Design and Implementation of Stream Cipher Using Neural Network

تصميم وتنفيذ تشفير انسيابي باستخدام الشبكات العصبية

د.صديق يوسف امين	د.مازن زكي عثمان	د.صفوان عمر حسون	د.مؤيد عبد الرزاق
استاذ	استاذ مساعد	مدرس	أستاذ مساعد
الجامعة التكنولوجية	الكلية التقنية/ موصل	جامعة الموصل	الهيئة العراقية للحاسبات والمعلوماتية

Abstract

The centaral problem in stream cipher cryptograph is the the difficulty to generate a long unpredicatable sequence of binary signals from short and random key. Unpredicatable sequence are desirable in cryptography because it is impossible, given a reasonable segment of its signals and computer resources, to find out more about them. Pseudorandom bit generators have been widely used to construct these sequences.

The paper presents a PN sequence generator that uses neural network. Computer simulation tests have been carried out to check the randomness of the generated through statistical tests. There tests have shown the successful PN sequence generator passes all the recommended tests. The paper also proposes and validates the data encryption and decryption process using neural network instead of using traditional methods (Exclusive or). This task increases the difficulty in the breaking the cipher.

الملخص

تتمركز مشكلة التشفير الانسيابي في صعوبة توليد اشارات ثنائية متتابعة غير متوقعة من مفتاح عشوائي قصير. ان توليد متسلسلات غير متوقعة يجعل من عملية التشفير عملية مرغوب ا لانه من خلال هذه المتسلسلات من غير الممكن التنبؤ بالحصول على مقاطع اشارات وتحديد موارد الحاسبات للوصول الى معلومات اكثر قد تؤدي الى فشل عملية التشفير. ان مولدات الارقام العشوائية الوهمية استخدمت بشكل واسع لبناء تلك المتسلسلات.

في هذا البحث تم بناء مولد ارقام عشوائي وهمي باستخدام الشبكات العصبية. استخدم برنامج محاكاة حاسوبي لفحص عشوائي المفتاح باستخدام الاختبارات الاحصائيه. تبين هذه الاختبارات نجاح مولد الارقام العشوائي الوهمي وكذلك بينت بوضوح نجاح النتائج باختبارات عشوائية مع عملية استرجاع البيانات. في هذا البحث ايضا تم التحقق من صلاحية استخدام عمليات التشفير وفك الشفرة باستخدام الشبكات العصبية عوضا عن الطرق التقليدية (Exclusive or) والتي زادت من صعوبة كسر الشفرة.

1. Introduction

Stream ciphers can be designed to be exceptionally fast, much faster than any block cipher. They are used in applications where plaintext comes in quantities of unknowable length, for example, a secure wireless connection (e.g. W-LAN and Bluetooth). If a block cipher were to be used in this type of application, considerable bandwidth would end up being wasted by padding, since block ciphers cannot work on blocks shorter than their block size. For example, if a 64-bit block cipher received separate 32-bit bursts of plaintext, half of the data transmitted would be padding. Stream ciphers eliminate this by operating on the smallest unit that can be transmitted[1].

Further advantages of stream cipher systems are:

1- Speed of encryption/decryption: since each bit of the message is encrypted independently of other plaintext bits, each bit can be encrypted as soon as it is read.

2- Low propagation error: since each bit is encrypted and decrypted separately, decryption of a bit that have been corrupted in transmission produces an error to the one corresponding message bit and no others.

An Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the way biological nervous systems, such as the brain. The key element of this paradigm is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements (neurones) working in unison to solve specific problems. ANNs, like people, learn by example. An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurones[3]. Researchers have also tried to use neural networks in Cryptography. In January 2002, the Physicists Kanter, Kinzel and Kanter proposeda new key exchange protocol between two parties *A* and *B.* It uses the new notion of chaotic synchronization, which makes it possible for two weakly interacting chaotic systems to converge even though each one of them (viewed individually)continues to move in a chaotic way[4]. Neural network now typically have size of hundreds input and output allowing a huge mapping function to be implemented.

Also the non-linearity produiced by neural network mapping is desired in cryptography. The implementation of neural networkcan be reduced to complexity of one using existing hardwre technologies[5].

2. Stream Cipher

Most stream ciphers consist of a pseudorandom number generator (PRNG).For encryption the PRNG is initialized with a key and provides its output as a sequence of bits known as the (pseudo) random key stream. Randomness is very important because it completely destroys any statistically properties in the message. Key stream generators are often constructed using linear feedback shift registers (LFSR). The LFSR method is an attempt to simulate a one-time pad by generating a long key sequence from a little information. As with any such attempt, if the key is shorter than the message itself, breaking part of the ciphertext gives the cryptanalyst information about other parts of the ciphertext. For an LFSR, a known plaintext attack can reveal parts of the key sequence[1].

The key stream is combined (XOR) with the plaintext bit by bit typically by an XOR gate. To decrypt the message the ciphertext bits are XORed with the corresponding key stream bits again. So an error in a single bit of ciphertext results in a single bit of plaintext error. This is very useful when the transmission error rate is high[6].
Some ciphers, called self-synchronizing stream ciphers, use several previous ciphertext bits to compute the key stream. They depend on the data and its encryption. A single-bit error will result in a long burst of garble, but the cipher will eventually recover from a lost bit after the damaged and incorrect bit falls off the shift register. The opposite are the synchronous stream ciphers, in which the key stream is generated independently of the plaintext and the ciphertext. They generate bits from a source other than the message itself. The simplest such cipher extracts bits from a register to use as the key. The contents of the register change on the basis of the current contents of the register. Most stream cipher designs are for synchronous stream ciphers[1].

The one-time pad is a form of stream cipher that can be proven secure. In this cipher each key character is randomly generated and is used only once[7]. The ciphertext gives opponent no additional information on the plaintext[8].

3. Artificial Neural Networks

Artificial neural networks (ANNs) are highly parallel interconnections of simple processing elements or neurons that function as a collective system. There exist various problems in pattern recognition that humans seem more efficient in solving as compared to computers. Neural networks may be seen as an attempt to emulate such human performance. These networks can be broadly categorized as those that learn adaptively by updating their connection weights during training and whose parameters are time-invariant.

Artificial neural network consists of a large number of simple processing elements (PEs) densely interconnected, analogous to neurons of human brain. The neural network is made up of several layers of processing elements connected together via unidirectional signal channels associated with weights, analogous to synapses. These processing elements work in unision to slove a problem. The knowledge of the networks represented by its weights[9]. Every useful artificial neural net has a minimum of three layers: an input layer through which data is given to the network, an output layer that holds the response relative to the input and optional layer between the input and output layers called the hidden layer where learning takes place. The number of neurons in the input and output layers can be determined by the number of input and output variables in the physical system. The number of hidden layers and the nunber of neurons in the hidden layers are arbitrary and can vary from zero to any finite number.

Figure(1) shows a simple network architecture with four inputs and one output. The circles in the figure denote the PEs arranged in layers. A processing element (PE) can have many input paths and combines, usually by a simple summation, the values of these input paths. The network learns by adapting the weights of its connections according to surrounding environment. The behavior of the output unit depends on the activity of the hidden and input units. They are data driven devices. Neural networks are exceptionally effective for predicting events when the networks have large training data sets to pull

on[3].

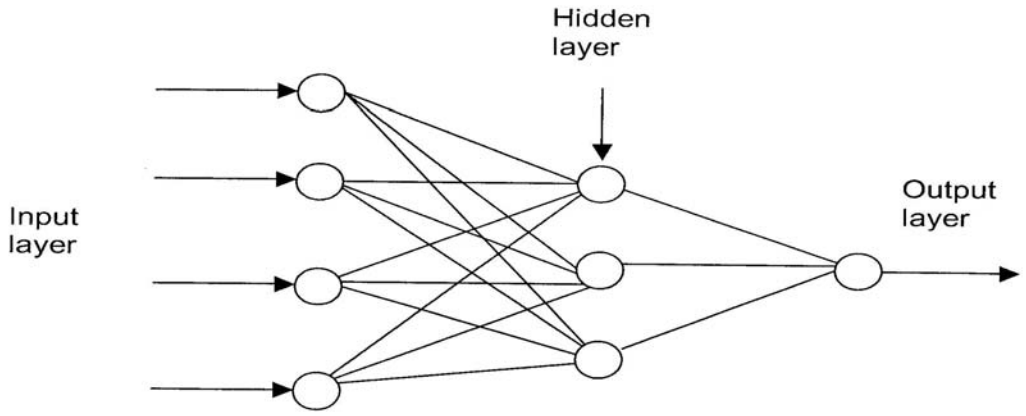

Figure.(1):A simple neural network architecture

With the given inputs and the related outputs, the network learns how the inputs of each data set are associated to the output. The network then continuously refines and organizes itself by adjusting the synaptic weights to fit the data, so that it produces relatively accurate response for a given input. Information processing of a single processing element is shown in Figure(2) Multiplying the input X*i* by weight W*i* can approximate the effect of a particular input unit. These weighted signals are added up to produce overall internal activation for the processing element. This activation level is passed through a transfer function, which produces an effective signal 'a'. If the activation is beyond a certain threshold, then the system gives the output response 'y'.In Figure(2) there are n inputs with signals $X_1, X_2, ..., X_n$ and corresponding weights $W_1, W_2, ..., W_n$.

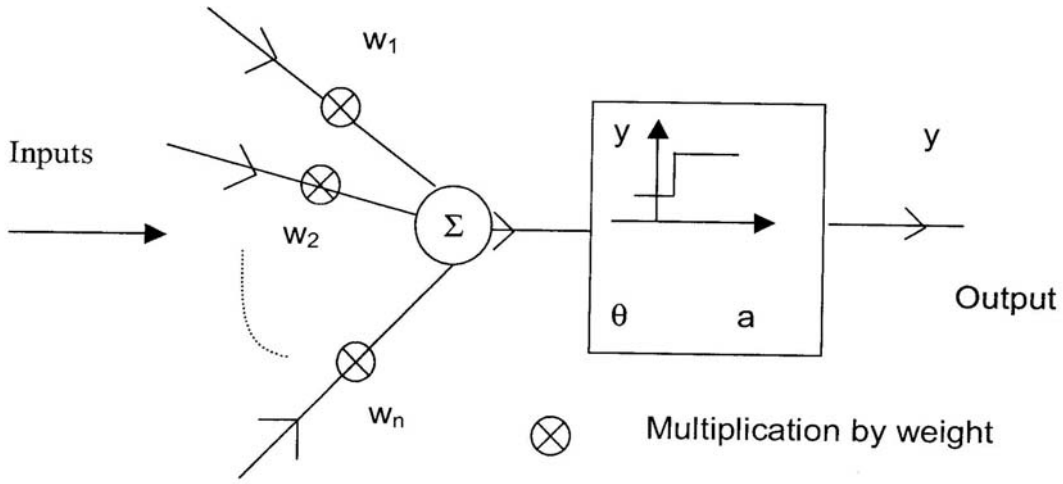

Figure.(2):Information processing of a single process element

The signals shown are Boolean value with values 'l' or '0' only. The system activation a is given by [10],

$$a = w_1x_1 + w_2x_2 + \ldots + w_nx_n \quad (1)$$

With this system activition, the output *y* is given by,

$$y = 0 \quad \text{if} \quad a \le \theta \quad (2)$$

$$y = 1 \quad \text{if} \quad a \ge \theta \quad (3)$$

where θ is the threshold that define the output. Generally there are two types of neural network archticecturte, Feed-forward networks and the Feedback networks.

In neural network, there are several network topologies. Each topology is designed to solve a specific problem or to represent certain point of view. Each topology has some drawbacks and benefits. These topologies can be categorized in three main categories: fully interconnected, feed forward connection, and mixed connectivity[11,12].

In Feed forward connection, the neurons are organized in multiple layers, so that a neuron in a certain layer is connected only to neurons on the next layer as shown in Figure (2). Multilayer perceptrons belong to this category [9].

4. The Proposed Artificial Neural Network PN Sequence Generator

The generation process of the key using backpropagation neural network cosist of there phases. The first phase is the feedforward and the second is a backpropagation process if there are errors. The third phase concerned with weights adjustments.

During a feedforward where input data is transmited from input unit P_i and calculate the activition function and sends its input data to each hidden unit h_i in hidden layer. In each hidden unit, the output of activition function is calculate to transform into output unit.

During training phase each output unit is compared with the target to determine the error rate. When the error is found, the error is propagate back to the input layer for weights adjusting(third phase). This process is repeated until the desired output obtained.

The algorithm below shows the procedure to achives the three phases:

1.Initiliazation network weight values

2.Sums weighted input and apply activition function to compute the output of the hidden layer using:

$$h_i = f(\sum_{i=1}^{n} P_i W_{ij} + b_j) \qquad (4)$$

where

h_i is the actual output of hidden neuron j.

P_i is the input signal of input neuron i.

W_{ij} is the weight between input neuron i and hidden neuron j.

b_j is the bias of hidden neuron j.

f is the activition function.

3. Sums weighted output of hidden layer and apply activition function to compute the output of output layer neurons using:

$$a_k = f(\sum_{k=1}^{n} h_i W_{ik} + b_k) \qquad (5)$$

where

a_k is the actual output of output neuron k.

W_{ij} is the weight between hidden neuron j and output neuron k.

b_k is the bias of the output neuron k.

4. Compute backpropagation error using:

$$\delta_k = (t_k - a_k) f'(\sum_{k=1}^{n} h_i W_{jk} + b_k) \quad (6)$$

where

f' is the derivative of the activation function.

t_k is the target output neuron k.

5. Calculate weight and bias correction output layer using:

$$\Delta W_{jk}(n) = \alpha \delta_k h_j \quad (7)$$

$$\Delta b_k(n) = \alpha \delta_k \quad (8)$$

where

α is the learning rate.

6. Add delta input for each hidden unit and calculate the error term using:

$$\delta_j = \sum_{j=1}^{n} \delta_k W_{jk} f'(\sum_{j=1}^{n} P_i W_{ij} + b_j) \quad (9)$$

7. Calculate the weight and the bais correction for the hidden layer using:

$$\Delta Wij(n) = \alpha \delta_j P_i \quad (10)$$

$$\Delta b_j(n) = \alpha \delta_j \quad (11)$$

8. Update weights and biases using:

$$W_{jk}(n) = W_{jk}(n-1) + \Delta W_{jk}(n) \quad (12)$$

$$b_k(n) = b_k(n-1) + \Delta b_k(n) \quad (13)$$

$$W_{ij}(n) = W_{ij}(n-1) + \Delta W_{ij}(n) \quad (14)$$

$$b_j(n) = b_j(n-1) + \Delta b_j(n) \quad (15)$$

9.Repeat from step 2 to 8 if error tolerance is not satisfied.

The randomness of the generated key from the backpropagation neural network can be justified using statistical tests. If the justification passees, the key can be used for data encryption.

5. Computer Simulation and Evalution of the PN Sequence Generator

In this section some of the statistical randomness tests have been applied to binary sequences to test these sequences. These tests include frequency, poker, serial and autocorrelation tests.

─ The Frequency Test

The frequency test involves the calculation of χ^2 using[2]:

$$\chi^2 = (N_0 - N_1)^2 / N \qquad (16)$$

where

N_0 is the number of occurrences of 0's in the N-bit sequence.

N_1 is the number of occurrences of 1's in the N-bit sequence.

The results showing the value of χ^2 for different values of N is as shown in Table(1).

Table (1): Frequency test result for the different length sequence.

Size of file	Result
100 bits	0.112
200 bits	0.154
300 bits	0.193
400 bits	0.211
500 bits	0.285

Here, if χ^2 is less than or equal 0.384 then the sequence is said to pass this test, otherwise it is failed. the results presented in Table(1) clearly shows that the proposed sequence passes the frequency test.

─ 5.2 The Serial Test

The serial test involves the calculation of χ^2 using[2]:

$$\chi^2 = [4/(N-1)\sum_{i=0}^{1}\sum_{j=0}^{1} N_{ij}{}^2 - (2/N)(\sum_{i=0}^{1} N_i{}^2)] + 1 \qquad (17)$$

where N_{00}, N_{01}, N_{10} and N_{11} be the number of occurrences of 00,01,10 and 11 respectively in the N-bit sequence. The results showing the value of χ^2 for different value of N as shown Table (2). Here, if χ^2 is less than or equal 5.99 then the sequence is said to pass this test, otherwise it is failed. The results presented in Table (2) clearly show that the proposed sequence passes the serial test.

Table (2): Serial test result for the different length sequences.

Size of file	Result
100 bits	2.91
200 bits	3.45
300 bits	3.86
400 bits	4.17
500 bits	4.43

─The Poker Test

The poker test involves the calculation of χ^2 using[2]:

$$\chi 2 = (2^m / K)\sum_{i=0}^{2^m-1}(Y_i{}^2 / C^m{}_i) - K. \qquad (18)$$

where

K is the number of blocks.

M is the length of block

Y_i is the number of m-bit subsequences having i 1's and (m-i) 0's, and so on. The calculated value should be compared table for χ^2 with 2^m-1 degrees of freedom. This will find the critical region as before using a statistical table. This test can be applied as

many times for different values of m. The results showing the value of χ^2 for different value of m is as shown Table(3) for different degrees of freedom.

Table (3): Poker test result for the different length sequences and different degree of freedom.

Size of file	m=3	m=4	m=5
100 bits	1.7	5.3	6.7
200 bits	2.0	6.7	7.2
300 bits	2.5	7.3	8.6
400 bits	2.7	7.9	9.1
500 bits	3.1	8.4	10.3

Here, if χ^2 is less than or equal 11.1 then the sequence is said to pass this test, otherwise it is failed. the results presented in Table(3) clearly show that the proposed sequence passes the poker test.

–The Autocorrelation Test

The autocorrelation test equation involves the calculation of χ^2 using :

$$A(d)=\sum_{i=1}^{N-d} U_i \oplus U_{i+d} \qquad (19)$$

Where N-bit is length of the sequence is (U) $=U_1\ U_2 \ldots U_N$. The factor of d is shift in bits between the sequences U_i and U_{i+d}. If there is correlation between the sequences, A(d) should satisfy on the following relations[2]:

$$A(d) > ((N-d)/2) + 1.96(\sqrt{(N-d)}/2) \qquad (20)$$

Or

$$A(d) < ((N-d)/2) - 1.96(\sqrt{(N-d)}/2) \qquad (21)$$

Table (4): Autocorrelation test results for the different length sequences .

No of shift	N=100	N=200	N=300	N=400	N=500
1	0.081	0.321	0.513	0.521	0.613
2	0.123	0.412	0.312	0.151	0.437
3	0.032	1.212	0.784	0.763	1.617
4	0.534	0.785	1.812	1.561	0.571
5	0.385	0.052	0.789	0.863	0.387
6	1.311	0.612	0.421	0.313	1.815
7	0.423	0.423	0.672	1.253	0.649
8	0.678	0.678	0.516	0.651	0.745
9	0.745	0.745	0.858	0.451	0.517
10	0.856	0.856	1.812	0.713	0.887

It is clear from Table (4) that all sequences tested pass the autocorrelation test since the value of A(d) for all are less than 1.96 as recommended by equation 21.

6. The Proposed Artificial Neural Network Scrambling Function

A new approach for data scrambling has been proposed from the standard backpropagation algorithm. The proposed approach uses the generated key from the neural network proposed in section 4 as input with the key weight instead of bias and plaintext as the second input as shown by the flowchart shown in Figure (3). The same technique is used for decrypting the ciphertext shown by the flowchart shown in Figure (4).The procedure used in the proposed algorithm is as follow:

1.Initiliazation network weight values

2.Sums weighted input and apply activition function to compute the output of the hidden layer using:

$$h_i = f(\sum_{i=1}^{n} P_i W_{ij} + \sum_{i=1}^{n} KiWk_{ij}) \qquad (22)$$

where

h_i is the the actual output of hidden neuron j.

P_i is the input signal of input neuron i.

W_{ij} is the weight between input neuron i and hidden neuron j.

K_i is the key of hidden neuron j.

Wk_{ij} is the key weight.

f is the the activition function.

3. Sums weighted output of hidden layer and apply activition function to compute the output of output layer neurons using:

$$a_k = f(\sum_{k=1}^{n} h_i W_{ik} + \sum_{k=1}^{n} K_i Wk_{ik}) \quad (23)$$

where

a_k is the the actual output of output neuron k.

W_{ij} is the weight between hidden neuron j and output neuron k.

4. Compute backpropagation error using:

$$\delta_k = (t_k - a_k) f'(\sum_{k=1}^{n} h_i W_{jk} + \sum_{k=1}^{n} K_i Wk_{jk} \quad (24)$$

where

f' is the the derivative of the activation function.

t_k is the the target output neuron k.

5. Calculate weight and key weight correction output layer using:

$$\Delta W_{jk}(n) = \alpha \delta_k h_j \quad (25)$$

$$\Delta Wk_{jk}(n) = \alpha \delta_k \quad (26)$$

where

α is the learning rate.

6. Add delta input for each hidden unit and calculate the error term using:

$$\delta_j = \sum_{j=1}^{n} \delta_k W_{jk} f'(\sum_{j=1}^{n} P_i W_{ij} + \sum_{j=1}^{n} K_i Wk_{ij}) \quad (27)$$

7. Calculate the weight and the key weight correction for the hidden layer.

$$\Delta Wij(n) = \alpha \delta_j P_i \quad (28)$$

$$\Delta Wk_{ij}(n) = \alpha\delta_j \quad (29)$$

8. Update weights and biases.

$$W_{jk}(n) = W_{jk}(n-1) + \Delta W_{jk}(n) \quad (30)$$

$$Wk_{jk}(n) = Wk_{jk}(n-1) + \Delta Wk_{jk}(n) \quad (31)$$

$$W_{ij}(n) = W_{ij}(n-1) + \Delta W_{ij}(n) \quad (32)$$

$$Wk_{ij}(n) = Wk_{ij}(n-1) + \Delta Wk_{ij}(n) \quad (33)$$

9.Repeat from step 2 to 8 if error tolerance is not satisfied.

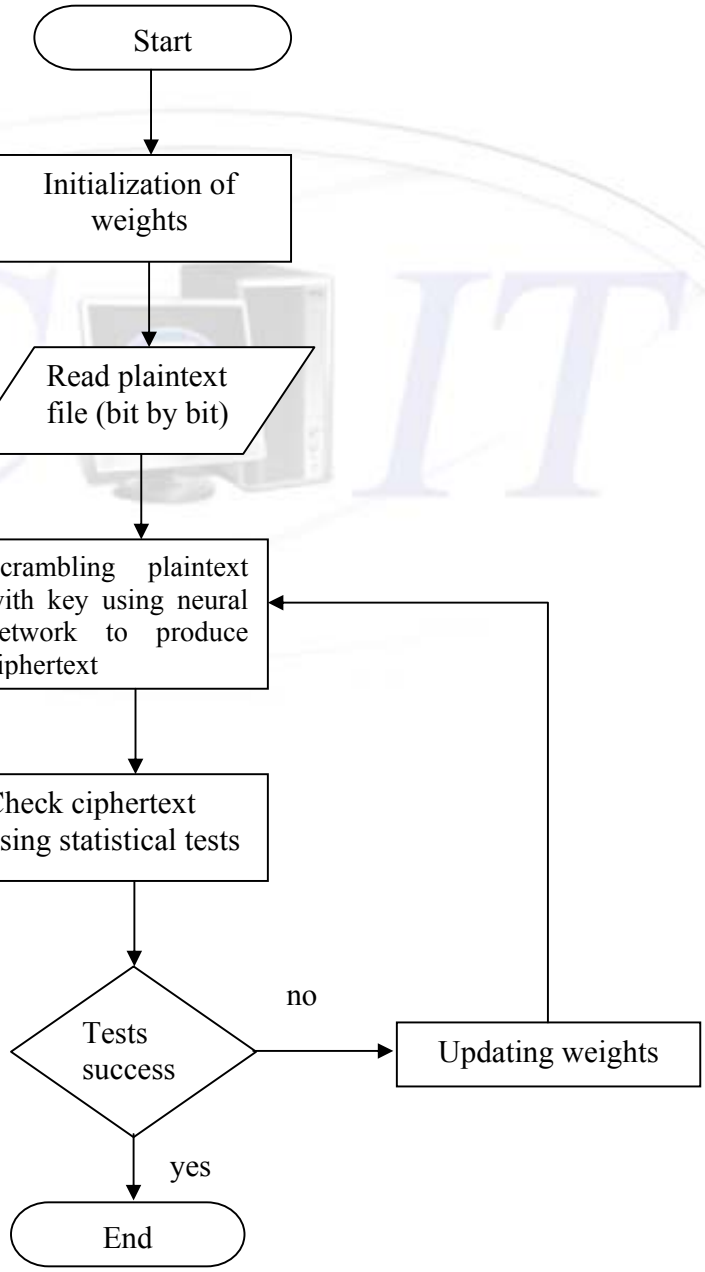

Figure.(3): Flowchat of proposed system for data encryption.

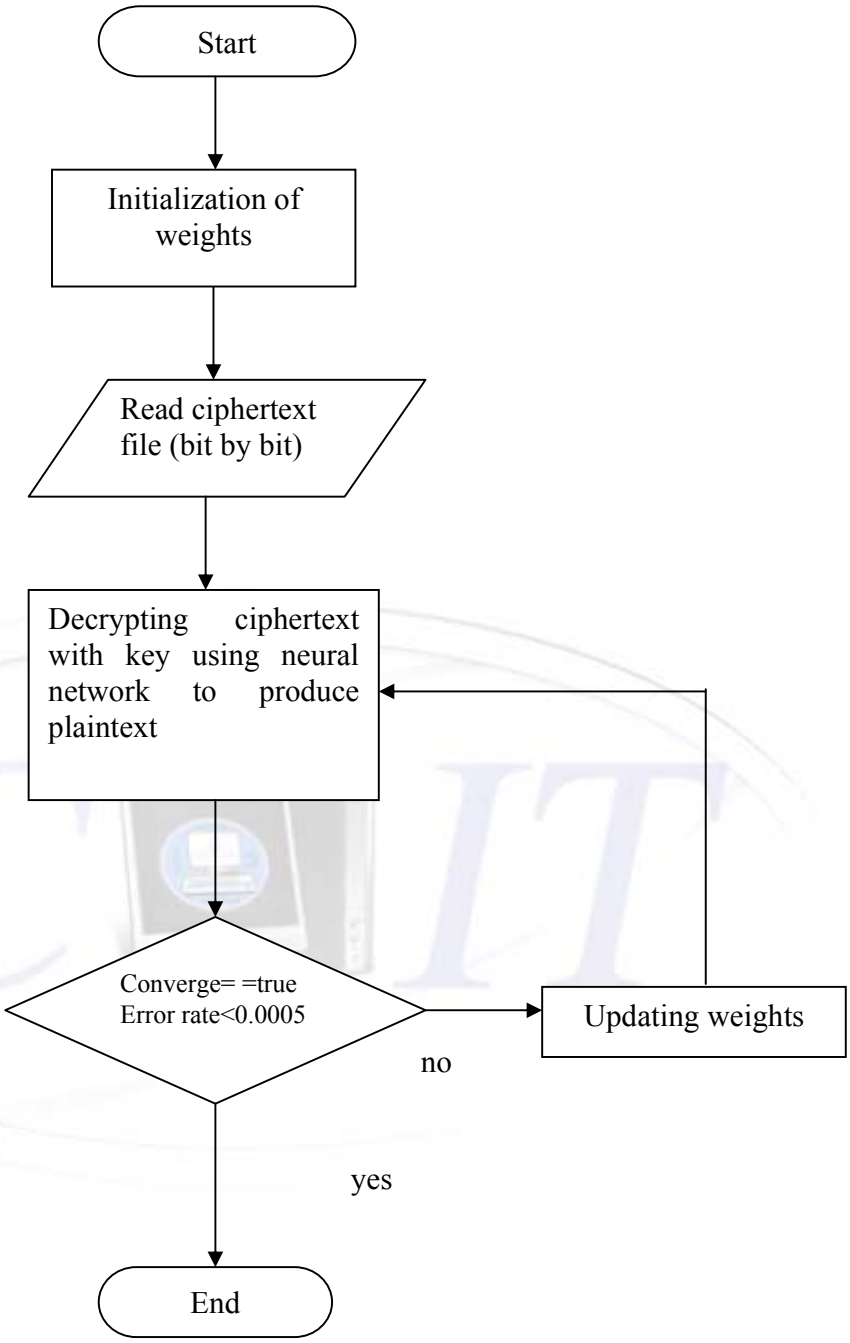

Figure.(4): Flowchart of proposed system for data decryption.

The proposed scrambling technique has been evaluated by encrypting the following plaintext with the generating key (Section 4). The following plaintext is adopted as input

There are several network topologies. Each topology is designed to solve a specific problem or to represent certain point of view. Each topology has some drawbacks and benefits. These topologies can be categorized in three main categories: Fully interconnected, feed forward connection, and mixed connectivity.

With the following key

101010001010000000000001100010111010010000000100100000100000000000000010010000110000010010001000010000000000
000100010010001000000100010000000010001001010011000000000100010100010000000111011001001000000001000101001010
011100101001010000001101000000000010000000100010001011100001000000000011010011000000000000000010000001000000
010010110100001000000100001101010110000100000100000100001000000010011000000001100001011100000100100000010000
100001010100000000100000000001000001010000100001100000100010010101110000001011000000000100001000101010011011 1
111000000101001000000001000011110000000000010100011010001110000010001000000111010010100000000000010100000101
000100000000010000001100011010000101100000010000010100100001100001001000010001001000000101010110000000010000
000111000001000101100000000000000000100000000100000001000000010000000001000010100000000000001010110000001000000
001010000010000000000000001000010001000000000001000000010000010001001100001010001100101000001100010000100010
000110000001000000001101100100010100000010100000000001000001000100110000010010100000010001010100010000011000
000100000011011000101001000000001001010010001011100000000010000010000100000000011110001000000011100010111010
000010001100010010001000101001000000001001000000000000100000000000000010100011001000111011000000000000000100
101101000000110100001001010011000010000111000001000100000000001100000000001000000101001100000000001000001
001100110001100000100000000000001011000000111010000001001100110000001110110011010000000001001100001011010000
000000000000000010111101001100101001001001100000000010010000010100000000010100001000000000011010010001100010 1
000000100000010101000111110010000100100010000000000000010001010000000010000000100000000101110010001000010010
010000000001000100110010000001001000001111000000000001000000010100000000000000001011001000101000011000000001
000000000000001110000000100011000010001101000000011100001000000000001000000100010000000110000000110010101100
000010000000000001000000010000000000001011001011000010100000011000010111000000010010010000000001010000011100
010000000000000000000011000100001001110001001000001000000001100000010000001000000001010100 11

This will produce the following ciphertext

4harm'oR qe|%n“Q k3ض3³‰ dO|nl{#éeِ)*TJihf?/ِoىë&[$iâ(ه;p'zdS)dت¼! mfuً@
ەqeâ)<aç zيrüdë.}zپvm0f§Pbi%و¾^أaz{ل0ً$hخش(OV6la/o< مؤè£opnnnf"(lw4woémن
RSc1Q!ïِF4b$ٔd8e *b"ەçچ °LŒ‡ەqى[!qalèى;cً b=/v°(~ەe(Am0 ®ReµàhSjY+C" e'@!
ûdW ٔ uٔt|"|ٔنmr'nll APd.•etpfïT}bpd'§ eVaم4mcِل6$ne€AH ژ$2#fne@گ!fTiژ

7. Conclusions

There are several techniques that can be used to generate a PN sequence. However, a new cryptosystem based on neural network is found very attractive in the field cryptosystem. Neural network technique is used for key generation and to encrypt data. The proposed technique offers more randomness for key generation, also to reduce the process time and to get high reliable secure system as compare as the traditional stream cipher methods.

References

[1] M. Prokop, 2004, " Stream cipher", http//www.michal-prokop.at.

[2] A. Kanso, 1999, "Clock-controlled generators", Ph.D. thesis, University of London.

[3] P. Inakollu, 2003, "A study of the effectiveness of neural network for elemental concentration from libs spectra", MS.C. thesis, Mississippi state university.

[4] A. Klimov, A. Milyaguine, and A. Shamir,2002,"Analysis of neural network", Computer science department, http://citeseer.ist.psu.edu/kinzel02neural.html.

[5] A. Albassal and Dr.A.Wahdan,2004,"A neural network based block cipher", Computer and system engineering Dept. faculty of engineering, Ain shams university.

[6] P. Ekdahl, 2003, "On LFSR based stream ciphers analysis and design",Ph.D. thesis, Lund university.

[7] J. Buchmann, 2001, "Introduction to cryptography", Edwards brothers publishing.

[8] T. Lindquist, M. Diarra, and B. Millard, 2004, "A java cryptography service provider implementing one-time pad", Proceedings of the 37th hawaii international conference on system sciences.

[9] C. Tegiou and D.Sigano,"Neural network", 2000, http://www.doc.ic.ac.uk/~nd/surprise96.

[10] M. Hagan, H. Demuth and M. Beale, 1996, "Neural network design", A division of international Thomson publishing Inc.

[11] R. Callan, 1999, "The essence of neural networkg", Prentice hall europ.

[12] M. Manic. 2008, "Neural networks design", http://husky.if.uidaho.edu/ ee578s08/hw.html