ISSN: 0067-2904

Towards Generating Robust Key Based on Neural Networks and Chaos Theory

Abeer Tariq Maolood*, Ala'a Talib Khudhair
Department of Computer Science, University of Technology, Baghdad, Iraq

Abstract

There are large numbers of weakness in the generated keys of security algorithms. This paper includes a new algorithm to generate key of 5120 bits for a new proposed cryptography algorithm for 10 rounds that combine neural networks and chaos theory (1D logistic map). Two methods of neural networks (NN) are employed as Adaline and Hopfield and the results are combined through several sequential operation. Carefully integrating high quality random number generators from neural networks and chaos theory to obtain suitable key for randomness and complexity.

Keywords: key generation, neural network, Adaline, Hopfield, chaos theory

توليد مفتاح قوي يعتمد على الشبكات العصبية ونظرية الفوضى

عبير طارق*، الاء طالب خضير
قسم علوم الحاسبات، جامعة التكنولوجيه، بغداد، العراق

الخلاصة

هناك عدد كبير من نقاط الضعف في المفاتيح المولده لخوارزميات الأمن .في البحث تم تصميم خوارزمية جديده أ لتوليد مفاتيح بطول ٥١٢ بت لخوارزمية جديدة مقترحة تتالف من (10 rounds) باستخدام تقنيات متعددة التي تجمع بين الشبكات العصبية ونظرية الفوضى (1D logistic map) .تم استخدام طريقتين للشبكات العصبية هي Adaline, Hopfield . تم تعديل نتائج ا لطريقتين بواسطه الدمج بين نتائج الطريقتين ولعدة خطوات متتالية. بعد دمج نتائج الشبكات العصبية ونظرية الفوضى للوصول إلى مفتاح أكثر عشوائية ومناسب للعشوائية والتعقيد.

1. Introduction

Cryptography is a technique which converts a given message into an unreadable format which can be seen as a noise by the third parties. It is a vital part of a safe communication process .Cryptography is a skill of sending the data in such a form that only those for whom it is intended can read it. Any cryptography algorithm needs to generate key that can be used in the steps of cryptography [1]. In this paper a new technique is employed for key generation by using Artificial Neural Network (ANN) as Hopfield and Adaline. Hopfield invented by the physician (John Hopfield) in 1982 which has these features: unsupervised learning, associative memory, full connection, single layer, feedback, fixed weight, bipolar, linear activation function and input nodes equal to output nodes. While Adaline NN invented by psychologist Donald Hebb postulated in 1949 [1]. Generating random numbers are mainly used to generate secret keys or random sequences and it can be carried out by many different techniques such as chaotic maps. In recent years many pseudo random numbers or sequences generators are proposed based on chaotic maps. The most important thing in any cryptography algorithms is having secure key and there are several requirements for achieving secure key. One of these requirements is randomize and key length, etc. [2].

* Email: 110032@uotechnology.edu.iq

Related work

This section discusses important previous works of Hopfield, Adaline and chaos:

1. Marat Akhmet & Mehmet Onur Fen, 2014, has applied a Generation of cyclic/toroidal chaos by Hopfield neural networks. Have proposed system to discuss the appearance of cyclic and toroidal chaos in Hopfield neural networks [3].
2. Terence Kwok & Kate Smith& Lipo Wang, 2013, Incorporating Chaos into the Hopfield Neural network for Combinatorial Optimization, the proposed method consisted of adding chaotic noise to a Hopfield network. The N-Queen problem is used as an application to test and compare the performance and robustness of the three methods. The traditional simulated annealing is also included for comparison in order to contrast the effectiveness of the various approaches [4].
3. Delhi Technological & Swati Mishra, 2013, Public key cryptography produced from neural networks and genetic algorithms. This paper made use of Hebbian learning rule to train the ANN of both sender and receiver machines. In the field of Public Key Cryptography (PKC), Pseudo Random Number Generator (PRNG) are widely used to generate unique keys and random numbers used in ANN which are found to possess many types of possible attacks. It is essential for a key to possess randomness for key strength and security. The PKC was generated by using of ANN through Genetic Algorithm (GA) [5].

2. Theoretical background

2.1 Hopfield NN

Hopfield Neural Network is an example of the network with feedback (so-called recurrent network), where outputs of neurons are connected to input of every neuron by means of the appropriate weights. Of course there are also inputs which provide neurons with components of test vector.[6]

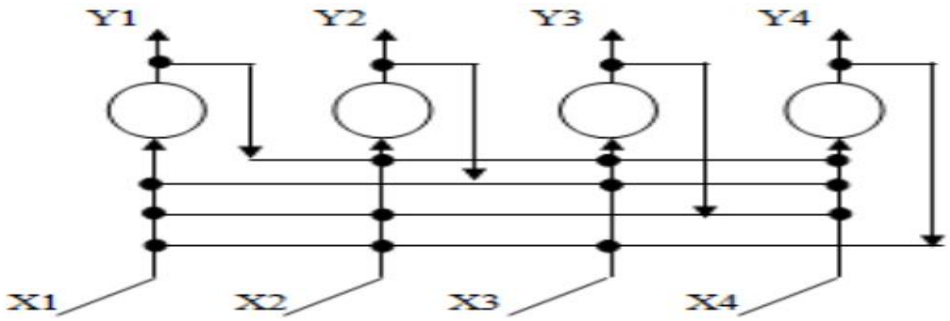

Figure 1- Hopfield NN Topology

Hopfield NN Learning

Initialize (N) no. of input node.

$$F(x) = \begin{cases} 1 & if\ x > 0 \\ -1 & if\ x \leq 0 \end{cases}$$

-Convert 0 to -1

-Compute weight matrix.

$$W_{i,j} = \begin{cases} 0 & if\ i = j \\ \sum_{i=1}^{p} x_i \times x_j & otherwise \end{cases}$$ where i,j =1.2….N[6]

Algorithm 1: Hopfield NN

Input: array 2D of size (3,3)
Output: array 2D of size (3,3) called wait
Begin
For i=0 to 2
For j=0 to 2
Step 1: if array (i, j) = 0 then
array (i, j) = -1
Step 2: for each input of vector x, do step 3
step 3: compute wait (i, j)
If i=j then
wait (i, j) =0
Else

$Wait\ (i,j)\ = \sum x_i\ x_j$
Step4:final wait(i, j)
Next j
Next i
End

2.2 Adaline neural network

Adaline which stands for Adaptive Linear Neuron is a network having a single linear unit. It was developed by Widrow and Hoff in 1960. Some important points about Adaline are as follows

- It uses bipolar activation function.
- It uses delta rule for training to minimize the Mean-Squared Error (MSE) between the actual output and the desired/target output.
- The weights and the bias are adjustable.[7]

Algorithm 2: Adaline NN

Input: input matrix size (4*3), matrix wait (3*1) and desired matrix (4*1)
Output: wait of size (10*3)
Begin
Step 1 − Initialize the following to start the training

- Weights
- Bias
- Learning rate α

For easy calculation and simplicity, weights and bias must be set to 0 and the learning rate must be set to 1.
Step 2 − Continue step 3-8 when the stopping condition is not true.
Step 3 − Continue step 4-6 for every bipolar training pair **s:t**.
Step 4 − Activate each input unit as follows −

$$X_i = s_i\ (i = 1\ to\ n)$$

Step 5 − Obtain the net input with the following relation −
Y_{in} =b+$\sum_i^n x_i * w_i$
Here **'b'** is bias and **'n'** is the total number of input neurons.
Step 6 − Apply the following activation function to obtain the final output −

$$F(\ y_{in}) = \begin{cases} 1 & if\ \ y_{in} \geq 0 \\ -1 & if\ \ y_{in} < 0 \end{cases}$$

Step 7 − Adjust the weight and bias as follows −
Case 1 − if **y** ≠ **t** then,
$w_i(new)= w_i\ (old)+\alpha(t- y_{in})x_i$
$b(new)=b(old)+\alpha(t- y_{in})$
Case 2 − if **y = t** then,
$w_i\ (new)= w_i\ (old)$
$b(new)=b(old)$
Here **'y'** is the actual output and 't' is the desired/target output.
$(t - y_{in})$ is the computed error.
Step 8 − Test for the stopping condition, which will happen when there is no change in weight or the highest weight change occurred during training is smaller than the specified tolerance.
END

2.3 Random Number Generator Using Chaos Theory

Chaos theory is an area of mathematical science that studies the dynamic systems behavior; it is a good choice to use in designing of random number generator. In fact, chaotic system behaviors like; highly sensitive to initial states, mix up attribute, deterministic nature and also cannot predict the long term returns , so these properties are valuable in cryptography. Logistic equation of chaos which is explained below can be used to generate large pseudo random numbers, increase the randomness of produced keys and make the keys more secure [8].

2.3.1 Chaos Logistic Equation

It is a dynamic system second order deference equation; the standard form of the logistic equation is given by:

F (Yi) = R Yi (1- Yi) (1)

Where Yi is called the iteration of Y0 (or population) and should by in subinterval [0, 1].
It all started to dawn on people when in 1960 a man named Edward Lorenz created a weather-model on his computer. Chaos is the study of complex systems with extreme sensitivity to slight changes in

conditions parameters. Chaos is the science of surprises, of the nonlinear and the unpredictable. It teaches us to expect the unexpected. While most traditional science deals with supposedly predictable phenomena like gravity, electricity, or chemical reactions, Chaos theory deals with nonlinear things that are effectively impossible to predict or control. [9].

Algorithm 3 : 1D logistic map chaos theory

Input:

- R =3.6
- y =0.2

Output:

- binary string of length 5120 bits

Processing:

Begin

For i=1 to 120

Step1: Applying equation of 1d logistic map

Step2: Remove the negative signal

Step3: Cut 17 numbers after the comma

Step4: Save the numbers of each execution in buffer

Next for

Step5: Cuts final result to sub strings each of length 3, each sub string take mod 256; convert the result to binary of 8 bits and aggregation the results in binary string

Step6: present final result of length 5120 bits

End

3. Proposed system design

Below the steps of the key generation method**:**

1: Input matrix size (4*3), matrix wait (3*1) and desired matrix (4*1) to apply adaline neural network for 10 rounds only to get the wait of adaline.

2: Input matrix size (3*3), applied Hopfield neural network on this matrix to present wait of Hopfield, The waiting from Hopfield can be broken or guessed so every number of waiting numbers is added with three serial numbers resulting from chaos theory (from the numbers collected from the chaos results saved in chaos buffer).

3: Create matrix k (10*3) by applying number of procedures will be carried out on the wait of adaline (step 1) .First, disposal from the negative signal. Second, cut two digits after comma. Third, apply the following equation to each number

K (i, j) = ((wait of adaline (i, j) +255)*(512 ^ f))

Where f=3, 4, and 5

If number of column is 1 then f=3

If number of column is 2 then f=4

If number of column is 3 then f=5

4: Create matrix u (3*3) by applying number of procedures will be carried out on the wait of Hopfield (step 2) .By apply the following equation to each number

U (i, j) = (wait of Hopfield (i, j) +255^6)

5: Create an array **f**(10*3) by performing several operations between the last modification on the wait of Hopfield (matrix **u** (3, 3)) and the last modification on the wait of adaline (matrix **k** (10, 3))

First row of f equals from adding first row of Hopfield with first row of adaline.

Second row of f equals from adding second row of Hopfield with second row of adaline.

Third row of f equals from adding third row of Hopfield with third row of adaline.

Four row of f equals from adding first column of Hopfield with fourth row of adaline.

Five row of f equals from adding second column of Hopfield with five row of adaline.

Six row of f is adding third column of Hopfield with six row of adaline.

Seven row of f equals from adding Hopfield array (1, 0) with seven row of adaline.

Eight row of f equals from adding Hopfield array (0, 1) with eight row of adaline.

Nine row of f equals from adding Hopfield array (1, 2) with nine row of adaline.

Ten row of f equals from adding Hopfield array (2, 1) with ten row of adaline.

6: Create an array m (10*1) consist of

First row of m equals to first row of f array.

Second row of m equals to second row of f accumulated with the previous row.

Third row of m equals to third row of f accumulated with the previous row.
Four row of m equals to four row of f accumulated with the previous row.
Five row of m array is adding five row of f accumulated with the previous row.
Six row of m equals to six row of f accumulated with the previous row.
Seven row of m equals to seven row of f accumulated with the previous row.
Eight row of m equals to eight row of f accumulated with the previous row.
Nine row of m equals to nine row of f accumulated with the previous row.
Ten row of m equals to ten row of f accumulated with the previous row.
7: Convert each row of m to binary of length 64 bits so the final result is 640 bits
8: Making many operation:
8.1 Save the result of step 7 in buffer
8.2 Make DNA complement on the result of step 8.1 and add the result in buffer
8.3 Make initial permutation of DES algorithm on the result of step 8.2 and add it to the buffer
8.4 XOR operation between the results of steps 8.2 and 8.3 then add it to the buffer
8.5 Add with mode 256 between the results of steps 8.3 and 8.4 then add it to the buffer
8.6 Make initial permutation of DES algorithm on the result of step 8.5 then add it to the buffer
8.7 XOR operation between the results of steps 8.5 and 8.6 then add it to the buffer
8.8 Make DNA complement on the result of step 8.7 then add it to the buffer
-Now the length of binary string saved in buffer is 5120 bit
9: Apply chaos theory (1D logistic map) to present result of length 5120 bits
10: XOR operation between the result of step 8 and 9 to present the final key
11: Present final key of length 5120 bits

Example: - about (add with mode 256)

Suppose a="11111111"
Suppose b="00000111"
Solve:
Convert a to integer=255 and Convert b to integer=7
= (255+7) mod 256=262 mod 256=6
Convert the result to binary="00000110"

Algorithm 4: New dynamic key generator using NN and chaos theory

Input:
- input matrix size (4*3) for adaline
- wait matrix (3*1) for adaline
- desired matrix (4*1) for adaline
- input matrix size (3*3) for Hopfield

Output:
- key of 5120 bits

Process:
Begin
Step1: Applied adaline algorithm to give the wait matrix
Step 2: Applied Hopfield algorithm to give the wait matrix which updated based on the result of chaos theory
Step 3: Create matrix k (10*3) by updating adaline wait
Step 4: Create matrix u (3*3) by updating Hopfield wait
Step 5: Create matrix f (10*3) by performing several operations between the last modification on the wait of Hopfield(u(3, 3)) and the last modification of the wait of adaline(k(10 ,3))
Step 6: Create matrix m (10*1)
Step 7: Convert each row of m matrix to binary of length 64 bits so the final result is 640 bits
Step 8: Making many operation
8.1 Save the result of step 7 in a buffer
8.2 Make DNA complement on the result of step 8.1 and add it to the buffer
8.3 Make initial permutation of DES algorithm on the result of step 8.2 and add it to the buffer
8.4 XOR operation between the results of step 8.2 and 8.3 then add it to the buffer
8.5 Add with mode 256 for the results of step 8.3 and 8.4 and add it to the buffer
8.6 Make initial permutation of DES algorithm on the result of step 8.5 and add it to the buffer
8.7 XOR operation between the result of step 8.5 and 8.6 then add the result to the buffer
8.8 Make DNA complement on the result of step 8.7 and add the result to the buffer

-Now the length of binary string saved in buffer is 5120 bit
Step 9: Apply chaos theory (1D logistic map) to give result of length 5120 bits
Step 10: XOR operation between the result of step 8 and 9 to present the final key
Step 11: Present the final key of 5120 bits
End

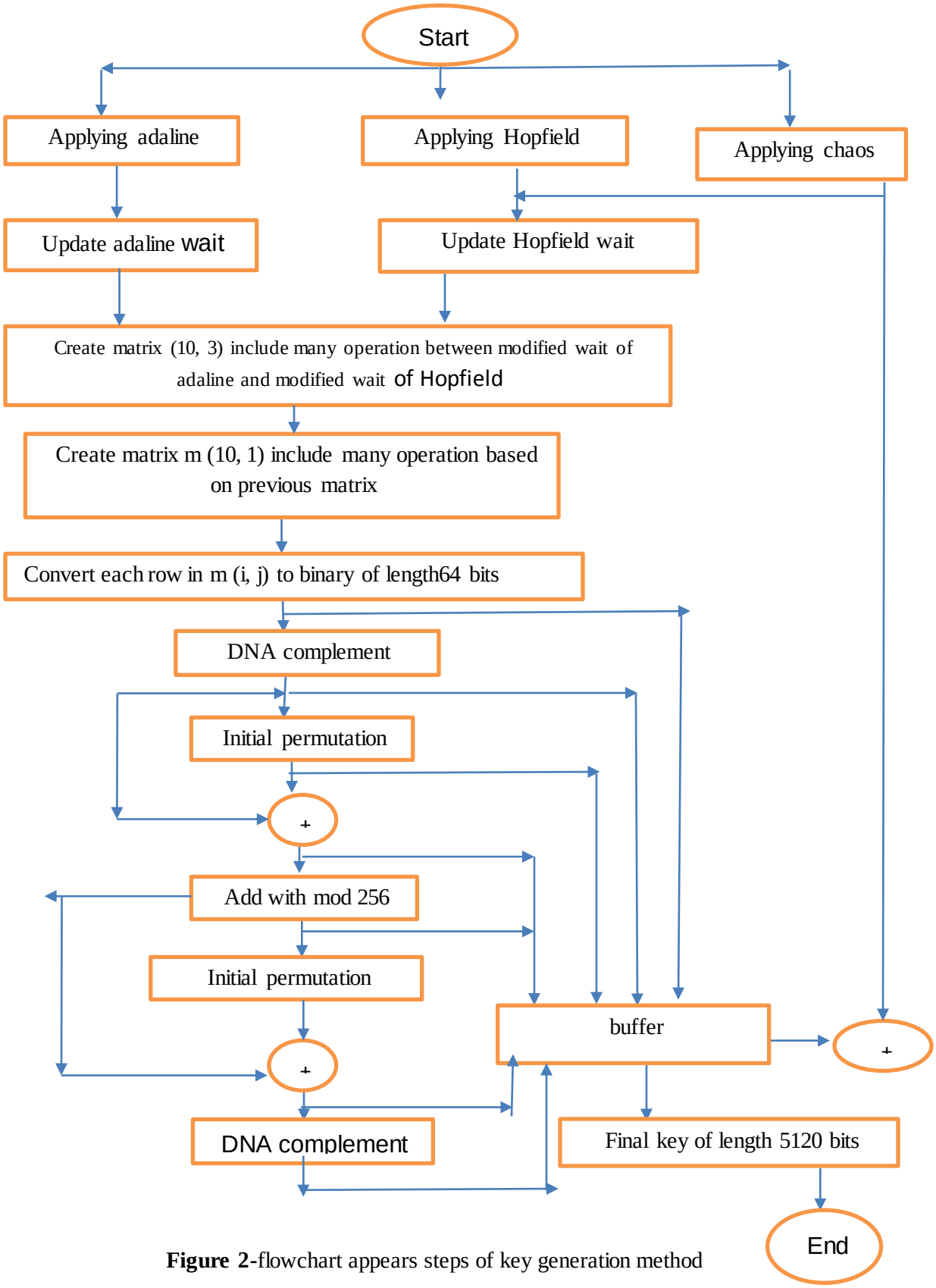

Figure 2-flowchart appears steps of key generation method

Example

Adaline is given the four different input and output combinations W0=-0.12, w1=0.4, w2=0.65

X_0	X_1	X_2	$Y_{desired}$
1	-1	-1	-1
1	-1	1	-1
1	1	-1	1
1	1	1	-1

1. First the input pattern: +1 -1 -1

Weights: -0.12 0.4 0.65

$$\text{net} = \sum_{i=1}^{n} x_i . w_i$$

= (+1*-0.12) + (-1*0.4) + (-1*0.65) = -1.17

d = desired output = -1 (for first pattern)

$$\therefore \delta = d - \text{net}$$

= -1-(-1.17) =0.17

$$\eta = 0.1$$

Also we must compute:-

$$\Delta w_{ij} = \eta \cdot \delta \cdot x_i$$

For convenience, these figures have been rounded to two places after the decimal point, so become:- $\eta \cdot \delta \cdot x_0 = (0.1 * 0.17 * +1) = 0.017 \approx = 0.02$

η □ x_1=0.1*0.17*(-1) =-0.017≈-0.02

η □ x_2=0.1*0.17*(-1) =-0.017≈-0.02

Note: - We continue to work with the rest of the input and thus we get the following results

								Δw_0	Δw_1	Δw_2
X_0	X_1	X_2	W_0	W_1	W_2	Net	d	$\eta\delta x_0$	$\eta\delta x_1$	$\eta\delta x_2$
+1	-1	-1	-0.12	0.40	0.65	-1.17	-1	0.02	-0.02	-0.02
+1	-1	+1	–0.12	0.40	0.65	0.13	-1	-0.11	0.11	-0.11
+1	+1	-1	-0.12	0.40	0.65	-0.37	+1	0.14	0.14	-0.14
+1	+1	+1	-0.12	0.40	0.65	0.93	-1	-0.19	-0.19	-0.19
total								-0.14	0.04	-0.46

$$meanj = total(\Delta wij) / p \qquad p = 4$$

$Mean_0$ = -0.14/4 =-0.035 = -0.04

$Mean_1$ = -0.04/4 =-0.01

$Mean_2$ = -0.46/4 =-0.115 = -0.12

$$\therefore W_{ij}^{new} = W_{ij}^{old} + \text{meanj}$$

W_0^{new} =-0.12+ (-0.04) =-0.16

W_1^{new} =0.40+ (0.01) = -0.41

W_2^{new} =0.65+ (-0.12) = 0.53

Continue until 10 round

No. of round	W0	W1	W2
Round1	-0.14	0.04	-0.46
Round2	-0.14	0.06	-0.46
Round3	-0.13	0.09	-0.49
Round4	-0.13	0.11	-0.49
Round5	-0.11	0.13	-0.51
Round6	-0.11	0.17	-0.53
Round7	-0.1	0.18	-0.54
Round8	-0.09	0.21	-0.57
Round9	-0.09	0.25	-0.59
Round10	-0.07	0.27	-0.61

2. Consider the following samples are inputs for Hopfield:

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

1-use Hebb rule to find the weights matrix

$$\begin{bmatrix} w_{00} & w_{01} & w_{02} \\ w_{10} & w_{11} & w_{12} \\ w_{20} & w_{21} & w_{22} \end{bmatrix}$$

$w_{00}=w_{11}=w_{22}$ =0 because i=j
w_{01}= (1*1) + (1*-1) = (-1*1) = -1=w_{10}
w_{02}= (1*1) + (1*1) = (-1*1) = 1=w_{20}
w_{12}= (1*1) + (-1*1) = (1*1) = 1=w_{21}

$$w= \begin{bmatrix} 0 & -1 & 1 \\ -1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

The waiting from Hopfield can be broken or guessed so every number of waiting numbers is add with three serial numbers resulting from chaos theory (from the numbers collected from the chaos results after the elimination of the negative signal and cut 17 rank after the comma)
The number produced from chaos is: 576879206438232902231654485 …….
The new wait is:
W_{11} =0 + 576=576
W_{12} =-1+879=878 and so on

$$\text{New } w= \begin{bmatrix} 576 & 878 & 207 \\ 437 & 232 & 903 \\ 232 & 655 & 485 \end{bmatrix}$$

3. Update the wait of adaline
W_{00}= (14+255)*(512^3) =269*134217728=36104568832
W_{10}= (14+255)*(512^3) =269*134217728=36104568832
W_{20}= (13+255)*(512^3) =268*134217728= 35970351104
W_{30}= (13+255)*(512^3) =268*134217728= 35970351104
W_{40}= (11+255)*(512^3) =266*134217728= 35701915648
W_{50}= (11+255)*(512^3) =266*134217728= 35701915648
W_{60}= (10+255)*(512^3) =256*134217728= 35567697920

W_{70}= (9+255)*(255^3) =264*134217728= 35433480192
W_{80}= (9+255)*(512^3) =264*134217728= 35433480192
W_{90}= (7+255)*(512^3) =262*134217728= 35165044736
W_{01}= (4+255)*(512^4) =259*68719476736= 17798344474624
W_{11}= (6+255)*(512^4) =261*68719476736= 17935783428096
W_{21}= (9+255)*(512^4) =264*68719476736= 18141941858304
W_{31}= (11+255)*(512^4) =266*68719476736= 18279380811776
W_{41}= (13+255)*(512^4) =268*68719476736= 18416819765248
W_{51}= (17+255)*(512^4) =272*68719476736= 18691697672192
W_{61}= (18+255)*(512^4) =273*68719476736= 18760417148928
W_{71}= (21+255)*(512^4) =276*68719476736= 18966575579136
W_{81}= (25+255)*(512^4) =280*68719476736= 19241453486080
W_{91}= (27+255)*(512^4) =282*68719476736= 19378892439552
W_{02}= (46+255)*(512^5) =301*35184372088832= 10590495998738432
W_{12}= (46+255)*(512^5) =301*35184372088832= 10590495998738432
W_{22}= (49+255)*(512^5) =304*35184372088832= 10696049115004928
W_{32}= (49+255)*(512^5) =304*35184372088832= 10696049115004928
W_{42}= (51+255)*(512^5) =306*35184372088832= 10766417859182592
W_{52}= (53+255)*(512^5) =308*35184372088832= 10836786603360256
W_{62}= (54+255)*(512^5) =309*35184372088832= 10871970975449088
W_{72}= (57+255)*(512^5) =312*35184372088832= 10977524091715584
W_{82}= (59+255)*(512^5) =314*35184372088832= 11047892835893248
W_{92}= (61+255)*(512^5) =316*35184372088832= 11118261580070912

4. Update Hopfield waits

W_{00}=w_{00} *(255^6) =576*(274941996890625) = 158366590209000000
W_{01}=w_{01} *(255^6) =878*(274941996890625)) = 241399073269968750
W_{02}=w_{02} *(255^6) =207*(274941996890625)) = 56912993356359375
W_{10}=w_{10} *(255^6) =437*(274941996890625)) = 120149652641203125
W_{11}=w_{11} *(255^6) =232*(274941996890625) = 63786543278625000
W_{12}=w_{12} *(255^6) =903*(274941996890625) = 248272623192234375
W_{30}=w_{30} *(255^6) =232*(274941996890625) = 63786543278625000
W_{31}=w_{31} *(255^6) =655*(274941996890625) = 180087007963359375
W_{32}=w_{32} *(255^6) =485*(274941996890625) = 133346868491953125

5. Create matrix (10, 3) include many operation between modified wait of adaline and modified wait of Hopfield

W_{00}= 36104568832+158366590209000000= 158366626313568832
W_{01}= 17798344474624+241399073269968750= 241416871614443374
W_{02}= 10590495998738432 + 56912993356359375= 67503489355097807
W_{10}= 36104568832+ 120149652641203125=120149688745771957
W_{11}= 17935783428096+63786543278625000 =63804479062053096
W_{12}=10590495998738432+ 248272623192234375 = 258863119190972807
W_{21}= 35970351104+63786543278625000=63786579248976104
W_{22}= 18141941858304+180087007963359375=180105149905217679
W_{23}= 10696049115004928+133346868491953125=144042917606958053
W_{30}= 35970351104+ 158366590209000000=158366626179351104
W_{31}=18279380811776+ 120149652641203125= 120167932022014901
W_{32}= 10696049115004928+ 63786543278625000=74482592393629928
W_{40}= 35701915648+ 241399073269968750=241399108971884398
W_{41}= 18416819765248 + 63786543278625000 = 63804960098390248
W_{42}= 10766417859182592 + 180087007963359375 =190853425822541967
W_{50}= 35701915648+ 56912993356359375 =56913029058275023
W_{51}= 18691697672192+ 248272623192234375=248291314889906567
W_{52}= 10836786603360256+ 133346868491953125= 144183655095313381
W_{60}= 35567697920+ 120149652641203125=120149688208901045
W_{61}= 18760417148928+120149652641203125=120168413058352053

W_{62}= 10871970975449088+120149652641203125= 131021623616652213
W_{70}= 35433480192+241399073269968750=241399108703448942
W_{71}= 18966575579136+241399073269968750=241418039845547886
W_{72}= 10977524091715584+241399073269968750 =252376597361684334
W_{80}= 35433480192+ 248272623192234375 =248272658625714567
W_{81}= 19241453486080+ 248272623192234375 =248291864645720455
W_{82}= 11047892835893248+ 248272623192234375 =259320516028127623
W_{90}= 35165044736+180087007963359375=180087043128404111
W_{91}= 19378892439552+180087007963359375= 180106386855798927
W_{92}= 11118261580070912+180087007963359375= 191205269543430287

6. Create matrix m (10, 1) include many operation based on previous matrix

$m_{00}=w_{00}+w_{01}+w_{02}$=158366626313568832+241416871614443374
+67503489355097807=467286987283110013
m10= $w_{10}+w_{11}+w_{12}+m_{00}$=910104274281907873
$m_{20}= w_{20}+w_{21}+w_{22}+m_{10}$=1298038921043059709
$m_{30}= w_{30}+w_{31}+w_{32}+m_{20}$=1651056071638055642
$m_{40}= w_{40}+w_{41}+w_{42}+m_{30}$= 2147113566530872255
$m_{50}= w_{50}+w_{51}+w_{52}+m_{40}$= 2596501565574367226
$m_{60}= w_{60}+w_{61}+w_{62}+m_{50}$= 2967841290458272537
$m_{70}= w_{70}+w_{71}+w_{72}+m_{60}$=3703035036368953699
$m_{80}= w_{80}+w_{81}+w_{82}+m_{70}$= 4458920075668516344
$m_{90}= w_{90}+w_{91}+w_{92}+m_{80}$= 5010318775196149669

7. Applying many operations such that:

7.1-Convert each of (m_{00}, m_{10}, m_{20}, m_{30}, m_{40}, m_{50}, m_{60}, m_{70}, m_{80}, m_{90}) to binary of 64 bits, the result is:
0000011001111100001000110000101000000010100……..

7.2- Applying DNA complement on the result of step 7.1, the result is:
1111100110000011110111001111010111111101011……...

7.3-Applying initial permutation of 64 bits on the result of step 7.2, the result is:-
0111110100111101010011000110101111111111010…..

7.4-Applying XOR operation between step 7.2 and step 7.3, the result is:
1000010010111110100100001001111000000101100…….

7.5-Applying add with mod 256 between result of step 7.3 and 7.4, the result is:-
0000000111111011110111000000100100000100111 0……

7.6-applying initial permutation of 64 bits on the result of step 7.5, the result is:-
1010011000000110010101000110101100100110101000 1…..

7.7-applying XOR operation between step 7.5 and 7.6, the result is:-
1010011111111101100010000110001000100010010000……

7.8-applying DNA complement on the result of step 7.7, the result is:
0101100000000010011101111001110111011011011111……..

8. Aggregation the results of steps (7.1, 7.2,… 7.8) to present the binary string of 5120 bit

9. Applying chaos method (1D Logistic map), the result is:
0100000000000000101011110110111100011000100110001100111……

10. Applying XOR operation between step 8 and 9, the result is:-
0100011001111100100011000110010100011101101101001 10……

11. Present key of length 5120 bits

12. Implementation and evaluation result

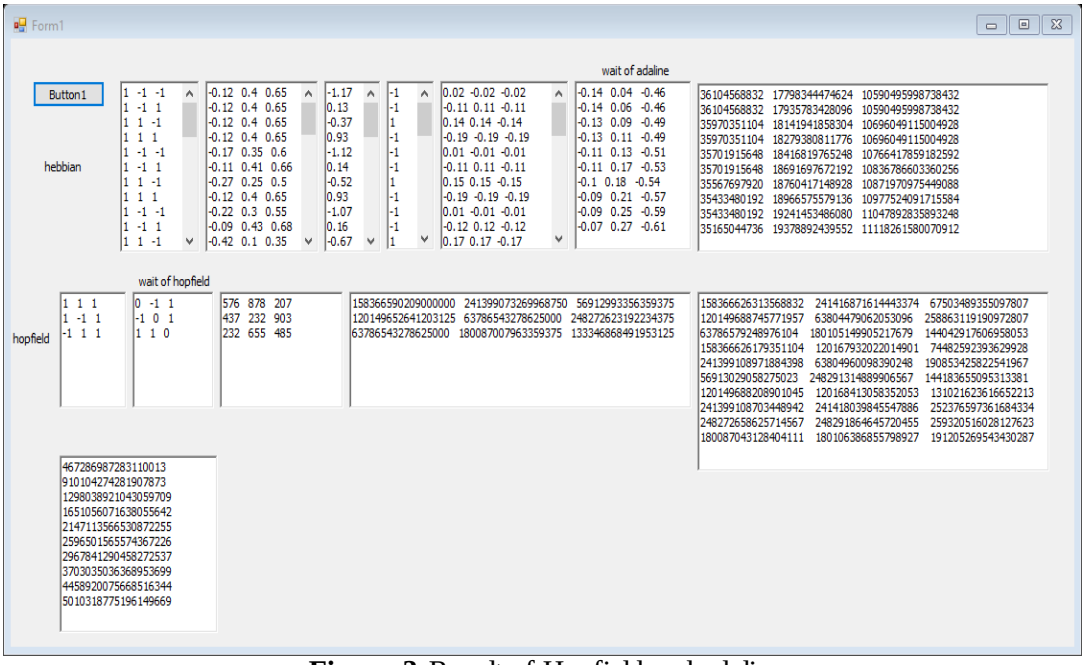

Figure 3-Result of Hopfield and adaline

This figure appears wait of adaline, wait of Hopfield, modified adaline wait, modified Hopfield wait and combine between two waits to generate matrix m (10*1)

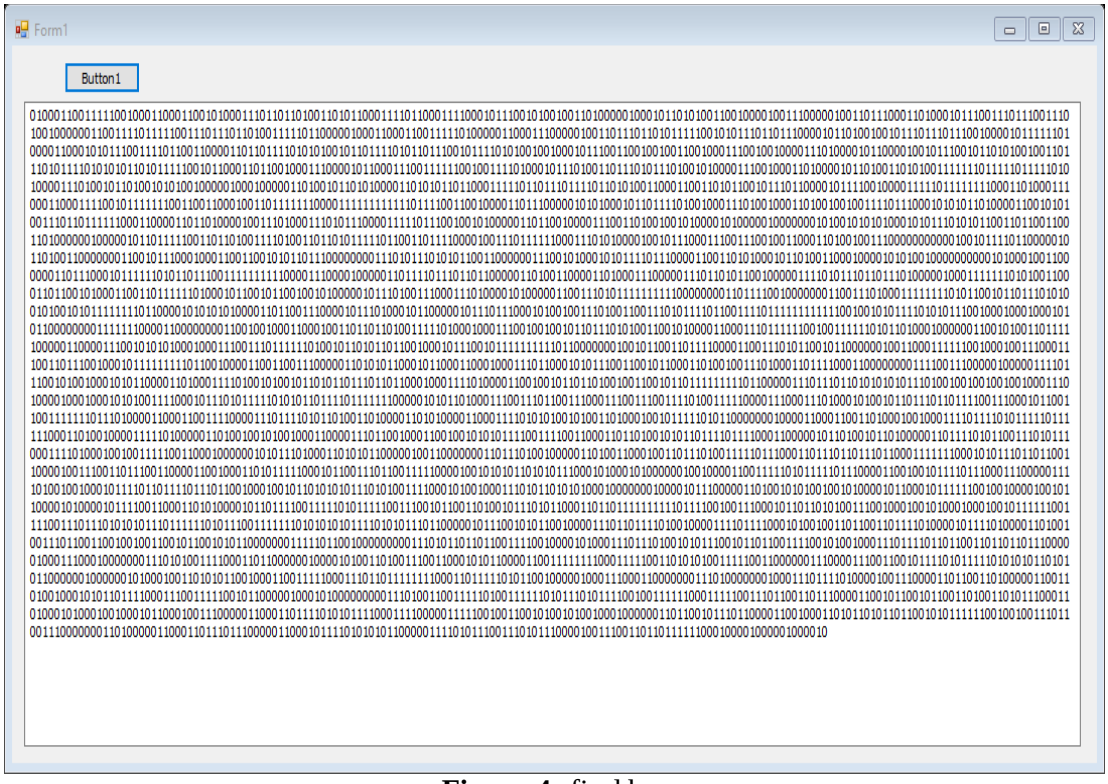

Figure 4- final key

This figure appears the final result of key generation in binary of length 5120 bits, after applied many operation in order to access the highest random key

Based on the results of the main 5 statistic tests which have been performed on (Towards generation robust key based on neural networks and chaos theory), as shown in table (1). The suggested modified method showed better results, particularly on the test of frequency, considered as the most important one of the five tests.

Table 1-Results of the 5 Tests that have been applied to the proposed key (5120 bits)

Statistical Test					
Tests		**Freedom Degree**	**First sample**	**Second sample**	**Third Sample**
Frequency Test		Must be <= 3.84	Pass =0.013	Pass =1.188	Pass =2.813
Run Test	**T0**	Must be <=19.391	Pass =3.856	Pass =11.731	Pass =11.450
	T1		Pass =5.622	Pass =11.686	Pass =9.264
Poker Test		Must be <=11.1	Pass =2.572	Pass =2.659	Pass =5.975
Serial Test		Must be <=7.81	Pass =0.219	Pass =5.034	Pass =3.269
Auto Correlation Test	**Shift No. 1**	Must be <= 3.84	Pass =0.086	Pass =0.877	Pass =0.267
	Shift No. 2		Pass =0.028	Pass =0.113	Pass =1.070
	Shift No. 3		Pass =2.408	Pass =0.016	Pass =1.618
	Shift No. 4		Pass =1.955	Pass =3.831	Pass =0.704
	Shift No. 5		Pass =1.159	Pass =1.220	Pass =0.057
	Shift No. 6		Pass =1.190	Pass =1.251	Pass =0.050
	Shift No. 7		Pass =0.728	Pass =2.863	Pass =0.071
	Shift No. 8		Pass =1.014	Pass =0.153	Pass =1.515
	Shift No. 9		Pass =0.240	Pass =1.284	Pass =0.002
	Shift No. 10		Pass =0.132	Pass =0.003	Pass =0.013

13. Conclusions

1. When the initial values of the chaos theory change, all the results change according to the initial values
2. It is impossible to think of the initial values of the chaos theory because its number of probability is too large
3. Each step of the key generation depends on the results of the previous step and this has a big role in the difficulty of breaking the key
4. It was concluded that neural networks did not give random results because they are trained so a chaos equation was used to generate clutter numbers to increase clutter in the key and to form a random and more secure key.
5. Artificial intelligence algorithms have been used with chaos in the producing of keys and are one of the new ways that not mentioned before.

References

1. Vittorio Capecchi, Massimo Buscema, Pierluigi Contucci and Bruno D'Amore. **2010.**"*Applications of Mathematics in Models, Artificial Neural Networks and Arts*", Springer Science+Business Media B.V, New York.
2. Ben Krose and Patrick van der Smagt. **1996.** "*An Introduction to Neural Networks*", University of Amsterdam, eight editions, November.
3. Marat Akhmet & Mehmet Onur Fen. **2014.** " *Generation of cyclic/toroidal chaos by Hopfield neural networks*", Department of Mathematics, Middle East Technical University, Ankara, Turkey.
4. Terence Kwok & Kate Smith& Lipo Wang. **2007**. "*Incorporating Chaos into the Hopfield Neural network for Combinatorial Optimization* ", School of Business Systems, Monash University Clayton, Victoria 3168, Australia., 2007.
5. Delhi Technological & Swati Mishra. **2013**. "Public key cryptography using neural networks and genetic algorithms ", Sixth International Conference on Contemporary Computing (IC3), Noida, India, 26 September 2013
6. Lakhmi Jain, Ph.D.,University of South Australia,Anna Maria Fanelli, Ph.D. **2012.** " *Recent Advances In Artificial Neural Networks Design and Applications*", International Series on Computational Intelligence.
7. Martin Anthony and Peter L. Bartlett, **2013.** "*Neural Network Learning: Theoretical Foundations*", Cambridge University Press, Published in the United States of America by Cambridge University Press, New York.
8. Alexander I. Galushkin, **2009.** "*Neural Networks Theory*", Moscow Institute of Physics & Technology, Department Neurocomputers, Russia.
9. Sprott, J. C. **1997.** " *Some simple chaotic jerk functions*", Department of Physics, University of Wisconsin, Madison, Wisconsin 53706.