Hybrid Rendezvous Algorithm against Jamming Attacks for Cognitive Radio Networks

Gabriel Anquetin and Yongchul Kim
Korean Military Academy, Seoul, South Korea
Email: gabriel.anquetin@st-cyr.terre-net.defense.gouv.fr; kyc6454@kma.ac.kr

Abstract —Rendezvous is a fundamental step in communicating with cognitive radio networks. Existing rendezvous algorithms using the *Channel Hopping* (CH) method perform well to guarantee a rendezvous between *Secondary Users* (SUs) quickly. Jump-Stay based algorithms can guarantee rendezvous for *asynchronous* and *asymmetric* models, but these algorithms are particularly vulnerable to jamming attacks, which reduce rendezvous probability significantly from 100% to 10%. In this paper, we will introduce the *Hybrid Rendezvous Algorithm* (HRA), combining the positive benefits of existing algorithms in order to perform in both jamming and non-jamming situations.

Index Terms—Cognitive radio networks, rendezvous algorithms, jamming attacks

I. Introduction

Massive use of wireless devices using unlicensed spectrum created congestion for unlicensed users. Spectrum sharing and regulation is an issue in most countries and it is far from being resolved even if solutions exist as Bhatarrai *et al.* prove it [1]. However, the *Opportunistic Spectrum Access* (OSA) technique allowed by cognitive radios is one of the solutions. OSA assumes that the network has a hierarchical structure, the licensed users, called *Primary Users* (PUs), share their network with unlicensed users, called *Secondary Users* (SUs), equipped with cognitive radios. These cognitive radios allow SUs to sense the spectrum and find idle channels and use them as opportunities to rendezvous with other SUs, unlike conventional radio which senses to detect noise only. OSA's main rule is that SUs must not interfere with PUs.

To communicate and exchange data, SUs must rendezvous with each other on the same channel. However, it is not simple for many reasons. The first is that SUs might not be aware of the presence of each other. Moreover, SUs often have a different available channel set, so that it is possible that SUs sense hundreds of available channels but only one is common to these SUs. Therefore, to do the first handshake, SUs must use a rendezvous algorithm.

Over the last decade, many rendezvous algorithms have emerged [2]. The main method used by every rendezvous algorithm is *Channel Hopping* (CH) where users assume that the network is time-slotted so that they jump among channels at every time-slot. These channels must be available so that SUs do not interfere with PUs. A rendezvous is said to be achieved if SUs jump on the same channel in the same time-slot.

Researchers have computed rendezvous algorithms using different tools. To begin with, some algorithms were working with servers called *centralized controller*. But this tool, initially used to facilitate rendezvous, has the opposite effect. Centralized systems are lack scalability, flexibility and versatility because a single problem on the server affects every user and the network is very vulnerable to jamming attacks. Thus, centralized controllers are not used in rendezvous algorithms, we only use *decentralized systems*. To create the best rendezvous algorithm, researchers also designed algorithms using *Common Control Channel* (CCC). The purpose of this method is to have a channel known by every user which allows them to exchange information to facilitate rendezvous thereafter. Unfortunately, a rendezvous algorithm using that kind of channel is not feasible because this CCC must be available for every user on the network, thus restraining this method to small areas. Moreover, local CCC for small areas might be designed, but will create congestion on these channels, and this congestion is what we want to avoid on unlicensed channels, so it is not feasible in practice. Therefore, the model we will use is called blind rendezvous algorithm, which means decentralized without CCC.

There are different cases a rendezvous algorithm can encounter, depending on network users or what the algorithm requires. Time-synchronization is one of these criteria. Some algorithms require time-synchronization between users to be able to establish rendezvous. However, in practice there is always a time-offset between users so it is very restrictive to use a synchronous algorithm. Thus, asynchronous algorithms are the most general and feasible alternatives. Another main criterion is symmetry. A model is said to be symmetric if every user has the same common available channels. This situation can occur in a small area. However, the most common case is the asymmetric one, in which users have different available channel sets.

In this study, we want to design a blind rendezvous algorithm working in any network environment, so we design the *Hybrid Rendezvous Algorithm* (HRA) to be able to perform with asynchronous and asymmetric schemes. Also, we want our algorithm to guarantee

Manuscript received December 16, 2017; revised April 23, 2018.
doi:10.12720/jcm.13.4.169-174

rendezvous without using any information from user (e.g., ID) or without using any pre-assigned role which could be restrictive. ID is mandatory in *Alternate Hop-and-Wait* (AHW) and *Ring-Walk* (RW) algorithms [3], [4]. This additional information allows these algorithms to generate parameters to rendezvous SUs as quickly as possible. As introduced by Guerra *et al.* and Oh *et al.* respectively, *Full Diversity Channel Hopping – Role-Based* (FDCH-RB) and *Role-based Channel Rendezvous* (RCR) algorithms are efficient algorithms but require pre-assigned roles of transmitter and receiver, which make these algorithms more restrictive [5], [6].

Then, our last constraint is to be jamming-resistant. Jamming attacks are a bane for rendezvous algorithms because rendezvous probability drops significantly, making any rendezvous virtually impossible to establish.

The main metric essential in rendezvous algorithm is *Time-To-Rendezvous* (TTR). This is the number of time-slots required to achieve a rendezvous between SUs. *Maximum Time-To-Rendezvous* (MTTR) is defined as the TTR in the worst scenario. A rendezvous algorithm can guarantee a rendezvous if it has a finite MTTR. An algorithm which selects randomly every channel where the SUs will hop has no finite MTTR because, in the worst-case rendezvous will not happen.

Expected Time-To-Rendezvous (ETTR) is used because an algorithm with a bad MTTR can be very effective with a good ETTR. Thus, the MTTR shows the theoretical limit of an algorithm whereas the ETTR can highlight whether a rendezvous algorithm is good in practice. The other important metric for our algorithm is the rendezvous probability because we will consider jamming attacks, so, even if a rendezvous algorithm can guarantee rendezvous between users, jamming attacks can completely thwart this scenario, because rendezvous probability has dropped to 10% for example.

We illustrated a case of channel hopping sequence of an asymmetric and asynchronous case in Fig. 1.

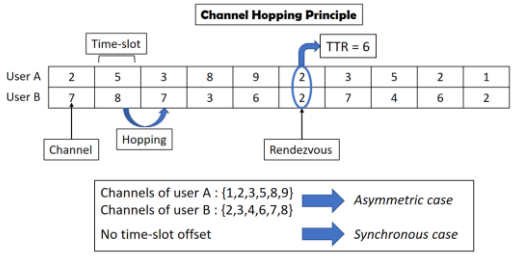

Fig. 1. Channel hopping principle

In the second part of this article, we discuss our selected algorithms and their respective jamming attacks, while in the third part we introduce the HRA, a solution to these jamming attacks, and its performance analysis.

II. Selected Algorithms and Jammming Attacks

To introduce our algorithm, we need to explain two existing algorithms which will be combined to create the HRA. We will use the following model:

-Let $M>1$ be the number of non-overlapping channels available for users

-Let P be the smallest prime number greater that M

-Let $N>1$ be the number of SUs.

-Let C be the whole available channel set, $C= \{c_1, c_2, \ldots, c_M\}$

-Let C_k be the available channel set for the kth user

-A channel is said to be available if SUs can join it without causing any interference to PUs

First, we will introduce the *Enhanced Jump-Stay* (EJS) rendezvous algorithm, an efficient algorithm with broad application and we will discuss the *Full Random* (FR) rendezvous algorithm. Then we will show the existing jamming attacks perturbing rendezvous algorithms.

A. Enhanced Jump-Stay and Full Random algorithms

A reliable rendezvous algorithm performing for asynchronous and asymmetric case is the *Enhanced Jump-Stay* (EJS) introduced by Lin *et al*, [7]. This algorithm uses the *Jump-Stay* (JS) method introduced by the same authors [8]. The JS method creates CH sequences with jump patterns and stay patterns. During a jump pattern, SUs hop among channels during P time-slots, starting from a channel whose index is i_0 (included in $[1,P]$). These SUs jump with a step-length r (included in $[1, M]$). A stay pattern is designed to stay P time-slots on the channel indexed r. The EJS rendezvous algorithm makes SUs jump 3 time-slots in a row and stay one time-slot on the channel r. Fig. 2 illustrates the EJS channel hopping sequence.

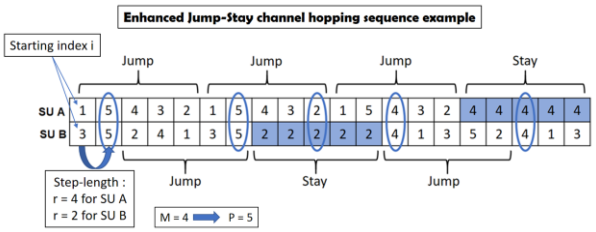

Fig. 2. EJS channel hopping example

This example is in a synchronous and symmetric case, but it also works in asynchronous and asymmetric cases. When a user jumps to a channel other than the available indexed channels, the hopping function remaps this channel index in available channels using modulo operation. Thus, for the channel j, if $j>M$ (j out of the available indexes), then $j=((j-1)\%M)+1$. This remapping operation is computed by the EJSHopping function in Fig. 3.

Line 4 shows that the jump pattern lasts $3P$ time-slots while line 5 is about the P time-slots for the stay pattern. This function is continuously called by the following EJS Algorithm in Fig. 4. The main difference between JS and EJS is that a round lasts $3P$ time-slots ($2P$ jump and P stay) in JS whereas a round lasts $4P$ time-slots in EJS.

We can note that the starting index i is switching at every round in round-robin fashion. Moreover, in the asymmetric scheme, a channel can be unavailable for a user so a replacement step is at line 10. It is different

from the remapping step in the EJSHopping function because here it is about channels and not index.

Function EJSHopping

```
1: Input M,P,r,i,t
2: Output channel index j
3: t=t%4P;                          //1 round = 4P time-slots
4: if (t<3P) j=((i+t*r-1)%P)+1   //jump pattern
5: else j=r;                         //stay pattern
6: if (j>M) j=((j-1)%M)+1        //remapping
7: return j;
```

Fig. 3. Pseudo code of EJSHopping function

EJS Algorithm

```
1: Input M, Ck                      //for kth user
2: P=the smallest prime number greater than M
3: r=step-length randomly selected from [1,M]
4: i0=starting index randomly selected from [1,P]
5: t=0;
6: while (not rendezvous)
7:   n= ⌊t/(4P)⌋                  // 1 round = 4P time-slots
8:   i=((i0+n-1)%P)+1        // switching i0 every round
9:   j= EJSHopping(M,P,r, i,t)  // hopping function
10:  if (Cj ∉Ck)      // out of available channel set
     replace Cj by the ((j-1)%|Ck|+1)th channel in Ck
//replacement for asymmetric case
11:  t=t+1;
12:  attempt rendezvous on channel Cj;
```

Fig. 4. Pseudo code of EJS algorithm

The experiment shows that, concerning TTR, EJS is efficient. Under symmetric model, its MTTR is $O(P)$, while under asymmetric model, its MTTR is $O(P^2)$. The fact that EJS has rounds of $4P$ time-slots instead of $3P$ for JS has important consequences because the MTTR of asymmetric JS is $O(P^3)$. Fig. 5 is the graph of the test between JS, EJS and the full random algorithm which is explained in the following paragraph.

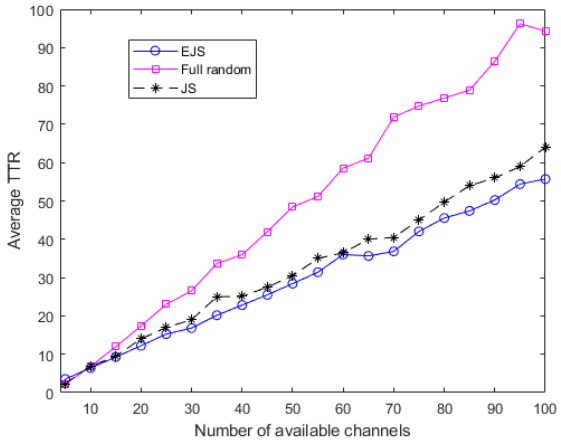

Fig. 5. Average TTR comparison between EJS, FR and JS rendezvous algorithms in symmetric scheme

The *Full Random* algorithm (FR) is a very simple algorithm. It randomly selects channels for users. This randomness has pros and cons. This algorithm has an infinite MTTR because in the theoretical worst scenario SUs will never meet each other on the same channel, but its ETTR is not that bad as we can see on Fig. 5. Moreover, this rendezvous algorithm can be used for asynchronous and asymmetric schemes, so it is a flexible one. Yet, the best characteristic of the FR rendezvous algorithm is its jamming resistance. As shown in the next part, randomness is the best way to protect algorithms from jamming attacks.

B. *Jamming Attacks*

In the literature, many conventional algorithms like EJS can guarantee a rendezvous very quickly without requiring any additional information. However, these conventional algorithms use repetitive channel hopping sequences that make them particularly vulnerable to jamming attacks. The basic strategy of a jamming attack is to find what is repetitive in the code in order to imitate it. Once the jammer has found the SUs' sequences, he will jump among channels using the step-length of the SU and begin dynamic jamming. Jamming attacks affect the rendezvous probability and mitigate it such that rendezvous is often impossible between SUs.

Proposed by Oh *et al*, the *Channel Detecting Jamming Attack* (CDJA) has been designed to attack the JS rendezvous algorithm [9]. This kind of jamming attacks uses the step-length r of the JS channel hopping sequence and the starting index i to compute the sequence to come. This method requires only one listening channel from the jammer, even if two listening channels give better results. You can refer to Figs. 6 and 7 to have examples of jamming attacks with one and two listening channels.

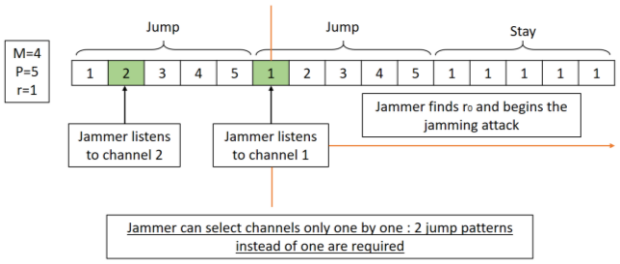

Fig. 6. Example of step-length research with one listening channel

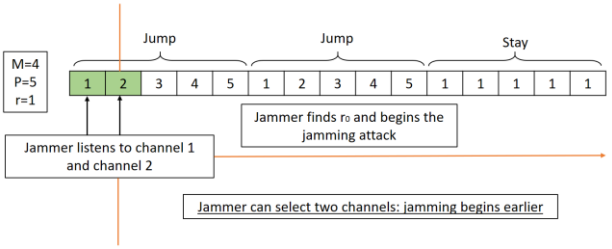

Fig. 7. Example of step-length research with two listening channel

To find the step-length of a user, the jammer must use the following formula which gives the channel at time t: $c_i = (i_0 + t_i \times r_0 - 1) \% P+1$. At the time-slot t_1, the jammer is listening to the channel c_1 and at the time-slot t_2, the jammer is listening to the channel c_2. So,

$$c_1 = (i_0 + t_1 \times r_0 - 1)\%P + 1 \quad (1)$$

$$c_2 = (i_0 + t_2 \times r_0 - 1)\%P + 1 \quad (2)$$

(2-1) gives:

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

Thus,

$$r_0 = (P \times k + (c_2 - c_1))/(t_2 - t_1)$$

The jammer uses r_0 to find i_0 with (1) or (2) and is able to prevent the rendezvous from being established. As a fact, the jammer only has to use (3).

$$C_{next} = (C_{last} + r_0)\%P + 1 \quad (3)$$

However, this case is the simplest one because it is synchronous scheme, which means that the beginning of the pattern is known by the jammer. But in asynchronous case, he cannot know it so i_0 is more difficult to compute. As explained in Fig. 8, the step-length is still easily found so the jammer can quickly begin a naive jamming process, but without i_0 this jamming is not perfect. The following figure explains the asynchronous CDJA.

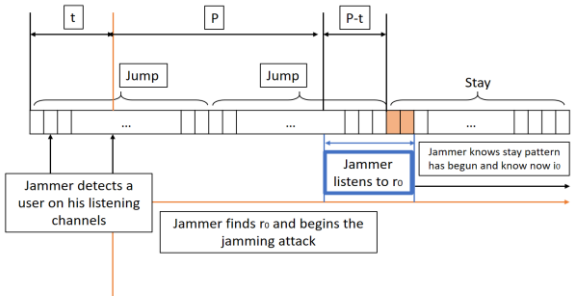

Fig. 8. Example of asynchronous research of step-length and starting index with two listening channels

CDJA is designed to jam JS algorithm. Oh *et al*, also proposed two jamming attacks against EJS [9]. These jamming attacks are the *Symmetric Channel Detecting Jamming* (SCDJ) and the *Asymmetric Channel Detecting Jamming* (ACDJ). Symmetric and asymmetric cases have been distinguished because of the replacement step in the asymmetric EJS which is not used by the symmetric EJS and so perturb SCDJ.

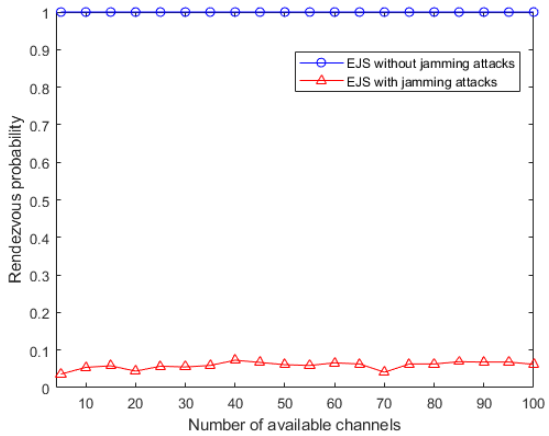

Fig. 9. Rendezvous probability for symmetric EJS with and without SCDJ attacks

Consequences of jamming attacks are in Fig. 9, in which the rendezvous probability of the EJS rendezvous algorithm significantly drops from 100% (guaranteed rendezvous) to 10% (rendezvous is compromised).

III. Hybrid Rendezvous Algorithm

A. HRA Theory

The HRA is designed to rendezvous SUs without jamming attacks with 100% probability, and be able to protect the rendezvous probability from the huge drop under jamming attacks. This algorithm works in the most general case, that is to say asynchronous, asymmetric, without pre-assigned roles or additional information to provide like ID or network size. In order to create that algorithm, we used the two selected rendezvous algorithm introduced in part II, EJS and FR algorithms. These two rendezvous algorithms have pros and cons, we combined them so that HRA will smartly use the pros of every algorithm. The main idea is that EJS is very efficient without jamming attacks, so it will be the base of HRA. Yet, FR cannot be jammed due to its random aspect and its ETTR is decent. What is interesting about FR is that its ETTR is the same in both jamming and non-jamming situations, so without jamming attacks this algorithm is not recommended but in the other case it is a very good choice. Therefore, HRA continuously switches from EJS to FR every MTTR of EJS, creating a channel hopping sequence made of EJS and FR rounds. This is illustrated in Fig. 10.

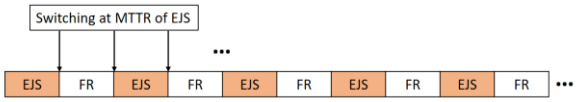

Fig. 10. HRA channel hopping sequence principle

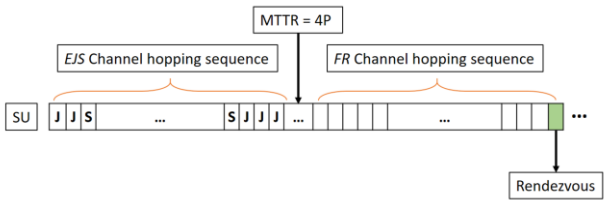

Fig. 11. HRA in symmetric mod

The switching between EJS and FR is crucial because if it is too early, EJS cannot be efficient even if there is no jamming attack, but if it is too late, some TTR is lost. Symmetric cases and asymmetric cases have to be distinguished. In a symmetric case, illustrated in Fig. 11, the MTTR is *4P* so rendezvous is theoretically achieved in at most *4P* time-slots and *P* is known by the concerned SU, so the switching is computed to happen at *4P*. Indeed, if the TTR is greater than *4P* for symmetric EJS, it means that jammers are on the network so the switching to FR is mandatory. Concerning asymmetric case, the MTTR is $4P^2$-*4P*-*4PG* with *G* the number of common available channels between SUs. Yet, *G* is not known by SUs, so

we cannot switch from EJS to FR at the right timing and the other problem is that EJS is less efficient than FR in asymmetric scheme, so we let the switching at *4P*, as if it was the symmetric case.

Theoretically, HRA cannot guarantee rendezvous in an asynchronous case, because an SU could perform an EJS round during the FR round of the other SU as shown in Fig. 12. However, the performance analysis in the next part shows that it is not a problem in practice.

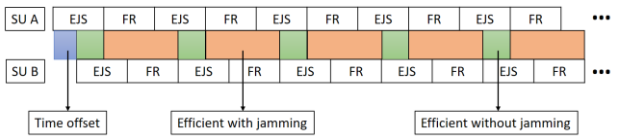

Fig. 12. Case of HRA in asynchronous scheme

B. HRA Performance Analysis

HRA has been computed in MATLAB and we tested it with and without jamming attacks. With jamming attacks, in Fig. 13, the experiment proves that the switching EJS/FR is effective: rendezvous is almost guaranteed for every number of available channels. Instead of dropping to 10% like EJS, HRA is reaching more than 95% of rendezvous probability during jamming attacks. When there are few channels, jamming is impossible to avoid because jammers are listening to two channels, so in the case of 5 available channels probability cannot be at 100%. HRA rendezvous probability is also better than FR alone, particularly when there are only few channels. For example, for 10 channels, this probability is 80% for FR and more than 90% for HRA, which is very good. We also tested TTR for HRA with and without jamming attacks, in Fig. 14 and Fig. 15 respectively.

We note that, in Fig. 15, HRA's TTR is a bit higher than EJS whereas it also performs EJS without any jamming attacks. Indeed, as it was highlighted in the theory and in Fig. 12, the asynchronous case does not allow SUs to perform EJS at the same time-slots, resulting in EJS/FR and FR/FR common time-slots, which are less efficient than EJS/EJS. However, Fig. 15 shows that HRA remains better than FR for TTR.

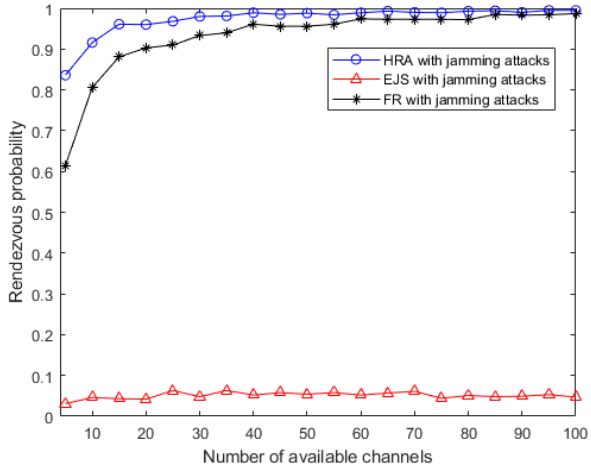

Fig. 13. Rendezvous probability comparison between HRA, EJS and FR algorithms with SCDJ in symmetric scenario

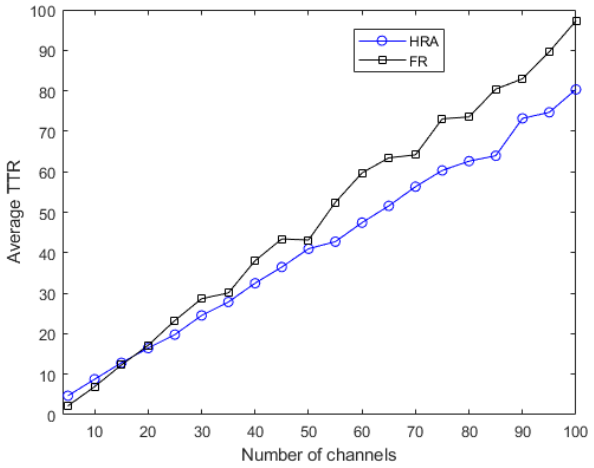

Fig. 14. Average TTR comparison between HRA and FR algorithms with jamming attacks in symmetric scenario

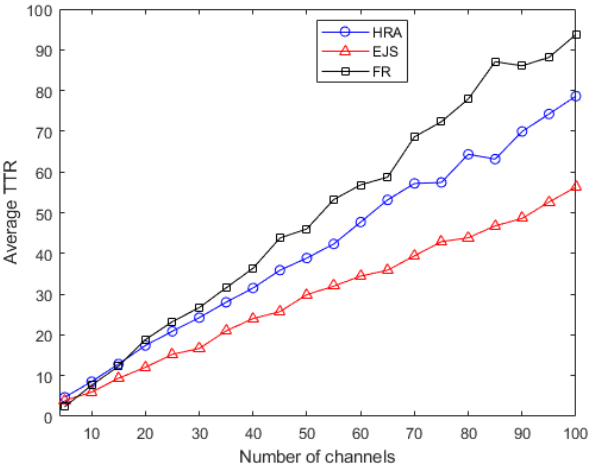

Fig. 15. Average TTR comparison between HRA, EJS and FR algorithms without SCDJ in symmetric scenario

IV. Conclusion

In this paper, the Hybrid Rendezvous Algorithm is proposed. It is intended to rendezvous SUs on cognitive radio networks in the most general situation. We combine the positive aspects of two existing rendezvous algorithms, the Enhanced Jump-Stay and the Full Random, in order to create the HRA. This rendezvous algorithm is jamming resistant, but also works in asymmetric and asynchronous schemes without requiring any pre-assigned roles or additional information. Our numerical results showed that HRA is more efficient than FR alone to solve the jamming attacks problem, with nearly 100% of rendezvous probability with jamming attacks. Its ETTR is also better than FR in every case, and is close to EJS in the non-jamming case.

References

[1] S. Bhattarai, J. M. J. Park, B. Gao, K. Bian, and W. Lehr, "An overview of dynamic spectrum sharing: Ongoing initiatives, challenges, and a roadmap for future research," *IEEE Transactions on Cognitive Communications and Networking*, vol. 2, no. 2, pp. 110-128, June 2016.

[2] H. Liu, Z. Lin, X. Chu, and Y. W. Leung, "Taxonomy and challenges of rendezvous algorithms in cognitive radio networks," in *Proc. ICNC*, 2012, pp. 645–649.
[3] I. Chuang, H. Y. Wu, K. R. Lee, and Y. H. Kuo, "Alternate hop-and- wait channel rendezvous method for cognitive radio networks," in *Proc. IEEE INFOCOM*, 2013.
[4] H. Liu, Z.Y. Lin, X. W. Chu, and Y. W. Leung, "Ring-walk based channel hopping algorithms with guaranteed rendezvous for cognitive radio networks," in *Proc. 2nd IEEE International Workshop on Wireless Sensor, Actuator and Robot Networks*, Dec, 2010.
[5] Y. Oh and D. J. Thuente, "Channel detecting jamming attacks against jump-stay based channel hopping rendezvous algorithms for cognitive radio networks," in *Proc. International Conference on Communications and Networks*, July 2013.
[6] E. O. Guerra, V. A. Reguera, R. D. Souza, E. G. Fernandez, and M. E. Pellenz, "Systematic construction of common channel hopping rendezvous strategies in cognitive radio networks," *Journal on Wireless Communication and Networking*, 2015.
[7] Z. Lin, H. Liu, X. Chu, and Y. Leung, "Enhanced jump-stay rendezvous algorithm for cognitive radio networks," *Communications Letters*, vol. 99, pp. 1–4, 2013.
[8] Z. Lin, H. Liu, X. Chu and Y. W. Leung, "Jump-stay based channel hopping algorithm with guaranteed rendezvous for cognitive radio networks," in *Proc. IEEE INFOCOM*, 2011.
[9] Y. Oh and D. J. Thuente, "Jamming and advanced modular-based blind rendezvous algorithms for cognitive radio networks," in *Proc. 17th International Symposium on A World of Wireless, Mobile and Multimedia Networks*, 2016.

Gabriel Anquetin was born in Rouen, Normandy, France, in 1994. Second Lieutenant at the French Military Academy, he is specialized in electrical engineering. He is currently pursuing the master degree.

Yongchul Kim received a B.E. from Korea Military Academy in 1998 and an M.S. from the University of Surrey (Surrey, UK) in 2001. He received a Ph.D. degree in Electrical and Computer Engineering from the North Carolina State University (Raleigh, USA). He is currently employed as an associate professor in the Department of Electrical Engineering at Korea Military Academy.