European Journal of Scientific Research
ISSN 1450-216X / 1450-202X Vol. 147 No 3 November, 2017, pp. 315-323
http://www. europeanjournalofscientificresearch.com

A New Pseudo Random Number Generator Based on Context-Free Grammars and Statistical Information

Abdulameer K.Hussain
AL-Salam University College-Iraq
E-mail: abdulameer.hussain@yahoo.com

Muthanna Abdalwhad
Dijlah University College-Iraq
E-mail: muthanna.khudhair@duc.edu.iq

Abstract

Pseudorandom numbers are used for generating strong cryptographic keys. This paper presents a new method for generating such numbers. There are different methods for pseudorandom generators but the proposed method implements the concept of context-free grammars to provide a secure string of alphabets. Once a string is generated by one type of a context-free grammar, it is submitted to another type of grammars with different rules. This procedure results in a secure relation among different strings and then the generated pseudorandom numbers will be in a stricter manner. In addition the proposed system enhances the string generation by using different statistical data and tests.

Keywords: Pseudorandom number generators, Context-Free Grammars, Agreement, Secrecy

Introduction

Pseudo-random number generators (PRNGs) are important and primary blocks that used in several applications such as Monte Carlo simulation algorithms, communications and many cryptographic systems. These types of applications depend primarily on the quality of the pseudo random sequences. There are many implementations of the generated numbers to be simple pseudo random sequences, private keys or secure signatures. The researchers developed PRNGs for few years and until now. Different techniques were studied for producing PRNGs [1-9]. These types of pseudo random generators must have an important property which is the robustness in order to provide secure applications in cryptography field and also they resist all the various and existing attacks. Different and large groups of PRNGs exist which are based on the generation of sequences numbers by a single chaotic system or a combination of chaotic maps [4, 8, 10, 11]. These PRNGS used one-way function. The aim of these groups of PRNGs, map by a one-way function, is to improve the security of these generators.

Random numbers, in cryptography, are implemented in several aspects. These aspects are key streams of one -time pads, secret keys of symmetric cipher systems, public key parameters, and session keys, nonce. The effect of these random numbers is to enhance modern cryptography and they are used to strengthen the primitives of cryptography in such a way to securing information by rendering it unknown, unguessable, unpredictable, and irreproducible for an adversary [12, 13, 14].

Random number generators (RNGs) are classified into two families. The first family is known as True Random Number Generators (TRNGs). This type depends on a physical phenomena that contain a part of incertitude. The second family of PRNGs is based on deterministic algorithm. The

later type is considered more appropriate for security applications [14, 15, 16]. The main objective of Pseudo Random Number Generator is producing numbers that are independent and intricately distributed. The easy and common method to generate random numbers depends on cryptographically secure PRNG [17].

PRNGs are considered an important part of any cryptosystem and the reason of this is that the secrecy of all cryptographic systems depends mainly on the generation of good pseudorandom sequences. There are different implementations used by these generated numbers such as keystreams, initial vectors, private keys, and private signatures that are destined for controlling or initializing cryptographic algorithms. The design of good and reliable pseudorandom generators still remains an open problem in cryptology. Some de facto standards that are regarded as secure in the past have now failed [18], [19], [20] and [21], while other generators such as the BBS generator— which is considered one of the few PRNGs with proven security [22] —, are of more little use for its slowness.

Related Works

In [31] a framework was proposed for designing experiments that test for structural and semantic patterns of simple or complex grammars as originally described by Chomsky. In this method , the authors discussed several and different limitations of recent experiments that had attempted to address this problem , and showed that these experiments have he objective of detecting patterns that apply a Fibonacci series had many advantages over other artificial context-free grammars.

A research was proposed to introduce a concept of what is known Structure Synthesis which is a 3D structure generator based on the design of grammar specifications. In this research, the user must specify a certain type of grammar, and then the program generates one of the several possible structures commitments to the grammar syntax. When this program is compared to general-purpose programming, the limitations of context-free systems encourage the user to discover and explore the systems. And even though the syntax limits the complexity of the rules, the resulting structures are often highly complex and nearly always unpredictable and surprising [32].

A patent in [33] designed efficient generators of pseudorandom numbers to fool languages (or equivalently, Boolean-valued functions). This pseudorandom generator fools advised context-free languages which is known as context-free languages which are assisted by external information known as advice. In addition, this generator is made almost one-to-one, stretching n -bit seeds to n+1 bits.

In [34] a paper proposed an efficient pseudorandom number generator (PRNG) based on the concept of a hybrid between one- and two-dimensional cellular automata (CA). The proposed CA PRNG consists of two phases according to dimension. The first phase is designated for the evolution of CA cells in two-dimension (2-D) with a von Neumann neighborhood. In order to achieve a high quality of randomness property, a different and distinguishable method is chosen from previous CA PRNGs. The second phase is the evolution of CA cells in one-dimension (1-D) with 2-state, 3-neighborhood. The results of this paper had shown that the high quality of randomness is better than previous proposed CA PRNGs.

Proposed System

This system generates numerical strings of pseudorandom numbers. To enhance the generator of such numbers, the system uses the concepts of grammars and their corresponding languages to establish secure relations among the different parts of pseudorandom numbers.

The system first selects a number of different types of grammars. In computational theory, each type of grammar generates strings depending on the level of parties and the number of iterations inside each grammar. The input generated from one grammar will be an output to a different type of grammar to produce complex interleaved strings.

Each grammar consists of three parts. First part is the non-terminal symbols (N) , the second part is the terminal symbols (T) which constitutes the symbols of string generated, the last part is a set of rules(P) .Each grammar will generate a correspond language.

For example, the following grammar is described below:

N={S} , T{a,b} , P={1.S→ aSb 2.S → ab}, here we have two rules.

The language corresponding for this grammar is generated as follows:

To see what is the language generated from this grammar, we need to know what the terminal symbols generated are. Any terminal strings derived from the start symbol (S) belongs to the language generated by this grammar.

From rule 2 which is 2.S → ab we can generate the string ab . This is described formally as:

$S \overset{2}{\Rightarrow} ab$

By applying rule 2 we get :

$S \overset{1}{\Rightarrow} aSb$, but S goes to ab then :

So we get the string aabb which is a^2b^2 and this belongs to the language of this grammar , denoted by L(G)

If we begin with rule 1 repeatedly:

$S \overset{1}{\Rightarrow} aSb$

$S \overset{1}{\Rightarrow} aaSb\ b$

$S \overset{1}{\Rightarrow} aaabbb$ so a^3b^3 belongs to L(G)

So we can use different strong types of grammars to generate secure pseudorandom numbers. For this purpose we will use hexadecimal systems to replace them by the set of alphabets.

Below the algorithm of the proposed system:

Let a set of n chosen grammars $G=\{G_1,G_2,\ldots\ldots,G_n\}$

For i=1 to n

{

1: Take G_i , with G_{im} rules and G_{ir} repletion of rules.

2: Generate the output as items(i_m) of string . This string takes the form $\{G_{im1},G_{im2},\ldots..,G_{1mk}\}$.

3: Set the input as $\{G_{im1},G_{im2},\ldots..,G_{1mk}\}$.

Count=$G_{imk}+G_{imk}+I$.

}

4: Convert each G_{imk} to hexadecimal representation.

5: Choose a secure module d .

6: Apply inverse function for each item in the form inv(G_{imk}) mod d .

7: Choose a secure seed s .

8: Choose a secure agreement factor r .

9: Let the current position of each item in the string p .

9: Calculate the value of the first item as $item_1=(s+hexvalue_1+p_1+r)$.

10: Let the predecessor hex value of the current item is pr .

11: Let the successor hex value of the current item is su.

10: For i=2 to k-1

{

ValueItemi=(pr(i)+su(i)+p(i)+r) mod d.

{

ValueItemk=(pr(k) +$ValueItem_1$+p(k)+r) mod d.

Results

The two grammars used to generate a strong pseudorandom numbers are listed below:

1: The first grammar G1 is :

S → ABC

A → aAb/ʎ , where ʎ represents empty list

B → cBd/cd

The language generated from this grammar is:

L(G1)= $\{a^n b^n c^m d^m a^i b^i; n,I,m^{\geq}0\}$

2: The second grammar G2 is:

S → aSA

S→ cAd/B

B→ bBa/ ʎ

The language generated from this grammar is:

L(G2)=$\{a^n c^m b^i a^i d^m b^n ; n. m, i \geq 0\}$

Now the system takes the hexadecimal set of numbers and divides them into groups .So a presented in the above two grammars will represent the first 4 numbers of hexadecimal numbers, b the next 4 numbers and so on . So the set of numbers will be as explained in table 1:

Table 1: Hexadecimal representation of the Generated Strings

Symbol	Hexadecimal sets
a	0123
b	4567
c	89AB
d	CDEF

The output generated from the first grammar is illustrated by assigning different parameters for each i , m , n which are found in the language corresponding to this grammar. For example suppose n=2, m=3, and i=3 the numerical string generated is explained below:

$(0123)^2$ $(4567)^2$ $(89AB)^3$ $(CDEF)^3$ $(0123)^3(4567)^3$

So the final string will be as follows :

0123 0123 4567 4567 89AB 89AB 89AB CDEF CDEF CDEF 0123 0123 0123 4567 4567 567

This string will be an input to the second grammar. Before this step, the system decomposes this string into 4 equal groups as explained in the table 2:

Table 2: Grouping of Hexadecimal Sets

Symbol	Hexadecimal sets
a	0123 0123 4567 4567
b	89AB 89AB 89AB CDEF
c	CDEF CDEF 0123 0123
d	0123 4567 4567 4567

Applying the language for grammar 2 which is :

L(G2)=$\{a^n c^m b^i a^i d^m b^n ; n. m, i \geq 0\}$, assuming n=3 , m=2 qnd i=4 we get the following string:

$(0123\ 0123\ 4567\ 4567)^3$ $(CDEF\ CDEF\ 0123\ 0123)^2$ $(89AB\ 89AB\ 89AB\ CDEF)^4$ $(0123\ 0123\ 4567\ 4567)^4$ $(0123\ 4567\ 4567\ 4567)^2$ $(89AB\ 89AB\ 89AB\ CDEF)^3$

So the final string generated is as follows:

0123 0123 4567 4567 0123 0123 4567 4567 0123 0123 4567 4567 - 0123 0123 4567 4567 0123 0123 4567 4567 - 89AB 89AB 89AB CDEF 89AB 89AB 89AB CDEF 89AB 89AB 89AB CDEF 89AB 89AB 89AB CDEF -0123 0123 4567 4567 0123 0123 4567 4567 0123 0123 4567 4567

0123 0123 4567 4567 - 0123 4567 4567 4567 0123 4567 4567 4567 - 89AB 89AB 89AB CDEF 89AB 89AB 89AB CDEF 89AB 89AB 89AB CDEF

The next step is to find the inverse of each redundant value in this generated string after converting each one to a decimal representation. The inverse is performed according to a secure choice of module number such as mod 16. The inverse function takes each value and finds a value between 0 and 15. The inverse is generated when we find a value when it is multiplied with each redundant value in the string and the product mod 16 must be 1.Table 3 contains the results of redundant values.

Table 3: Decimal Representation of Redundant Values

Original value in the string	Decimal representation	Inverse value mod 16
123	291	11
4567	17767	7
89AB	35243	3
CDEF	52719	15

After that , each element of the original string is replaced by its corresponding inverse. Then we choose a secure seed applied only to the first element in the string. So the first part, 0123 0123 4567 4567 0123 0123 4567 4567 0123 0123 4567 4567, of the string is look like the following:

11 11 7 7 11 11 7 7 11 11 7 7

Now the first element is added to a secure seed, for instance 3. To enhance the security of such pseudorandom number generator, the system implements a secure agreement parameter. The parameter must be a prime number; in this example the parameter is 31.

To generate the pseudorandom numbers, the first element is calculated as follows:

1stelement= (I,S,P) mod n, where I is the inverse value , S is the seed, P is the agreement parameter, and is the total number of elements in the original string which is 72 in this example. So the 1stelement = (11+3+31) mod 72= 46.

The steps of calculating the remaining elements are explained as follows:

2ndelemen= (PV +ISUC+POS+P) mod n , where PV is the predecessor of this element , ISUC, the successor value of 2ndelement , POS, is the position of this element ,and P is the agreement parameter, so :

2^{nd} element= (46+7+2+31)mod 72=86 mod 72=14 . Table 4 explains the results of the remaining elements of the first part with agreement parameter 31.

Table 4: Results of Pseudorandom items of the First Part of the Original Generated Strings

Current Element	predecessor value	successor value	Position	Results
3d	14	7	3	(14+7+3+31) mod 72=55 mod 72=55
4^{th}	55	11	4	(55+11+4+31)mod 72= 101 mod 72=29
5^{th}	29	11	5	(29+11+5+31) mod 72=76 mod 72=4
6^{th}	4	7	6	(4+7+6+31) mod 72= 388 mod 72=48
7^{th}	48	7	7	(48+7+7+31) mod 72= 93 mod 72=21
8^{th}	21	11	8	(21+11+8+31)mod 72= 71 mod 72=71
9^{th}	71	11	9	(71+11+9+31) mod 72= 122 mod 72=50
10^{th}	50	7	10	(50+7+10+31) mod 72= 98 mod 72= 26
11^{th}	26	7	11	(26+7+11+31) mod 72= 106 mod 72=75mod 72=3

The next group of the string is calculated in the similar way as explained in table 5.

Table 5: Results of Pseudorandom Items of the Second Part of the Original Generated Strings

Current Element	Predecessor Value	Successor value	Position	Results
12th	3	11	12	(3+11+12+31)mod 72=57 mod 72= 57
13th	57	11	13	(57+11+13+31)mod 72=112 mod 72=40
14th	40	7	14	(40+7+14+31) mod 72= 92 mod 72=20
15th	20	7	15	(20+7+15+31) mod 72= 73 mod 72= 1
16th	1	11	16	(1+11+16+31) mod 72= 59 mod 72= 59
17th	59	11	17	(59+11+17+31) mod 72=118 mod 72=46
18th	46	7	18	(46+7+18+31) mod 72=102 mod 72= 30
19th	30	7	19	(30+7+19+31)mod 72=87 mod 72 =15

Now we continue with the third group of the original string as explained in table 6.

Table 6: Results of Pseudorandom Items of the Third Part of the Original Generated Strings

Current Element	Predecessor Value	Successor value	Position	Results
20th	15	3	20	(15+3+20+31)mod 72=69mod 72= 69
21th	69	3	21	(69+3+21+31)mod 72=124 mod 72=52
22th	52	3	22	(52+3+22+31) mod 72=108 mod 72=36
23th	36	15	23	(36+15+23+31) mod 72=105 mod 72=33
24th	33	3	24	(33+3+24+31) mod 72= 91 mod 72= 19
25th	19	3	25	(19+3+25+31) mod 72=78 mod 72=6
26th	6	3	26	(6+3+26+31) mod 72=66 mod 72=66
27th	66	15	27	(66+15+27+31) mod 72=139 mod 72=67
28th	67	3	28	(67+3+28+31) mod 72= 129 mod 72=57
29th	57	3	29	(57+3+29+31) mod 72= 120 mod 72= 48
30th	48	3	30	(48+3+30+31) mod 72= 112 mod 72= 40
31th	40	15	31	(40+15+31+31) mod 72=117 mod 72= 45
32th	45	3	32	(45+3+32+31) mod 72=111 mod 72= 39
33th	39	3	33	(39+3+33+31) mod 72=106 mod 72=34
34th	34	3	34	(34=3+34+31) mod 72= 102 mod 72= 30
35th	30	15	35	(30+15+35+31) mod 72= 111 mod 72= 39

In order to calculate the 36 element we continue with the fourth group of the original string. Table 7 illustrates these calculations.

Table 7: Results of Pseudorandom Items of the Fourth Part of the Original Generated Strings

Current Element	Predecessor Value	Successor value	Position	Results
36th	39	11	36	(39+11+36+31) mod 72= 117 mod 72= 45
37th	45	11	37	(45+11+37+31) mod 72= 124 mod 72= 52
38th	52	7	38	(52+7+38+31) mod 72=128 mod 72= 56
39th	56	7	39	(56+7+39+31) mod 72= 133 mod 72= 61
40th	61	11	40	(61+11+40+31) mod 72= 143 mod 72= 71
41th	71	11	41	(71+11+41+31) mod 72= 154 mod 72= 10
42th	10	7	42	(10+7+42+31) mod 72= 90 mod 72= 18
43th	18	7	43	(18+7+43+31) mod 72= 99 mod 72= 27
44th	27	11	44	(27+11+44+31) mod 72= 113 mod 72= 41
45th	41	11	45	(41+11+45+31) mod 72= 128 mod 72= 56
46th	56	7	46	(56+7+46+31) mod 72= 140 mod 72= 68
47th	68	7	47	(68+7+47+31) mod 72= 153 mod 72= 9
48th	9	11	48	(9+11+48+31) mod 72= 99 mod 72= 27
49th	27	11	49	(27+11+49+31) mod 72= 118 mod 72= 46
50th	46	7	50	(46+7+50+31) mod 72= 134 mod 72= 62
51th	62	7	51	(62+7+51+31) mod 72= 151 mod 72= 7

In the same manner we take the fifth group of the original string to complete the calculations of other elements. Table 8 contains these calculations.

Table 8: Results of Pseudorandom Items of the Fifth Part of The Original Generated Strings

Current Element	Predecessor Value	Successor value	Position	Results
52^{th}	7	11	52	(7+11+52+31) mod 72= 101 mod 72= 29
53^{th}	29	7	53	(29+7+53+31) mod 72= 120 mod 72= 48
54^{th}	48	7	54	(48+7+54+31) mod 72= 140 mod 72= 68
55^{th}	68	7	55	(68+7+55+31) mod 72= 161 mod 72= 17
56^{th}	17	11	56	(17+11+56+31) mod 72= 115 mod 72= 43
57^{th}	43	7	57	(43+7+57+31) mod 72= 138 mod 72= 66
58^{th}	66	7	58	(66+7+58+31) mod 72= 162 mod 72= 18
59^{th}	18	7	59	(18+7+59+31) mod 72= 115 mod 72= 43

The final group of the original string will be used to calculate the remaining elements as shown in table 9.

Table 9: Results of Pseudorandom Items of the Sixth Part of the Original Generated Strings

Current Element	Predecessor Value	Successor Value	Position	Results
60^{th}	43	3	60	(43+3+60+31) mod 72= 137 mod 72= 65
61^{th}	65	3	61	(65+3+61+31) mod 72= 160 mod 72= 16
62^{th}	16	3	62	(16+3+62+31) mod 72= 112 mod 72= 40
63^{th}	40	15	63	(40+15+63+31) mod 72= 149 mod 72= 5
64^{th}	5	3	64	(5+3+64+31) mod 72= 103 mod 72= 31
65^{th}	31	3	65	(31+3+65+31) mod 72= 130 mod 72= 58
66^{th}	58	3	66	(58+3+66+31) mod 72= 158 mod 72= 14
67^{th}	14	15	67	(14+15+67+31) mod 72= 127 mod 72= 55
68^{th}	55	3	68	(55+3+68+31) mod 72= 157 mod 72= 13
69^{th}	13	3	69	(13+3+69+31) mod 72= 116 mod 72= 44
70^{th}	44	3	70	(44+3+70+31) mod 72= 148 mod 72= 4
71^{th}	4	15	71	(4+15+71+31) mod 72= 121 mod 72= 49
72^{th}	49	11	72	(49+11+72+31) mod 72= 163 mod 72=19

Note: The predecessor of the last element is rounded to take the value of the first element.

Conclusion

This proposed system generates a more secure pseudorandom numbers. The importance of providing more secrecy is that these numbers are used to generate the cryptographic keys especially in sensitive applications. The main step for generating secure pseudorandom numbers is that using different context-free grammars. Each grammar has its own rules for generating a grammar corresponding to that grammar. This language generates infinite items of strings but the proposed system implement an agreement factor to choose the designated items. So agreement factor is one of the most methods used to enhance the secrecy. Then this output from the first chosen grammar is submitted as an input to another grammar with different rules. In the step also the authenticated user uses another agreement factor. Another agreement factor is used also to choose the number of grammars. Then these alphabetic strings are converted to numerical values. To hide the relations among these numeric items, the system takes the inverse for each group of items modular the number of generated items. The best step is that in order to generate each item, the system depends on the predecessor value and successor value of that item with agreement factor. So we have different agreement factors which in turns strengthen the secrecy of the system three times multiplied by the nature of each factor.

Refrences

[1] R. Anderson, "Two remarks on public key cryptology," Invited Lecture, In: the 4th ACM Conference on Computer and Communications Security, 1997.

[2] M. Bellare, S. Miner, "A forward-secure digital signature scheme," In: Wiener, M.J. (ed.) Advances in Cryptology-CRYPTO'99. LNCS, vol. 1666, pp. 431–448. Springer, Heidelberg, 1999.

[3] M. Abdalla, L. Reyzin, "A new forward-secure digital signature scheme," In: Okamoto, T. (ed.) Advances in Cryptology- ASIACRYPT 2000. LNCS, vol. 1976, pp. 116-129. Springer, Heidelberg, 2000.

[4] H. Krawczyk, "Simple forward-secure signatures for any signature scheme," In: the 7th ACM Conference on Computer and Communications Security. pp. 108-115. ACM Press, New York, 2000.

[5] G. Itkis, L. Reyzin, "Forward-secure signatures with optimal signing and verifying," In: Kilian, J. (ed.) Advances in Cryptology-CRYPTO 2001. LNCS, vol. 2139, pp. 499–514. Springer, Heidelberg, 2001.

[6] A. Kozlov, L. Reyzin, "For ward-secure signatures with fast key update," In: Cimato, S., Galdi, C., Persiano, G. (Eds.) the Proc of security in communication Networks. LNCS, vol. 2576, pp. 247-262. Springer, Heidelberg, 2002.

[7] T. Maklin, D. Micciancio, S. Miner, "Efficient generic forward-secure signatures with an unbounded number of time periods," In: Knudsen, L. (ed.) Advances in Cryptology-EUROCRYPT 2002. LNCS, vol. 2332, pp. 400-417. Springer, Heidelberg, 2002.

[8] C. Gentry, A. Silverberg , "Hierarchical ID-based cryptography," In: Zheng, Y. (ed.) Advances in Cryptology-Asiacrypt 2002. LNCS, vol. 2501, pp. 548- 566. Springer, Heidelberg, 2002.

[9] F. Hu, C. H. Wu, J. D. Irwin, "A new forward-secure signature scheme using bilinear maps," Cryptology ePrint Archive, Report 2003/188, 2003.

[10] B.G. Kang, J. H. Park, S.G. Halm, "A new forward-secure signature scheme," Cryptology ePrint Archive, Report 2004/183, 2004.

[11] J. Yu, F. Y. Kong, X. G. Cheng, R. Hao, G. W. Li, "Construction of Yet Another Forward-secure Signature Scheme Using Bilinear Maps," In: the second international conference on provable security (ProvSec 2008). LNCS, vol. 5324, pp. 83-97. Springer, Heidelberg, 2008.

[12] X. Boyen, H. Shacham, E. Shen, B. Waters, "Forward- secure Signatures with Untrusted Update," In: the 13th ACM conference on Computer and communications security. pp. 191-200. ACM Press, New York, 2006.

[13] B. Libert, J. Jacques, M. Yung, "Forward-Secure Signatures in Untrusted Update Environments: Efficient and Generic Constructions," In: the 14th ACM conference on Computer and communications security. pp. 266-275. ACM Press, New York, 2007.

[14] M. Bellare, B. Yee, "Forward-security in private-key cryptography," In: Joye, M. (Ed.) Topics in Cryptology-CT-RSA 2003. LNCS, vol. 2 612, pp. 1-18. Springer, Heidelberg, 2003.

[15] R. Canetti, S. Halevi, J. Katz, "A forward-secure public-key encryption scheme," In: Biham, E. (ed.) Advances in Cryptology-EUROCRYPT 2003. LNCS, vol. 2656, pp. 255-271. Springer, Heidelberg, 2003.

[16] M. Abdalla, S. Miner, C. Namprempre, "Forward-secure threshold signature schemes," In: Naccache, D. (ed.) Topics in Cryptology–CT-RSA 2001. LNCS, vol. 2020, pp. 441-456. Springer, Heidelberg, 2001.

[17] Z. J. Tzeng, W.G. Tzeng, "Robust forward signature schemes with proactive security," In: Kim, K. (ed.) Public-Key Cryptography (PKC 2001). LNCS, vol. 1992, pp. 264- 276. Springer, Heidelberg, 2001.

[18] H. Wang, G. Qiu, D. Feng, G. Xiao, "Cryptanalysis of Tzeng-Tzeng Forward-Secure Signature Schemes," IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences E89-A(3), 822-825, 2006.

[19] J. Yu, F. Y. Kong, R. Hao, "Forward-secure Threshold Signature Scheme from Bilinear Pairings," In: Wang, Y., Cheung, Y., Liu, H. (Eds.) the Second International Conference on Computational Intelligence and Security. LNAI, vol. 4456, pp. 587-597. Springer, Heidelberg, 2007.

[20] Y. Dodis, J. Katz, S. Xu, M. Yung, "Key-insulated public key cryptosystems," In: Knuds en, L. (ed.) Advances in Cryptology- Eurocrypt 2002. LNCS, vol. 2332, pp. 65-82. Springer, Heidelberg, 2002.

[21] Y. Dodis, J. Katz, S. Xu, M. Yung, "Strong key-insulated signature scheme," In: Desmed t, Y. (ed.) Advances in Public key Cryptography-PKC 2003. LNCS, vol. 2567, pp. 130-144. Springer, Heidelberg, 2003.

[22] Y. Zhou, Z. Cao, Z. Chai, "Identity Based Key Insulated Signature," In: Chen, K., Deng, R., Lai, X., Zhou, J. (Eds.) the Second International Conference Information Security Practice and Experience (ISPEC 2006). LNCS, vol. 3903, pp. 226-234. Springer, Heidelberg, 2006.

[23] B. Libert, J. Quisquater, M. Yung, "Parallel Key-Insulated Public Key Encryption With out Random Oracles," In: Okamoto, T., Wang, X. (Eds.) Advances in Public Key Cryptography– PKC 2007. LNCS, vol. 4450, pp. 298-314. Springer, Heidelberg, 2007.

[24] Y. Dodis, J. Katz, S. Xu, and M. Yung, "Strong key-insulated signature scheme," in Proceedings of the Public-Key Cryptography , 2003, pp. 130-144.

[25] J. Weng, X. X. Li, K. F. Chen, and S. L. Liu, "Identity-based parallel key-insulated signature without random oracles," Journal of Information Science and Engineering Vol. 24, 2008, pp. 1143-1157.

[26] G. Itkis and L. Reyzin, "SiBIR: Signer-base intrusion-resilient signatures," in Proceedings of Cryptology – Crypto , 2002, pp. 499-514.

[27] Z. Gong, X. X. Li, D. Zheng, and K. F. Chen, "A generic construction for intrusion- resilient signatures from linear feedback shift register," Journal of Information Science and Engineering, Vol. 24, 2008, pp. 1347-1360.

[28] X. Boyen, H. Shacham, E. Shen, and B. Waters, "Forward-secure signatures with untrusted update," in Proceedings of the 13th ACM Conference on Computer and Communications Security , 2006, pp. 191-200.

[29] B. Libert, J. Quisquater, and M. Yung, "Forward-secure signatures in untrusted up- date environments: efficient and generic constructions," in Proceedings of the 14th ACM Conference on Computer and Communications Security , 2007, pp. 266-275.

[30] T. Malkin, D. Micciancio, and S. Miner, "Efficient generic forward-secure signatures with an unbounded number of time periods," in Proceedings of Cryptology –Eurocrypt, 2002, pp. 400-417.

[31] U. Juan and A. Reggia" A Framework for the Comparative Study of Language", Evol Psychol July 2013 vol. 11 no. 3 , 2013.

[32] C. Mikael," Structural Synthesis using a Context Free Design Grammar Approach",Generative Art Conference GA, 2009.

[33] Y. Tomoyuki, "Pseudorandom generators against advised context-free languages", Theoretical Computer Science, Volume 613, 1 February 2016, Pages 1–27.

[34] V. Sang-Ho and Y. Kee-Young," An Efficient PRNG Based on the Hybrid between One- and Two-Dimensional Cellular Automata",Information Technology: New Generations, ITNG '09. Sixth International Conference on, 2009.