A New Scheme for Pseudo Random Numbers Generator Based on Secret Splitting

Abdulameer K. Husain, **Ayman A. Rahim A. Rahman**

Abstract— This paper presents a secure scheme for generating a pseudo random numbers. The scheme is based on secret splitting of a piece of secure information which is used as a seed to the generator. In this scheme the procedure of splitting the secure information is performed according to a specified weight in such a way that each segment of the information piece takes a special weight depending on the priority of each part of the random number sequence. Another important concept used in this method is the agreement strategy applied to the secure information. By comparison of this scheme with other methods of generating pseudo random generation, it is found that there are a secure complex relationships among the elements of the random number sequences which are difficult to be discovered by most active attacks.

Index Terms—Pseudo-Random, secret splitting, Cryptography, Secrecy, agreement.

I. INTRODUCTION

Random numbers have many advantages especially in cryptography as they are used for key streams of one - time pads, secret keys of symmetric cipher systems, public key parameters, session keys, nonce, initialization vectors and salts. In addition random numbers sequences are of crucial importance in almost every aspect of modern digital cryptography, having a significant impact on the strength of cryptographic primitives in securing secret information by rendering it unknown, unusable, unpredictable, and irreproducible for an adversary [1,2,3] . Random Number Generators (RNG) are divided into two main families:(1) the True Random Number Generators (TRNGs) which exploit the physical phenomena that contain part of incertitude and (2) Pseudo Random Number Generators which are based on deterministic algorithm and they are considered more appropriate for security applications [3, 4,5] .The main objective of RNG is producing certain independent and intricately distributed numbers [6] . Some standards that are regarded as secure in the past have recently failed [7], [8], [9] and [10], other generators which are one of the few PRNGs with proven security [11]. These standards are of little use for their slowness. In recent years new generators which are called Lagged Fibonacci pseudo-random number generators have become increasingly popular generators for serial as well as scalable parallel machines.

Manuscript Received on December 2014.

Abdulameer K. Husain, his PhD in Computer Science, Computer Security, Presently Working as Assoc. Prof., in Jerash University –Jordan.

Ayman A. Rahim A. Rahman, his PhD in Computer Information System, Presently Working as Asst. Prof., in Jerash Uiversity - Jordan.

These generators are of common use because they are easy to implement, cheap to compute and they perform reasonably well on standard statistical tests [12], [13] and [14] especially when the lag is sufficiently high.

II. RELATED WORKS

In [15], the authors were proposed for a new splittable pseudorandom number generator based on a cryptographic hash function. In this scheme, the authors showed that the currently known and used splittable PRNGs are either not efficient enough, have inherent flaws, or lack formal arguments about their randomness. The authors provided proofs giving strong randomness guarantee under assumptions commonly made in cryptography. A patent method of generating pseudo-random numbers by means of an iteration applied to a one way function which is based on a start value and a key for generating a part of the pseudo-random number and wherein the iteration is initialized with a random start value and a random key, and wherein, in each iteration step, both the start value and the key for an iteration step are determined from the part of the pseudo-random number determined in the previous iteration step using the one-way function [16] . In [17] a new Iteration Function System (IFS) had been used for generating a Pseudo -Random Number Generator. In this research. The authors measured the sensitivity of the Iterated Function System (IFS) to the initial condition. In this system and at a certain initial value, the iterated function (IFS) can generate a chaotic random numbers.

A paper introduced an interesting hierarchy of random number generators based on the review of random numbers characteristics and chaotic functions theory. The main aim of this paper is to produce a dynamical system which can be implemented in random number generators. The authors had carried out certain statistical tests on a series of numbers obtained from the introduced hierarchy [18].

III. PROPOSED SYSTEM

This method generates pseudo random numbers by selecting a certain piece of secure information. The users involved in this system to generate these random numbers must agree upon this information in advance to enable them to generate their own random numbers and then used them for cryptographic systems. The users also agree about the predefined structure of this secret information. The structure in this method consists of a certain number of digital information or in a binary representation. One of the constraints of this structure is the agreement upon the size (n) of this secure information. The other constraint is the way which is used to split this information piece. The secure

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication Pvt. Ltd.

information is split into different segments. In this scheme we used a secure procedure for splitting information piece by using a specified different weight for each segment. Figure 1 illustrates the structure of this secure information.

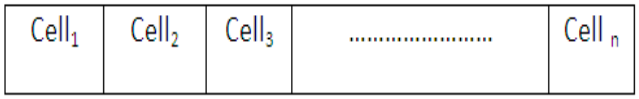
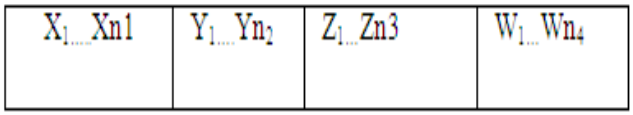
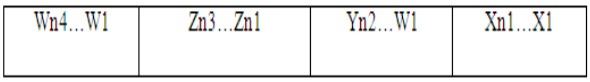


Figure 1: The Structure of Secret Information Piece.

Every segment represents a certain cell of one number. Each cell contains different digital numbers. The splitting method of information is based on decomposition of the total contents of the cells with weighted segments. The first random segments generated are done by taking x_1% as a weight of the whole secret information. The second random segment is generated by a different percentage of x_2 % as a weight with the remaining size of the information piece and so on. For this reason the users must have a knowledge about these percentage values which is considered an agreement factor among these users. The next step is that each segment is Applying XOR with its corresponding weight. At the end of this step, the system generates the first set of pseudo random numbers. To generate the next set of random numbers, first, each segment is reversed and the total set of first generation is also reversed. Secondly we apply XOR operations to this reversed set with the corresponding weight and thus we get the second set of pseudo random numbers, Figure 2 illustrates a segment after applying the reverse procedure.



A: Segments with XOR operations.



B: Reversed Segments.

Figure 2: XOR and Reverse Operations.

At this stage the first set of pseudo random numbers is generated. The same procedure is repeated to generate further sets of random numbers

IV. ALGORITHM

Let w be an array of all available weights where w= $[w_1,w_2,\ldots..,w_m]$.
Let A be an array of size n which represents the information piece where A=[1..n].
Let B be an array of size n_1 which contains the segmented elements of array A where B=[1..n_1].
Let C be an array of size n_2 which contains the results of XOR operations on the elements of the array B where C=[1..n_2] .

Initialization (splitting the information piece) :
j=1
n =length. A
Repeat
for (i=1 to w. length)
B[j]=n*w[i]
n=n- B .length
j=j+1
until n=0 or n=1
Step 1:
k=1
m=1
for (x=1 to n_1)
C[k]=B[x]XOR w[m]
k=k+1
m=m+1
if w. length= 0 then w. length=w.length+1
Step 2:

A: Reverse each element of size n digits:

```
for (i=1 to C. length )
{
   count=0
      while (n>0)
        count=count+1
        n=n/10 ;
       temp=n
{
C[i]= temp
End reverses each element
```

B: Reverse the entire array

```
      int start, end, temp
      While (star<end)
temp = C[start]
C[start]==C[end]
C[end]=temp
start=strat+1
end=end-1
```

V. IMPLEMENTATION AND RESULTS

Initialization:
We choose a secure agreement information piece which is stored in an array A of size n=14 as shown below:

The weights set assigned for splitting this information piece are chosen into 4 weights which are: 15 %, 20%, 25% and 30%). Now elements of the information piece are gathered into groups by multiply each element by the its corresponding weight until all elements were exhausted as explained bellow:

$R1 = \lfloor n * 15\% \rfloor = \lfloor (14 * 15\%) \rfloor = 2$ digit
$R2 = \lfloor (n - R1) * 20\% \rfloor = \lfloor (14 - 2) * 20\% \rfloor = 2$ digit
$R3 = \lfloor (n - (R1+R2)) * 25\% \rfloor = \lfloor (14 - (2+2)) * 25\% \rfloor = 2$ digit
$R4 = \lfloor (n - (R1+R2+R3)) * 30\% \rfloor = \lfloor (14 - (2+2+2)) * 30\% \rfloor = 2$ digit
$R5 = \lfloor (n - (R1+R2+R3+R4)) * 15\% \rfloor = \lfloor (14 - (2+2+2+2)) * 15\% \rfloor = 1$ digit
$R6 = \lfloor (n - (R1+R2+R3+R4+R5)) * 20\% \rfloor = \lfloor (14 - (2+2+2+2+1)) * 20\% \rfloor = 1$ digit
$R7 = \lfloor (n - (R1+R2+R3+R4+R5+R6)) * 25\% \rfloor = \lfloor (14 - (2+2+2+2+1+1) * 25\% \rfloor = 2$ digit
$R8 = \lfloor (n - (R1+R2+R3+R4+R5+R6+R7)) * 30\% \rfloor = \lfloor (14 - (2+2+2+2+1+1+2) * 30\% \rfloor = 1$ digit
$R9 = \lfloor (n - (R1+R2+R3+R4+R5+R6+R7+R8)) * 15\% \rfloor = \lfloor (14 - (2+2+2+2+1+1+2+1) * 15\% \rfloor = 1$ digit

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication Pvt. Ltd.

The new generated segments are stored in an array B as explained below:

63	98	13	54	2	6	19	5	4

Step1:
Now we apply the XOR for each element of the new generate segment with the corresponding weights as explained below:

```
63 XOR 15 = 48
98 XOR 20 = 118
13 XOR 25 = 20
54 XOR 30 = 40
2  XOR 15 = 13
6  XOR 20 = 18
19 XOR 25 = 10
5  XOR 30 = 27
4  XOR 15 = 11
```

In this case we get the first information and will be stored in the array C as the initial value for the first set of the pseudo random number as explained below:

48	118	20	40	13	18	10	27	11

Step 2:
Then we reverse each element and the entire elements of the array C and the result is stored in array C itself, this procedure generates the first set of pseudo random numbers as explained below:

11	72	01	81	31	04	02	811	84

Similarly we apply step 1 and step 2 to generate the second set of pseudo numbers as shown below:

```
11 XOR 15 = 4
72 XOR 20 = 92
01 XOR 25 = 24
81 XOR 30 = 79
31 XOR 15 = 16
04 XOR 20 = 16
02 XOR 25 = 27
811 XOR 30 = 821
84 XOR 15 = 91
```

4	92	24	79	16	16	27	821	91

At the end of this procedure we get the second set of pseudo numbers as shown below:

19	128	72	61	61	97	42	29	4

The same steps are repeated to generate the third set of pseudo random numbers as explained below:

```
19 XOR 15 = 28
128 XOR 20 = 148
72 XOR 25 = 81
61 XOR 30 = 35
61 XOR 15 = 50
97 XOR 20 = 117
42 XOR 25 = 51
29 XOR 30 = 3
4 XOR 15 = 11
```

28	148	81	35	50	117	51	3	11

11	3	15	711	05	53	18	841	82

Note: we can continue with the same procedure to get further sets of pseudo random numbers.

VI. Conclusion

In this Paper we use the concept of secret splitting to produce sequences of pseudo-random number generation because secret splitting is the most common secure technique to enhance system's secrecy. The relations among any random number are based on an agreement between the sender and the receiver so both must have the knowledge of the size of secure agreement which is the seed, the secret information and also the weights assigned for splitting this seed. Based on this agreement procedure, the proposed system will generate secure random numbers and then can be used as cryptographic keys. In analyzing this system we find that the information piece is secure because it is not transmitted but it can be distributed in house especially for sensitive applications. Another enhancement factor is that the splitting of such information piece is designated in a more strict procedure and these weights are considered as an agreement and secure factors .These weights must be selected carefully depending on the priority of each segment and those are used later for generating successive sets of random sequences. In comparison of this system with other pseudo random generators it is found that it changes the method of seed selection of the generator in that instead of using a fixed and a public seed, the seed in this system is private and it does not take a whole part but it is composed of different segments with complex and secure relations to recover the original piece of information. Finally we had found that most pseudo random generators rarely depend on secret splitting and agreement except those which are used in parallel processing.

References

[1] D. Mathilde, T Jessy, R.Philiippe," Methodology for the Fault Analysis end Evaluation of True Random Number Generator", hal 00678001 version 1,Monaco , France, 11thMar 2012.
[2] A. Kinga, F Aline, E Christain," Generation and Testing of Random Numbers for Cryptographic Application", Proceeding of Romania Academy, vol. 13, number 4/2012,pp 368 -377.
[3] D. Dilli, S. Madhu, "Design of a New Cryptography Algorithm using Reseeding -Mixing Pseudo Random Number Generator", IJITEE, vol. 52, Is sue 5, 2013.

[4] S. Martain, “Testing of True Random Number Generator Used in Cryptography”, IJCA, vol.2, issue4, 2012.
[5] W. Edward, Q. Christphor, T. Boateng, “ Comparative Analysis of Efficiency of Fibonacci Random Number Generator Algorithm and Gaussian Random Number Generator Algorithm in Cryptography”, Computer Engineering and Intelligent System, vol. 4, No. 10, 2013.
[6] S. Juan, ” Statistical Testing of Random Number Generators “, NIST company, 2012.
[7] A. Klein, “Attacks on the RC4 stream cipher,” Designs, Codes and Cryptography, vol. 48, no. 3, pp. 269–286, 2008.Cryptography, vol. 48, no. 3, pp. 269–286, 2008.
[8] I. Goldberg and D. Wagner, “Randomness and the Netscape browser,” Dr.Dobb’s Journal, pp. 66–70, 1996.
[9] Z. Gutterman, B. Pinkas, T. Reinman, “Analysis of the Linux Random Number Generator”. Proceedings of the 2006 IEEE Symposium on Security and Privacy, pp. 371–385, 2006.
[10] L. Dorrendorf, Z. Gutterman, B. Pinkas. “Cryptanalysis of the random number generator of the Windows operating system”. ACM T. Inform. System vol. 13(1), pp. 10:1–10:32 ,2009.
[11] L. Blum, M. Blum, and M. Shub, “A simple unpredictable pseudorandom number generator,” SIAM Journal on Computing, vol. 15, pp. 364–383, 1986 .
[12] G. Marsaglia, “A current view of random number generators”. Computing Science and Statistics: Proceedings of the XVIth Symposium on the Interface, pp. 3–10, 1985.
[13] G. Marsaglia, Diehard battery of tests of randomness, The Marsaglia random number CDROM, Department of Statistics, Florida State University, 1995.
[14] G. Marsaglia and W.W. Tsang, "Some difficult-to-pass tests of randomness", Journal of Statistical Software, Vol. 7, Issue 03, 2002.
[15] K. Claessen, M. Palka (2013) "Splittable Pseudorandom Number Generators using Cryptographic Hashing". Proceedings of Haskell Symposium 2013 pp. 47-58
[16] B. Heike. N., Steffen Scholze,Matthias Voegeler, Method of generating pseudo-random numbers, US 20090150467 A1 , Jun 11, 2009.
[17] E. Mohamed Nageb, R. Mahmud, Pseudo –Random Number Generator Using Deterministic Chaotic System , INTERNATIONAL JOURNAL OF SCIENTIFIC & TECHNOLOGY RESEARCH VOLUME 1, ISSUE 9, OCTOBER 2012 .
[18] B. Sohrab, A. Amir, A. Afshin, A.Samsudin, A novel dynamic model of pseudo random number generator, Journal of Computational and Applied Mathematics –J COMPUT APPL MATH, vol. 235, no. 12, pp. 3455-3463, 2011 .

Dr. Abdulameer K. Husain, Jerash University- Jordan. He has completed Master degree in computer science , university of Sadam , Iraq , in 1991 and his PhD in computer science , computer security from Al-Neelain University ,Sudan . He has total 20 years teaching experience and presently working as Associate professor in Jerash University –Jordan.

Dr. Ayman A. Rahim A. Rahman, Jerash University-Jordan. PhD,MSc,BSC. He has completed Master degree in Information Technology, and his PhD in Computer Information System , and presently working as Assistant Professor in Jerash Uiversity - Jordan.