2010.

[44] Luk Wai-Shing and Wong Tien-Tsin. Two new quorum based algorithms for distributed mutual exclusion. In *Proceedings of the 17th International Conference on Distributed Computing Systems (ICDCS '97)*, ICDCS '97, pages 100–106. IEEE Computer Society, 1997.

[45] Liu Yao, Ning Peng, Dai Huaiyu, and Liu An. Randomized differential DSSS: jamming-resistant wireless broadcast communication. In *INFOCOM'10: Proceedings of the 29th conference on Information communications*. IEEE Press, 2010.