Mobile Information Systems 9 (2013) 281–294
DOI 10.3233/MIS-130163
IOS Press

The evolutionary random interval fingerprint for a more secure wireless communication

Kuo-Kun Tseng[a], Jeng-Shyang Pan[a], Wei Wei[a] and Hui-Huang Hsu[b,*]
[a]*Department of Computer Science and Technology, Harbin Institute of Technology, Shenzhen Graduate School, Guangdong, China*
[b]*Department of Computer Science and Information Engineering, Tamkang University, Taipei, Taiwan*

Abstract. In this paper, we propose a novel evolutionary Random Interval Fingerprint (RIF) for active RFID and ZigBee systems. This new approach can enable more secure multi-party communication since, if the wireless packets are forged by another wireless communication party, the interval fingerprint can provide another way to detect the spoofing packet. Moreover, the random evolutionary algorithms, both genetic and memetic, are also proposed as a means to generate the random interval fingerprint. Compared to the conventional random generator, our approach is flexible in generating uniform random and long cycle numbers, and more robust for the anti-cracking. It is difficult for the forged party to produce the fake random intervals. Finally, we provide an application example, a completed work survey, pseudo-code and analysis result to prove that our concept is feasible for the Wireless communication.

Keywords: Random Interval, evolutionary algorithm, genetic algorithm, memetic algorithm, wireless communication

1. Introduction

1.1. The concept of random interval fingerprint in pervasive communication

With the local area wireless communication having become increasingly popular, Wi-Fi, ZigBee, Bluetooth, RFID and relevant wireless networks technologies [1] have become more common in our daily lives. Security issues [2] sometimes viewed as a standalone component, however, must be emphasized, since systems designed without security can become weakness. Security is one of the most important issues in wireless communication given that data are transmitted through the air. Data can be sniffed or modified by those people who have knowledge of the wireless techniques. Although multiple encryption approaches can encrypt your data over the air, considering the complexity and heavy cost of encryption operations, randomizing transmission interval can be an optimum solution. For example, if the transmission interval is not the same as the default packet intervals, the receiver can detect the trans-

*Corresponding auhtor: Hui-Huang Hsu, Department of Computer Science and Information Engineering, Tamkang University, Taipei, Taiwan. E-mail: huihuanghsu@gmail.com.

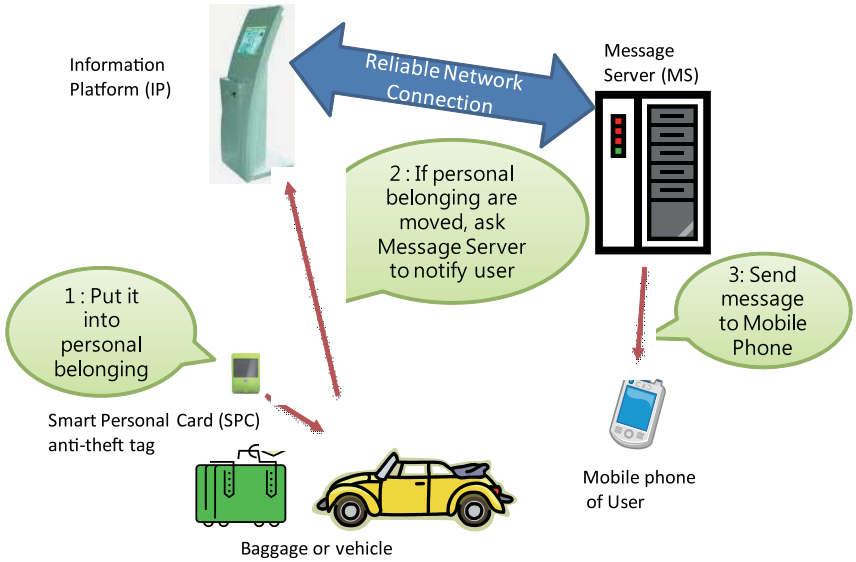

Fig. 1. Application of random interval fingerprint.

mission error or forged data. Therefore, we propose a new research direction, namely Random Interval Fingerprint (RIF), to assist the security of wireless communication.

1.2. The application example of RIF

We give an application of RIF which, as shown in Fig. 1, allows a more secure wireless commutation that consists of a wireless local area network – Smart Personal Card (SPC) with active RFID, mobile phone, Information Platform (IP) and Message Server (MS) to develop an anti-theft system with the concept of fixed wireless network. This application can be widely used for personal belongings or automobiles, such as parking a car or leaving luggage in a public area. Users should firstly put their SPC inside the luggage or car and then activate the anti-theft mode.

When the car or luggage is moved, SPC will notify the IP with a warning message to MS. After MS receives it, it will send out the warning message to the users' mobile phones. Besides, SPC will send a regular beacon to the information platform to report its status. Signal interference can be prevented by this intensive beacon signal and a more reliable communication can be achieved. The reason why is that active RFID/ZigBee has lower power consumption than mobile phone communication, and is connected to a reliable wire network connection, such as Ethernet for communication between IP and MS. Compared to the existing GPRS anti-theft systems, this approach is more feasible.

In this application, SPC needs to send beacons back to the Information Platform. In addition to conventional encryption, if the RIF is employed on this application, it can help the Information Platform to indentify users in order to enhance their communication security.

1.3. Why we apply evolutionary approach for RIF

For the generation of RIF, we propose a new evolutionary approach. This provides a new way to generate the scope of the adjacent interval of random fingerprint. It starts with the external seed having some primary random numbers, and then uses genetic/memetic algorithms to perform the digital crossover, mutation, and finally comply with the specified scope with a series of random numbers.

Our proposed evolutionary generator is a nonlinear random generator and, compared to conventional linear random number generators such as LSFR, it is not easy to track and crack. So, this method can be applied to prevent eavesdropping and forgery packets. Moreover, our proposed evolutionary generator helps avoid packet collisions since it is flexible in providing the random sequence with uniform distribution. In comparison to the later related work study, our proposed approach and its innovative features should be a unique design.

The sections below are organized as follows: Section 2 involves related work whereas our proposed design and evaluation are described in Sections 3 and 4 respectively. Finally, we state our conclusion in Section 5.

2. Related work

For this section, we have made further investigation on some other topics, relevant to random intervals, and methods of random sequence production, classification for random generator, and genetic/memetic random works.

Firstly, we study the random interval of packet related investigations. So far, we haven't found similar research in this area. The random interval has appeared to research on mathematics [3]. In addition, we found some application of the random interval, they are more common in the applications of applied operations research [4,5], avoided packet collisions in the network [6], and studied the analysis of the random interval in the attacked distribution network [7]. However, as we mentioned, the random interval as the network fingerprint concept has not yet been found in this research area.

Also, in the second part of the survey and because our proposed method had to generate the sequence of random intervals, we did further investigation on related applications and two classifications for the random sequence generation. Traditionally, random sequence has many applications, such as on games, the lottery and articles mentioned in biological genetic [8], security encryption [9], communication coding [10] and biological genetic study; they are DNA or RNA [11] series. In the survey, we should be the first to apply the random sequence in the network interval generation as it has not been found in other similar research.

In order to design our work, we also made some pre-study on some related work such as following issues, binary or decimal, pseudo-random or real random. There are two common random sequences in the classification: the numerical binary random sequence [12] and the decimal random sequence [13]. Because interval is a decimal value, our method should be a decimal-based random sequence.

Another classification for random sequence generation is related to the distinction between the pseudo-random sequence and the real random sequence [14]. Since our random intervals are generated on a server platform – a common personal computer- and in order to have a wide range of applications, we will use pseudo-random sequence to design our approach.

A pseudorandom generator is also known as a deterministic random generator. The sequence is not truly random but it is rather determined by a small set of initial values. Common classes of these algorithms are linear random generators, such as linear congruential generators (LCG), multiply-with-carry (MWC), Lagged Fibonacci generators and linear feedback shift registers (LFSR) and uniform distribution generators [15]. The weakness of the linear random generators is that their random numbers can be tracked and predicted. For instance, LFSR- a popular pseudo-random generator which will be used for later evaluation of our proposed approach- is characterized by a shift register with a linear function of its previous state; the new inserted bit of shift register is given by the XOR of some bits of the shift register.

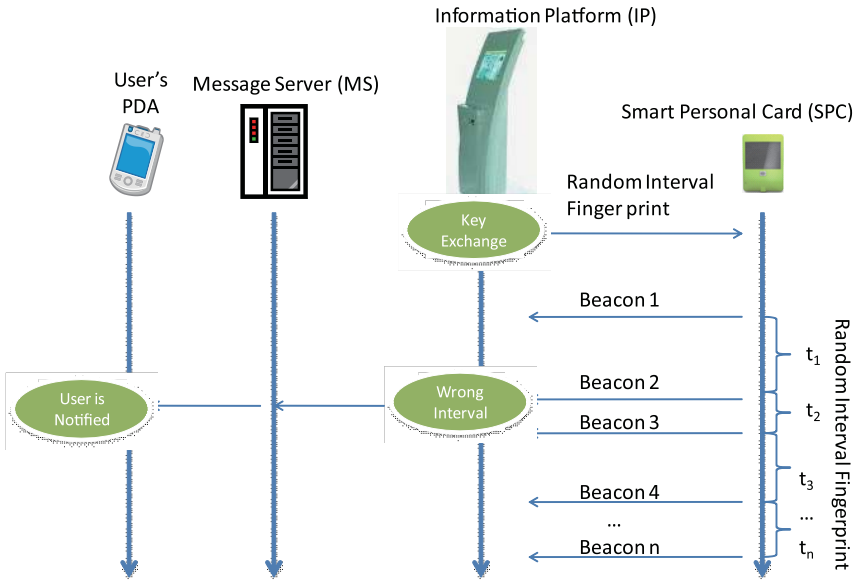

Fig. 2. Concept of random interval fingerprint.

LFSR can be implemented on hardware and software. Since the register has a finite number of possible states, it must eventually enter a repeating cycle.

Our proposed approach is a nonlinear random generator, which is related to evolutionary population-based metaheuristic algorithms that use the genetic/memetic mechanisms inspired by biological evolution, reproduction, mutation, recombination, and selection. Our approach is based on such algorithms as it is the evolutionary algorithms that create the genetic random number difficultly of tracking and cracking. Genetic and memetic algorithms are common for optimization problems [16,17]. According to our survey, the most related papers to mention about genetic and memetic algorithms with random number generation are a few and they usually include research on improving randomization of numbers for genetic and memetic algorithms [18,19]. Thus, the specific genetic or memetic algorithms for random number generation have not been found yet. We should be the first to use these algorithms to generate random numbering in wireless applications.

In the algorithm applications compared [20,21] to our research, a memetic algorithm (MA) is an extension of the traditional genetic algorithm, which uses a local search technique to reduce the convergence time of the genetic algorithm (GA). In this paper, in addition to the genetic algorithm, we apply local search algorithm of memetic algorithm to reduce the time of random number generation.

3. Proposed design

3.1. How does random interval fingerprint work?

As we mentioned above, RIF technology is applied in the fixed wireless security. As shown in Fig. 2, in the beginning, the user brings the SPC to IP to buy the anti-theft service. Then the setup operation and the key exchange between SPC and IP are performed. The key exchange contains the encryption key and a key of RIF. This RIF key is generated by the information platform. After SPC gets the key, it sends

a sequence of beacons with intervals $\{t_1, t_2 \ldots \ \mathrm{t}_n\}$ in RIF behavior to notify IP, when the information platform detecting the beacons is not in RIF mode. IP will inform the user through the MS when an exception condition occurs.

To successfully produce the RIF, it should consist of Random Order Switching (ROS) and Genetic Random Sequence (GRS) to achieve robust anti-cracking. Their detailed description follows.

3.2. Random order switching (ROS)

To make more changes in the random interval so as to make it more difficult to be guessed by a third party, we plan to let SPC obtain a set of random sequences from the IP. SPC will perform the ROS to select a new random sequence in the new round.

We can define this proposed solution formally as follows: F representing a RIF matrix, which is a collection of the random sequences $\mathrm{t}_{n,m}$.

The entire random interval fingerprint of SPC is defined as follows:

$$\mathrm{F} = \begin{bmatrix} t_{1,1} & t_{1,2} & \ldots & t_{1,m} \\ t_{2,1} & t_{2,2} & \ldots & t_{2,m} \\ \ldots & \ldots & \ldots & \ldots \\ t_{n,1} & t_{n,2} & \ldots & t_{n,m} \end{bmatrix} \tag{1}$$

Where n is the length of random sequence, and m is the maximum number of supported smart personal cards in information platform. Besides, n also represents the value of the change cycle of random sequences.

For the exchange of random sequence, we initially used a ROS method in which the random intervals were exchanged in random order and in each round. The exchange system is also equivalent to the number of rounds m value.

$$\mathrm{F} = \begin{bmatrix} t_{1,1} & t_{1,2} & \ldots & t_{1,m} \\ t_{2,1} & t_{2,2} & \ldots & t_{2,m} \\ \ldots & \ldots & \ldots & \ldots \\ t_{n,1} & t_{n,2} & \ldots & t_{n,m} \end{bmatrix} \tag{2}$$

Round 1 Round 2 Round m

Therefore, the time T_i is required for each round, where i is the number of rounds, and in each round T_i is a random sequence, where m is the maximum value, so t represents the sending point in each round, and t cannot be duplicated in each round.

$$T_i = \max |\{t_{i,1}, t_{i,1}, \ldots, t_{i,m}\}| \tag{3}$$

For ROS, we apply a simple effective approach. The new random switching order j derives from the sum of current random sequence and next random sequence. The sum can be carried on to send the random sequence. As a result, even a less powerful processor can handle this process easily. Moreover, since

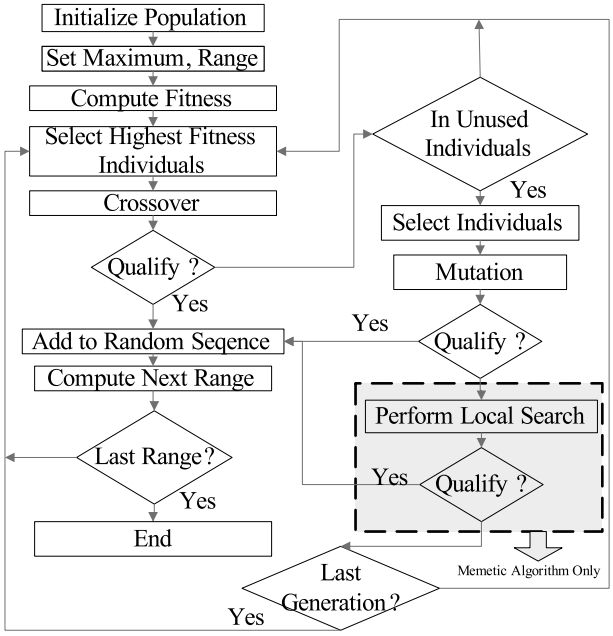

Fig. 3. The flow of genetic/menetic random sequence.

current and next random sequences contain many high random values, simple additions are effective to provide a good random value for the ROS as in the following equation.

$$j = (\sum_{k=1}^{k=n} tk, i + \sum_{k=1}^{k=n} tk, i + 1)\%m + 1\,, \tag{4}$$

Where i is the card number of SPC, after the$t_{n,1}$ interval, SPC starts the ROS. After the card number is given, its next sequence is obtained by modulo operation with m.

3.3. Genetic/menetic random sequence (GRS)

Two versions of our random sequence generators are proposed: the memetic algorithm version is a modified version of our genetic algorithm and improves the computation speed by a local search algorithm.

The generation of random sequence $\{t_1 \ldots t_n\}$ requires the properties of a uniform distribution and nonlinear uncertainty. As mentioned above, compared with LFSR, uniform distribution generators and linear congruential generator, the genetic generator has more advantages. Therefore, we will apply the genetic/memetic algorithms to design our RIF. In our proposal, the main flow is depicted in Fig. 3.

First, as in the operation flow shown in Fig. 3, the genetic and memetic algorithms are similar except for the fact that the memetic ones perform local search algorithm to reduce the evolutionary time. Evolutionary random algorithm (ERA) requires three main parameters to generate the random sequence.

It initializes the population from external seed *SD*, and incrementally generates random numbers from zero to the maximum random value *MAX* within a given incremental range *RG*.

After the initialization, the algorithm takes a computing of genetic/memetic algorithm as shown in the following steps:

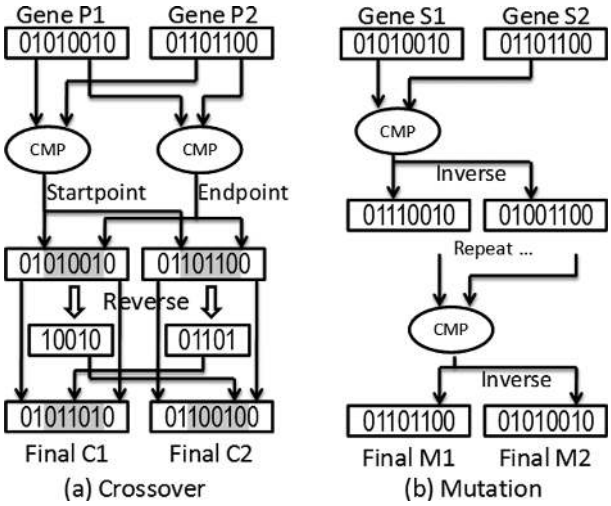

Fig. 4. The crossover and mutation operations.

- Computing and selecting fitness for crossover. The fitness function can be defined as generating different types of random distributions. For instance, our fitness function is designed to generate a uniform distribution.
- Performing crossover operation to generate new random number by adding generated into random sequence if they meet the fitness function. For details, please see the crossover operation in Fig. 4.
- Performing mutation operation if the crossover cannot generate the fitness random number. For details, please see the mutation operation of Fig. 4.
- (For memetic algorithm only) Performing local search algorithm if the mutation operation cannot obtain satisfactory random numbers. For details, please see the pesudo- code of memetic local search.
- Checking the last generation, the algorithm performs specified iterations for crossover and mutation operations to avoid infinite running. 30 iterations are given in the evaluation examples. More iterations may have less non-fitness values, and they increase the computation time.
- The algorithm terminates the program when it has the last range; the maximum random range is reached.

The crossover and mutation examples are given in Fig. 4. As shown in the left side diagram, the gene parents $P1$ and $P2$ are compared (*CMP* operation) to distinguish between different bits in order to find the start point and endpoint. Then, two substrings of $P1$ and $P2$ are obtained, reversed and exchanged to generate the two final genes $C1$ and $C2$. The new random value is selected from the fitness of these two values.

If the crossover operation cannot generate a satisfactory random value, it will perform the mutation operation as depicted in the right side diagram of Fig. 4. In this operation, it first performs a comparison (*CMP*) between the two genes $S1$ and $S2$ from left to right bit position. If two bits are different, the bit inverse is performed. The above mutation operations are repeatedly performed until the bit comparisons reach the rightmost bit. After two final mutation strings $M1$ and $M2$ are generated, the new random value is selected from the fitness of these two values as well.

When both crossover and mutation operations fail to generate fitness values, we perform a local search algorithm using a binary tree as shown in Fig. 5. Symbols are denoted as follows:

```
BF(): Best faintness function
R: root node of search tree
R.V: Root value.
U :    Unused population.
CR: Current range:
RI: Range interval
CBP: Change bit position
LB: Lower bound of bit position.
UB: Upper bound of bit position.
GT(): Generate search tree function
MTL: Maximum tree level
LC: Left child node
```

```
1   R.V ← BF(U)
2   CBP ← (LB + UB) /2 where LB ← CR; UB ← CR+RI
3   I ← 0
4   GT(I, R , CBP)
5   Static CF
6   CF←ST(C)
7
8   GT(I, R, CBP, MTL) {
9   If I = MTL; Exit
10  LC.V ← CB1(R, CBP); RC.V ← CB2(R,CBP)
11  CBP ← CBP + 1; I ← I + 1
12  GT(I, LC, CBP); GT(I, RC, CBP)
13  }
14
15  ST(C ) {
16  If   C=NULL; Exit;
17  If FR(CF) <   FR(C)
18      CF ← C.V
19  ST(C.L);
20  ST(C.R);
```

```
LC.V: left child value
RC: Right child node
RC.V: Right child value
CB1(): first changed bit function
CB2():    second changed bit function
ST(): Search tree function
C: Current node
C.V: Current node value
C.L: Left node of current node
C.R: Right node of current node
FR: Fitness Ranking
CF : Closet fitness value
```

Fig. 5. The pesudo algorithm of memetic local search.

First, it selects the root value R for binary search tree based on the best fitness function *BF()* with unused population U. After R is selected, it computes the current changed bit *CBP* according to upper bound *UB* and lower bound *LB* values. During the generation search tree *GT()*, the left and right two children values *LC.V* and *RC.V* are generated by the changed bit functions *CB1()* and *CB2()* respectively. The next *CBP* is increased to select a neighbor bit of a previous *CBP*. Then, the *GT()* performs child nodes generation recursively until the maximum tree level *MTL* parameter is reached.

In the search tree function *ST()*, it performs a left to right order searching. If the fitness ranking function *FR()* of current node C is higher than current fitness value CF, then it updates the value of current node $C.V$. *ST()* perform in a recursive way as well; the next left child *ST(C.L)* and right *ST(C.R)* are performed in a left to right order. It exits the search function when C is *NULL*.

4. Evaluation

In this evaluation section, we first compare our genetic and memetic algorithm and then compare our algorithm with the conventional random generator.

4.1. The comparison between genetic and memetic random generator

Since we want to know the merit of memetic algorithm (MA) as in the last paragraph of Section 2, the non-fitness, execution time, and different range three issues are studied for the genetic algorithm (GA) and MA. In these experiments, we run the maximum range of GA and MA for 500, 1000, 10000 and 100000. This can show the differences between the maximum ranges.

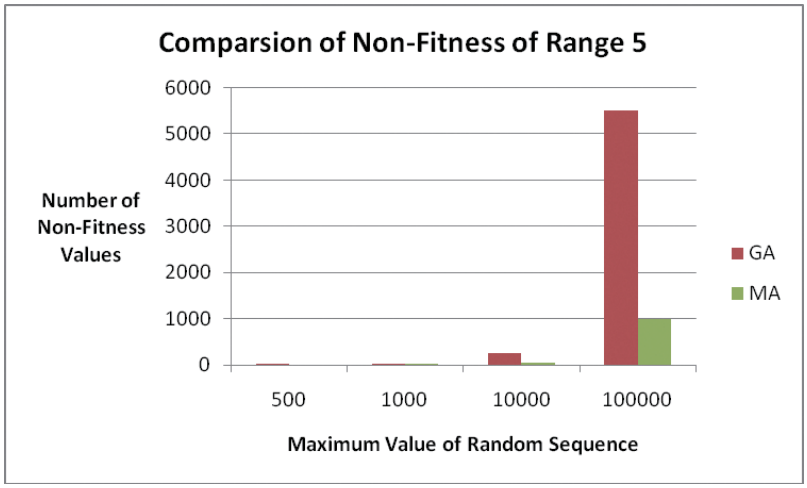

Fig. 6. The non-fitness comparison of range 5 for genetic/menetic random sequence.

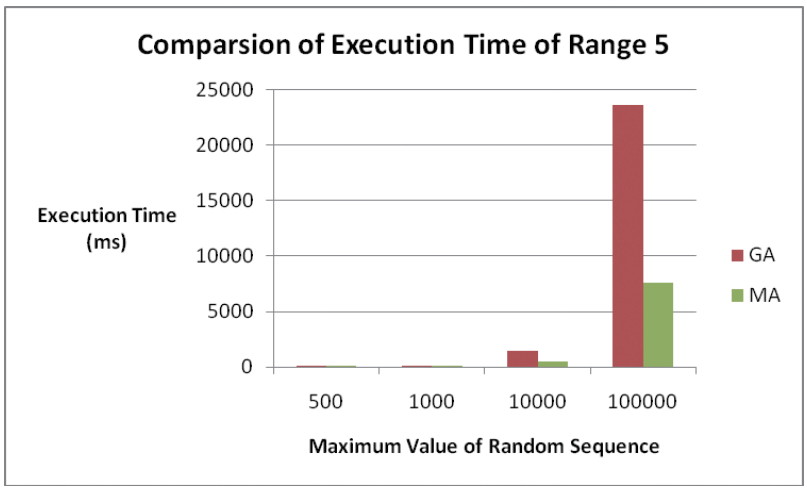

Fig. 7. The execution time comparison of range 5 for genetic/menetic random sequence.

Figures 6 and 7 depict the non-fitness comparison and execution time between MA and GA algorithms. The results reveal that MA can reduce non-fitness values and execution time from GA. These results are improved by the local search algorithm of MA.

A range 5 random interval is used in this comparison, which means the generation of random sequence is increased by 5 from 0 to maximum values. Figure 6 is the non-fitness comparison of range 5 for both genetic and memetic random sequences. This comparison demonstrates that MA is effective in reducing non-fitness values.

The second comparison is the execution time for GA and MA of random algorithm. The result shows MA reduce execution time from GA as in Fig. 7.

The next comparisons are non-fitness and execution time comparison between range 5 and 10. Figures 8 and 9 shown the non-fitness and execution time and be improved and reduced by relaxed the range condition as in Fig. 3. Since the GA and MA algorithm are the non-linear evolutionary algorithms, if it

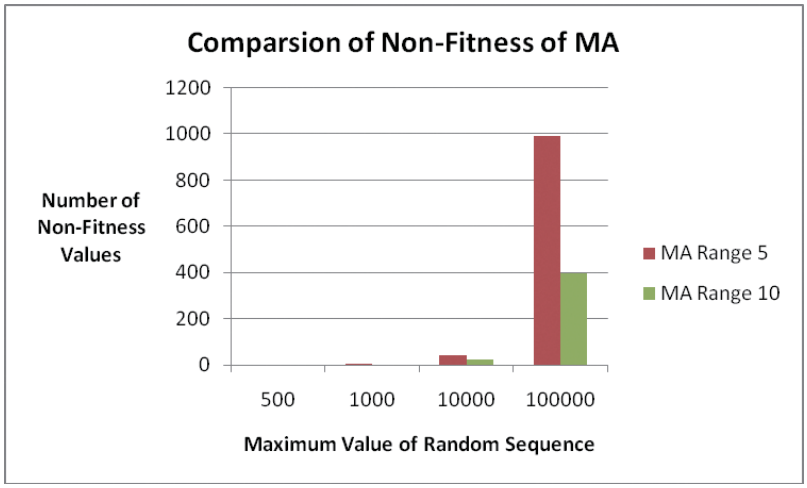

Fig. 8. The non-fitness comparison of range 5 and range 10 for menetic random sequence.

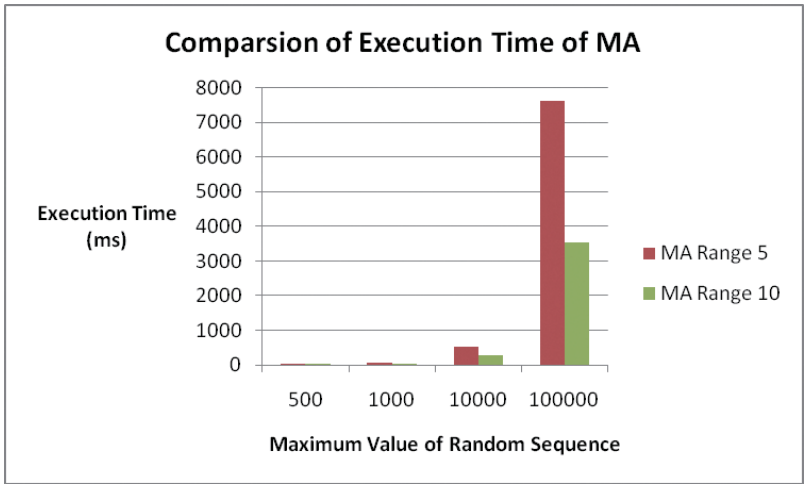

Fig. 9. The execution time comparison of range 5 and range 10 for menetic random sequence.

is possible to limit their restrictions, then this will be helpful for their performance.

In the comparison between the two different ranges for the MA as Figs. 8 and 9, the result shows range 10 as a less strict range which has better performance in reduced non-fitness values and reduced execution time.

For instance, it generated 100 pseudo-random numbers from 0 to 999. One number in a span of 10 numbers is expected to fall on. The pseudo-random number should follow the distribution as follows: the first should fall between 0 and 10, the second between 11 and 20, and the last between 990 and 999. During the 100 pseudo-random, local search was applied for 32 pseudo-random. Similarly, for 1000 pseudo-random generated between 0 and 10000, local search was applied for 622 pseudo-random. For 10000 pseudo number between 0 and 100000, local search was applied for 7345 pseudo-random.

As the population of individual genes generating random numbers from the corresponding genes varies widely, choosing the individual, if it is limited to the population, will result in local convergence, and

after many generations it will not be able to evolve into the appropriate individual. If the best individuals from the fitness area of outstanding representatives of the search are found in order to replace this individual and if then the evolutionary process is run, the individual fitness in the local search process will be improved. This local search mechanism of the population aims at improving the direction of evolutionary efficiency.

The results prove that the proportion of suitable pseudo-random based on MA method has been greatly improved. The MA-based local search method based on genetic algorithm efficiency is about 3 times the genetic algorithm GA.

4.2. The comparison with the conventional random generators

During the generation of the random number of our GA/MA, the individual in the processes of crossover and mutation operations is hard to be predicted. The reason for the crossover result is random. After the crossover, if the number is not obtained within specified intervals, the mutation performed based on this number is less than or greater than the upper limit. This can turn 0 to 1 or 1 to 0. The purpose of this mutation is that it helps avoid the certainty of the genetic variation of the algorithm and improve the efficiency of generating random numbers. Although the mutation results can be predicted, the final numbers are more random than LFSR, linear congruential generator and uniform random generator as the following analysis.

4.2.1. LFSR

We intend to introduce the commonly used random number generator LFSR to compare with our method. Comparing our methods to the conventional linear algorithm LFSR method of generating random numbers, our generated random numbers exhibit better performance on the uncertainty, which is harder to be derived by the generation method. Although the crossover and mutation operation of genetic/memetic algorithms have the higher complexity, our methods of crossover and mutation have a purpose to increase the uncertainty in the process, thus ensuring a new generation to meet the objectives of the chromosome the direction of conversion.

LFSR is based on the $X_{i=}(\lambda X_{i-1} + \mu)$ mod P, in this equation λ and μ are constant, and $P = 2^{\beta}$, $X_i \in \quad \left[0, 1, \ldots, 2^{\beta} - 1\right]$. The quality of random number depends on λ, μ, P. We can derive β, λ, μ, and P according to the random numbers. For example, if $\{X_1 = 1X_2 = 2X_3 = 3X_4 = 0X_5 = 1\}$, we can obtain $\beta = 2$, and $\mu = 4a+1$, $\lambda = 4b+1$, where a and b are the integers. If we put these parameters into the equation, we can predict the future random numbers. Thus we can conclude that the randomness quality of LFSR random numbers is relatively poor.

The linear feedback shift register (LFRS) method is based on the states of the shift register and the registers bitwise operation of XOR. Since it has a limited number of states, so after the state to state change in a repeated loop. The generated random numbers of LFSR method can be determined by two aspects: 1). Register state. 2). Feedback function. Shift register is to register the content from the current location to the location adjacent to the last one will be out of register. The register is shifted from low to high bits and then vice versa. The vacated bits will be out through the feedback function to calculate the result of filling. Feedback function plays a key role because the final outcome can be computed from this register status and feedback functions.

There is an example using the LFSR random number generation with the register of the initial state being 001000. The role of feedback function in practice on the first 3 and 5 bits is to perform the XOR operation on these two bits, as *1 XOR* $0 = 1$. Then it performs the rotation operations on the register. So, the next state would be 100100. In the same approach, the next state is 010010. Beginning from the

001001 state, the shift of $2^6 - 1 = 63$ time follows and then it goes back to the 001001 state. Since the role of feedback function and the shift state transition are not changeable, and the state cannot be 000000. LFSR can determine its next state register with a 50% possibility of success. For example, the next state of 001001 can only be 100100 or 000100. So the next state has a 50% chance to be 100100. According to this analysis, LFSR produces the predictable random possibility of 50%. This quality of the generated random number is not ideal.

LFSR is used in the example to generate pseudo-random numbers where the random number will be repeated after a period of just a random number generated series. If the random number series is 001000,100100,010010, no matter what the LFSR method implementation details are, we can conclude that after the random number 001000, the followed by random numbers will be 100100. In other words, the two adjacent pseudo-random values have dependence.

4.2.2. Linear congruential generator

Linear congruential generator can be implemented as $X_{n+1} = (aX_n + c)\%b$, where a, b, and c are assumed to be constant. The disadvantage of this method is that the generator numbers can be derived from the generated following two derivations.

For the first derivation, let $b = \max\{X_1, X_2, X_3, \ldots\}$, $X_{\max} = \max(X_i, X_{i+1})$ and $X_{\min} = \min(X_i, X_{i+1})$. We designed two equations for $X_{n+1} = (aX_n + c)\%b$ as $aX_{i+1} + c = e*b + X_{i+2}$ and $aX_i + c = f*b + X_{i+1}$, where $(e - f)$ are the nature numbers. With the first equation minus the second equation, we can obtain $a = ((e - f)*b + X_{i+1} - X_i)/(X_{\max} - X_{\min})$.

In the second derivation, if $aX_{i+1} + c = e*b + X_{i+2}$, and let $e = 1, 2, \ldots$ until $e*b + X_i + 2 > a*X_{i+1}$, we can get $c = e*b - aX_{i+1}$.

From the deduction we apply for the above two derivations of the linear congruential generator. We can assert that according to the series numbers generator by the two derivations above. To test 10 number sequences $\{X_i, X_{i+1}, \ldots X_{i+9}\}$, if all these numbers fit the equation, we consider the equation to be correct. Otherwise, if the equation is not correct, we leave $b = b + 1$ and repeated processing two above derivations until the correct answer is obtained.

4.2.3. Uniform random generator

A common method of generating uniform random number is *number = low + (high − low) ∗ random ()* with the range of random numbers. This method of generating random numbers needs to rely on another generated random number. If we overlook the random number, this function is not well to make a well random number.

5. Conclusion

We proposed a new concept – evolutionary random interval fingerprint- to enhance security of wireless communication. Besides the packet encryption, it is another way to provide a more secure communication. Moreover, the new nonlinear genetic/memetic algorithms are designed for the random interval fingerprint which is more robust than conventional pseudo-random generators. In this paper, we also provided an application example, completed related work survey, pseudo-code and analysis result to prove that our concept is feasible for the wireless communication.

References

[1] M.M. Organero, P.J.M. Merino and C.D. Kloos, Using bluetooth to implement a pervasive indoor positioning system with minimal requirements at the application level, *Mobile Information Systems* **8**(1) (2012), 73–82.
[2] S. Parvin, F.K. Hussain and S. Ali, A methodology to counter DoS attacks in mobile IP communication, *Mobile Information Systems* **8**(2) (2012), 127–152.
[3] J. Žilinskas and I. Bogle, Evaluation ranges of functions using balanced random interval arithmetic, *Informatica* **14**(2) (2003), 403–416.
[4] C.J. Pietras, A.E. Brandt and G.D. Searcy, Human responding on random-interval schedules of response-cost punishment: the role of reduced reinforcement density, *Journal Of The Experimental Analysis Of Behavior* **93**(1) (2010), 5–26.
[5] H. Takagi and R.M. Rodríguez-Dagnino, Counting the number of renewals during a random interval in a discrete-time delayed renewal process, *Operations Research Letters* **35**(1) (2007), 119–124.
[6] A. George, K. Thanasis and T. Leandros, An 802.11 k compliant framework for cooperative handoff in wireless networks, *EURASIP Journal on Wireless* **23** (2009), 14.
[7] V. Shmatikov and M.H. Wang, Timing analysis in low-latency mix networks: attacks and defenses, *Computer Security–ESORICS* (2006), 18–33.
[8] A.D. Keefe and J.W. Szostak, Functional proteins from a random-sequence, *Nature* **410** (2001), 715–718.
[9] C. Yuan, Y. Zhong and S. Yang, Composite chaotic pseudo-random sequence encryption algorithm for compressed video, *Tsinghua Science and Technology* **9**(2) (2004), 234–241.
[10] S. Verdú and S. Shamai, Spectral efficiency of cdma with random spreading, *IEEE Transactions On Information Theory* **45**(2) (1999), 622–640.
[11] C.Y. Chan and Y. Ding, Boltzmann ensemble features of rna secondary structures: a comparative analysis of biological rna sequences and random shuffles, *Journal of Mathematical Biology* **56**(1–2) (2008), 93–105.
[12] Z. Ren and G.G. Zhu, Pseudo-random binary sequence closed-loop system identification error with integration control, *Journal of Systems and Control Engineering* **223**(6) (2009), 877–884.
[13] N. Mandhani and S. Kak, Watermarking using decimal sequences, *Cryptologia* **29** (2005), 50–58.
[14] C. Jianxiao, R. Lixin and C. Kangsheng, A random sequence generator based on chaotic circuits, *Journal of Electronics* **18**(1) (2007).
[15] F. Koeune, Pseudo-random number generator, *Encyclopedia of Cryptography and Security* (2005).
[16] K. Iba, Reactive power optimization by genetic algorithm, *IEEE Transactions on Power Systems* **9**(2) (1994), 685–692.
[17] J. Knowles and D. Corne, M-PAES: A memetic algorithm for multiobjective optimization, *Proceedings of the 2000 Congress on Evolutionary Computation* **1** (2000), 325–332.
[18] R. Lande, Natural Selection and Random Genetic Drift in Phenotypic, *Evolution* **30**(2) (1976), 314–334.
[19] F. Samanlioglu, W.G. Ferrell and M.E. Kurz, A Memetic Random-Key Genetic Algorithm for a Symmetric Multi-Objective Traveling Salesman Problem, *Computers and Industrial Engineering* **55**(2) (2008), 439–449.
[20] P. Garg, A Comparison Between Memetic Algorithm and Genetic Algorithm for the Cryptanalysis of Simplified Data Encryption Standard Algorithm, *International Journal of Network Security & Its Applications* **1**(1) (2009), 34–42.
[21] Y. Cengiz and H. Tokat, Linear Antenna Array Design With Use Of Genetic, Memetic and Tabu Search Optimization Algorithms, *Progress In Electromagnetics Research C* **1** (2008), 63–72.

Kuo-Kun Tseng received his PhD. degree at Computer Science and Information Engineering Department from Taiwan National Chiao Tung University in year 2007. He is an associate professor at Computer Science and Technology Department in Harbin Institute of Technology Shenzhen Graduate School. From 1998 to present, he has over 10 years R&D and 7 years teaching experiences. Currently, his research areas are the pattern matching, image processing, intelligent classification and network algorithms for mobile embedded system, cloud computing and Internet of Things.

Jeng-Shyang Pan received the B. S. degree in Electronic Engineering from the National Taiwan University of Science and Technology in 1986, the M. S. degree in Communication Engineering from the National Chiao Tung University, Taiwan in 1988, and the Ph.D. degree in Electrical Engineering from the University of Edinburgh, U.K. in 1996. Currently, he is a Professor in the Department of Electronic Engineering, National Kaohsiung University of Applied Sciences, Taiwan. He is also invited to be the Doctoral advisor both in University of South Australia and Harbin Institute of Technology. He jointed the editorial board of International Journal of Innovative Computing, Information and Control, LNCS Transactions on Data Hiding and Multimedia Security, International Journal of Hybrid Intelligent System, Journal of Information Assurance and Security, International Journal of Computer Sciences and Engineering System, Journal of Computers, International Journal of Digital Crime and Forensics, ICIC Express Letters, and Computational Intelligence and Its Applications Book Series (IGI Publishing). His current research interests include soft computing, information security and signal processing.

Wei Wei is a master student of Computer Science and Technology Department in Harbin Institute of Technology Shenzhen Graduate School. His research interests are wireless communication and embedded system.

Hui-Huang Hsu is a professor in the Department of Computer Science and Information Engineering at Tamkang University, Taipei, Taiwan. He received his PhD and MS Degrees from the Department of Electrical and Computer Engineering at the University of Florida, USA, in 1994 and 1991, respectively. Prof. Hsu has been very active in international academic activities. He has published over 100 referred papers and book chapters. His current research interests are in the areas of data mining, ambient intelligence, services computing, and bio-medical informatics. Prof. Hsu is a senior member of the IEEE.