Design and Implementation of Block Cipher Using Neural Network

Prof. Siddeq Y. Ameen & **Dr. Mazin Z. Othman**

Dept. of Computer Eng. and I.T
University of Technology/Baghdad

Technical College of Mosul
Commission of Technical Education

&

Dr. Safwan O. Hasson

College of Computers Sciences and Mathematics
University of Mosul

Received 04 / 11 / 2009 **Accepted** 27 / 06 / 2010

الـمـلـخص

شهد العقد الاخير تطور كبير في حقول الحاسبات ، الذكاء الاصطناعي، الاتصالات وتقنيات نقل المعلومات. جذب هذا التطور اهتمام المختصصين بضرورة الحاجة لتصميم أنظمة تشفير تعتمد في اداءها على سلوك الشبكات العصبية والتي تجعل من الصعب بل من المستحيل في بعض الحالات كسر شفرة هذه الأنظمة.

يتناول البحث استغلال الشبكات العصبية في بناء خوارزميات التشفير الكتلي اخذين بنظر الاعتبار حجوم مختلفة من النصوص الصريحة . بين استخدام اختبار التردد الكتلي نجاح النصوص المشفرة مع صعوبة كسر عملية التشفير.

ان استخدام الشبكات العصبية في علم التشفير توفر سرعة ومستوى امني عاليين في التشفير الكتلي مقارنة مع الطرق التقليدية . أستخدمت لغة C++ لبناء برامج الشبكات العصبية الخاصة بخوارزميات التشفير وفك الشفرة.

Abstract

The last decade witnessed a great evolution on the fields of computer science, artificial intelligence, communication and data transmission. This evolution draw the attention of specialists to design a modern cryptosystem for data encryption based on neural networks methodologies that are very hard, if not impossible, to be broken.

This paper employs cryptography scheme utilizes the neural networks in block cipher algorithms. Taking into consideration different plaintexts block size. The block frequency test has been show the successful of ciphertexts with difficulty to break the cipher process.

The application of the neural network in cryptography provides fast and high security system in block ciphering in comparison with traditional methods. The C++ language is used for designing the programs of neural network to encryption and decryption algorithms.

1. Introduction

A Block Cipher is a form of encryption algorithm that operates on the input data in blocks of a fixed size. A Block Cipher takes a plaintext block of a specified size and an encryption key, and then operates on this data to produce a ciphertext block of the same size. Similarly, the decryption algorithm takes as input the fixed size ciphertext block and the decryption key, identical to the encryption key. It then performs the function of retrieving the plaintext block of data. The nature of this description necessarily implies that a particular block of plaintext will always encrypt to the exact same ciphertext block given the same encryption key [1].

The main objectives of this work to design adaptive block cipher algorithms based on ANN techniques. The common issue between cryptography system and ANN techniques is to enhance secure for encrypting/decrypting data. Combine the two approaches or techniques will enrich the cryptography process.

2. Block Cipher

In block cipher, also called secret-key or symmetric-key encryption, one key is used both for encryption and decryption. The Data Encryption Standard (DES) is an example of a conventional cryptosystem that is widely employed by the Federal Government. In 1997, the U.S. National Institute of Standards and Technology (NIST) announced the requirements for a new encryption standard called Advanced Encryption Standard (AES) [1], with the goal of creating a new national standard (Federal Information Processing Standard) (FIPS), for encryption with a symmetric algorithm.

The general principles used in most block ciphers can be summed up using the terminology confusion and diffusion. Confusion, in relation to encryption, attempts to obscure any relationship between the plaintext and the ciphertext. This can usually be performed with simple substitution of bits within the plaintext message, in fact the one time pad cipher is a confusion based cipher where each bit in the plaintext is either inverted or left untouched depending on the random key used. Diffusion, in relation to encryption, tries to remove any redundancies and statistical relationships in the plaintext by spreading the effect of the plaintext over as much ciphertext as possible. In reference to an n-bit block cipher, diffusion would attempt to spread the effect of a single bit of plaintext over all n bits of the ciphertext block produced by the cipher. A strong cipher would succeed in this whilst an insecure cipher would not be able to ensure that each plaintext bit affected all ciphertext bits in the block. Ciphers that employ diffusion techniques only are not particularly secure and can be easily cryptanalysis and broken. Most Block Ciphers use both confusion and diffusion techniques in their algorithmic designs [2]. The details of diffusion and confusion are shown in below section.

2.1 Diffusion and Confusion

The terms confusion and diffusion were introduced by Claude Shannon to capture the two basic building blocks for any cryptographic system. Shannon's concern was to thwart cryptanalysis based on statistical analysis. The reasoning is as follows: Assume the attacker has some knowledge of the statistical characteristics of the plaintext. For example, in a human-readable message in some language, the frequency distribution of the various letters may be known. Or there may be words or phrases likely to appear in the message. If these statistics are in any way reflected in the ciphertext, the cryptanalyst may be able to deduce the encryption key, or part of the key, or at least a set of keys likely to contain the exact key [2].

Shannon proposes two methods frustrating statistical cryptanalysis: diffusion and confusion. In diffusion, the statistical structure of the plaintext is dissipated into long-range statistics of the ciphertext. This is achieved by having each plaintext digit affect the value of many ciphertext digits, which is equivalent to saying that each ciphertext digit is affected by many plaintext digits. An example of diffusion is to encrypt a message M = $m_1, m_2, m_3, \ldots, m_n$

Of characters with an averaging operation:

$$Y_n = \sum_{n=1}^{k} m_{i+1} (\mathrm{mod}\, 26) \qquad (1)$$

Adding k successive letters to get a ciphertext letter Y_n. One can show that the statistical structure of the plaintext has been dissipated. Thus the letter frequencies in the ciphertext will more nearly equal than in the plaintext; the diagram frequencies will also be more nearly equal, and so on. On the other hand, confusion seeks to make the relationship between the statistics of the ciphertext and the value of the encryption key as complex as possible, again to thwart attempts to discover the key. Thus, even if the attacker can get some handle on the statistics of the ciphertext, the way in which the key was used to produce that ciphertext is so complex as to make it difficult to deduce the key. This is achieved by use of complex substitution algorithm. In contrast, a simple linear substitution function would add little confusion.

So, successful are diffusion and confusion in capturing the essence of the desired attributes of a block cipher that they have become the cornerstone of modern block cipher design [2].

3. Artificial Neural Nework

Artificial neural networks are parallel computing devices consisting of many interconnected simple processors. These processors are quite simplistic, especially when compared with the type of processor found in a computer. Each processor in a network is only aware of signals it periodically receives and the signal it periodically sends to other processors, and yet such simple local processors are capable of performing complex tasks when placed together in a large network of orchestrated cooperation.

Artificial neural networks have their roots in work performed in the early part of the twentieth century, but only during the 1990s, after the breaking of some theoretical barriers and the growth in available computing power, have these networks been widely accepted as useful tools. The word “artificial” is sometimes used to make it clear that discussion is about an artificial device and not about the real biological neural networks found in humans. It is the human brain that has inspired the creation of artificial neural networks and no doubt will influence further development. However, in comparison to the human brain, artificial neural networks are at present highly simplistic abstractions. It is common to drop the prefix 'artificial' when it is clear in which context these networks are being discussed. Also, artificial neural networks are often referred to as connectionist networks when computing ability is emphasized rather than biological fidelity. In other words, connectionists aim to make neural networks solve a task rather than attempt to mirriic faithfully some part of a biological process[3].

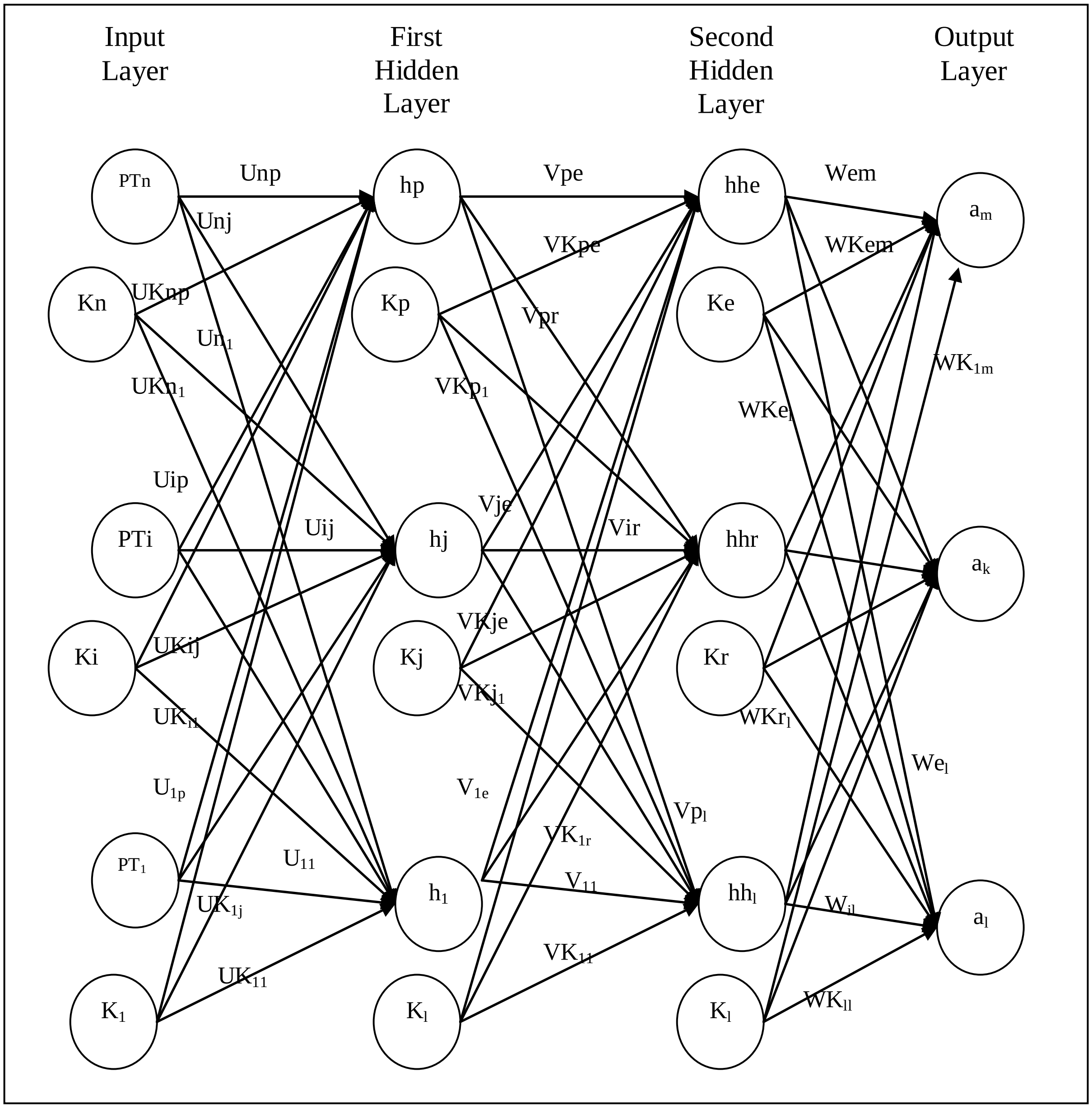
Although neural networks can be implemented as fast hardware devices, much research is performed using a conventional computer running software simulations. Software simulation provides a somewhat cheap and flexible environment in which to research ideas and for many real-world applications simulation provides adequate performance. Although a neural network solution might have the look and feel of any conventional piece of software, there is a key difference in that most neural solutions are “learnt” and not programmed: the network learns to perform a task rather than being directly programmed. Indeed, many neural network solutions exist either because it is impossible to write a program or because the neura network “learnt solution” provides improved performance. The Backpropagation algorithm is type of neural networks uses gradient descent to tune network parameters to best fit a training set of input-output pairs in the least-squares sense. The method is quite robust to the errors in the training data, and hence, it is well suited for pattern recognition (PR) tasks with noisy, complex sensor data [4].

4. The Proposed System

The proposed neural network that is used for data encrypting in term of block cipher techniques consists of four layers, the first layer is the input layer, the second and third represented the hidden layers while as the fourth is the output layer.

The file that contains the data to be encrypted divided into set of blocks; each block consists of set of bits. The number of neurons available in the input layer must be as the same size of block such that each neuron represents a bit in the specific block.

The secret key that used in encryption process is inserted to neural network instead of the bias so the inputs of the neural network will be a combination between plaintext and secret key. The weights of the proposed neural network is generate in random way. Figure (1) illustrates the proposed structure of the system for data encryption.



Figure(1): Proposed ANN for block cipher

The procedure used in the proposed algorithm is as follow:

1. Initiliazation network weight values randomly from 0 or 1.
2. Exclusive_or weighted input and apply activition function to compute the output of the first hidden layer.

$$hi = f((PT_i \oplus U_{ij}) \oplus (K_i \oplus Uk_{ij})) \qquad (2)$$

where

i =1,2,…,n, j = 1,2,…,n, and $\oplus$ denote XOR(modulo 2 addition).

h_i is the the actual output of hidden neuron j.

PT_i is the input signal of input neuron i.

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

K_i is the key of hidden neuron j.

Uk_{ij} is the key weight.

f is the the activition function.

3. Exclusive_or weighted output of first hidden layer and apply ctivition function to compute the output of second layer neurons using:

$$hh_j = f((h_j \oplus V_{jk}) \otimes (K_j \oplus Vk_{jk})) \qquad (3)$$

where

j =1,2,…,p, k = 1,2,…,p, and $\oplus$ denote XOR (modulo 2addition).

hh_k is the the actual output of second hidden layer neuron k.

V_{jk} is the weight between hidden neuron j in second layer and hidden neuron k in third layeUk_{ij} is the key weight.

Vk_{jk} is the key weight.

4. Exclusive_or weighted output of second hidden layer and apply the activition to compute the output of the output layer neurons using:

$$a_l = f((hh_l \oplus W_{lk}) \oplus (K_l \oplus Wk_{lk})) \qquad (4)$$

k=1,2,…,m, l = 1,2,…,m, and $\oplus$ denote XOR(modulo 2 addition).

a_l is the the actual output of output neuron l.

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

5. Compute backpropagation error using:

$$\delta_l = a_l f'(\sum_{k=1}^{m} hh_k W_{kl} + \sum_{k=1}^{m} K_k Wk_{kl}) \qquad (5)$$

where

f' is the the derivative of the activation function.

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

$$\Delta W_{kl}(n) = \alpha \delta_l hh_k \qquad (6)$$

$$\Delta Wk_{kl}(n) = \alpha \delta_l k_k \qquad (7)$$

where
α is the learning rate.

7. Add delta input for each hidden unit in third layer and calculate the error term using:

$$\delta_k = \sum_{l=1}^{m} \delta_l W_{kl}\, f'(\sum_{j=1}^{e} h_j V_{jk} + \sum_{j=1}^{e} K_j VK_{jk}) \qquad (8)$$

8. Calculate the weight and the key weight correction for the second hidden layer.

$$\Delta V_{jk}(n) = \alpha \delta_k h_j \qquad (9)$$

$$\Delta VK_{jk}(n) = \alpha \delta_k K_j \qquad (10)$$

9. Add delta input for each hidden unit in second layer and calculate the error term using:

$$\delta_j = \sum_{k=1}^{e} \delta_k V_{jk}\, f'(\sum_{i=1}^{p} PT_i U_{ij} + \sum_{i=1}^{p} K_i UK_{jk}) \qquad (11)$$

10. Calculate the weight and the key weight correction for the first hidden layer.

$$\Delta U_{ij}(n) = \alpha \delta_j X_i \qquad (12)$$

$$\Delta UK_{ij}(n) = \alpha \delta_j K_i \qquad (13)$$

11. Update weights.

$$U_{ij}(n) = U_{ij}(n-1) + \Delta U_{ij}(n) \qquad (14)$$

$$UK_{ij}(n) = UK_{ij}(n-1) + \Delta UK_{ij}(n) \qquad (15)$$

$$V_{jk}(n) = V_{jk}(n-1) + \Delta V_{jk}(n) \qquad (16)$$

$$VK_{jk}(n) = VK_{jk}(n-1) + \Delta VK_{jk}(n) \qquad (17)$$

$$W_{kl}(n) = W_{kl}(n-1) + \Delta W_{kl}(n) \qquad (18)$$

$$WK_{kl}(n) = WK_{kl}(n-1) + \Delta WK_{kl}(n) \qquad (19)$$

12. Repeat step 2 to 11 until ciphertext passed block frequence test.

The process of the neural network passes through three phases, feedforward, and backpropagation and weight adjustment.

In the first phase the neural network receiving the inputs block by block, each block is encrypted with the secret key, after that the output of the first layer passing to first hidden layer then the of activation function of this layer pass to second hidden layer, where as the activation function of second layer send to the output layer.

In the training phase, the output of the proposed neural network is tested using block frequency test as shown in Figure (2). If its passes then the output will be ciphertext, if not, extra weight adjustment is required. This will be considered in the third stage. The same technique is used for decrypting the ciphertext as shown by the flowchart in Figure (3).

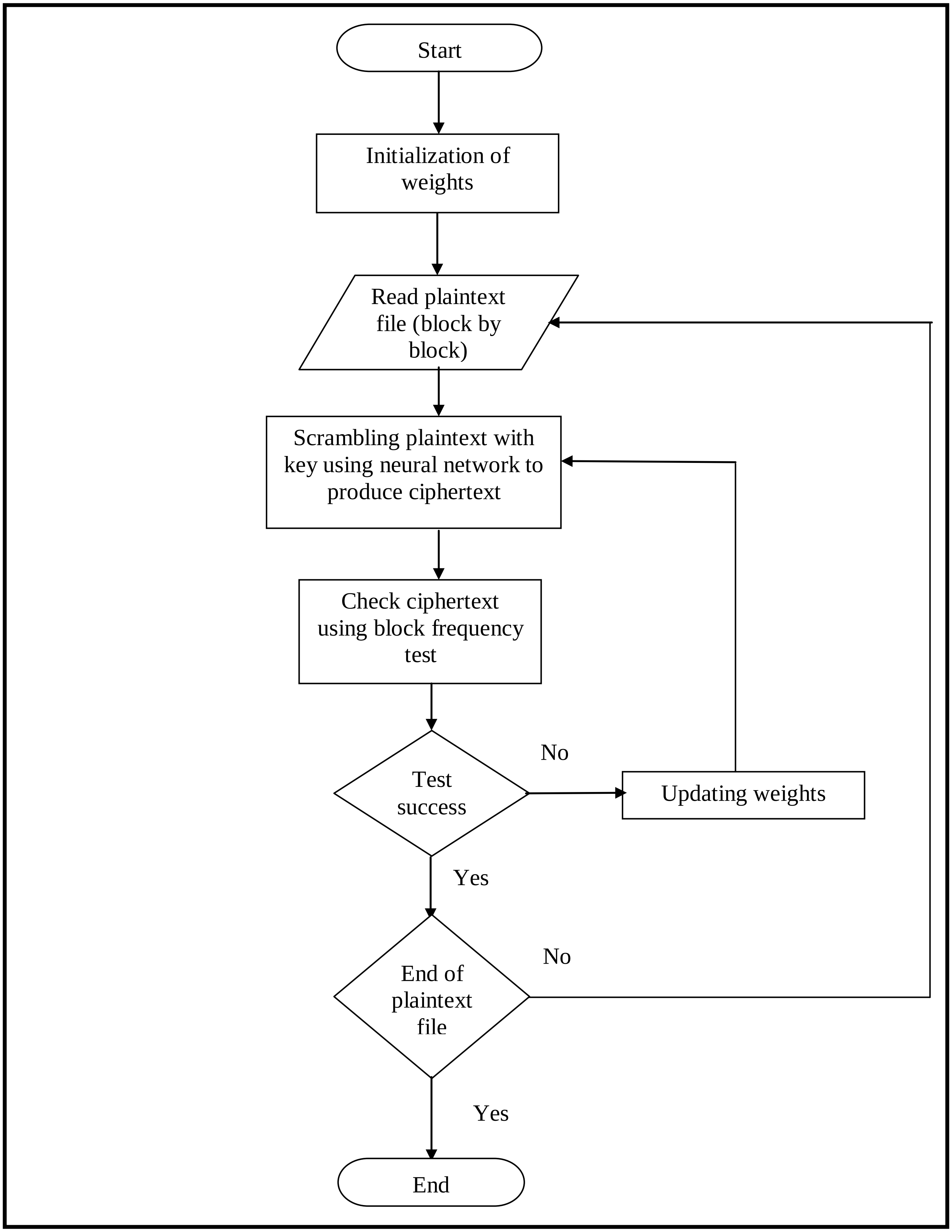

Start
Initialization of weights
Read plaintext file (block by block)
Scrambling plaintext with key using neural network to produce ciphertext
Check ciphertext using block frequency test
Test success
No
Updating weights
Yes
End of plaintext file
No
Yes
End

Figure (2): Flowchart of the proposed system for block data encryption.

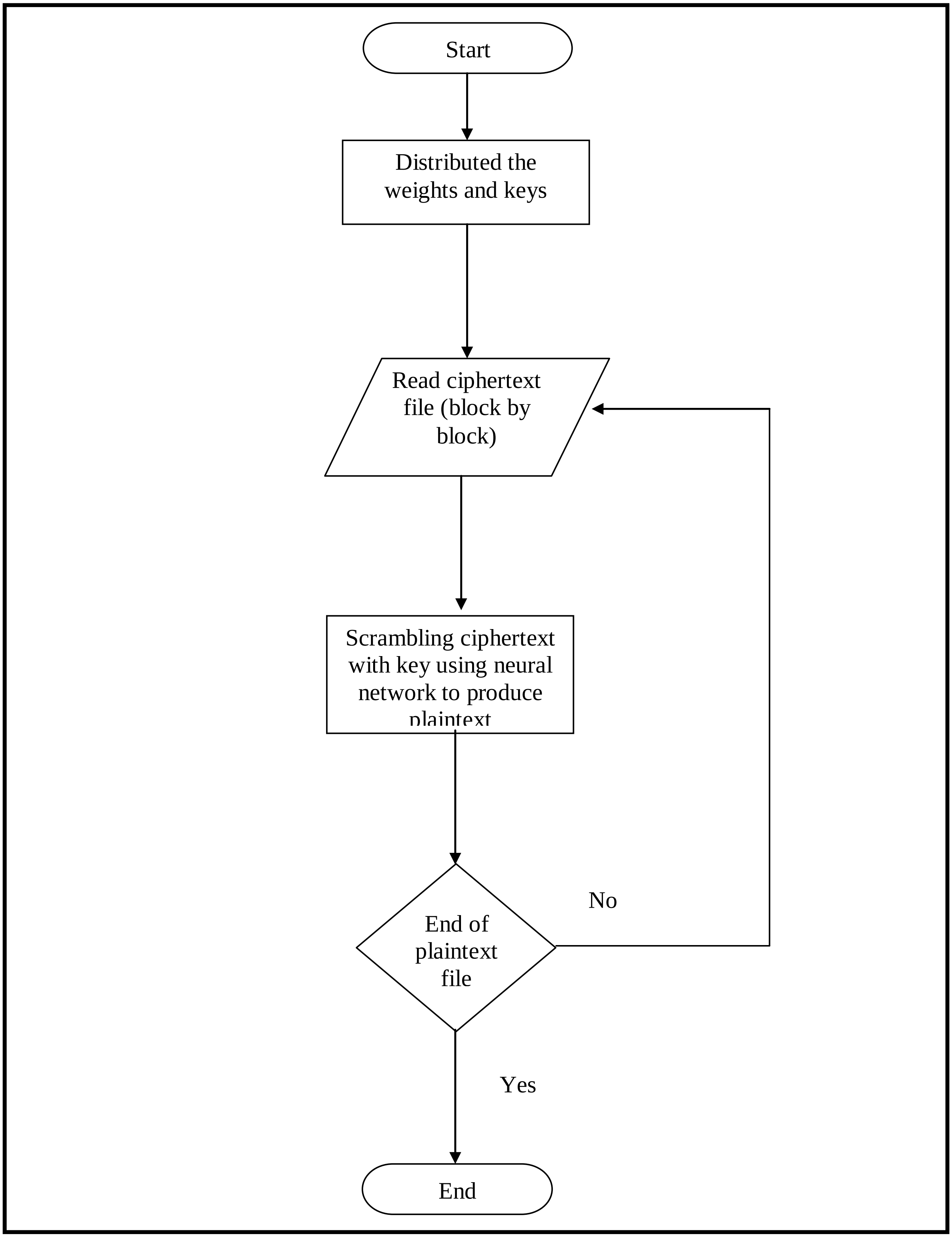

Start
Distributed the weights and keys
Read ciphertext file (block by block)
Scrambling ciphertext with key using neural network to produce plaintext
End of plaintext file
No
Yes
End

Figure (3): Flowchart of the proposed system block data decryption.

5. Block Frequency Test

The Definition of block frequency test show as below:

"The Hamming weight W(C) of a code word C is equal to the number of nonzero components in the codeword". Let (O_i) denotes the number of n-bit block vectors hamming weight is (i) from the set of (IND_m) to be examined for randomness. There are altogether (C_i^n) binary vectors of length (n) and weight (i). It should be noted that:

$$C_i^n = n! / [(n - i)! * i!] \tag{20}$$

in the case where the (m) binary vectors are random. The expected number of binary vectors from this set of weight (i) is :

$$E_i = C_i^n * m / 2^n \tag{21}$$

If the (m) binary vectors are random then the values of (O_i) (E_i) should be approximately the same. Furthermore the values of (E_i) and (O_i) may be compared using the Chi-squared χ^2 (Appendix A) [5]:

$$\chi^2 = \sum_{i=1}^{k} (Oi - Ei)^2 / E_i \tag{22}$$

where (O_i) is the observed number of blocks whose weight (i), Σ denote the summation cover all possible runs (k) of length (i) such that ($E_i \geq 5$). That means the (n) should be taken large enough this is compared with a Chi-squared distribution with k-1 degrees of freedom and significance region P, (0.01,0.05,0.001). If all the values of the Chi-squared calculated are greater than P, then there is indicated a possible weakness in the cipher.

The results presented in Table (1) clearly show that the each ciphertext generate using proposed neural network passes the block frequency test.

Table (1): Block frequency test results

Block no.	χ Bock size (64 bits)	χ Bock size (128 bits)
block0-block4	0.001	0.05
block5-block9	0.05	0.01
block10-block14	0.05	0.01
block15-block19	0.01	0.05
block20-block24	0.001	0.01
block25-block29	0.01	0.01
block30-block34	0.05	0.001
block35-block39	0.001	0.05

Here if all chi-square calculated greater than significance region P,(0.01,0.05,0.001), then is said to fail this test, otherwise it is pass.

6. Avalanche Effect Criteria

Extra property used to measure the strength of block ciphers is the avalanche effect. This can be applied by measuring the plaintext avalanche effect or the key avalanche effect. A block cipher satisfies the plaintext (key) avalanche effect if for a fixed key (plaintext) a small change in the plaintext (key) causes a large change in the resulting ciphertext block. A more specialized property is defined, namely the strict plaintext avalanche criteria, which will be denoted be (SPAC). A block cipher satisfies the SPAC if for a fixed key each bit of the ciphertext block changes whenever any bit of the plaintext block is complemented. This property can also be applied to key changes where a block cipher satisfies the strict key avalanche criteria (SKAC) if, for a fixed plaintext block, each bit of the ciphertext block changes whenever any bit of the key changes [6], A description of the method for analyzing the SPAC will be given below and the method for analyzing the SKAC is similar.

6.1 The Strict Plaintext Avalanche Criteria (SPAC)

To measure the SPAC for a block cipher of length (n) we follow the steps:

1- Generate a large number of random plaintext blocks P_r, For r = 1, 2,…, R.

2- Let P_{rj} for j = 1, 2,… ,n be the plaintext vectors that differ in the j^{th} position.

3- Using the fixed key k, let C_r and C_{rj} denote the ciphertext vectors that result from enciphering P_r and P_{rj} respectively.

4- Define avalanche vectors such as:

$$AV_{rj} = C_r \oplus C_{rj} \qquad (23)$$

Where j =1,2,…,n, r = 1,2,…,R, and $\oplus$ denote XOR(modulo 2 addition).

The hamming weight of each vector indicates the number of bits that changed in the ciphertext vectors when one bit of plaintext vectors is changed. The number of changed bits in the output vector must be large and differ from vector to other and randomly distribution [7]. The distribution of the number changes in ciphertext bits may also be investigated, since for a given plaintext / ciphertext bit complemented, the number of ones in the avalanche vector represents the number of changes in ciphertext, then a frequency test may be applied to the number of ones in the avalanche vector. This test is applied separately for each plaintext / key bit complemented.

If the block cipher algorithm passes the avalanche criteria test, the independence of the avalanche variable may also be investigated. The test of independence of the avalanche variables determines whether a change in value in ciphertext position (i) is independent of a change in value in ciphertext position (j) for all (n) positions [7].

6.2 The Strict Key Avalanche Criteria (SKAC):

As we stated previously, the block cipher satisfies the strict key avalanche criterion (SKAC), if for a fixed plaintext block, each bit of the ciphertext block changes whenever any bit of the key changed. The same work in sec (6.1) can be applied to the SKAC by defining a fixed plaintext blocks and encrypt it using the key K and changing the key bit position K_j and encrypt the same plaintext blocks by using the key changed.

The following plaintext encrypted using the proposed neural network with a specific key, when the block size is 64 bits.

Plaintext:

Neural network consist of set of neurons (nodes), each node represents a bit in plaintext and key. These are modeled after neurons, with weighted links interconnecting the units together. The main difference between ANNs and together learning mechanism is that it is composed of these units they work together in a highly parallel manner.

Key:

p u t g k q l i

The proposed neural network will generate the following ciphertext depended on specific key.

Ciphertext:

```
Û&iš.u@oð7k‡=r@bú-o • <m@nóco • ;9_gµ-y□ =vrµkr ‡ + | _ ( ¹ c y
‰,q@oú'yÈ=|_sð0y†;j@`µ!uœop!å/}□ !m_yác}†+9dìm<œ'|_dµ"n□ o9nñ&p
 • +9_gá&nÈ!|_sú-oÄon  uýck□ &~_uð'<„&wrµ*rœ*k_nû-y ‹ ; p
fµ7t • olhá0<œ~_uý&nÆoM_dµ.}□ !9_hó%yš*w_dµ!yœ8|_oµ_R¦<9_oñch‡
(|_ið1<„*x_oü -{È"|_iô-u›"9      rµ7t‰;9u µ * o È , v q ú 0 y Œov_!á+y›*9
oü7oÈ;q_xµ4 sš $ 9 _ n ò & h € * k @ h û c } È ' p _ i ù : < ˜.
k_mù&pÈ"xoð12
```

Figure (4a) illustrates the avalanche effect when one bit changes in plaintext and its effect on ciphertext.

Figure (4b) illustrates the avalanche effect when one bit changes in key and its effect on ciphertext.

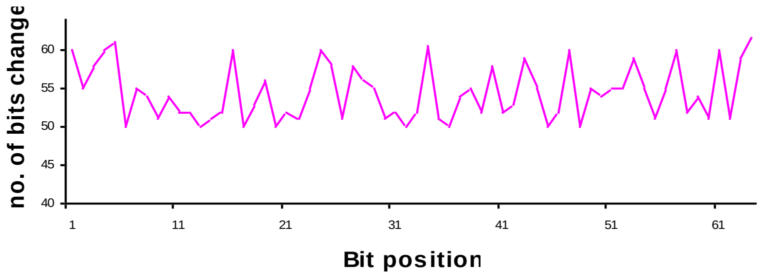


Figure (4a): Number of bits changed in C_i for each bit change in PT_i

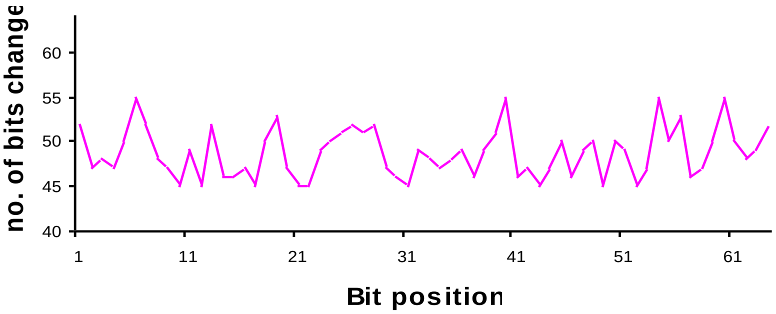


Figure (4b): Number of bits changed in C_i for each bit change in Ki

The following plaintext encrypted using the proposed neural network with a specific key, when the block size is 128 bits.

Plaintext:

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.

Key:

f y i s g h k w p d i h b z j v

The proposed neural network will generate the following ciphertexts depended on a specific key.

Ciphertext:

UÊé»áÓ_ò]ÂðÈµ□P•x˜ó·óÍ_éWÝðŽ‘Dl‡=˜ü âš_ò[Æ¼ßðzC†uÔñ·ëšõHË¢Å
¿dL‘wÌô½éÉ@ôZŽ£Ï½zN‘4Èï½äß_èUÀ·†µfG™qÖé¡§Õ_»RË¥Ô¿dQÔ`Ðü¦
§Ü_õ_Ú¹É¾*C‡4Ù½±èÖþ_Ú¹Ðµ*Q□ gÌø¿©š4óYÜµ†µrK‡`˜ë³õÓ_îOŽÔ¿

hN‘yË½»éš_úHÚµÔ¾*P‘w×ú¼îÎôRŽ¤Î±~_œaÕü¼ôš_þYÃðË¿xGÔqÞû»ä
Ó_õHŽ¹ÈðyM˜bÑóµ§Û_»_Á½Ö±xG • 4ÌòòäÕëIÚµÔ

Figure (5a) illustrates the avalanche effect when one bit changes in plaintext and its effect on ciphertext.

Figure (5b) illustrates the avalanche effect when one bit changes in key and its effect on ciphertext.

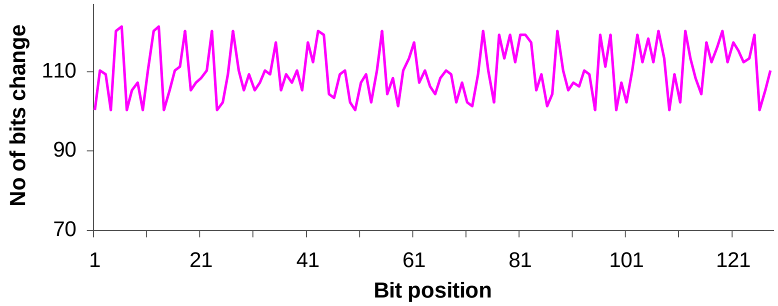


Figure (5a): Number of bits changed in C_i for each bit change in PT_i

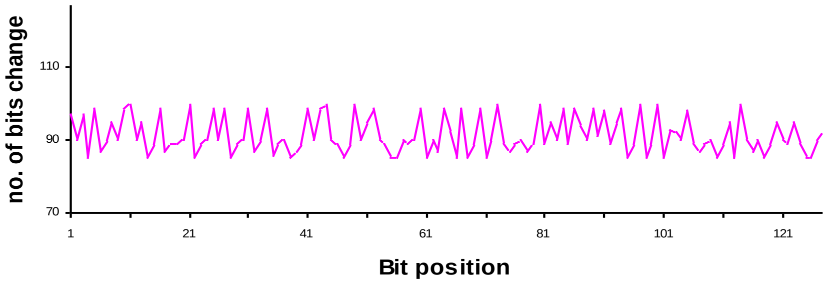


Figure (5b): Number of bits changed in C_i for each bit change in PT_i

When compared the results of proposed system explained in Figure (4a) to Figure (5b) with the results of the traditional methods, it show a great improvement have been happened because of the high diffusion that occurs on the results by changing specified bit either in plaintext or in key and its effects on the number of bits changed in ciphertext.

Table (2) illustrates the difference of diffusion between the Data Encryption Standard (DES) (see Appendix B) traditional method and the Neuron-block ciphering.

Table (2): The Difference of diffusion between DES and Neuron-Ciphering.

	DES [9]		Neuron-block cipher	
	Min bits change	Max bits change	Min bits change	Max bits change
Plaintext	1	32	50	61
Key	0	35	45	55

The min bits change and max bits change in plaintext and key of neuron-block cipher concluded from Figure (4a) and Figure (4b).

7. Conclusions

There are several conclusions gain from the research, that can be explain in the following point:

1. Organization can gain competitive advantages through the proper use of neural network approaches; however neural networks are being increasingly used for problems involving function approximation. Many researchers believe that neural networks offer the most promising approach to building truly intelligent computer systems. As has been informed in the introduction, neural networks or artificial neural networks are algorithms that can be used to perform nonlinear statistical modeling and provide a new alternative to logistic regression of which is commonly used method for developing predictive models for any business application.
2. The block cipher algorithms base on neural network techniques will process the plaintext from sources and learn from it to produce the ciphertext. This ability differs from traditional block cipher because it does not depend upon the prior knowledge of rules. Besides, ANNs can reduce the development time by learning the underlying relationships even when they are difficult to find and describe. The proposed cryptosystem itself will be able to solve the problems of the lack of traditional cryptosystem.
3. There are many modern block cipher methods, but the present algorithms based on neural network techniques. These algorithms offer a high security system compared with traditional algorithms through a high diffusion resulted from using the new approach. The high diffusion comes from the strategy of neural network. Neural network consist of set of neurons (nodes), each node represents a bit in plaintext and key. These are modeled after neurons, with weighted links interconnecting the units together. The main difference between ANNs and together learning mechanism is that it is composed of these units which are work together in a highly parallel manner.
4. The new algorithms may be considered adaptive, because of its ability to generate, different schemes for the same plaintext and secret key with weights variations of neural network.

References

1) Carol C., "Baltimore Technologies Announces Integration of (ASE) Algorithm into Baltimore Product Set", http://www.baltimore.com/ devzone/ase/index.html, 2000.

2) Stallings W., "Cryptography and Network Security: Principles and Practice" , New Jersey: Prentice Hall, 2003.

3) R. Callan R., "The Essence of Neural Networks", Prentice Hall Europ, 1999.

4) J. Kortelainen "Pattern Recognition and Neural Networks", http://www.ee.oulu.fi /research /tklab/courses /521497S/progex/Instructions_BP.pdf,2009

5) Razavi-arazavi A., "Analysis and Evaluation of Cryptographic Pseudo Random Number Generators", http://www.swen.uwaterloo.ca/~arazavi/papers/ece628proj.pdf ,2004

6) Jorstad N., "Cryptography Algorithm Metrics", http://csrc.nist. gov/nissc/1997 proceedings/128.pdf,1997

7) Gustafson H., dawson E., Nielson L. and Caell W., "Computer & Security", No. 8, Vol. 13, 1994

8) Tannenbaum A. S, "Computer Network," Prentice Hall, 1996.

9) Kenneth. R., "Data Network Handbook,", Galgotia Publications (P) LTD, 1998.

Appendix A

χ^2 - Distribution Table

Table (A.1): χ^2 - distribution table

ν	α							
	0.99	**0.975**	**0.95**	**0.05**	**0.025**	**0.01**	**0.005**	**0.001**
1	**0.0157**	**0.0982**	**0.0393**	**3. 841**	**5.024**	**6.635**	**7.879**	**10.827**
2	**0.0201**	**0.0506**	**0.103**	**5.991**	**7.378**	**9.210**	**10.597**	**13.815**
3	**0.115**	**0.216**	**0.352**	**7.815**	**9.348**	**11.345**	**12.838**	**16.266**
4	**0.297**	**0.484**	**0.711**	**9.488**	**11.143**	**13.277**	**14.860**	**18.466**
5	**0.554**	**0.831**	**1.145**	**11.070**	**12.832**	**15.086**	**16.750**	**20.515**
6	**0.872**	**1.237**	**1.653**	**12.592**	**14.449**	**16.812**	**18.548**	**22.457**
7	**1.239**	**1.690**	**2.167**	**14.067**	**16.013**	**18.475**	**20.278**	**24.3219**
8	**1.646**	**2.180**	**2.733**	**15.507**	**17.535**	**20.090**	**21.955**	**26.124**
9	**2.088**	**2.700**	**3.325**	**16.919**	**19.023**	**21.666**	**23.589**	**27.877**
10	**2.558**	**3.247**	**3.940**	**18.307**	**20.483**	**23.209**	**25.188**	**29.588**
11	**3.053**	**3.816**	**4.575**	**19.657**	**21.920**	**24.725**	**26.757**	**31.264**
12	**3.571**	**4.404**	**5.226**	**21.026**	**23.337**	**26.217**	**28.300**	**32.909**
13	**4.107**	**5.009**	**5.892**	**22.362**	**24.736**	**27.688**	**29.819**	**34.528**
14	**4.660**	**5.629**	**6.571**	**23.685**	**26.119**	**29.141**	**31.319**	**36.123**
15	**5.229**	**6.262**	**7.261**	**24.996**	**27.488**	**30.578**	**32.801**	**37.697**
16	**5.812**	**6.908**	**7.962**	**26.296**	**28.845**	**32.000**	**34.267**	**39.252**
17	**6.408**	**7.564**	**8.672**	**27.587**	**30.191**	**33.409**	**35.718**	**40.790**
18	**7.015**	**8.231**	**9.390**	**28.869**	**31.526**	**34.805**	**37.156**	**42.312**
19	**7.633**	**8.907**	**10.117**	**30.144**	**32.852**	**36.191**	**38.582**	**43.820**
20	**8.260**	**9.591**	**10.851**	**31.410**	**34.170**	**37.566**	**39.997**	**45.314**

Appendix B

Data Encryption Standard (DES)

In January 1977, the U.S. government adopted a product cipher developed by IBM as its official standard for unclassified information. This cipher, DES, was widely adopted by the industry for use in security products. It is no longer secure in its original form, but in modified form it is still useful.

An outline of DES is shown in Figure.(B.1a). A plaintext is encrypted in block of 64-bits, yielding 64-bits of ciphertext. The algorithm, which is parameterized by a 56-bit key, has 19 distinct stages. The first stage is a key independent transposition on the 64-bit plaintext. The last stage is the exact inverse of that transposition. The stage prior the last one exchanges the leftmost 32-bits with the rightmost 32-bits. The remaining 16 stages are functionally identical but are parameterized by different functions of the key. The algorithm has been designed to allow decryption to be done with the same key as encryption. The steps are just run in the reverse order [8].

The operation of one these intermediate stages is illustrated in Figure.(B.2b). Each stage takes two 32-bit inputs and produces two 32-bit outputs. The left output is simply a copy of the right input. The right output is the bitwise XOR of the left input and a function of the right input and the key for this stage, K_i. All the complexity lies in this function [8].

The function consists of four steps, carried out in sequence. First, a 48-bit number, Expanding (E) , is constructed by expanding the 32-bit R_{i-1} according to a fixed transposition and duplication rule. Second, E and Key ($K_{i)}$ are XOR together. This output is then partitioned into eight groups of 6 bits each, each of which is fed into a different Substation-box (S-box). Each of the 64 possible inputs to an S-box is mapped onto a 4-bit output. Finally, these 32 bits are passed through a Permutation-box (P-box).

In each of the 16 iteration, a different key is used. Before the algorithm starts, a 56-bit transposition is applied to the key. Just before each interaction, the key is partitioned into two 28-bit units, each of which is rotated left by a number of bits dependent on the iteration number, K_i is derived from this rotated key by applying another 56-bit transposition to it. A different 48-bit subset of the 56 bits is extracted and permuted on each round [8].

Because DES is standard, and because it is quite computer-intensive to encode and decode data, it has been embedded in silicon (a specific chip). The DES chip accepts a 64-bit block of plaintext and the key; it then outputs the ciphertext. The same chip can be used to reverse the process by inputting the ciphertext and the key to produce the plaintext [9].

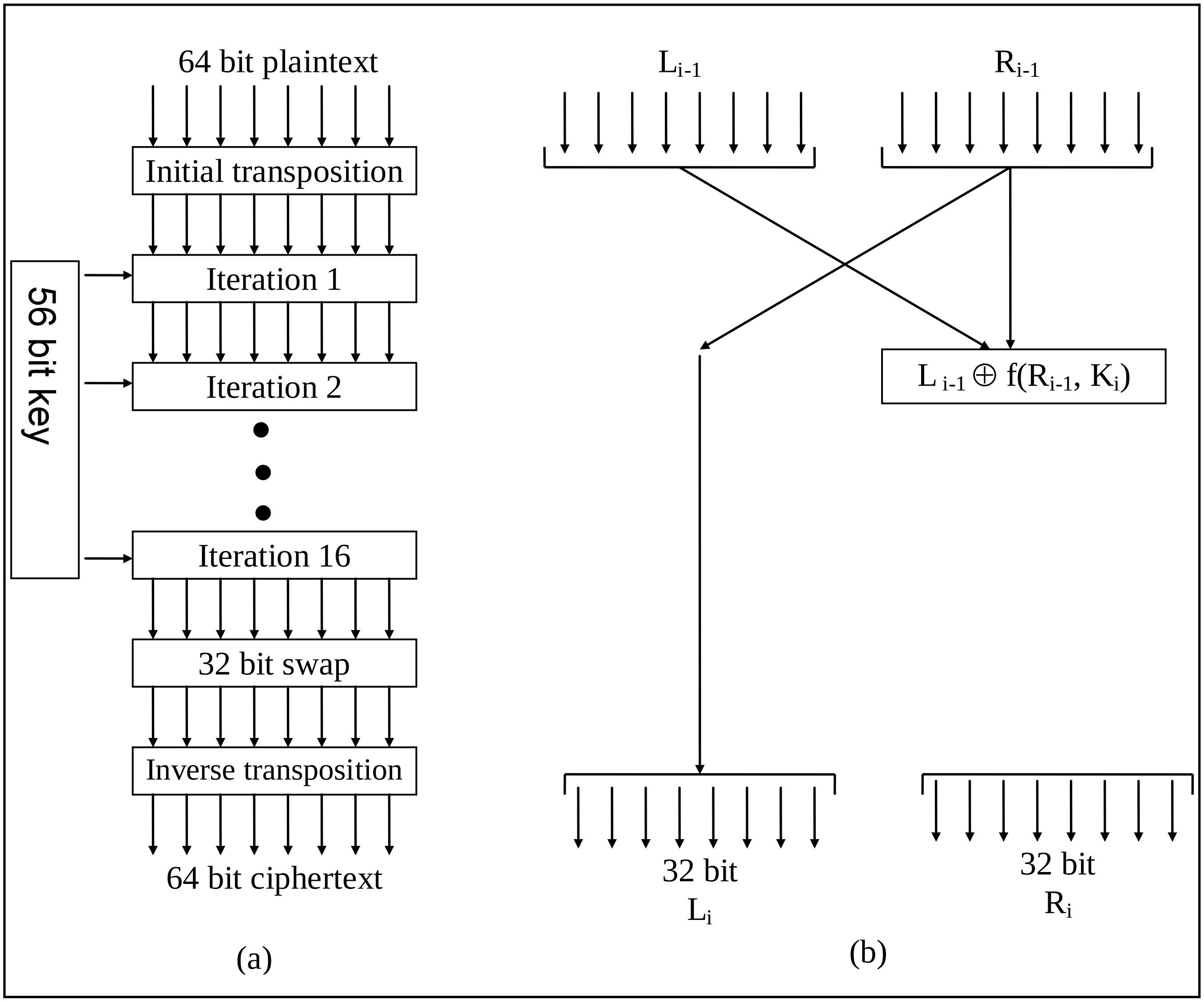


Figure (B.1): The data encryption standard, (a) General outline. (b) Detail of one iteration

The following plaintext encrypted using Data Encryption Standard (DES) with a specific key.

Plaintext:

Neural network consist of set of neurons (nodes), each node represents a bit in plaintext and key. These are modeled after neurons, with weighted links interconnecting the units together. The main difference between ANNs and together learning mechanism is that it is composed of these units they work together in a highly parallel manner.

Key:

p u t g k q l i

The Data Encryption Standard (DES) will generate the following ciphertext depended on specific key.

Ciphertext:

젝皤麹□陁□□讟ಹ쬀弅□넋蚒罹Ꙩ冇趦몢覍□뉛껀ꭗ坘Ꝩ鏗뾥Ǉ瞰□뾥뾜
皎ണ□竞髻檆嵂鑁薹鏗뾥蔓箣흰斡⋟現□□ಹ□v̇ 罹㷫轞郪崞鋧遒觳丟┅➡ꟾ
味레θ̂楖脫□찞鲆譚諭□퐕毷ɋ□唯헝鏍晡□衺렚曙□薹G絊□㚟□け豥\ 邬
蔓踋鏗ʒ駖□焯乌蕙燋□뛘霏뿔□沼敍□鼅꺖□刼薹□邬⌗毣道ㅆ簹□□陁
□豨꾕훮□态릦□蔓睚⍰⍍陁쏴駊陁□摟鏗□藞紓ɯ죴ㄲ웃陁□됬宰ℴ慼鏗✉
扉ي