B14559754

A BLOCK CIPHER DESIGN USING RECURRENT NEURAL NETWORKS

by

Shuwei Wu

A project
presented to Ryerson University
in partial fulfillment of the
requirements for the degree of

Master of Engineering
in the Program of
Electrical and Computer Engineering

Toronto, Ontario, Canada, 2003

© Shuwei Wu 2003

PROPERTY OF
RYERSON UNIVERSITY LIBRARY

UMI Number: EC53453

INFORMATION TO USERS

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleed-through, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

UMI®

UMI Microform EC53453
Copyright 2009 by ProQuest LLC
All rights reserved. This microform edition is protected against unauthorized copying under Title 17, United States Code.

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

AUTHOR'S DECLARATION

I hereby declare that I am the sole author of this project.

I authorize Ryerson University to lend this project to other institutions or individuals for the purpose of scholarly research.

Signature

I further authorize Ryerson University to reproduce this project by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Signature

BORROWER'S PAGE

Ryerson University requires the signatures of all persons using or photocopying this project. Please sign below, and give address and date.

A BLOCK CIPHER DESIGN USING RECURRENT NEURAL NETWORKS

ABSTRACT

As security has become a necessary component for business applications in many areas, research of new cryptography technology is desirable, especially the highly secure and efficient data encryption technique. A new block cipher designed based on recurrent neural networks is proposed for first time in the project. Recurrent neural networks have dynamics characteristics and can express functions of time. By introducing recurrent neural networks to cryptography, the proposed block cipher releases the constraint on the length of secret key. The inherited high by parallel processing capability of neural networks can also improve the encryption performance greatly. The recurrent neural networks make the block cipher strong to resist different cryptanalysis attacks and to provide data integrity and authentication service at the same time. The design of the proposed block cipher is presented and analyzed in detail. Simulation results provide illustrations. The proposed block cipher is flexible to be implemented either in software or in hardware for efficient data encryption purpose.

ACKNOWLEDGMENTS

The author wishes to thank my academic supervisor Dr. Alireza Sadeghian, who helps me, encourages me and guides me towards my academic as well as my personal success. He keeps on reminding me that research needs dedicated research-oriented attitude, hard working and courage to face the risk to explore new direction. The project would also not be successful without other members of the committee. They provide me a lot of help and suggestion throughout my work to solve problems.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF ILLUSTRATIONS

Chapter 1

INTRODUCTION

1. Introduction

1.1. Motivation

As security has become a necessary component for business applications in many areas, research of new cryptography technology is valuable. The requirement of the performance and security of data encryption keeps on increasing so that new data encryption technology is always desirable.

The research of new cryptography technology is under pressure. The development of both hardware and software of computer has entered an accelerative period. Consequently, new cryptanalysis techniques keep on coming up and the cost for cryptanalysis attack decreases dramatically. Previous cryptography techniques find themselves no longer secure. How to increase the security level without degrading the performance is an imperative problem that needs to be solved. Innovative technique to improve the performance and security of the data encryption will have very important practical significance.

1.2. Scenarios

There are several scenarios where a new cryptography technique needs to be considered. The one most important is that strong cryptography must ensure the confidentiality of the data. To protect the privacy of the information, confidential is the basic service of cryptography technique.

Besides confidentiality, data authentication is another factor. Nowadays, there are more and more denial of service attacks happening. One important reason is that many systems have no authentication mechanism. The identity of the other side of network cannot be identified so that the system cannot ensure the service request coming from legal customer.

Although the development of communication technology can help to remove the noise of the data communication, it is still possible for the data to be changed during transit by purpose. Cryptography technique needs to provide the data integrity service to assure that no one can change messages without being detected. Data integrity is the third scenario to consider.

All of the above aspects need to be considered together so that the cryptography technique can have the capability to resist different types of cryptanalysis attack.

1.3. Summary

In summary, cryptography research can bring up new tools to ensure the confidentiality, authentication and integrity of the data. New cryptography technique has to find an innovative method to increase the level of security without significant performance degrading of the system.

In the project, a review of cryptography and neural network will uncover that neural network as a parallel computing technique can be used for possible cryptography usages. Next my new block cipher design will be described in detailed followed by modeling and analysis of the design. The simulation experiments are also presented to provide illustrations. Finally, in the conclusion section, it will show that the project contributes to present a new block cipher design for high secure and high performance data encryption.

Chapter 2

PRELIMINARIES

2. Preliminaries

2.1. Cryptography

2.1.1. Cryptography Overview

Cryptography is the art and science to protect a secret message from anyone who does not know the secret key. As the importance of network security increases, cryptography becomes a critical technique for the secure communication. Cryptology includes two aspects, cryptography and cryptanalysis. Cryptology is the study of cryptography and cryptanalysis. These two aspects are so close to each other just as the two surfaces of one coin.

Cryptography has a long and fascinating history. In the past, cryptography was used mainly to secure the communications of the military and royalty. The widespread use of computers has expanded the need for secure communications around the globe. DES, the Data Encryption Standard, is the most well known cryptographic mechanism in history. It remains the standard method for securing electronic commerce for many financial institutions around the world. A lot of cryptanalysis has been emphasized on this algorithm. The most striking development in the history of cryptography is Diffie and Hellman published "New Directions in Cryptography" in 1976 [16]. It introduced the revolutionary concept of public-key cryptography and also provided a new and ingenious method for key exchange, the security of which is based on the discrete logarithm problem[16]. The public key cryptography is usually used for key distribution and authentication. The key length of public key cryptography is much larger than symmetrical key cryptography so that the public key cryptography is not suitable to encrypt actual data. Table 2.1 is a comparison of the key length between symmetric and public key [1].

Table 2.1 Symmetric and Public key lengths with similar resistances to Brute-force attacks

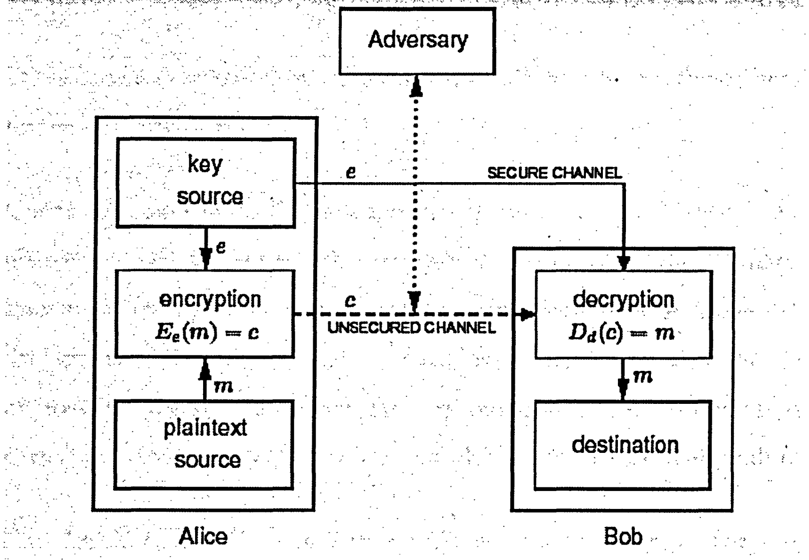
Symmetric Key Length	Public Key Length
56 bits	384 bits
64 bits	512 bits
80 bits	768 bits
112 bits	1792 bits
128 bits	2304 bits

In this report, emphasis will be placed on those aspects that are most practical and applied, the principles, techniques, and algorithms of interest in symmetrical key cryptographic practice, especially the block cipher design.

2.1.2. Symmetric-key Encryption

Consider an encryption scheme consisting of the sets of encryption and decryption transformations $\{E_e : e \in K\}$ and $\{D_d : d \in K\}$, where K is the key space. The encryption scheme is said to be symmetric-key if for each associated encryption/decryption key pair (e, d), it is easy to determine d knowing only e, and to determine e from d. Fig. 2.1 is the diagram illustrating the symmetrical-key encryption scheme where the secure channel may set up the technique such as public-key encryption scheme for key exchange.

Fig. 2.1 Symmetric-key Encryption Scheme



There are two classes of symmetric-key encryption schemes: block ciphers and stream ciphers. A block cipher is an encryption scheme that breaks up the plaintext messages to be transmitted into strings (called blocks) of a fixed length, and encrypts one block at a time. Most well-known symmetric-key encryption techniques are block ciphers. Stream ciphers can be thought as the block ciphers with block length equal to one.

Two important classes of block ciphers are substitution ciphers and transposition ciphers. Product ciphers combine these two classes. Substitution ciphers are block ciphers that replace symbols (or groups of symbols) by other symbols or groups of symbols. Transposition cipher simply permutes the symbols in a block. Simple substitution and transposition ciphers individually do not provide a very high level of security. However, by combining these transformations it is possible to obtain strong ciphers. The composition of a substitution and a transposition will be called a round. A substitution in a round is said to add confusion to the encryption process whereas a transposition is said to add diffusion. Most modern block cipher systems apply a number of rounds to encrypt plaintext.

Diffusion can eliminate the statistical relationships between the ciphertext and the underlying plaintext.

Confusion can break the relationship between the ciphertext and the secret encryption key to prevent the cryptanalyst from using the ciphertext and the known cipher method to figure out the encryption key. Confusion usually means the cryptographer introduces a complex nonlinear substitution so that the cryptanalyst cannot figure out the key but only some ciphertext patterns. An example of confusion is the well-known S-boxes of the DES algorithm.

The Polybius cipher from ancient Greece is an example for the diffusion. Suppose the shared secret is the following Table 2.2

Table 2.2 Shared Secret Key

	1	2	3	4	5
1	A	B	C	D	E
2	F	G	H	I/J	K
3	L	M	N	O	P
4	Q	R	S	T	U
5	V	W	X	Y	Z

For example, the message "STEPHEN" will be encrypted as "43 44 15 35 23 15 33" (the blanks between numbers are not included in the actual ciphertext). This step is the substitution. Since the most frequent character "E" also has the most frequent appearance "15" in ciphertext, the statistical relationship between the ciphertext and the plaintext is not totally hidden. Based on the result of substitution, a step of transposition is introduced to add diffusion to the ciphertext as following Fig. 2.2.

Split the ciphertext "43441535231533" as

Fig. 2.2 Ciphertext Split

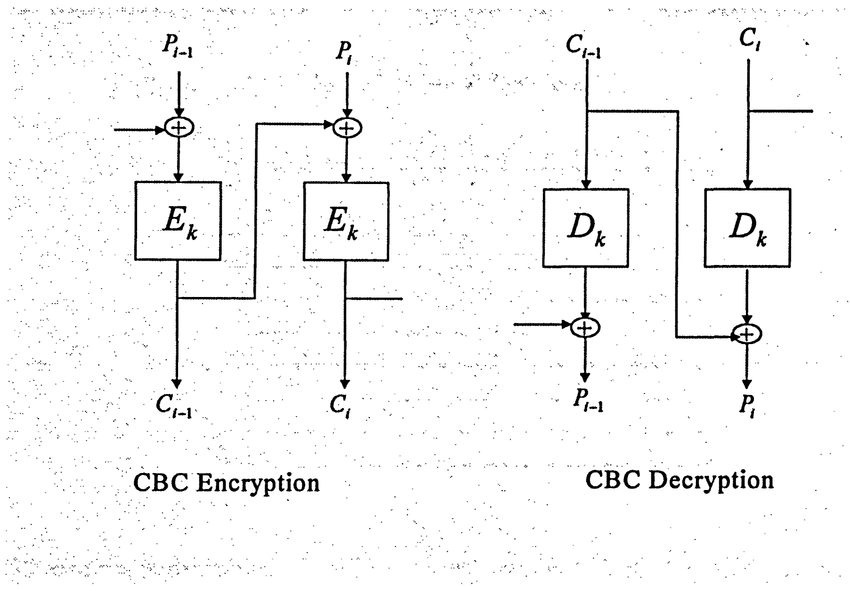
4	4	1	3	2	1	3
3	4	5	5	3	5	3

The result of the transposition will be "44132133455353". Now the relationship between statistical pattern of the ciphertext and the plaintext is broken. The transformation can have the effect of diffusion.

Besides security, data integrity and authentication are also necessary for secure communication. Ciphertext will usually be attached with a message authentication code (MAC) for that purpose. MAC is a key-dependent one-way hash function. MACs can be used to authenticate cipher text between users and to determine whether the ciphertext has been altered or replaced. Most of symmetric-key algorithms prepare MAC by encrypting the hash value of ciphertext. When a block cipher works in CBC (Cipher Block Chaining) modes as Fig. 2.3, where P_i are the plaintext blocks and C_i are the cipher text blocks, a simplest way to make

MAC is to use the last encrypted block to be the MAC, and this is referred to as "CBC-MAC".

Fig. 2.3 CBC Mode



In the following section, we will investigate two important block cipher designs in detail.

2.1.3. Block Cipher DES and Rijndael

DES[17] and Rijndael[18] are the previous and current encryption standard selected by National Institute of Standards and Technology (NIST). They are the most important symmetric-key block ciphers. The review of them will uncover some design principles of the block cipher design.

DES is a block cipher encrypting data in 64-bit blocks with 56-bit key length. Suppose that DES operates on a 64 bits block of plaintext M $= m_1 ... m_{64}$, the input key is $k_1...k_{64}$ and the ciphertext block will be C $= c_1...c_{64}$, the algorithm can be described as following:

1) An initial permutation(IP) to break the block into a left half and a right half (L_0, R_0) according to Table 2.3;

Table 2.3 Initial Permutation

IP							
58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

After initial permutation, L_0 will be $m_{58}m_{50}...m_8$ and R_0 will be $m_{57}m_{49}...m_7$.

2) Computing 16 round keys K_i from 64-bit input key K;

At first, the 64-bit input key K is reduced to a 56-bit key K2 that is divided into two 28-bit halves (C_0, D_0) according to Table 2.4.

Table 2.4 Transfer 64-bit key to 56-bit key

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

After this step, the C_0 will be $K_{57}K_{49}...K_{36}$ and D_0 will be $K_{63}K_{55}...K_4$.

Next v_i, $1 \le i \le 16$ is introduced as following: $v_i = 1$ for $i \in \{1,2,9,16\}$, otherwise $v_i = 2$. For i from 1 to 16, $C_i \leftarrow (C_{i-1} << v_i), D_i \leftarrow (D_{i-1} << v_i)$, where << denotes left circularly shift.

After the left circularly shift, the 48-bit round keys K_i can be selected according to Table 2.5. This step is called a compression permutation (CP).

Table 2.5 Compression Permutation

14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

3) For i from 1 to 16, computing L_i and R_i as following:

$$L_i = R_{i-1}$$
$$R_i = L_{i-1} \oplus f(R_{i-1,}K_i)$$

At the final round (16th round), exchange block L_{16} and R_{16}.

The function f can be denoted as $f(R_{i-1}, K_i) = P(S(E(R_{i-1}) \oplus K_i))$ and explained as following:

At first, R_{i-1} is expending from 32 to 48 bits using E according to Table 2.6:

Table 2.6 Expend 32-bit key to 48-bit key

E					
32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

Let $T \leftarrow E(R_{i-1})$, compute $T^1 \leftarrow T \oplus K_i$ and represent T^1 as eight 6-bit character strings: $(B_1, ..., B_8)$.

Next perform the S-Box Substitution S (T^1): $T^2 \leftarrow (S_1(B_1), S_2(B_2), ..., S_8(B_8))$, where $S_i(B_i)$ maps $B_i = b_1 b_2 ... b_6$ to the 4-bit entry in row r and column c of S_i in Table 2.7, where $r = 2 \cdot b_1 + b_6$ and $b_2 b_3 b_4 b_5$ is the radix-2 representation of c.

Table 2.7 S-Box Substitution

row	column number															
	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]	[13]	[14]	[15]
S_1																
[0]	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
[1]	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
[2]	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
[3]	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
S_2																
[0]	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
[1]	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
[2]	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
[3]	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
S_3																
[0]	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
[1]	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
[2]	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
[3]	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
S_4																
[0]	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
[1]	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
[2]	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
[3]	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14
S_5																
[0]	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
[1]	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
[2]	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
[3]	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
S_6																
[0]	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
[1]	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
[2]	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
[3]	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
S_7																
[0]	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
[1]	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
[2]	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
[3]	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12
S_8																
[0]	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
[1]	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
[2]	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
[3]	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

The final step of function f is the P-Box permutation P (T^2) according to Table 2.8, which maps each bit of the 32-bit output of the S-box substitution to an output position.

Table 2.8 P-Box Permutation

P			
16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

4) After the above 16 rounds operations, the final permutation (IP^{-1}) is performed as the inverse of the initial permutation (IP) according to table 2.9.

Table 2.9 Final Permutation

IP^{-1}							
40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

The 64-bit ciphertext block will be $C \leftarrow IP^{-1}(R_{16,}L_{16})$.

The DES decryption uses the same function as encryption with the same key but using the key in the reversed order. The effect of IP^{-1} will be cancelled out by IP so that (R_{16}, L_{16}) will be the output of IP^{-1} in decryption. Next the round 16 of encryption procedure will be cancelled out by the round 1 in decryption as following:

The round 1 in decryption will be:

$$L_1' = R_0' = L_{16} = R_{15}$$
$$R_1' = L_0' \oplus f(R_0', K_1') = R_{16} \oplus f(L_{16}, K_{16}) = L_{15} \oplus f(R_{15}, K_{16}) \oplus f(R_{15}, K_{16}) = L_{15}$$

Thus the round 1 decryption yields (R_{15}, L_{15}). This is the inverse of round 16 in encryption. The remaining 15 rounds will be cancelled out likewise one by one in reversed order.

From the above review of DES, it shows that the S-box performs the substitutions step for DES and is the only nonlinear operation in DES. This step determines the security of DES. To resistant the linear and differential attack, triple DES was proposed and effectively extending the DES key from 56 bits to 112 bits. The alternative of DES, Rijndael, releases this type of constraint on key length.

Rijndael is a byte-oriented block cipher which symmetric and parallel structure is derived from the square block cipher. In October 2000, Rijndael is announced as Advance Encryption Standard (AES). Rijndael supports variable key length: 128-bit, 192-bit and 256-bit and variable block length: 128-bit, 192-bit and 256-bit.

The round transformation of Rijndael contains four different transformations and can be expressed in the following pseudo C notation:

```
Round(State,RoundKey)
{
ByteSub(State);
ShiftRow(State);
MixColumn(State);
AddRoundKey(State,RoundKey);
}
```

The final round of the cipher is slightly different. It is defined by:

```
FinalRound(State,RoundKey)
{
ByteSub(State) ;
ShiftRow(State) ;
AddRoundKey(State,RoundKey);
}
```

Each step of the round function has its own particular character:

- ByteSub is nonlinear
- ShiftRow is inter-column diffusion
- MixColumn is inter-byte diffusion within columns
- Round key addition is a simple EXOR operation of the round key to the intermediate state.

From the above review of two important block ciphers we can see, although different block cipher designs has different round operations, most of them performs a nonlinear transformation to provide the security protection, such as the S-box of DES. The nonlinear transformation can make the cryptanalysis much more difficult when it is combined with other linear permutation operations. After the following review of neural networks, we can find out that neural networks and its learning procedure has this type of nonlinear dynamic feature and can possibly be constructed for cryptography purpose.

2.2. Neural networks

2.2.1. Neural networks Structure

Artificial neural networks, commonly referred to as "neural networks", have been motivated from its inception by the recognition that the brain computes in an entirely different way from the conventional digital computer. The brain contains billions of neurons with massive interconnections between them. Similarly, a neural networks is a massively parallel-distributed processor that is made up of neurons with interconnections between them.

A neuron is an information-processing unit of neural networks. A neuron k can be modeled by writing the following pair of equations:

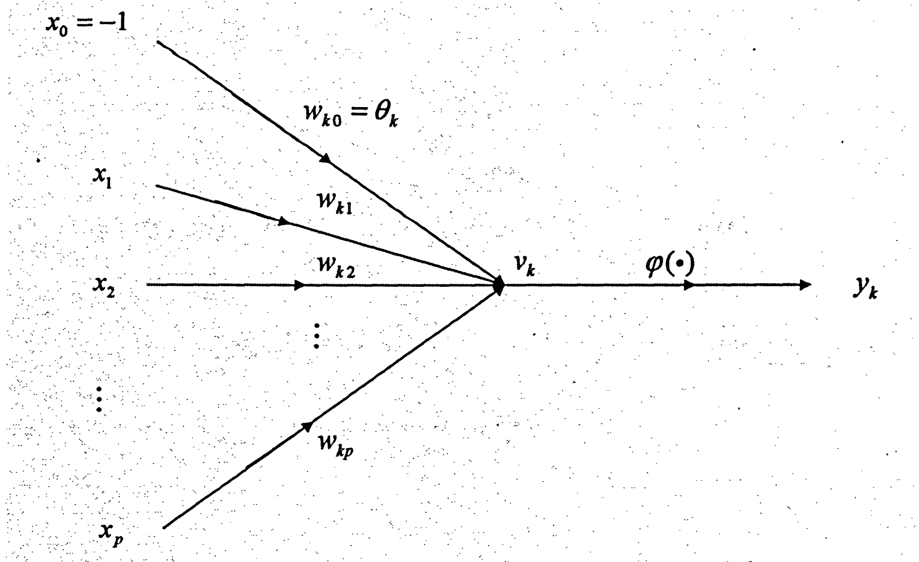
$$v_k = \sum_{j=1}^{p} w_{kj} x_j \tag{1.1}$$

and

$$y_k = \varphi(v_k) \tag{1.2}$$

where $x_1, x_2, \ldots, x_p$ are the input signals; $w_{k1}, w_{k2}, \ldots, w_{kp}$ are the synaptic weights of neuron k; v_k is the linear combiner output; $\varphi(\bullet)$ is the activation function; and y_k is the output signal of the neuron.

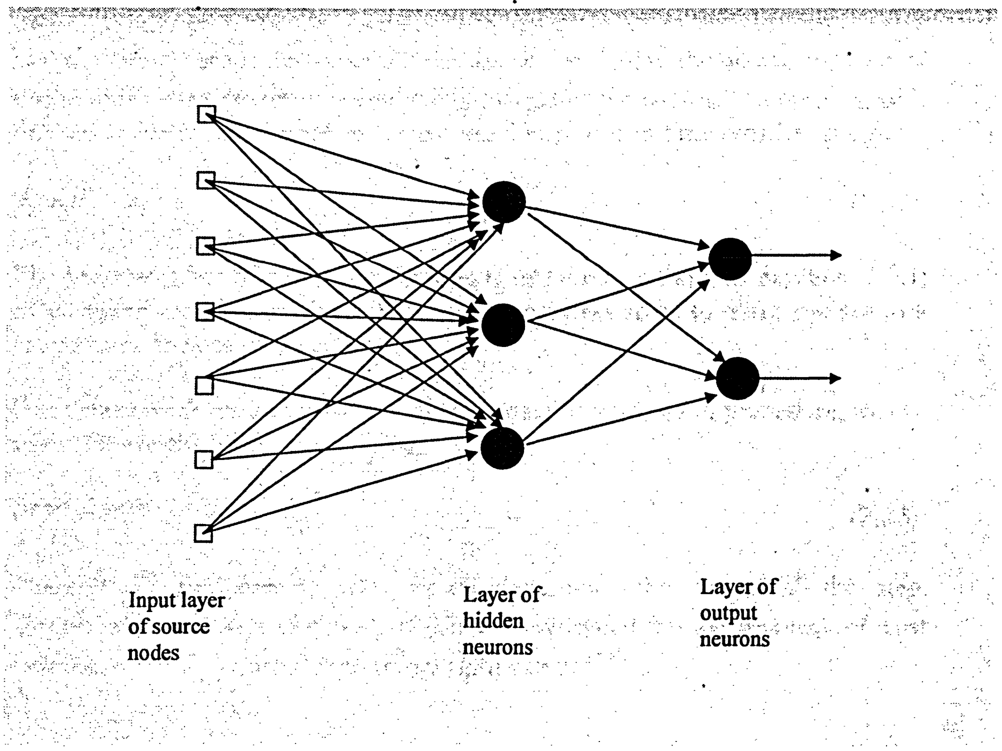
Fig. 2.4 Neuron Model



A neuron model can be illustrated as Fig. 2.4. The threshold θ_k of neuron k here is represented by a synaptic link with an input signal fixed at a value of -1.

Typically, the network consists of a set of sensory units that constitute the input layer, one or more hidden layers and an output layer of neurons. A fully connected feedforward Multiple Perceptrons networks (MLPs) with one hidden layer and output layer can be illustrated as Fig. 2.5.

Fig. 2.5 MLP



The neurons of one or more hidden layers of MLPs are not part of the input or output of the network. These hidden neurons enable the network to learn complex tasks by extracting progressively more meaningful features from the input patterns (vectors). This procedure is referred to as the learning procedure of neural networks.

2.2.2. Error-correction Learning

Suppose that the learning procedure of neural networks selects (2.2.1) as the cost function to minimize.

$$J = \frac{1}{2}\sum_k e_k^2 \tag{2.2.1}$$

Let d_k denote desired response for neuron k, y_k denote the actual response of this neuron when an input vector x is presented to the neuron. An error signal is defined as the difference between the desired response and the actual response:

$$e_k = d_k - y_k \tag{2.2.2}$$

The purpose of error-correction learning is to minimize the cost function (2.2.1) by changing the weight values W of neural networks so as to make the network to approach the desired response.

If the steepest descent method is applied for this minimization procedure, we can adjust the weight values as following:

$$W^{new} = W^{old} + \eta P \tag{2.2.3}$$

where η is the learning rate, which determines the length of the step, P represents a search direction, which is determined by the gradient of cost function value J evaluated at the old weight value W^{old}:

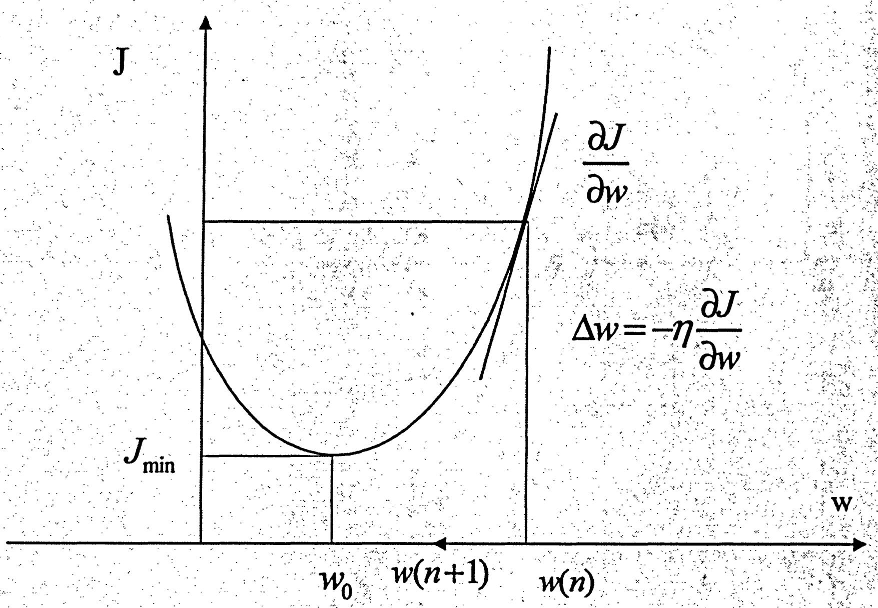
$$P = -\frac{\partial J}{\partial W^{old}} \tag{2.2.4}$$

According to the method of steepest descent, the adjustment of weight Δw should be in a direction opposite to the gradient vector so that we have:

$$\Delta w = -\eta \frac{\partial J}{\partial w} \tag{2.2.5}$$

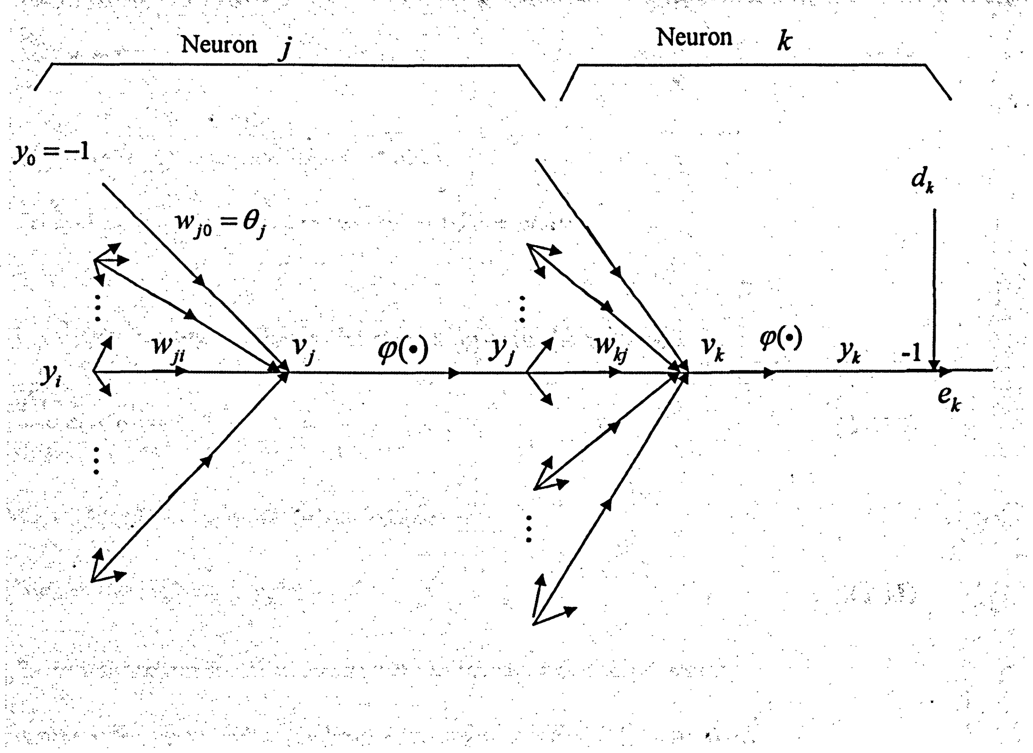
This relationship is illustrated as Fig. 2.6.

Fig. 2.6 Weight Adjustment by Steepest Descent Method


J
$\frac{\partial J}{\partial w}$
$\Delta w = -\eta \frac{\partial J}{\partial w}$
$J_{\min}$
w
w_0
$w(n+1)$
$w(n)$

To compute $\frac{\partial J}{\partial w}$ for (2.2.5), let us consider a neural networks as Fig. 2.7.

Fig. 2.7 Neural networks Sample



According (2.2.5), we set

$$\Delta w_{kj} = -\eta \frac{\partial J}{\partial w_{kj}} \tag{2.2.6}$$

Using chain rule, we have

$$\frac{\partial J}{\partial w_{kj}} = \frac{\partial J}{\partial v_k}\frac{\partial v_k}{\partial w_{kj}} = \frac{\partial y_k}{\partial v_k}\frac{\partial J}{\partial y_k}\frac{\partial v_k}{\partial w_{kj}} \tag{2.2.7}$$

According to (1.1), we have

$$\frac{\partial v_k}{\partial w_{kj}} = x_j \tag{2.2.8}$$

According to (1.2), we have

$$\frac{\partial y_k}{\partial v_k} = \varphi'(v_k) \tag{2.2.9}$$

where $\varphi'(v_k)$ is the derivation of $\varphi(\bullet)$.

To compute $\frac{\partial J}{\partial y_k}$, there are two cases to consider:

1) when the neuron k is in the output layer of the network:

$$\frac{\partial J}{\partial y_k} = -(d_k - y_k) \tag{2.2.10}$$

then (2.2.6) has been solved as following:

$$\Delta w_{kj} \approx \eta\varphi'(v_k)(d_k - y_k)x_j \tag{2.2.11}$$

2) when the neuron k is not in the output layer of the network:

Suppose the right hand side layer of neuron k is the l th layer, then

$$\frac{\partial J}{\partial y_k} = \sum_l \frac{\partial J}{\partial v_l}\frac{\partial v_l}{\partial y_k} = \sum_l \frac{\partial J}{\partial v_l}\left(\frac{\partial}{\partial y_k}\sum_i w_{li}y_i\right) = \sum_l \frac{\partial J}{\partial v_l} w_{lk} \tag{2.2.12}$$

Then (2.2.6) will be

$$\Delta w_{kj} \approx -\eta\varphi'(v_k)\sum_l \frac{\partial J}{\partial v_l} w_{lk}x_j \tag{2.2.13}$$

Now if we introduce δ_k as following, we can conclude the recursive computation for the weight values update.

$$\delta_k = \frac{\partial J}{\partial v_k} \tag{2.2.14}$$

Then (2.2.7) can be rewritten:

$$\Delta w_{kj} = -\eta \delta_k x_j \tag{2.2.15}$$

According to (2.2.11), when lth layer is the output layer, we can compute δ_l directly:

$$\delta_l = -\varphi'(v_l)(d_l - y_l) \tag{2.2.16}$$

Otherwise, according to (2.2.13), calculate the δ_k recursively as following:

$$\delta_k = \varphi'(v_k)\sum_l \delta_l w_{lk} \tag{2.2.17}$$

So we back-propagate the sensitivity of error signal layer by layer according to (2.2.16) and (2.2.17) and update the weight values according to (2.2.15). This is the standard back-propagation algorithm for the error-collection learning.

2.2.3. Recurrent Neural networks

In general, a network that has closed loops in its topological structure is considered a recurrent network. A recurrent network is a network with feedback, some of its outputs are connected to its inputs. There are many types of recurrent neural networks that have attracted lots of research such as Hopfield network[19], Elman network[9], real-time recurrent neural networks[2] etc. Characterized by the use of nonlinear processing units and utilization of feedback, recurrent neural networks are nonlinear dynamic systems. Because recurrent networks have feedback paths, they can demonstrate temporal behaviour, that is, both spatial and temporal patterns can be generated and stored in the network. The use of feedback connections in the recurrent network makes it less sensitive to noise and permits it to learn faster.

2.3. Summary

The review of the cryptography and neural networks uncovers that there are several similarity between these two different fields. In fact there is research work to apply neural networks for cryptography purpose recently. Su, S., Lin, A. and Jui-Cheng Yen [3] propose to use the chaotic output of neural networks to encrypt signal. Liew Pol Yee and Liyanage C. De Silva [4] propose to use neural networks to design a block cipher. Their proposed block cipher is based on complex bit operation.

According to the review, although there are many different cryptography algorithms, the fundamental principle of block cipher does not change. The combination of diffusion and confusion forms the round operation to provide security for the data. Multiple round operations tend to perform complex nonlinear substitution and permutation over the plain text so as to make the cryptanalysis extraordinary difficult. On the other hand, neural networks are a nonlinear dynamic machine that expands the expression of input data as a linear combiner of the inputs to the synapses, and then perform a nonlinear transformation. While the data is calculated feedforward through the multilayer of neural networks, it has the similar effect of multiple round operations in block cipher. That is linear permutation followed by nonlinear transformation. Without the knowledge of the weight matrix, the analysis of the data based only on the output will be very difficult. If the learning procedure can be controlled, it is possible to change neural networks to be an infinite state machine. As a result, the pattern is revealed but never perfectly repeated and the neural system becomes a complex system to generate complex variation.

Chapter 3

PROPOSED BLOCK CIPHER BY RECURRENT NEURAL NETWORKS

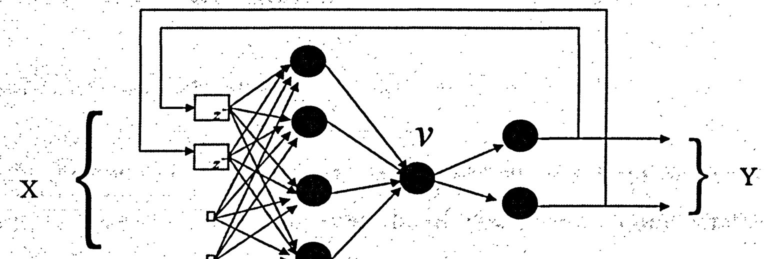
3. Proposed Block Cipher By Recurrent Neural networks

3.1. Block Cipher Design

The proposed block cipher is based on a recurrent multilayer neural networks. The structure of the block cipher is presented in details as following.

Suppose that Alice and Bob are the users of the communication, both Alice and Bob will have an identical multilayer perceptron neural networks (MLP) that is known by public. One example is illustrated as Fig. 3.1.

Fig. 3.1 Block Cipher Structure



The neural networks is a multilayer neural networks with two constraints:

- The dimension of the input vector, n, is two times of the dimension of the output vector, m;
- The number of neuron is equal to one in one of the hidden layers and the output of this neuron is notated as v.

The weight and bias of the neural networks are initialized to one identical value, which is also known to public.

Alice and Bob will exchange a secret key S by some public key or key exchange algorithm such as Diffie-Hellman algorithm. There is no constraint of the secret key length but the secret key S must contain three parts of information:

- Input vector $X=[x_1,x_2,...,x_n]^T$;
- Training target $Y=[y_1,y_2,...,y_m]^T$;
- A critical value of learning rate self-adaptive procedure α.

The first two parts of the secret key S will be used to train the neural networks for key extension purpose. The learning algorithm for key extension can simply adopt the standard back-propagation algorithm (2.2.11, 2.2.14, 2.2.15, 2.2.16 and 2.2.17) instead of recurrent learning.

3.1.1. Key Extension

The input vector X and the training target Y will be presented to the neural networks for training. The neural networks will train for the same iterative times, which will be referred to as "epoch". The number of epoch is open and it will be set to be a relatively large number even it will make the neural networks over-trained. In fact, all of the learning parameters such as learning rate, training function etc, can be known by public except for secret key S. The purpose of the training process here is to make the neural networks detect and store or even "remember" the features information of the secret key S. After enough training, the neural networks will initialize itself according to the secret key S. Now the feature information of the secret key S is stored in the synaptic weights of the hidden layers. Due to the feature of the standard Back-Propagation learning algorithm, it is commonly assumed the hidden layers of neural networks are chaotic and unpredictable. This will help to keep the secret key S secret perfectly by the synaptic weights of the hidden layers. The well-trained neural networks will keep secret and become the extended secret key for the following encryption and decryption procedures. The last actual output of the neural networks during the key extension procedure will be the initial vector for the encryption procedure, which is notated as M_0.

3.1.2. Data Encryption

Basically the encryption procedure performs three steps:

- Cipher text generation;
- One epoch training of neural networks;
- Learning rate Self-adaptation.

The one epoch training step keeps on updating the weight matrix of neural networks. This makes the analysis of the weight matrix becomes difficult to cryptanalyst without the knowledge of initial state of the network.

If the feed-forward operation of the neural networks is notated as function f, f can be decomposed into two parts f_1 and f_2 according to the structure of the block cipher. As mentioned before, the neural networks must contain one hidden layer that has only one neuron the output of this neuron is notated as v. If this hidden neuron is notated as neuron z, the feed-forward operation over the weight and bias matrix on the left hand side of neuron z (from the input layer to neuron z) and the neuron z itself will be function f_1. The feed-forward operation over the weight and bias matrix on the right hand side of neuron z (from neuron z to output layer) will be function f_2. f_1 will be used for MAC verification purpose. Both f_1 and f_2 will help to decrypt the cipher text and will be described as following section.

3.1.2.1. Cipher text generation

A plaintext will be transformed to vectors $M_i = [m_1, m_2, ..., m_n]^T, i = 1,2,3,...$ according to the dimension of input vectors. The first message vector M_1 will combine with the initial vector from key extension procedure M_0 to build up the first input vector $X_1 = \begin{bmatrix} M_0 \\ M_1 \end{bmatrix}$. $\begin{bmatrix} M_0 \\ M_1 \end{bmatrix}$ means the two n by 1 vectors M_0 and M_1 concatenate with each other to form another $2n$ by1 vector X_1. X_1 will present to the neural networks to produce the intermediary neuron output V_1 in the hidden layer and the output Y_1 of the neural networks. Next error signal E_1 will be calculated as $E_1 = M_1 - Y_1$. Here the neural networks M_1 is the target of the identity mapping. Finally E_1 and V_1 will be the first block of the cipher text $C_1 = \{V_1, E_1\}$.

3.1.2.2. One epoch training

After the cipher text block is constructed, the neural networks will be trained for one epoch with $X_1 = \begin{bmatrix} M_0 \\ M_1 \end{bmatrix}$ as input vector and M_1 as the training target.

From the second plain text block on, the previous output of the neural networks Y_{i-1} will combine next plain text block M_i to build up the current input vector

$X_i = \begin{bmatrix} Y_{i-1} \\ M_i \end{bmatrix}$. The above two steps of the encryption procedure will repeat to generate the values V_i and Y_i and train the neural networks for one epoch.

In summary, the cipher text blocks C_i will be constructed as following:

$$V_i = f_1(X_i) \tag{3.1.1}$$

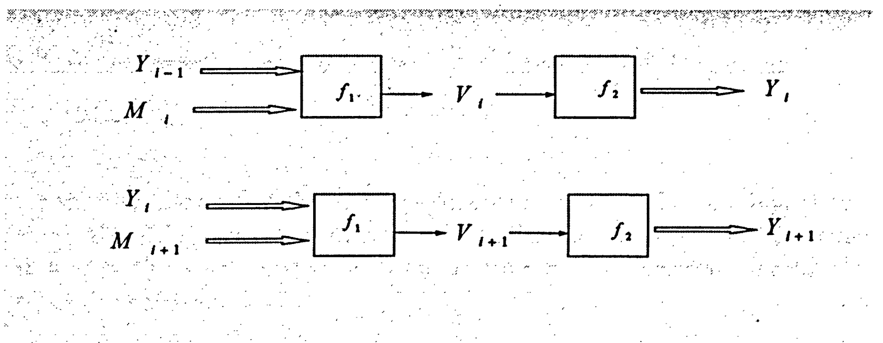
$$Y_i = f_2(V_i) \tag{3.1.2}$$

$$E_i = M_i - Y_i \tag{3.1.3}$$

$$C_i = \{V_i, E_i\} \tag{3.1.4}$$

In fact, the above two-step encryption procedure makes the block cipher working in CBC mode implicitly as Fig. 3.2.

Fig. 3.2 Implicit CBC Mode



In order to ensure the security of the block cipher, the learning rate need to be set to a large value. The detailed analysis of the learning rate control will be provided in the following analysis chapter. The following learning rate self-adaptive algorithm is introduced as a necessary step after each one epoch training to adapt the learning rate of the block cipher for arbitrary function.

3.1.2.3. Learning rate self-adaptation

At first the algorithm detects the trend of the learning procedure by monitoring the mean square error (MSE) performance function, which is the average of the sum of the square of the error E_i. Accordingly, the learning procedure adjusts the learning rate by a multiplicative-increase gradual-decrease (MIGD) method.

A low-pass filter for the MSE as following will perform the learning trend detection:

$$T(k) = \gamma T(k-1) + (1-\gamma) MSE(k) \tag{3.1.5}$$

γ is a coefficient that is between 0 and 1, for example $\gamma = 0.5$. $T(k)$ is the output of the low-pass filter of MSE at time k and the initial state $T(0)$ is set to be zero . Let α be the critical value of $T(k)$. If MSE^{stop} is the learning stop condition, the learning goal, then:

$$\alpha \geq MSE^{stop} \tag{3.1.6}$$

The learning rate will adapt itself according to following MIGD method:

1) If $T(k) \leq \alpha$

The condition shows that the learning procedure tends to be convergent to the learning goal. To avoid the stability of the learning and restore the chaotic behaviour of the learning procedure, the learning rate is increased aggressively by λ times, for example $\lambda = 2$:

$$\eta = \lambda \cdot \eta \tag{3.1.7}$$

2) If $(T(k) > \alpha)$ *and* $(T(k) > T(k-1))$

The condition shows that the learning procedure tends to be oscillating. To maintain the learning rate close to the maximum allowable learning rate and prevent the learning rate from being broken, the learning rate is decrease gradually by θ times, for example $\theta = 0.9$:

$$\eta = \theta \cdot \eta \tag{3.1.8}$$

3) If $(T(k) > \alpha)$ *and* $(T(k) \leq T(k-1))$

The learning rate will keep the same value.

The above self-adaptive procedure can be performed after each one-epoch training step in both encryption and decryption procedure. The critical value α can guarantee the learning procedure will not settle down to stable points. At the same time, the learning rate self-adaptation helps to maintain the learning rate close to the maximum allowable learning rate so that the learning trajectory is closely related to different training data. It will make the learning trajectory behave more random and the analysis of the learning procedure more difficult without the knowledge of the initial state of the neural networks.

3.1.3. Data Decryption

The decryption procedure works in similar manner as the encryption procedure. When the block cipher is used to decrypt the received cipher text blocks C_i, the neural networks need to perform the identical key extension procedure so that its weights and bias value will be initialized to be the same state as the encryption procedure. According to the encryption procedure, the decryption procedure will perform three steps:

- Cipher text decryption;
- One epoch training of the neural networks;
- Learning rate Self-adaptation.

These two steps are described as following sections.

3.1.3.1. Cipher text decryption

After the block cipher receives cipher text $C_i = \{V_i, E_i\}$, the message block can be restored as following:

$$Y_i = f_2(V_i) \tag{3.1.9}$$

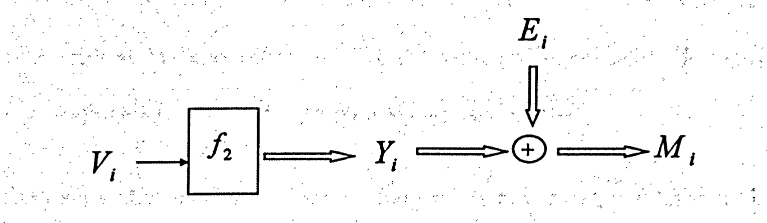
$$M_i = Y_i + E_i \tag{3.1.10}$$

3.1.3.2. One epoch training

After the plain text block M_i has been restored, the neural networks can be trained for one epoch with $X_i = \begin{bmatrix} Y_i \\ M_i \end{bmatrix}$ as input vector and M_i as the training target.

The cipher text decryption is illustrated as following Fig. 3.3.

Fig. 3.3 Cipher Text Decryption



3.1.3.3. Learning rate self-adaptation

A learning rate self-adaptation will follow the above one epoch training and it is the same as it is in the encryption procedure.

3.1.4. MAC Preparation

The final V_i of the final block can be the MAC for the whole cipher text. The MAC verification procedure and reason to use V_i as MAC is presented as following.

After the decryption procedure calculates Y_i from V_i, it can produce the M_i and construct $X_i = \begin{bmatrix} Y_i \\ M_i \end{bmatrix}$ again, then he can compute $V_i^{'}$ as following:

$$V_i^{'} = f_1(X_i)$$

Next it can compare $V_i^{'}$ with V_i to verify the data integrity and authentication.

Now suppose cipher text block $C_i = \{V_i, E_i\}$ has been modified to $*C_i$, then either E_i or V_i will be changed. Therefore the decryption will produce $*M_i$ from $*C_i$ according to (3.1.9) and (3.1.10), then the user will construct input vector $*X_i$ and calculate $*V_i$ according to (3.1.1). Because the value of $*V_i$ and V_i will not match each other, the data corruption can be detected.

The following analysis of the block cipher will revisit the block cipher design to find out how the block cipher can provide different cryptographic services.

3.2. Block Cipher Analysis

Although the new block cipher can be constructed by a common MLP network, it works as a real-time recurrent neural networks (RRNN). The encryption and decryption procedure of the block cipher is the real-time temporal supervised learning procedure of RRNN.

The new block cipher has at least following advantages by introducing RRNN:

- High performance;
- Ensure data integrity and authentication;
- High security.

The detailed analysis of the block cipher is divided into following corresponding sections.

3.2.1. Performance Improvement

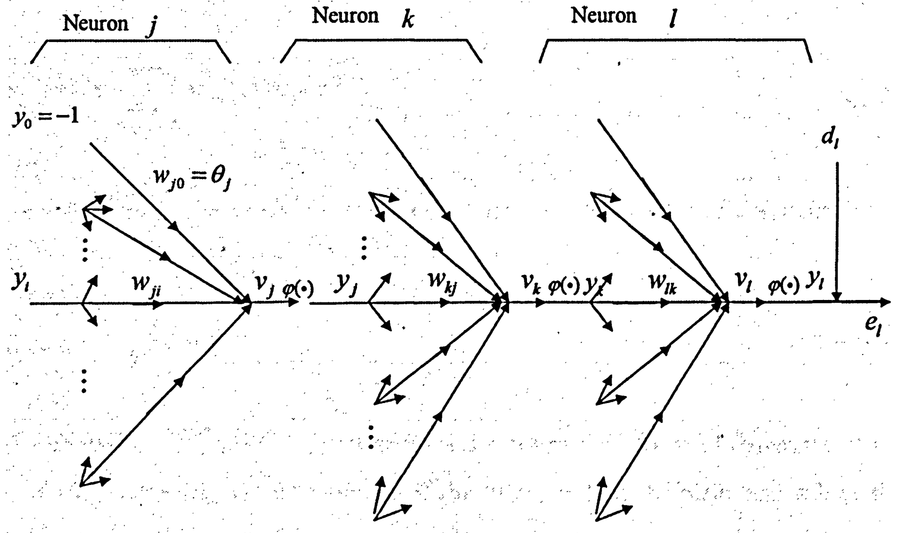
According to the technical review of neural networks structure, a neural network is a mass parallel computing machine. Different neurons can work independently and simultaneously. By increasing the dimension of the input vector, the block cipher can handle large message blocks so as to increase the speed of encryption. Furthermore the encryption and decryption are performed within one feed-forward calculation without multiple iterative round operations. Due to this type of parallel processing capability of neural networks, the block cipher can improve the performance for encrypting and decrypting tasks greatly. If the block cipher is implemented by software, it can make use of multiple processes or multiple threads programming for parallel computing. If it is implemented by hardware, the parallel computing advantage will be more obvious.

3.2.2. Data Integrity and Authentication Service

The patterns that the input vector and output vector represent will have two types of information, spatial and temporal. If the block cipher is expected to provide MAC for the cipher text, the data integrity relationship between the two message blocks needs to be detected. In other words, as these two message blocks will be generated by the block cipher sequentially in time, both spatial and temporal relationship between input and output data need to be detected. Comparing to standard neural networks, RRNN and its recurrent learning algorithm have this type of temporal processing capability. Its related learning algorithm provides network dynamic properties and makes the network responsive to time-varying signals.

A sample of the block cipher is illustrated as Fig. 3.4.

Fig. 3.4 Block Cipher Sample



RRNN will have two types of input data, external input $x(n)$ and one-step delayed output $y(n)$. They will concatenate to each other to form the input vector $u(n)$ for the RRNN. Let A denote the set of index i for external input $x_i(n)$ and B denote the set of index i for which $y_i(n)$ is the output of a neuron, the input vector for neuron i, $u_i(n)$, can be represented as following:

$$u_i(n) = \begin{cases} x_i(n) & \text{if } i \in A \\ y_{i-1}(n) & \text{if } i \in B \end{cases} \tag{3.2.1}$$

If neuron i is in the input layer of RRNN, $y_{i-1}(n)$ denotes the output vector of output layer of RRNN. If neuron i is in one of the hidden layers of RRNN, A set will be an empty set and $y_{i-1}(n)$ will denote the output vector of the neuron that is on the left-hand side of i th layer.

The internal neuron activity $v_j(n)$ is given by:

$$v_j(n) = \sum_{i \in A \cup B} w_{ji}(n) u_i(n) \tag{3.2.2}$$

Let $\varphi(\bullet)$ denote the nonlinear activation function, the output of the neuron will be

$$y_j(n+1) = \varphi(v_j(n)) \tag{3.2.3}$$

The above (3.2.2) and (3.2.3) constitutes the entire feed-forward dynamics of the network. According to the structure of the block cipher, the plain text will be the external input $x(n)$ and the error signal will be the second part of the cipher text:

$$e(n) = x(n) - y(n) \tag{3.2.4}$$

The (3.2.4) can rewrite as following:

$$e_j(n) = x_j(n) - y_j(n) \tag{3.2.5}$$

At time n, the instantaneous sum of squared error is:

$$\varepsilon(n) = \frac{1}{2}\sum_j e_j^2(n) \tag{3.2.6}$$

The objective of the learning is to minimize the following cost function, the sum $\varepsilon(n)$ over time n:

$$\varepsilon_{total} = \sum_n \varepsilon(n) \tag{3.2.7}$$

To apply steepest descent method for the minimization task, the gradient matrix of ε_{total} with respect to the weight matrix W need to be computed:

$$\nabla_W \varepsilon_{total} = \frac{\partial \varepsilon_{total}}{\partial W} \tag{3.2.8}$$

In order to learn in real-time, we have to use an instantaneous estimate of the gradient, the gradient of $\varepsilon(n)$ with respect to the weight matrix W, namely, $\nabla_W \varepsilon(n)$, to approximate the above gradient matrix as following:

$$\nabla_W \varepsilon_{total} = \sum_n \frac{\partial \varepsilon(n)}{\partial W} = \sum_n \nabla_W \varepsilon(n) \tag{3.2.9}$$

Now consider the incremental change $\Delta w_{ji}(n)$ made at time n for a particular weight $w_{ji}(n)$. Let η denote the learning rate, from (3.2.5) and (3.2.6), we have:

$$\Delta w_{ji}(n) = -\eta \frac{\partial \varepsilon(n)}{\partial w_{ji}(n)} = \eta \sum_{l} e_l(n) \frac{\partial y_l(n)}{\partial w_{ji}(n)} \tag{3.2.10}$$

To obtain $\frac{\partial y_l(n)}{\partial w_{ji}(n)}$, we use the chain rule:

$$\frac{\partial y_l(n+1)}{\partial w_{ji}(n)} = \frac{\partial y_l(n+1)}{\partial v_l(n)} \frac{\partial v_l(n)}{\partial w_{ji}(n)} = \varphi'(v_l(n)) \frac{\partial v_l(n)}{\partial w_{ji}(n)} \tag{3.2.11}$$

According to (3.2.2), we have:

$$\begin{aligned} \frac{\partial v_l(n)}{\partial w_{ji}(n)} &= \sum_k \frac{\partial (w_{lk}(n) u_l(n))}{\partial w_{ji}(n)} \\ &= \sum_k \left[w_{lk}(n) \frac{\partial u_l(n)}{\partial w_{ji}(n)} + \frac{\partial w_{lk}(n)}{\partial w_{ji}(n)} u_l(n) \right] \end{aligned} \tag{3.2.12}$$

As we know $\frac{\partial w_{lk}(n)}{\partial w_{ji}(n)}$ equals 1 only when $l = j$ and $k = i$, otherwise, it is zero.

We also note that

$$\frac{\partial u_l(n)}{\partial w_{ji}(n)} = \begin{cases} 0 & \mathit{if}\ i \in A \\ \frac{\partial y_k(n)}{\partial w_{ji}(n)} = \frac{\partial y_k(n)}{\partial w_{ji}(n)} & \mathit{if}\ i \in B \end{cases} \tag{3.2.13}$$

We may combine (3.2.11), (3.2.12) and (3.2.13) to have:

$$\frac{\partial y_l(n+1)}{\partial w_{ji}(n)} = \varphi'(v_l(n)) \left[\sum_k w_{lk}(n) \frac{\partial y_k(n)}{\partial w_{ji}(n)} + \kappa_{jl} u_l(n) \right] \tag{3.2.14}$$

Here κ_{jl} is a Kronecker delta equal to 1 when $j = l$ and zero otherwise.

We may introduce a triply indexed set of variables π_{ji}^{k} as following:

$$\pi_{ji}^{k} = \frac{\partial y_k(n)}{\partial w_{ji}(n)} \tag{3.2.15}$$

The equation (3.2.14) can be simplified as:

$$\pi_{ji}^{l}(n+1) = \varphi'(v_l(n))\left[\sum_k w_{lk}(n)\pi_{ji}^{k}(n) + \kappa_{jl}u_l(n)\right] \tag{3.2.16}$$

We can assume the initial state of the network at time $n = 0$ is zero because there is no function dependence on the synaptic weights

$$\pi_{ji}^{k}(0) = 0 \tag{3.2.17}$$

So all the $\pi_{ji}^{l}(n)$ can be calculated recursively.

Therefore, we can calculate (3.2.10) as:

$$\Delta w_{ji}(n) = \eta\sum_l e_l(n)\pi_{ji}^{l}(n) \tag{3.2.18}$$

Now we can repeat the computation to update weight w_{ji} in accordance with

$$w_{ji}(n+1) = w_{ji}(n) + \Delta w_{ji}(n) \tag{3.2.19}$$

According to the above analysis, the block cipher makes use of the forward dynamics to generate cipher text and MAC. At the same time, the block cipher keeps on updating the weight matrix by the backward dynamics described by (3.2.18) and (3.2.19). According to s (3.2.16), (3.2.17), (3.2.18) and (3.2.19), it shows that the backward dynamics is varying with time so that the learning procedure of RRNN has the capability to detect the temporal pattern of the training data. As a result, the cipher text generated by the learning procedure will represent both the spatial and temporal pattern of the training data. Every modification of the cipher text during transmission will affect the following cipher text block and will reflect different temporal pattern comparing to the original training data. So this modification can be detected during the decryption process. Consequently, the block cipher can prepare MAC to maintain both the data integrity and authentication if the weight matrix is considered as the secret extended key.

3.2.3. Security Guarantee

According to the design of the new block cipher, it also can resist different cryptanalysis attack and provide high security.

At first, the attack based on cipher text analysis is difficult. The cipher text contains the intermediary output of hidden layer v and the error signal e. v is the compress image of the plain text and e is the difference between network output and the plain text. Without the knowledge of the initial weight matrix, he cannot update the weight matrix recursively along with the learning procedure. Since he cannot analyze the backward dynamic of the block cipher, he will loose hint of the feed-forward dynamic characters of the RRNN. If he has no knowledge of the weight matrix and the output of neural networks, it is difficult to obtain the plain text based only on cipher text and the encrypted data get protected.

It is also infeasible for cryptanalyst to analyse the extended key, the weight matrix of the neural networks itself, because it is commonly assumed that the weight distribution of the hidden layers are chaotic and unpredictable without the knowledge of the training data, the original secret key. By changing the length of secret key and the dimension or the hierarchy of the hidden layers, user can flexibly adjust the security level. This is another advantage of the proposed block cipher that it can release the constraint on the secret key length.

Obviously, the encryption and decryption of the block cipher is relied on the feed-forward dynamics of RRNN. The encryption procedure can be a nonlinear mapping function and the cipher text is the nonlinear transform result of the plain text. If this nonlinear transform function is static, the nonlinear equations are possible to be solved if the cryptanalyst has large volumes of plain text and its corresponding cipher text available. So the nonlinear transform function should be dynamic when it is applied for data encryption. Consequently, the feed-forward dynamics of RRNN must keep vary with time to provide security protection of the plain text. As the real-time recurrent learning procedure can update the weight matrix while the feed-forward procedure is proceeding and this change of weight is a type of feedback to the network, the learning procedure can change the feed-forward dynamics of RRNN with the variety of time and help to generate cipher text.

But there is a potential known-text attack. Consider the following attack.

Let G denote the set of plain text, let Z denote the set of local and global minimum points and let L denote the largest invariant set in Z. L will contains all of the possible points at which the solution might converge and the trajectory will be trapped. Assume L contains only one fixed-point y, or the fixed points are closed enough to be consider as one point, cryptanalyst will train the block cipher with the known plain text repeatedly until the block cipher convergent to L. One possible method for cryptanalyst to achieve this is to insert a large amount of known plain text before the secret plain text as input. After the block cipher is stable, all the secret plain text input that belongs to G will convergent to this fixed point. Although cryptanalyst has no knowledge of the weight matrix and the

PROPERTY OF
RYERSON UNIVERSITY LIBRARY

initial state of the block cipher, he can get the convergent point y in L by the known plain text. Then the cryptanalyst can restore other following secret plain text M by the error signal e: M = y + e.

It shows that the stability of the neural networks will eventually help cryptanalyst break the block cipher without the knowledge of the weight matrix. To resist this attack, the learning procedure needs to guarantee that it will not become convergent to the invariant set L after the training of large volume of plain text. This consideration is directly related to the stability problem of neural networks.

A lot of research has been carried out on the stability problem on nonlinear dynamical systems. In fact, the RRNN can be modeled as nonlinear dynamical system. Let us continue the above analysis of the RRNN as following.

We can write the (3.2.2) in another form:

$$0 = -v_j(n) + \sum_i w_{ji}(n)\varphi(v_i(n)) + I_j \qquad (3.2.20)$$

I_j is the bias which weight value is always 1. We observe that (3.2.20) can be the fixed points of an associated dynamical system and it can transfer to the widely used additive model:

$$\frac{\partial v_j(n)}{\partial n} = -v_j(n) + \sum_i w_{ji}\varphi(v_i(n)) + I_j \qquad (3.2.21)$$

As the neural networks can be modeled as additive dynamical model, the stability theory, known as the direct method of Liapunov, is directly applicable to the stability analysis of neural networks. The key to apply the direct method of Liapunov is to find out the Liapunov function. In general, it is not possible to construct a Lyapunov function for the recurrent back-propagation algorithm.

However, the direct method of Liapunov will not help the security analysis of the block cipher here because the inability to find a suitable Liapunov function does not prove instability of the system. The existence of a Liapunov function is sufficient but not necessary condition for stability.

Alternatively, a "local" stability analysis of the network is performed here.

Let the state vector $Y(\infty)$ represent a fixed point of the recurrent network, the j th element of $Y(\infty)$ is defined by

$$y_j(\infty) = \varphi(v_j(\infty)) \tag{3.2.22}$$

$v_j(\infty)$ is the activation potential of neuron j at time $n = \infty$, it can be

$$v_j(\infty) = \sum_i w_{ji} y_i(\infty) + I_j \tag{3.2.23}$$

If we perform a local stability analysis on the forward, we can express the state variable $y_j(n)$ as:

$$y_j(n) = y_j(\infty) + \Delta y_j(n) \tag{3.2.24}$$

$\Delta y_j(n)$ is a small deviation of the state variable $y_j(n)$ from the coordinate $y_j(\infty)$. Correspondingly, we have:

$$v_j(n) = v_j(\infty) + \Delta v_j(n) \tag{3.2.25}$$

Next, we denote $F(v_j(n))$ to stand for (3.2.21) and we have:

$$\frac{\partial v_j(n)}{\partial n} = F(v_j(n)) \tag{3.2.26}$$

and

$$F(v_j(n)) = -v_j(n) + \sum_i w_{ji}\varphi(v_i(n)) + I_j \tag{3.2.27}$$

Consider to use the Taylor series expansion of $F(v_j(n))$ to approximate it by retaining the first two terms in the Taylor series:

$$F(v_j(n)) \approx F(v_j(\infty)) + A\Delta v_j(n) = A\Delta v_j(n) \tag{3.2.28}$$

where the matrix A is defined by:

$$\begin{aligned} A &= \frac{\partial}{\partial(v_j(n))} F(v_j(n)) \bigg|_{v_j(n)=v_j(\infty)} \\ &= -\sum_i \kappa_{ji} + \sum_i w_{ji}\varphi'(v_j(\infty)) \\ &= -\sum_i (\kappa_{ji} - w_{ji}\varphi'(v_j(\infty))) \\ &= -\sum_i L_{ji} \end{aligned} \tag{3.2.29}$$

Here κ_{ji} is a Kronecker delta equal to 1 when $j = i$ and zero otherwise. L_{ji} is defined as

$$L_{ji} = \kappa_{ji} - w_{ji}\varphi'(v_j(\infty)) \tag{3.2.30}$$

According to (3.2.25), (3.2.26), (3.2.30) and the definition of $v_j(\infty)$, we have the following linear differential equations:

$$\frac{\partial}{\partial n}\Delta v_j(n) \simeq -\sum_i L_{ji}\Delta v_j(n) \tag{3.2.31}$$

Using matrix notation, we may rewrite (3.2.31) as:

$$\frac{\partial}{\partial n}\Delta v(n) = -L\Delta v(n) \tag{3.2.32}$$

Next we consider backward dynamic. According to (3.2.14), (3.2.15) and (3.2.18), we can rewrite (3.2.18) as following more general form:

$$\Delta w_{ji}(n) = \eta\sum_l e_l(n)\pi^l_{ji}(n) = \eta\sum_l e_l(n)\frac{\partial y_l(n)}{\partial w_{ji}(n)} \tag{3.2.33}$$

We also have:

$$\begin{aligned}\frac{\partial y_l(n)}{\partial w_{ji}} &= \frac{\partial y_l(n)}{\partial v_l(n)}\frac{\partial v_l(n)}{\partial w_{ji}} \\ &= \varphi'(v_l(n))\left[\sum_k\left(\frac{\partial w_{lk}}{\partial w_{ji}}u_l(n) + w_{lk}\frac{\partial u_l(n)}{\partial w_{ji}}\right)\right] \\ &= \varphi'(v_l(n))\left[\sum_k\left(\kappa_{lj}\kappa_{ki}y_k(n) + w_{lk}\frac{\partial y_k(n)}{\partial w_{ji}}\right)\right] \\ &= \varphi'(v_l(n))\left[\kappa_{lj}y_i(n) + \sum_k w_{lk}\frac{\partial y_k(n)}{\partial w_{ji}}\right]\end{aligned} \tag{3.2.34}$$

The left hand side of (3.2.34) will be:

$$\frac{\partial y_l(n)}{\partial w_{ji}} = \sum_k \kappa_{lk}\frac{\partial y_k(n)}{\partial w_{ji}} \tag{3.2.35}$$

Combine (3.2.34) and (3.2.35), we have:

$$\sum_k (\kappa_{lk} - w_{lk}\varphi'(v_l(n)))\frac{\partial y_k(n)}{\partial w_{ji}} = \kappa_{lj}\varphi'(v_l(n))y_i(n) \qquad (3.2.36)$$

Compare to (3.2.29), we can define the elements of the state vector $Y(\infty)$ for the fixed point as:

$$\sum_k L_{lk}\frac{\partial y_k(\infty)}{\partial w_{ji}} = \kappa_{lj}\varphi'(v_l(\infty))y_i(\infty) \qquad (3.2.37)$$

If denote $(L^{-1})_{kj}$ as the kj th element of the inverse matrix L^{-1}, we have:

$$\frac{\partial y_k(\infty)}{\partial w_{ji}} = (L^{-1})_{kj}\varphi'(v_j(\infty))y_i(\infty) \qquad (3.2.38)$$

so that (3.2.33) will be:

$$\Delta w_{ji}(\infty) = \eta\sum_l e_l(\infty)(L^{-1})_{lj}\varphi'(v_j(\infty))y_i(\infty) \qquad (3.2.39)$$

We introduce $z_j(\infty)$ as following:

$$z_j(\infty) = \sum_l e_l(\infty)(L^{-1})_{lj} \qquad (3.2.40)$$

To avoid the matrix inversion, rewrite (3.2.40) as following:

$$\sum_k L_{kl}z_k(\infty) = e_l \qquad (3.2.41)$$

According to (3.2.30), rewrite (3.2.41) as following:

$$\begin{aligned} 0 &= -\sum_k L_{kl} z_k(\infty) + e_l \\ &= -\sum_k \{\kappa_{kl} - w_{kl}\varphi'(v_k(\infty))\} z_k(\infty) + e_l \\ &= -z_l(\infty) + \sum_k w_{kl}\varphi'(v_k(\infty)) z_k(\infty) + e_l \end{aligned} \tag{3.2.42}$$

We observe that the solutions of (3.2.42) are indeed the fixed points of an associated dynamical system defined by:

$$\frac{\partial z_l(n)}{\partial n} = -z_l(n) + \sum_k w_{kl}\varphi'(v_k(\infty)) z_k(n) + e_l \tag{3.2.43}$$

The backward propagation (3.2.43) can be expressed as following matrix form:

$$\frac{\partial z(n)}{\partial n} = -L^T z(n) + e \tag{3.2.44}$$

where L^T is the transpose of L.

From equation (3.2.32) we note that the local stability of the forward propagation equation depends on the eigenvalues of L. Because L^T has the same eigenvalues of L, we can draw the following conclusion:

The local stability of the forward propagation (3.2.21) is a sufficient condition for the local stability of the backward propagation (3.2.43), and vice versa[5].

Let us recall that the possible cryptanalysis attack before. If the block cipher will convergent to an invariant set L that contains fixed point, cryptanalyst will possible to break the security of the block cipher. According to the conclusion we just got above, if we can guarantee the backward propagation equation unstable, we can guarantee the forward propagation equation unstable so that we can resist the above cryptanalysis attack.

According to (3.2.39), the instability of the backward propagation equation is related to both the error signal and the weight matrix. During the recurrent back-propagation learning procedure, the learning keeps on moving the fixed points toward the desired values and the Euclidean distance between the fixed point and the desired pattern is diminished. As a result, the error signals will have smaller values. That will cause the dynamic of the backward propagation decreased. According to (3.2.39) and (3.2.40), the stability of the backward propagation will cause the stability of the weight matrix and will allow the recurrent neural networks settle down finally. But this type of convergence is not guaranteed for RRNN because of two reasons. The first reason is that from (3.2.8) to (3.2.9), we have used an instantaneous estimate of the gradient to approximate the above gradient curve in order to perform real-time learning so that the learning is not following the exact curve of the gradient. The other reason is the learning rate η, an artificial parameter, is introduced in (3.2.10) to calculate the update of the weight matrix. Because the learning procedure is not following the true gradient curve exactly, it will miss the true stable point and large error signals can be injected into the back-propagation process. If the learning rate is large at the same time, a near miss may have a dramatic effect on the weight update process and possibly cause the forward propagation unstable.

Another reason to have to choose large learning rate is to hide the temporal pattern of the input plain text. According to (3.2.10), the learning rate η defines the time scale over which the weights of the network change. If the learning rate η is large, the weight is changing so rapidly that the procedure to minimize the cost function (3.2.6) is no longer move along the learning trajectory in the weight space that descends against the gradient vector of the error signal. Because the weight changes produced by the learning can be viewed as another source of feedback, large learning rate will make the time-varying change of the weight dominate the phase state of the network so that the temporal structure of the plain text input can be hidden. This near chaotic oscillation is by purpose generated here and tends to provide security protection for the data.

3.3. Summary

In this chapter, the design of the new block cipher is described in detail. The analysis of the block cipher illustrates that the new design can provide high performance, high security data encryption and decryption. At the same time the block cipher can also ensure the data integrity and authentication with the help of recurrent neural networks. The learning rate self-adaptation function of the block cipher can help the block cipher work robustly and resist possible cryptanalysis attack.

Chapter 4

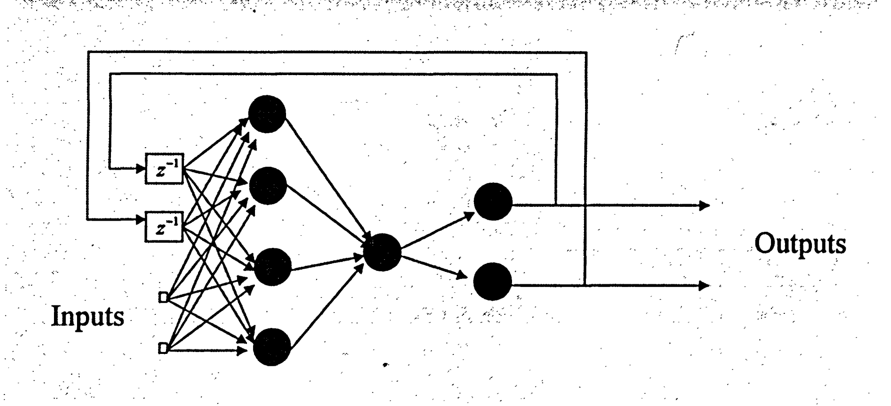
SIMULATION

4. Simulation

4.1. Simulation Set up

The block cipher sample used for simulation experiment is illustrated as Fig. 4.1.

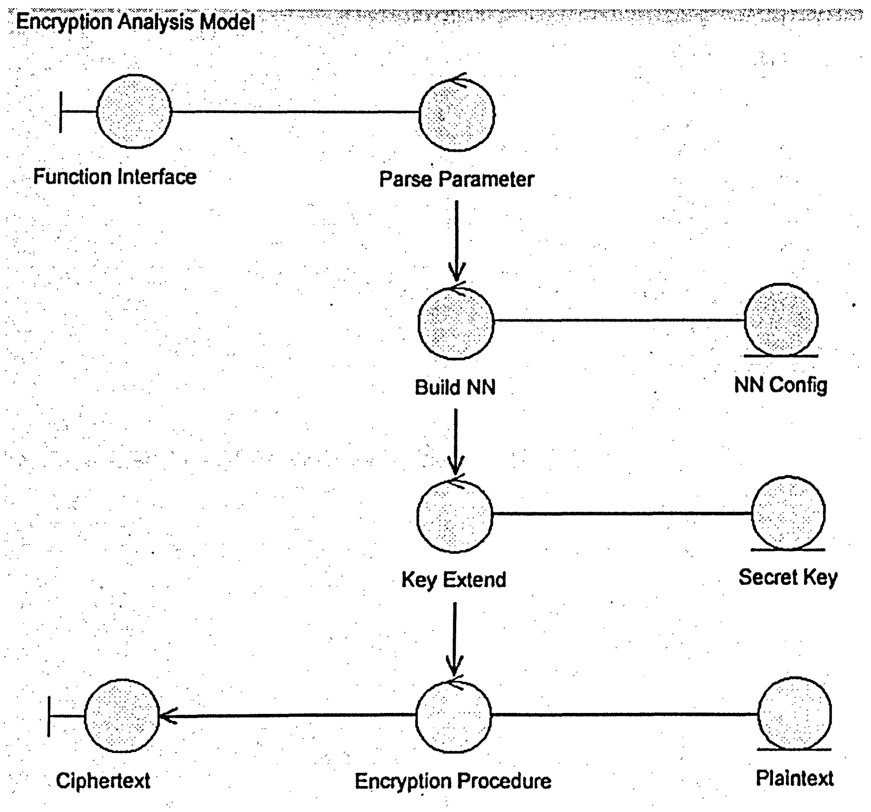
Fig. 4.1 Block Cipher Sample for Simulation Experiments



The block cipher is constructed by a common MLP network. In order to perform the recurrent real-time learning, the output vector of the block cipher is fed back as part of the input vector. The dimension of the input vector is 4 and the dimension of the output vector is 2. The hidden layer has only one neuron and its output will be the first part of the cipher text.

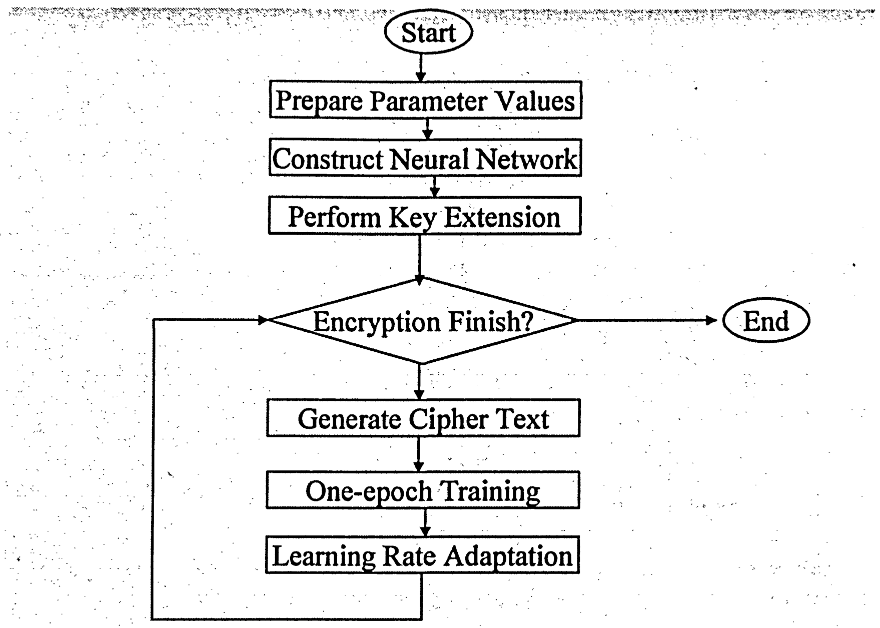
Corresponding simulation software program is coded in Matlab. It contains the encoder and decoder. The visual modeling of the encoder is illustrated as Fig. 4.2.

Fig. 4.2 Encoder model



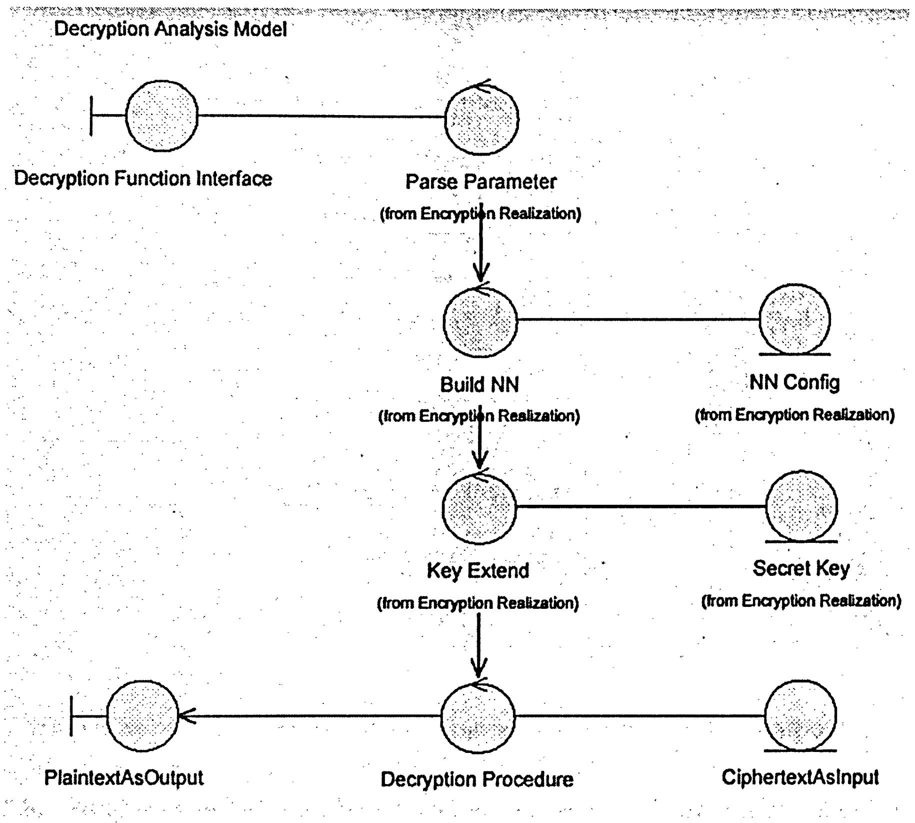
The flowchart of the encoder is illustrated as Fig. 4.3.

Fig. 4.3 Encoder Flowchart



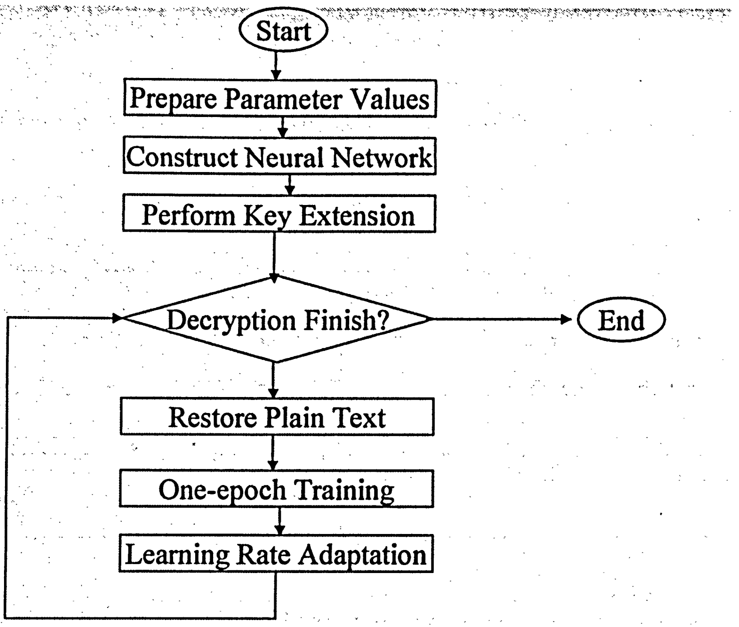
The visual modeling of the decoder is illustrated as Fig. 4.4.

Fig. 4.4 Decoder model


Decryption Analysis Model
Decryption Function Interface
Parse Parameter
(from Encryption Realization)
Build NN
(from Encryption Realization)
NN Config
(from Encryption Realization)
Key Extend
(from Encryption Realization)
Secret Key
(from Encryption Realization)
PlaintextAsOutput
Decryption Procedure
CiphertextAsInput

The flowchart of the decoder is illustrated as Fig. 4.5

Fig. 4.5 Decoder Flowchart



There are also other programs for data scaling, analysis and plotting purpose.

The configuration of the block cipher is controlled by script file. The configuration script file controls the parameters such as the dimension of the input and output vector, the dimension of the hidden layer, the learning stop condition and learning rate etc. The activation function is sigmoid function. The plaint text and secret key input of the block cipher is common text file. The weight and bias are all set to an identical initial value before training. At the same time the input is represented in their ASCII form and scaled to between 0 and 1 so that the neural networks will be sensitive to the input pattern.

4.2. Simulation Experiments

Simulation experiments are carried out to illustrate the effect of learning rate to the learning procedure and how learning rate adaptation help to control the proposed block cipher to generate chaotic cipher text.

The configuration parameters for the first experiment are listed as Table 4.1

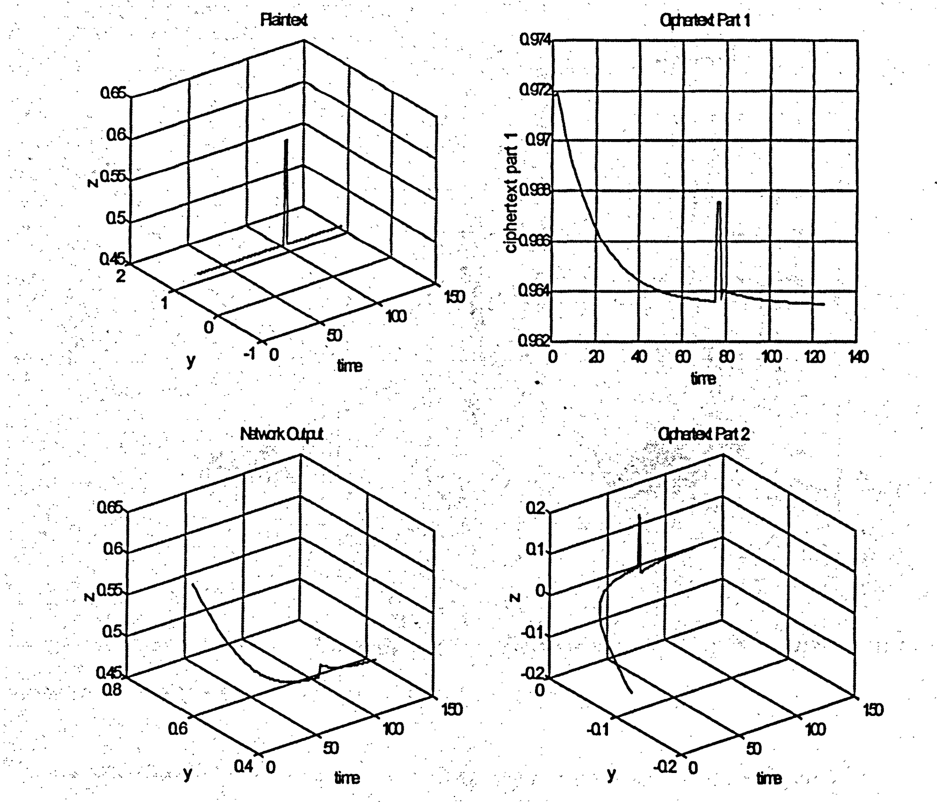
Table 4.1 Configuration Parameters for the first experiment

Weight Initial Value	Dimension of Input Vector	Dimension of Output Vector	Learning rate	Learning stop condition	Learning rate adaptation
0.5	4	2	0.05	1e-50	Disabled

The first experiment is to try to perform the cryptanalysis attack mentioned before. The plain text contains a long string of single character "a", then followed by a different character "z" and next followed by a long string of "a" again. For example, the plain text will look like this: "aaaaaaaaaaaa aaaaaaaaaaaaaaaaaaaaaaaazzaaaaaaaaaaaaaaaaaa". The learning rate is set to be a small value, 0.5, and the learning rate self-adaptation function is disabled for the first experiment. The result of the experiment is illustrated as Fig. 4.6.

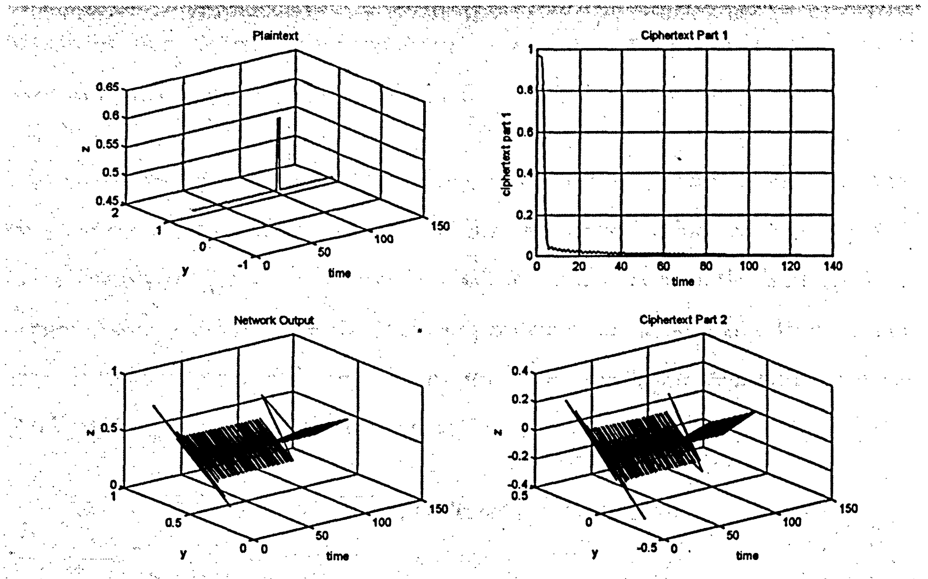
The experiment shows that after the network is convergent to a single point, new input pattern cannot change the network output far enough from the stable point because the learning rate is small. Consequently, the second part of the cipher text will expose the temporal pattern of the plain text. The cryptanalyst can perform the attack mentioned in this paper to guess the new character "z".

Fig. 4.6 Small Fixed Learning Rate Effect



The same experiment environment as above is used for the second experiment except that the learning rate is changed to a larger value. The result of the experiment is illustrated as Fig. 4.7. The learning rate is set to be 35 in this experiment. The result of the experiment illustrates that if the learning rate is changed to a larger value, the learning procedure can be prevented from convergence. The temporal structure of the plain text input can be protected because the cipher text is chaotic.

Fig. 4.7 Large Fixed Learning Rate Effect



In the third experiment, the learning rate self-adaptation function is turned on to investigate its effect to the block cipher.

The configuration parameters for the third experiment are listed as Table 4.2

Table 4.2 Configuration Parameters for the third experiment

Weight Initial Value	Dimension of Input Vector	Dimension of Output Vector	Learning rate initial value	Learning stop condition	Critical value α	Increase factor λ	Decrease factor θ
0.5	4	2	1	1e-50	0.01	2	0.9

The result of the experiment of the learning rate self-adaptive procedure is illustrated as Fig. 4.8. The left-hand side of Fig. 4.8 is the signal of $T(k)$ and the right-hand side is the learning rate signal of η. The corresponding signal of two parts of cipher text is illustrated as Fig. 4.9.

The experiment results shows that every time the learning rate is set to a large rate, the error signal will miss far away from the critical value and learning goal. Consequently the parameter α can be used as knob to control the learning to be unpredictable and guarantee the learning procedure unstable. Large learning rate help to hide the temporal structure of the plain text input data and force the block cipher generate chaotic cipher text.

Fig. 4.8 Learning Rate Self Adaptation

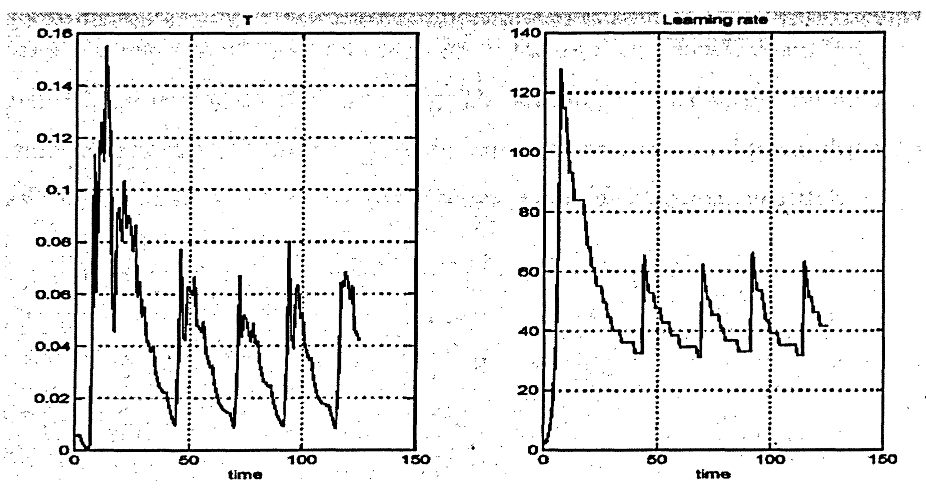
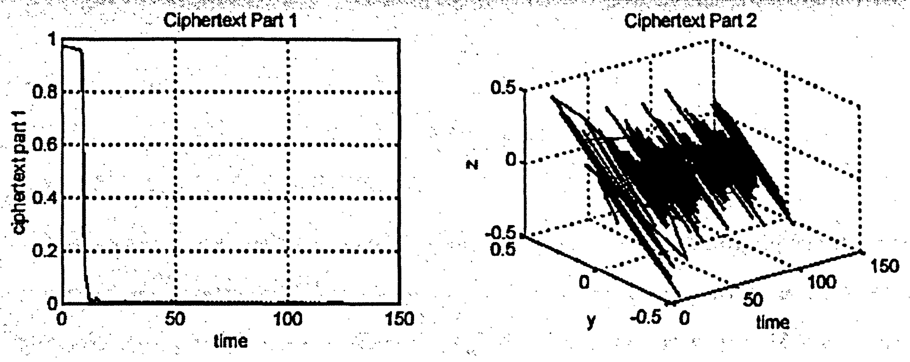


Fig. 4.9 Chaotic Cipher Text



4.3. Summary

The simulation experiments provide illustration to verify some conclusions of the security analysis of the block cipher. The results of the simulation shows that large learning rate will inject much noise to the learning procedure and force the network generating chaotic cipher text. The learning rate self-adaptation function helps to guarantee the instability of the learning procedure so that the dynamics of the block cipher can be maintained to resist possible cryptanalysis attack.

Chapter 5

CONCLUSION

In the project, the source of the problem is defined by considering different types of security requirements. The review of both cryptography and neural network illustrates the potential possibility to apply neural network for block cipher design. The project contributes to propose a new block cipher design based on recurrent neural networks. The analysis of the design shows that the hidden layers of neural network can be used as the secret extended key, different services can be provided through a integrity scheme and the learning procedure of the recurrent neural network can be controlled to provide the secure protection for the data by adapting the learning rate. Simulation experiments also help to verify some of the analysis conclusions by examples.

In summary, the new block cipher design has several advantages by introducing recurrent neural networks for symmetric-key block cipher design. First of all, the proposed block cipher releases the limitation of the length of secret key and the length of the message blocks. This breakthrough unbinds many limitations of the implementation for the block cipher. The block cipher can flexibly adjust the secret key and message length to accommodate different security and performance requirements. Secondly it is capable to provide both high secure data encryption and data integrity service. Different cryptographic services are provided by an integrity scheme with relatively simple architecture. The simplified design can make it possible to perform the analysis of the block cipher based on the error-correction learning theory of the recurrent neural networks. Furthermore, the inherent parallel computing capability of the block cipher can accommodate high performance data encryption requirements such as secure point-to-point file transfer between gateways. In fact there are many applications that will consider the proposed block cipher practical. Every time high efficient secure data communication is desired, the proposed block cipher will be greatly helpful.

Further research work about the knowledge representation of neural networks and stability analysis of recurrent neural network will provide more valuable information for the analysis of the block cipher. More pressuring cryptanalysis against the proposed block cipher are also desirable to uncover the risk of possible attacks. As cryptography is both a science and an elegant art, this type of attack-and-defence attempts is the motivity of development of cryptology all along.

REFERNCES

[1] Bruce Schneier. Applied cryptography, 2nd edition. Published by John Wiley & Sons Inc., 1996.

[2] Simon Haykin. Neural networks, a comprehensive foundation. Published by Macmillan College Publishing Company, 1994.

[3] Su, S.; Lin, A.; Jui-Cheng Yen. Design and realization of a new chaotic neural encryption/decryption network. In*The 2000 IEEE Asia-Pacific Conference on Circuits and Systems*, Page(s): 335 –338, 2000.

[4] Liew Pol Yee and Liyanage C. De Silva. Application of MultiLayer Perceptron Networks in Symmetric Block Ciphers. In *Proceedings of the 2002 International Joint Conference on Neural networks*, Volume 2, Page(s):1455–1458, 2002

[5] Almeida, L.B. A learning rule for asynchronous perceptrons with feedback in a combinatorial environment. In *1st IEEE International Conference on Neural networks*, Vol. 2, 1987.

[6] Williams, R.J., and D. Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation 1* , Page 270-280, 1989.

[7] Kenneth H. Rosen. Elementary Number Theory and its Applications, 3rd edition, published by Addison-Wesley, 1993.

[8] Menezes, A., van Oorschot, P., Vanstone, S.. Handbook of Applied Cryptography. Published by CRC Press, 1997.

[9] Jeffrey L. Elman. Finding Structure In Time. *Cognitive Science, 14, 179-211.* 1990.

[10] Kam-Chuen Jim, C. Lee Giles, Bill G. Home. An Analysis of Noise in Recurrent Neural networks: Convergence and Generalization. *IEEE Transactions On Neural networks*, Vol 7, No. 6, November, 1996.

[11] Hirsch, M.W. Convergence in neural nets. *1st IEEE International Conference on Neural networks*, Vol. 2. Page(s): 115-125. 1987.

[12] H.X. Mel, Doris Baker. Cryptography Decrypted. Published by Addison-Wesley, 2001.

[13] Martin T. Hagan, Howard B. Demuth, Mark Beale. Neural networks Design. Published by PWS Pub., 1996.

[14] Wan, E.A. Temporal backpropagation for FIR neural networks. *IEEE International Joint Conference on Neural networks*, Vol. 1, pp. 575-580. 1990.

[15] S. Townley, A.Iichmann, M. G. Weib, W. Mcclements, A. C.Ruiz, D. H. Owens, and D. Pratzel-Wolters. Existence and Learning of Oscillations in Recurrent Neural networks. *IEEE Transactions on Neural networks*, Vol. 11, No. 1, January 2000.

[16] Whitfield Diffie, Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, Vol. 22, Issue: 6, Page 644-654, Nov 1976.

[17] National Bureau of Standards. Data Encryption Standard. *FIPS Pub 46, US, Washington DC,* Jan, 1977.

[18] Joan Daemen and Vincent Rijmen. Rijndael. The advanced encryption standard. *Dr. Dobb's Journal,* Vol. 26, No. 3, Page 137-139, March 2001.

[19] John J. Hopfield. Rijndael. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences of the U.S.A.,* 79, Page 2554-2558, 1982.

[20] M. W. Hirsch. Convergent activation dynamics in continuous time networks. *Neural Networks 2,* Page 331-349, 1989.

[21] Jinde Cao and Jun Wang. Global asymptotic stability of a general class of recurrent neural networks with time-varying delays. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications,* Vol 50, Issue 1, Page 34-44, Jan 2003.

[22] Sanchez, E.N. and J.P. Perez. Stabilization of stochastic recurrent neural networks. *Proceedings of the 2002 IEEE International Symposium on Intellegent Control,* Page 445-447, 2002.