Multi-Radio Channel Detecting Jamming Attack Against Enhanced Jump-Stay Based Rendezvous in Cognitive Radio Networks

Yang Gao[1(✉)], Zhaoquan Gu[1], Qiang-Sheng Hua[2], and Hai Jin[2]

[1] Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing, People's Republic of China
y-gao13@mails.tsinghua.edu.cn

[2] Services Computing Technology and System Lab/Cluster and Grid Computing Lab, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, People's Republic of China

Abstract. Rendezvous problem has been attracting much attention as a fundamental process to construct the Cognitive Radio Network (CRN). Many elegant algorithms have been proposed to achieve rendezvous on some common channel in a short time, but they are vulnerable to the jamming attacks where a jammer may exist listening or blocking the channels[14]. In this paper, we propose the Multi-Radio Channel Detecting Jamming Attack (MRCDJA) problem where the jammer can access multiple channels simultaneously. We assume the users adopting the Enhanced Jump Stay (EJS)[8] algorithm, which guarantees rendezvous by generating a channel hopping sequence, and our goal is to determine the hopping sequence as quickly as possible.

1 Introduction

Cognitive Radio Network (CRN) has become a new paradigm to alleviate the spectrum scarcity problem since the unlicensed spectrum is overcrowded while the licensed spectrum has low utilization[15]. There are two kinds of users in CRN, the so-called primary users (PUs) who own the licensed spectrum and the secondary users (SUs) who can use the particular licensed spectrum that are not occupied. Unless otherwise specified, 'users' mentioned hereafter refers to SUs.

In constructing such a CRN, *rendezvous* is a fundamental procedure for the users to establish a link on a common licensed channel for communication[10]. Many algorithms have been proposed to reduce the time needed to rendezvous[2,4,8,9] by constructing a hopping sequence. Let U be the set of the available channels and $|U| = M$, the state-of-the-art results guaranteed rendezvous in $O(M^2)$ time slots[8]. However, these algorithms are vulnerable to Channel Detecting Jamming Attack (CDJA)[14] where a jammer exists trying to listen or block the

This work was supported in part by the National Basic Research Program of China Grant 2011CBA00300, 2011CBA00301, the National Natural Science Foundation of China Grant 61103186, 61033001, 61361136003.

D. Xu et al. (Eds.): COCOON 2015, LNCS 9198, pp. 759–770, 2015.
DOI: 10.1007/978-3-319-21398-9_59

channels[14]. The intuition of the CDJA is to determine one user's channel hopping sequence based on the listened channels and to jam the predicted channels against rendezvous.

Recently, multiple radios architecture has been widely used in wireless networks[1,7,12,13] and many problems can be solved efficiently. In this paper, we propose Multi-Radio Channel Detecting Jamming Attack (MRCDJA) where the jammer can listen or block n ($n > 1$) channels simultaneously and the goal is to determine the user's channel hopping sequence as quick as possible. Since Enhanced Jump Stay (EJS)[8] is one representative state-of-the-art algorithm, we design multi-radio jamming algorithm against EJS.

In this paper, we first review the EJS algorithm and the CDJA algorithm as the cornerstones, then we design an efficient algorithm to determine the channel hopping sequence when the users can use any channel in the set U. We show that the sequence can be figured out in $O(\frac{M}{n})$ expected time, which is n times better than the result in [14]. Moreover, when some channels are occupied by the PUs, we provide another efficient algorithm for the MRCDJA problem, we show that if the ratio of the available channels is more than 40%, our algorithms can guarantee fast successful attack with probability more than 80%. We also evaluate our proposed algorithms and the results show that our algorithms can determine the hopping sequence quickly.

The rest of the paper is organized as follows. Preliminaries are provided in the next section. In Section 3, we review the EJS algorithm. In Section 4, we show our main results by providing efficient algorithms to figure out the channel hopping sequences. In Section 5, we conduct simulations for our algorithms. Finally, we conclude the paper in Section 6.

2 Preliminaries

We consider a CRN with two users that are trying to rendezvous with each other and a jammer who aims to break the rendezvous process by blocking some licensed channels. We suppose the licensed spectrum is divided into M non-overlapping channels labeled $U = \{1, 2, \ldots, M\}$ and the labels are known to the users and the jammer.[1]

Time is supposed to be divided into slots of equal length of $2t$, where t is the time duration for establishing a connection. In each time slot, the user can access one channel for rendezvous attempt, while the jammer can choose $n > 1$ channels for listening (detecting) or blocking. Here listening or detecting means that the jammer can check some channels to see if the user is accessing these channels in the same time. The users can achieve rendezvous only if they access the same channel in the same time slot and the jammer doesn't block the channel. We assume the users start the rendezvous process asynchronously, i.e. the two users doesn't need to start in the same time slot. Obviously, we can

[1] It is reasonable to make this assumption since the channels are easy to be distinguished by frequency, however there are some research on the case when the labels are not known to the users[5,6].

consider the rendezvous process as slot-aligned even in the asynchronous setting since the duration of each time slot is $2t$[6]. In this paper, we assume the jammer starts detecting the channels before all users starting to access channels.

Before the users start the rendezvous process, they sense the licensed spectrum to check whether the channels in U are occupied. We say that a channel is *available* for the user if it is not occupied by any nearby PUs. After the spectrum sensing stage, each user i has a set of available channels $C_i \subseteq U$ and they can begin rendezvous attempt by accessing the available channels. The users are said to be *symmetric* if the available channel set is U for both of them, i.e. $C_1 = C_2 = U$[8], otherwise, they are *asymmetric*, i.e. $C_1 \subseteq U, C_2 \subseteq U$[8]. In this paper, we design efficient algorithms for the jammer to detect the channel hopping sequence to prevent rendezvous between the users. We assume the users adopting the Enhanced Jump Stay (EJS)[8] algorithm and formulate the problem as:

Problem 1 (MRCDJA). Suppose two users run the EJS algorithm for rendezvous, while the jammer who can access $n > 1$ channels simultaneously tries to determine the channel hopping sequence of one user. The goal is to design an efficient algorithm to detect the sequence as quickly as possible.

3 Review of the Enhanced Jump-Stay Algorithm

Since Enhanced Jump Stay (EJS)[8] is one representative rendezvous algorithm, we review the intuition of the algorithm in this section. The algorithm works as follows: each user generates its own channel hopping sequence in rounds. Each round consists of three jump patterns and one stay pattern where each pattern lasts for P time slots (P is the smallest prime number such that $P \geq M$). In the first round, the user chooses a random channel $i \in [1, P]$ as a start channel and a random number $r \in C$ as the step length, where C is the available channel set of this user. In the jump patterns of the first round where time t suits $0 \leq t < 3P$, the user chooses channel $(i + tr - 1) \mod P + 1$, then in the stay pattern ($3P \leq t < 4P$), the user stays on channel r. The j-th ($j > 1$) round is generated in the same way, except that the start channel is $(i + j - 2)\%P + 1$ and the step length remains r. For example $M = 5, P = 5, i = 3, r = 1$, we show the first two rounds of the channel sequence:

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

Notice that (3 + 1) represents the start channel of the second round as described above.

There are two special situations we need to consider. The first one is $P \neq M$, channel x ($x > M$) doesn't exist and thus we map it to $x - M$. The second one is the asymmetric situation, where some channels in U may be occupied by the PUs, and thus the user just choose a random channel in its available channel set to replace the unavailable channel.

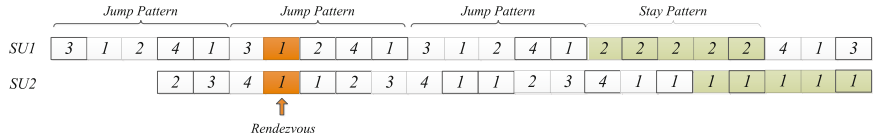


Fig. 1. Example for rendezvous of two users

For example, suppose $M = 4, P = 5$, the available channel set $C = \{1, 3, 4\}$, $i = 3, r = 1$. It is obvious that channel 5 doesn't exist and it should be mapped to channel 1. Since channel 2 doesn't belong to C, the user chooses a random channel from C once it should access channel 2. Therefore, the new sequence is:

$$3,4,1,1,3,3,4,1,1,4,3,4,1,1,1,1,1,1,1,1,(3+1),1,1,1,3,4,1,1,4,3,4,1,1,4,3,1,1,1,1,1,...$$

In this paper, we use two important lemmas in [11] as:

Lemma 1. *Given a positive integer P, if $r \in [1, P)$ is relatively prime to P, i.e., the common factor between them is 1, then for any $x \in [0, P)$ the sequence $S =< x\%P + 1, (x + r)\%P + 1, ..., (x + (P - 1)r)\%P + 1 >$ is a permutation of $< 1, 2, ..., P >$.*[2] *[11]*

Lemma 2. *Given a prime number P, if r_1 and r_2 are two different numbers in $(0, P)$, then for any $x_1, x_2 \in [0, P)$, there must be an integer $k \in [0, P)$ such that $(x_1 + kr_1)\%P = (x_2 + kr_2)\%P$. [11]*

Combining these two Lemmas, the following theorem is proved.

Theorem 1. *Under the symmetric setting, any two users performing EJS achieve rendezvous in at most $4P$ time slots, under asymmetric setting the time is at most $4P(P + 1 - G)$ time slots, where P is the smallest prime number greater or equal to M and G is the number of channels commonly available to the two users. [8]*

Fig. 1 is an example of rendezvous under symmetric setting. Here $U = \{1, 2, 3, 4\}, P = 5$, user 1 chooses channel 3 as its start channel and 2 as its step length, user 2 chooses channel 2 as its start channel and 1 as its step length. User 2 starts 3 time slots later than user 1, they rendezvous at the 4th time slot (the start time is based on the later user's clock) on channel 1, here channel 1 is mapped from channel 5 for both users.

Remark 1. The Expected Time To Rendezvous (ETTR) of EJS is proved to be no more than $\frac{3}{2}P + 3$ in [8]. However, it is a very loose bound. In our work, we derive a much more accurate estimation ($\frac{13}{24}P + O(1)$). Due to the page limits, we left the proof and simulation in the full version [3].

[2] We use the symbol % as the meaning of mod function in this paper.

4 Multi-Radio Jamming Attack Algorithm

In this section, we design efficient algorithms for the jammer to attack the rendezvous process. Clearly, if the jammer can figure out one user's hopping sequence, it can block the predicted channels to prevent rendezvous between the users. First, we review the existing work of CDJA[14] problem, then we design efficient algorithms for both symmetric and asymmetric situations when the jammer can access n channels. We show that we can improve the efficiency to figure out the hopping sequence by adopting multiple radios.

4.1 The Existing Work

In [14], the attack scheme against the Jump Stay (JS) algorithm[11] under symmetric setting is proposed and the basic idea is to find the start channel and the step length of one user. The key intuition is that if the jammer detects that the user has accessed channel c_1 at time t_1, and c_2 at t_2 (t_1, t_2 are based on the jammer's own clock), where $(t_1 - t_2)\%P \neq 0$, it can calculate the step length r used by this user. This is due to the reason that c_1 and c_2 can be written as:

$$c_1 = (i + (t_1 + \Delta t)r - 1)\%P + 1$$
$$c_2 = (i + (t_2 + \Delta t)r - 1)\%P + 1$$

here Δt is the time difference between the jammer and the user's clocks. We have $((t_2 - t_1)r)\%P = (c_2 - c_1)\%P$ and the r is unique. Formally, see Alg. 1.

Algorithm 1. StepLength

```
Input: t1, c1, t2, c2
Output: the step length r;
for r = 1 to P do
    if ((t2 − t1)r)%P = (c2 − c1)%P then
        Return r;
    end if
end for
```

However, the algorithm in [14] restricts that the jammer can choose channels only in $\{P-M+1, P-M+2, ...M\}$ since this wouldn't cause any ambiguity. For example in Fig. 1, the jammer shouldn't choose to listen on channel 1, otherwise it cannot tell whether it is the actual channel 1 or it is mapped from channel 5 in the user's original sequence when it detects that the user is accessing channel 1.

In the next subsections we give our attack scheme to the Enhanced Jump Stay Algorithm (EJS) under both symmetric and asymmetric settings. We assume the jammer can access n ($n > 1$) channels simultaneously and our scheme doesn't need the channel selection restriction. The two improvements make the attack more efficient.

Algorithm 2. Multi-Radio Attack Algorithm Under Symmetric Setting

1: **Input:** M, P, n;
2: **Output:** the step length r;
3: Choose n random channels $A = \{a_1, a_2 ... a_n\}$ from $[1, M]$;
4: Keep listening on channels in A until the first signal appears, say it is on channel b_x at time t_x;
5: Randomly select another channel d in $[1, M]$ but not in A, $A = A \setminus \{b_x\} \bigcup d$;
6: Keep listening on channels in A until the first signal appears, say it is on channel b_y at time t_y;
7: **if** $b_x > P - M$ and $b_y > P - M$ **then**
8: Return $StepLength(t_x, b_x, t_y, b_y)$;
9: **else if** $b_x \leq P - M$ and $b_y > P - M$ **then**
10: $r_1 = StepLength(t_x, b_x, t_y, b_y)$, $r_2 = StepLength(t_x, b_x + M, t_y, b_y)$;
11: Check $((b_y + r_1 - 1)\%P)\%M + 1$ and $((b_y + r_2 - 1)\%P)\%M + 1$ at time $t_y + 1$
12: Return the right step length;
13: **else if** $b_x > P - M$ and $b_y \leq P - M$ **then**
14: $r_1 = StepLength(t_x, b_x, t_y, b_y)$, $r_2 = StepLength(t_x, b_x, t_y, b_y + M)$;
15: Check $((b_y + r_1 - 1)\%P)\%M + 1$ and $((b_y + M + r_2 - 1)\%P)\%M + 1$ at time $t_y + 1$
16: Return the right step length;
17: **else**
18: $r_1 = StepLength(t_x, b_x, t_y, b_y)$, $r_2 = StepLength(t_x, b_x + M, t_y, b_y)$, $r_3 = StepLength(t_x, b_x, t_y, b_y + M)$, $r_4 = StepLength(t_x, b_x + M, t_y, b_y + M)$;
19: Check the four channels at time $t_y + 1$: $((b_y + r_1 - 1)\%P)\%M + 1$, $((b_y + r_2 - 1)\%P)\%M + 1$, $((b_y + M + r_3 - 1)\%P)\%M + 1$, or $((b_y + M + r_4 - 1)\%P)\%M + 1$, if the jammer doesn't have enough radios, keep on checking at time $t_y + 2$;
20: Return the right step length;
21: **end if**

4.2 Symmetric Situation

We first give the intuition for finding the step length r: the jammer chooses n channels randomly from $[1, M]$ and keeps listening on these channels, one radio to one channel. Once getting a signal from some channel b_x at time t_x, the jammer randomly chooses another channel replacing b_x and keeps on detecting until the second signal appears on channel b_y at time t_y. If both b_x and b_y are greater than $P - M$, which means they won't cause any ambiguity, the jammer can find r immediately. If one channel is not greater than $P - M$, for example $b_x < P - M$, then the jammer needs to consider two cases: $\{b_x, b_y\}$ and $\{b_x + M, b_y\}$, for either of them the jammer can get a step length. Since the jammer can access more than one channels each time slot, it can check which of the two cases is right in the next time slot. It is similar when b_x and b_y are all not greater than $P - M$, in this case the jammer should check out 4 step lengths in the next 1 or 2 time slots. Formally, please refer to Alg. 2.

In Alg. 2, the input M denotes the number of available channels, P is the smallest prime that $P \geq M$ and n is the number of radios the jammer has. The function *StepLength* refers to Alg. 1. In line 3-6 the jammer gets the two

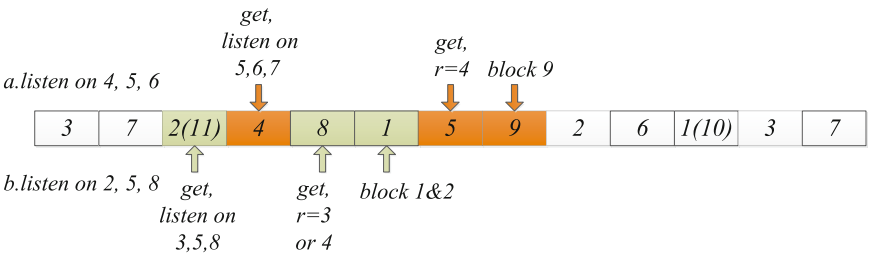


Fig. 2. Example of Alg. 2

signals $(t_x, b_x), (t_y, b_y)$, in line 7-21 the jammer calculates the step length based on different situations of b_x, b_y.

We show two examples of Alg. 2 in Fig. 2. The blocks represent the first jump pattern of the user. Here $M = 9, P = 11$, the user's start channel is 3 and the step length is 4. The jammer has 3 radios.

Case a is shown above the blocks. From the beginning, the jammer chooses channel 4,5,6. When the user chooses channel 4 (the dark yellow block labeled 4) in the 4-th time slot, the jammer gets the signal and changes to listen on channel 5,6,7. In the 7-th time slot a signal on channel 5 appears. Thus the jammer can calculate the step length of the user, it knows the next channel chosen by the user is 9, and so on.

Case b is shown below the blocks. The jammer chooses channel 2,5,8 from the beginning. In the 3-rd time slot, it gets the signal from channel 2 (the light green block labeled 2(11) which means it is mapped from 11), then it changes to channel 3,5,8 until a signal on channel 8 appears in the 5-th time slot. But the jammer cannot get the step length immediately because it doesn't know whether channel 2 is the actual 2 or it is mapped from 11. If it is mapped from 11, the step length should be 4, and the next channel should be 1, otherwise the step length should be 3 and the next channel should be 2. So in the 6-th time slot the jammer listens on channel 1 and 2 and makes sure that 4 is the real step length.

After getting the step length, another important problem we should solve is the start time of the user, because without knowing the start time the jammer cannot decide when will the user enter the stay pattern. From Lemma 1, we can easily derive that each available channel will appear in each jump pattern. If the jammer starts detecting earlier than the user starts, it can get the step length or some alternative step lengths in the first jump pattern of user, i.e. in P time slots after the user starts. So the jammer can confirm the right step length r before the user entering the stay pattern. After the jammer gets r, it allocates one radio staying on channel r. When it gets 3 consecutive signals on channel r, say at time $t, t+1, t+2$, it knows the user has entered the stay pattern, either t, $t+1$ or $t+2$ is the start time of the stay pattern. Thus the jammer can calculate the alternative start time and start channels of the user, when the stay pattern ends and the next round starts, the jammer can make sure which is right.

4.3 Analysis of Alg. 2

In this section, we analyze the expected time and maximum time of Alg. 2. There are two important lemmas as follows.

Lemma 3. *Given $m \geq n \geq 3$, if we sample n numbers from $\{1, 2...m\}$ without repeating, say the numbers are $a_1 < a_2 < a_3... < a_n$, then the expected value of a_2 and a_3 are $E(a_2) = \frac{2(m+1)}{n+1}$,$E(a_3) = \frac{3(m+1)}{n+1}$.*

Due to the page limits, the proof of the lemma can be found in the full version [3].

Remark 2. Actually we can prove that $E(a_i) = \frac{i \times (m+1)}{n+1}$ for all i in $\{1, 2, 3...n\}$.

Lemma 4. *Given $m > n \geq 1$, If we sample $n+1$ numbers from $\{1, 2...m\}$ without repeating, say they are $S = \{a_1, a_2, a_3...a_{n+1}\}$, let $p = min\{a_1, a_2...a_n\}$, $q = min\{x : x \in S \setminus \{p\}, x > p\}$, then $E(q) = \frac{(m+1)(2n+3)}{(n+1)(n+2)}$.*

Proof. We divide the problem into two cases. First, $a_{n+1} < p$, which means a_{n+1} is the smallest in S, so q is the third smallest number in S, from Theroem 3, $E(q) = \frac{3(m+1)}{n+2}$. Second, $a_{n+1} > p$, in this case, q is the second smallest number in S, $E(q) = \frac{2(m+1)}{n+2}$. So we have:

$$\begin{aligned} E(q) &= E(q|a_{n+1} < p)Pr\{a_{n+1} < p\} + E(q|a_{n+1} > p)Pr\{a_{n+1} > p\} \\ &= \frac{3(m+1)}{n+2}\frac{1}{n+1} + \frac{2(m+1)}{n+2}\frac{n}{n+1} \\ &= \frac{(m+1)(2n+3)}{(n+1)(n+2)} \end{aligned}$$

According to the two lemmas, we can derive the theorem as:

Theorem 2. *By using Alg. 2 for finding the sequence generated by the user, the maximum time is $P - n + 4$, and the expected time is $\frac{(M+1)(2n+3)}{(n+1)(n+2)} + O(1)$. M is the number of channels, P is the smallest prime that $P \geq M$, n is the number of radios the jammer has.*

Proof. From Section 4.2 we know that Alg. 2 ends in at most 2 time slots after detecting two signals, and all the channels appear in the first jump pattern of the sequence generated by the user. So the worst case is that the jammer chooses n channels which appears at the end of the first jump pattern, thus the maximum time is no more than $P - n + 2 + 2 = P - n + 4$.

Then we prove the expected time, we focus only on the first jump pattern of the sequence because Alg. 2 ends in the first jump pattern. Since $P - M$ is much smaller comparing to P and M, we assume $M = P$, then we know the jump pattern is a permutation of $\{1, 2, ..., P\}$. Notice that even though each channel appears on each position of the permutation with equal probability, the permutation is not a completely random permutation. However, since the

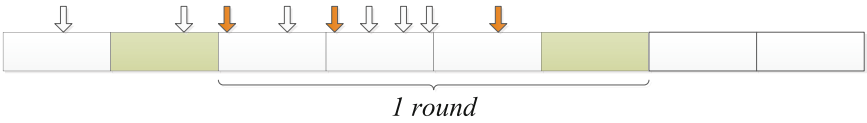


Fig. 3. Example of Alg. 3

jammer chooses n completely random channels to detect, we can just consider the case when the permutation is $1, 2, 3, ..., P$. Suppose the jammer chooses channel set $S = \{a_1, a_2, ..., a_n\}$ to detect, the first signal will appear on channel p where $p = min\{S\}$, then the jammer changes the channel from p to a_{n+1}, now it listens on $S \setminus \{p\} \bigcup \{a_{n+1}\}$, and the second signal should be on channel q where $q = \{x : x = min\{S\}, x > p\}$. We can see that the problem is the same with Lemma 4, so the expected time is $\frac{(M+1)(2n+3)}{(n+1)(n+2)} + O(1)$.

4.4 Asymmetric Situation

Due to the spectrum sensing technique, the user may find some channels occupied by the PUs which means the user cannot access these channels, this is the so-called asymmetric setting. We assume that the jammer knows the available channel set of the user due to the same spectrum sensing technique. In Remark 3 we briefly discuss the situation where the jammer doesn't know the set at all.

Similar to the symmetric setting, the purpose of the jammer is to find the step length and the start time chosen by the user. The main intuition is the same as Alg. 2: the jammer waits on some channels and then calculates the step length along with the start time. The difference is that the jammer may detect some channels that are randomly chosen by the user, this may interfere the jammer and the jammer may get the wrong step length.

Our solution is that the jammer keeps detecting until there are 3 signals, say channel b_1 at time t_1, b_2 at t_2, b_3 at t_3, that satisfy:

$$StepLength(t_1, b_1, t_2, b_2) = StepLength(t_2, b_2, t_3, b_3) = StepLength(t_3, b_3, t_1, b_1)$$

The jammer considers $StepLength(t_1, b_1, t_2, b_2)$ as the step length, and $\{b_1, b_2, b_3\}$ as the channels not randomly chosen by the user. It uses $StepLength(t_1, b_1, t_2, b_2)$ and b_3 for generating the remain sequence. We call $\{b_1, b_2, b_3\}$ "good channels".

Take Fig. 3 as an example, the arrows point to the channels that the jammer gets, the dark yellow arrows point to the "good channels". If they are not chosen randomly by the user and are in the same round as the picture shows, the jammer can get the right step length from the 3 signals. Formally, please refer to Alg. 3. In Alg. 3, the input P, M, n are defined the same way with Alg. 2. Initially, the jammer holds a set B in line 3, when it hears channel b at time t, it adds (t, b) into B. In line 6, the jammer judges all the triplets in B, and if it fails to find the step length, in line 9 it changes the channel from b to another channel and keeps on listening.

Algorithm 3. Multi-Radio Attack Algorithm Under Asymmetric Model

1: **Input:** P, M, n, the available channel set $C = \{c_1, c_2, ..., c_k\}$ of the user;
2: **Output:** the step length r;
3: Set $B = \emptyset$;
4: Sample n channels $A = \{a_1, a_2 ... a_n\}$ without repeating from C;
5: Keep listening on channels in A until the first signal appears, say it is on channel b at time t, $B = B \bigcup \{(t, b)\}$;
6: **if** there exists $(t_x, b_x), (t_y, b_y), (t_z, b_z)$ in B such that $StepLength(t_x, b'_x, t_y, b'_y) = StepLength(t_x, b'_x, t_z, b'_z) = StepLength(t_y, b'_y, t_z, b'_z)$, here $b'_x = b_x$ or $b_x + M$(only when $b_x \leq P - M$), so are b'_y and b'_z **then**
7: Return $StepLength(t_x, b'_x, t_y, b'_y)$;
8: **else**
9: Randomly select another channel $d \in C \setminus A$, $A = A \setminus \{b\} \bigcup \{d\}$, goto Line 5.
10: **end if**

Notice that "good channels" are not always good. Sometime three channels not in the same round or containing random channels can also satisfy the above equation. This means our algorithm cannot guarantee finding the right step length. We discuss why our algorithm still works well in the next subsection.

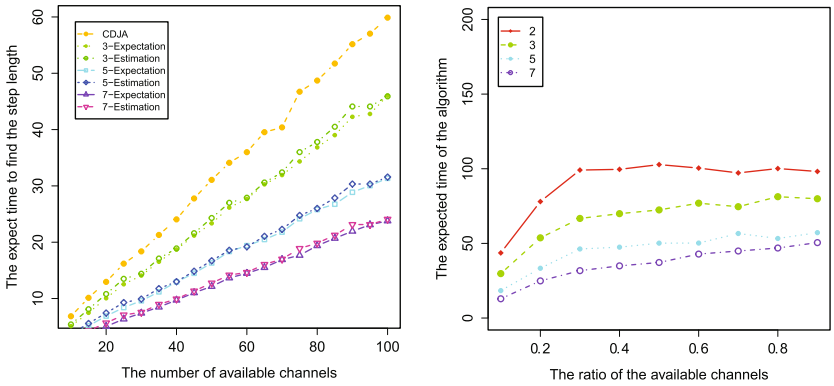
When the jammer gets the step length r, another important problem is the start time and we tackle it similarly as the symmetric situation, where the jammer allocates one radio on channel r to find the time of entering the stay pattern, and then it can know the start time of the user.

Remark 3. When the jammer doesn't know the available channel set of the user, it can still use Alg. 3 to make the attack, by just choosing n random channels to listen. However, this makes the algorithm slower, and we leave more study of this situation to the future work.

4.5 Analysis of Alg. 3

In this section we discuss why Alg. 3 works well even though sometime "good channels" may not be "good". Suppose the number of all channels is M and the number of available channels for the user is λM, P is the smallest prime no less than M, the jammer has n radios and starts detecting earlier than the user starts. In the jump patterns of the first round which last for $3P$ time slots, there are about $3P\lambda$ channels which are not chosen randomly. For each of these channels, the probability of being detected by the jammer is about $n/(P\lambda)$, so on average, the jammer can detect about $3P\lambda \times n/(P\lambda) = 3n$ channels which are not random channels in the jump patterns of the first round. Thus we can expect that Alg. 3 ends in the first round with great probability, which means there is great chance that the jammer can find really good "good channels" in the first round. Our simulation results in the full version [3] shows that, the probability of successful attack in the first round is more than 80% when the ratio of available channels is more than 40%.

A good way to make the calculation of the step length more accurate is to run Alg. 3 multiple times. In each time, the jammer can get a step length r and



(a) Expectation and estimation of the time of Alg. 2

(b) The expected time of running Alg. 3 once

Fig. 4. Simulation Results

it is easy to see that the real step length appears more times than the other false r. Notice that under the asymmetric setting, the rendezvous time is no longer within $4P$ time slots, so the jammer can try to run Alg. 3 several times to get an accurate r. Moreover, our simulation result in Section 5 shows that the average time for Alg. 3 is always within P time slots.

5 Performance Evaluation

In this section we conduct extensive simulations to evaluate our algorithms. We use R language to implement the simulation, in each experiment we get the result as the average of more than 20000 separate runs. Due to page limits, we only show part of the simulations results, you can refer to the full version [3] for all the simulation results.

Fig. 4(a). shows the expected time of Alg. 2. The x-bar represents different numbers of available channels, while the y-bar represents the expected time. We do experiment in case the jammer can access 3, 5 or 7 channels. We also use the CDJA[14] to attack the EJS users. From the figure, we can see that our result is better than CDJA. In each case our estimation which is based on Theorem. 2 is also shown in the figure, they are almost the same as the experiment result. The gap is no more than 2.

Fig. 4(b) shows the expected time of Alg. 3. Here we assume there are totally 100 channels and the jammer starts detecting earlier than both users, the x-bar represents the ratio of available channels of the user. The y-bar represents the expected time. We can see that there is no much difference when the ratio changes, but the number of radios the jammer has is very important. However, even if the jammer can detect only 2 channels each time slot, the expected time is around 100, this means in expectation Alg. 3 can end in the $1st$ pattern of the $1st$ round. It coincides with our analysis in Section 4.5.

6 Conclusion

In this paper, we propose the Multi-Radio Channel Detecting Jamming Attack (MRCDJA) problem where the jammer can listen or block $n \geq 2$ channels simultaneously to prevent rendezvous between the users. We assume the users adopting the Enhanced Jump Stay (EJS)[8] algorithm for rendezvous attempt and we design efficient algorithms to determine the users' channel hopping sequence. For the symmetric users that can use all channels in the channel set, our algorithm is n times faster than the current methods. For asymmetric users, our algorithm can also work well. Finally we conduct extensive simulations for evaluation. In the future, we are to explore efficient attack algorithms when the users are also equipped with multiple radios for rendezvous attempt.

References

1. Draves, R., Pahye, J., Zill, B.: Routing in multi-radio, multi-hop wireless mesh networks. In: MobiCom (2004)
2. Gu, Z., Hua, Q.-S., Wang, Y., Lau, F.C.M.: Nearly Optimal Asynchronous Blind Rendezvous Algorithm for Cognitive Radio Networks. In: SECON (2013)
3. Gao, Y., Gu, Z., Hua, Q.-S., Jin, H.: Multi-Radio Channel Detecting Jamming Attack Against Enhanced Jump-Stay Based Rendezvous in Cognitive Radio Networks. http://grid.hust.edu.cn/qshua/cocoon15_full.pdf
4. Gu, Z., Hua, Q.-S., Dai, W.: Local Sequence Based Rendezvous Algorithms for Cognitive Radio Networks. In: SECON (2014)
5. Gu, Z., Hua, Q.-S., Wang, Y., Lau, F.C.M.: Oblivious Rendezvous in Cognitive Radio Networks. In: Halldórsson, M.M. (ed.) SIROCCO 2014. LNCS, vol. 8576, pp. 165–179. Springer, Heidelberg (2014)
6. Gu, Z., Hua, Q.-S., Dai, W.: Fully Distributed Algorithms for Blind Rendezvous in Cognitive Radio Networks. In: MOBIHOC (2014)
7. Li, G., Gu, Z., Lin, X., Pu, H., Hua, Q.-S.: Deterministic Distributed Rendezvous Algorithms for Multi-Radio Cognitive Radio Networks. In: MSWiM (2014)
8. Lin, Z., Liu, H., Chu, X., Leung, Y.-W.: Enhanced Jump-Stay Rendezvous Algorithm for Cognitive Radio Networks. IEEE Communications Letters (2013)
9. Liu, H., Lin, Z., Chu, X., Leung, Y.-W.: Jump-Stay Rendezvous Algorithm for Cognitive Radio Networks. IEEE Transactions on Parallel and Distributed Systems **23**(10), 1867–1881 (2012)
10. Liu, H., Lin, Z., Chu, X., Leung, Y.-W.: Taxonomy and Challenges of Rendezvous Algorithms in Cognitive Radio Networks. In: ICNC (2012)
11. Liu, H., Lin, Z., Chu, X., Leung, Y.-W.: Jump-Stay Based Channel-Hopping Algorithm with Guaranteed Rendezvous for Cognitive Radio Networks. In: INFOCOM (2011)
12. Paul, R., Jembre, Y.-Z., Choi, Y.-J.: Multi-interface rendezvous in self-organizing cognitive radio networks. In: DySPAN (2014)
13. Yang, D., Shin, J., Kim, C.: Deterministic Rendezvous Scheme in Multichannel Access Networks. Electronics Letters (2010)
14. Oh, Y.-H., Thuente, D.-J.: Channel Detecting Jamming Attacks Against Jump-Stay Based Channel Hopping Rendezvous Algorithm for Cognitive Radio Networks. In: ICCCN (2013)
15. Zhao, Q., Sadler, B.-M.: A Survey of Dynamic Spectrum Acess (signalprocessing, networking and regulatory policy). IEEE Signal Processing Magazine (2007)