Encrypted Voice Over TCP/IP by Using Neural Encryption Algorithm (NEA).

by

Tubtim Sanguanwongthong

Faculty of Engineering
July 2003

Encrypted Voice Over TCP/IP By Using Neural Encryption Algorithm (NEA)

A thesis
submitted to the Faculty of Engineering

by

Tubtim Sanguanwongthong

in partial fulfillment of the requirements
for the degree of
Master of Engineering in Broadband Telecommunications

Advisor: Dr. Thiraphong Charoenkhunwiwat

Assumption University
Bangkok, Thailand
July 2003

"Encrypted Voice Over TCP/IP by Using Neural Encryption Algorithm (NEA)"

by

Ms.Tubtim Sanguanwongthong

A Thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Engineering
Majoring in Broadband Telecommunications

Examination Committee:

1. Dr. Thiraphong	Charoenkhunwiwat	(Advisor)	[illegible]
2. Dr.Gennady	Veselovsky	(Member)	[illegible]
3. Dr.Somkiat	Sampan	(Member)	[illegible]
4. Assoc.Prof.Dr.Watit	Benjaponlakul	(MUA Representative)	Watit

Examined on: May 20, 2003
Approved for Graduation on: July 1, 2003

Faculty of Engineering, Assumption University
Bangkok, Thailand

Encrypted Voice Over TCP/IP by Using Neural Encryption Algorithm (NEA)

By

Tubtim Sanguanwongthong

Abstract

This thesis proposes a new encryption, Neural Encryption Algorithm (NEA), designs and develops a software of voice/text chat over TCP/IP with NEA including Data Encryption Standard (DES), Triple DES, Secure And Fast Encryption Routine with a Key of length 64 bits (SAFER K-64), Rivest Shamir and Adleman (RSA) and no encryption for comparison in secure voice/data communication over TCP/IP under Windows operating system.

Since there are a lot of well-known cryptographies with both symmetric-key and public-key encryptions, the NEA, a symmetric-key encryption is proposed to provide the new alternative way for secure communication. This proposed encryption has been applied in the proposed software for voice and text chat.

The software development testing process is divided into 3 steps.

1. Study how to transmit/receive voice/text with no encryption in LAN.
2. Design server as the authentication center of user-and-password login.
3. Design client for voice and text chat with applying DES, Triple DES, SAFER K-64, RSA, NEA and none of encryption.

In the proposed software, PCM is being applied for digitizing voice with 8 kHz sampling rate and 512-byte buffer size. In testing security, the packet sniffer is applied to detect the packet segment and read data message in network.

Acknowledgements

This thesis has been finished by the very special kindness and helpfulness from Dr.Somkiat Sampan, who always gives me guidance and idea as well as his great support and follows up my thesis. I also wish to thank Dr. Thiraphong Charoenkhunwiwat, my advisor, who has assisted me to fulfill my thesis.

I also thank to my cousin, Mr. Kittiwut Choochote, for his very useful idea.

Special appreciation is due to my family for their fervent and continuous encouragement. Above all, I am forever grateful to my parents whose willingness to invest in my future has enabled me to achieve my educational goal.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

1.1 Background and Motivation

With rapid growth of modern technology, more and more people rely on computer networks to exchange knowledge, access information, and process data in a distributed environment e.g. telephony, conference and fax. To meet the demand of people who seek secret communications, various kinds of data transformation, or encryption/decryption techniques, have been developed so that the data through open media become meaningless to intruders, but the original data can be recovered when data reaches the destination. Motivation of this research is the study of encryption techniques for secure communication over computer network e.g. Data Encryption Standard (DES), triple DES, Secure And Fast Encryption Routine with a Key of length 64 bits (SAFER K-64) and Rivest Shamir and Adleman (RSA). Using the structure of feed forward multilayer neural network, the new encryption is then proposed and called Neural Encryption Algorithm or NEA. Unlike other encryptions, NEA has flexible block size of plaintext and flexible key length. With equal block size in other block-enciphering algorithms, the strength of NEA is the multiple of keys and multiple of hidden layers. Finally, NEA is initiative for further modified or designed algorithms.

1.2 Previous Works

In recent years, the importance of security in the information technology has increased significantly. To guarantee confidentiality and authenticity of information, cryptographic algorithms and methods are used. There have been a lot of research implementations on cryptography e.g. both hardware [1-3] and software [4-9].

In 1988, Davis designed a security radio system [10]. This system uses digital voice encryption throughout a system of receiver voting. The radios are encrypted with DES algorithm, which is a digital voice-encryption technique approved for transmission of sensitive information.

In 1990, Sridharan, Dawson and Goldburg presented a comparison of five discrete orthogonal transforms in speech encryption systems and proposed a new method for analog speech encryption [11-12]. Five of discrete orthogonal transforms are discrete Fourier transform (DFT), the discrete prolate spheroidal transform (DPST), the discrete cosine transform (DCT), Walsh Hadamard transform (WHT) and the Karhunen Loeve transform (KLT). This research worked only the first two transforms since there have been considered on encryption of speech. Then in 1991, they proposed Fast Fourier transform based speech encryption system [13]. It also presented the two fundamentally distinct approaches to achieve voice security in speech communication systems, e.g. digital encryption and analog encryption. In the same year, Duraiappan and Zheng proposed Triple-DES to improve speech security and authentication in mobile communications instead of the A5 encryption algorithm used in the existing security system [14].

In 1992, Schulzrinne introduced Network Voice Terminal (NEVOT) as a tool to support audio conferences across local and wide area networks, including the Internet [15]. The Feature of NEVOT (version 0.95) is DES-based voice encryption and audio encodings supported: 16 bit linear encoding, 64 kb/s G.711 μ-law PCM, 32 kb/s G.721 ADPCM, 32 kb/s Intel/DVI ADPCM, 24 kb/s G.723 ADPCM and 4.8 kb/s LPC.

In 1996, Troulllnos presented a flexible software based approach for implementing secure voice systems as well as implementation for speech coding (2.4 kbps, 4.8 kbps, 9.6 kbps LPC) and encryption algorithms (DES) [16].

In 2000, Wu and Kuo proposed the selective encryption (DES or SSL protocol) and MHT (multiple-Huffman-table) for ITU G.723.1 speech coding on the fast encryption methods for audiovisual data confidentiality [17]. In the same year, Borujeni presented permutation of FFT coefficients in speech encryption system [18].

In 2002, Khonkaen University had the project on software development for data encryption by voice [19]. RSA is only one encryption chosen to implement in this project.

Besides the presented researches, this thesis proposes the new alternative encryption for voice and data communication in computer network and develops the software for comparison with other encryptions.

1.3 Research Objective

The objective of this research is to propose the new encryption called Neural Encryption Algorithm (NEA) and develop software for voice/text chat over TCP/IP applying Data Encryption Standard (DES), triple DES, Secure And Fast Encryption Routine with a Key of length 64 bits (SAFER K-64), Rivest Shamir and Adleman (RSA) and no encryption for comparison with NEA under Windows operating system.

1.4 Scope of the Study

1.4.1 NEA has been applied only in voice/text chat over computer network.

1.4.2 DES, Triple DES, SAFER K-64, RSA and no encryption are implemented in the proposed software for comparison with NEA.

1.4.3 The proposed software is developed with Borland Delphi Enterprise, version 6.0 under Windows XP operating system.

1.4.4 The software is to test the encryption techniques in only voice/text chat over computer network.

1.4.5 For voice/text chat, there is only one couple of people at a time.

1.4.6 The packet sniffer is the software to detect the data flow in the network.

1.5 Methodology

1.5.1 Study how to transmit/receive voice in LAN.

1.5.2 Develop the voice/text chat software over TCP/IP.

1.5.3 Study DES algorithm and then apply in the software.

1.5.4 Study Triple DES algorithm and then apply in the software.

1.5.5 Study SAFER K-64 algorithm and then apply in the software.

1.5.6 Study RSA algorithm and then apply in the software.

1.5.7 Design the NEA algorithm and then apply in the software.

1.5.8 Test and compare all encryption techniques.

1.5.9 Conclude and compose the thesis.

1.6 Expected Benefits

1.6.1 The proposed encryption, NEA, can be applied in other applications or in other network e.g. wireless network.

1.6.2 Using encryption techniques, data flow in network is ciphertext or unknown message so the third person cannot read data using packet sniffer.

1.6.3 Digitizing voice technique can be applied in further development.

1.6.4 The proposed software can be applied in further application.

1.6.5 The proposed NEA is the new encryption technique for protecting the data.

1.6.6 The proposed software can be used in the real life with no hardware required in the computer network.

1.6.7 The software is the alternative way for communication over existing computer network with no cost like telephone call.

1.7 Thesis Overview

This thesis is to propose a new encryption, Neural Encryption Algorithm (NEA), and develop a testing software for voice/text chat applying NEA as well as DES, Triple DES, SAFER K-64, RSA and no encryption for comparison.

Chapter 1 provides background and motivation for this research, the related previous works in speech encryption, objective, scope, methodology of the thesis and expected benefits.

In chapter 2, all encryption algorithms, NEA, DES, Triple DES, SAFER K-64 and RSA as well as PCM technique for digitizing voice analog and background in computer network are described.

Chapter 3 provides the software implementation for testing the encryption techniques on voice/text chat.

In chapter 4, NEA, DES, Triple DES, SAFER K-64, RSA and no encryption are applied for testing the quality of voice, the continuity of voice and the security of data flow with the packet sniffer.

Chapter 5 concludes overall implementation of this thesis and also the recommendation for future works.

Chapter2

Related Theory

To have the secure communication, lots of cryptography techniques have been implemented. In this work, the NEA is proposed and compared with DES, Triple DES, SAFER K-64 and RSA in the proposed software over computer network. Then, the algorithm of DES, Triple DES, SAFER K-64, RSA and proposed NEA is described. The technique of PCM for digitizing voice analog and the background of computer network are also included.

2.1 Cryptography

The cryptography can be divided into two parts: symmetric algorithms (conventional algorithms) and public-key algorithms (asymmetric algorithms) [20-21]. Then, DES, triple DES and SAFER K-64 are in the symmetric algorithms part and RSA is in the public algorithms part.

2.1.1 Symmetric Algorithms

Symmetric algorithms sometimes are called conventional algorithms, secret-key algorithms, single-key algorithms, or one-key algorithms. For symmetric algorithms, there is only one key using for both encryption and decryption. Then, the key must be kept secret. The problem is how the sender sends this secret key to the receiver securely as well as how the key remains secret. By the way, the symmetric algorithms are not complicated and then fast for computation.

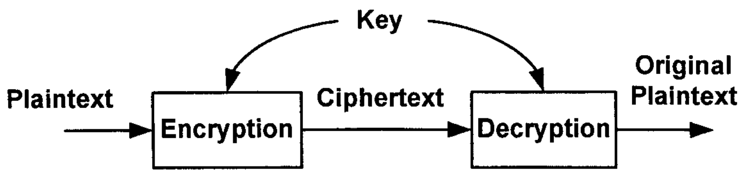


Figure 2.1 Symmetric algorithm

2.1.1.1 Data Encryption Standard (DES)

DES is the block algorithm with one secret key [20]. The length of plaintext and ciphertext are 64 bits equally with 56-bit key length after ignorance of every 8th bit (parity bit) with key permutation by table 2.2. The overall algorithm has 16 rounds for each encryption and decryption.

Algorithm outline is shown in figure 2.2. First, the input of 64-bit block of plaintext is into an initial permutation (table 2.1) and then it is broken into a right half and left half, each 32 bits long. Then there are 16-round identical f operation round, in which the data are combined with the key. After the sixteenth round, the right and left halves are joined and then the result goes to the inverse permutation (table 2.8). The ciphertext is finally generated.

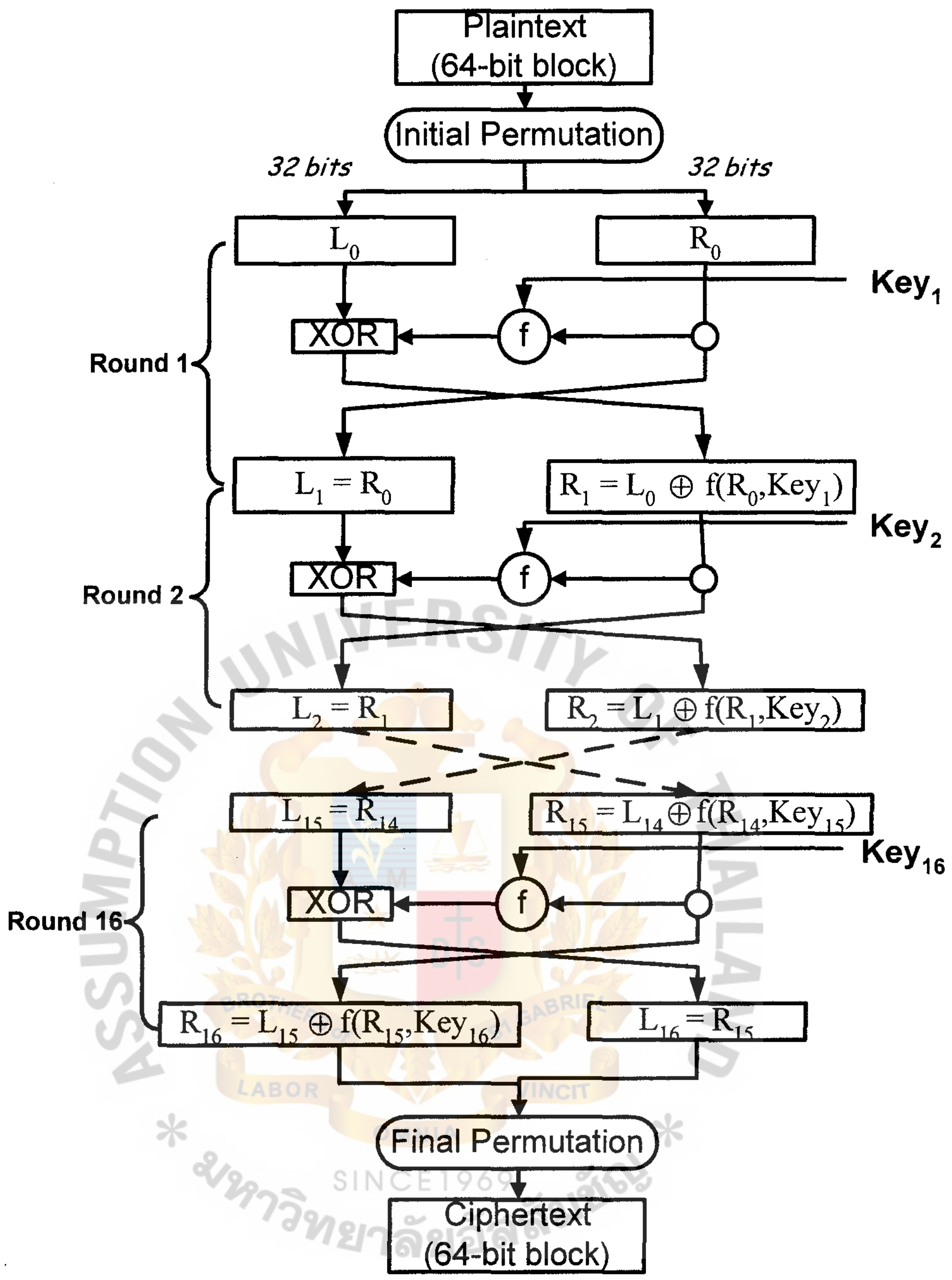


Figure 2.2 DES Algorithm

In each round operation (see figure 2.3), the 56-bit key is circularly left shifted (table 2.3) and then the result key is selected from the compression permutation (table 2.4) to 48 bits. The 32-bit right half of the data is expanded due to expansion permutation (table 2.5) to 48 bits, combined with 48 bits of a shifted and permuted key via an XOR operation, sent through S-Box (table 2.6) becoming 32 new bits, and permuted with P-box (table 2.7). These four operations make up function f and its output is then combined with the left half via another XOR. The result becomes the new right half while the old right half becomes the new left half. These operations are repeated 16 times, or called 16 rounds. The conclusion can be written as:

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

If B_i is the result of the ith operation, the left and right halves of B_i are L_i and R_i respectively, the 48-bit key for round i is K_i, the function f is f for all the substituting and permuting and XORing with the key.

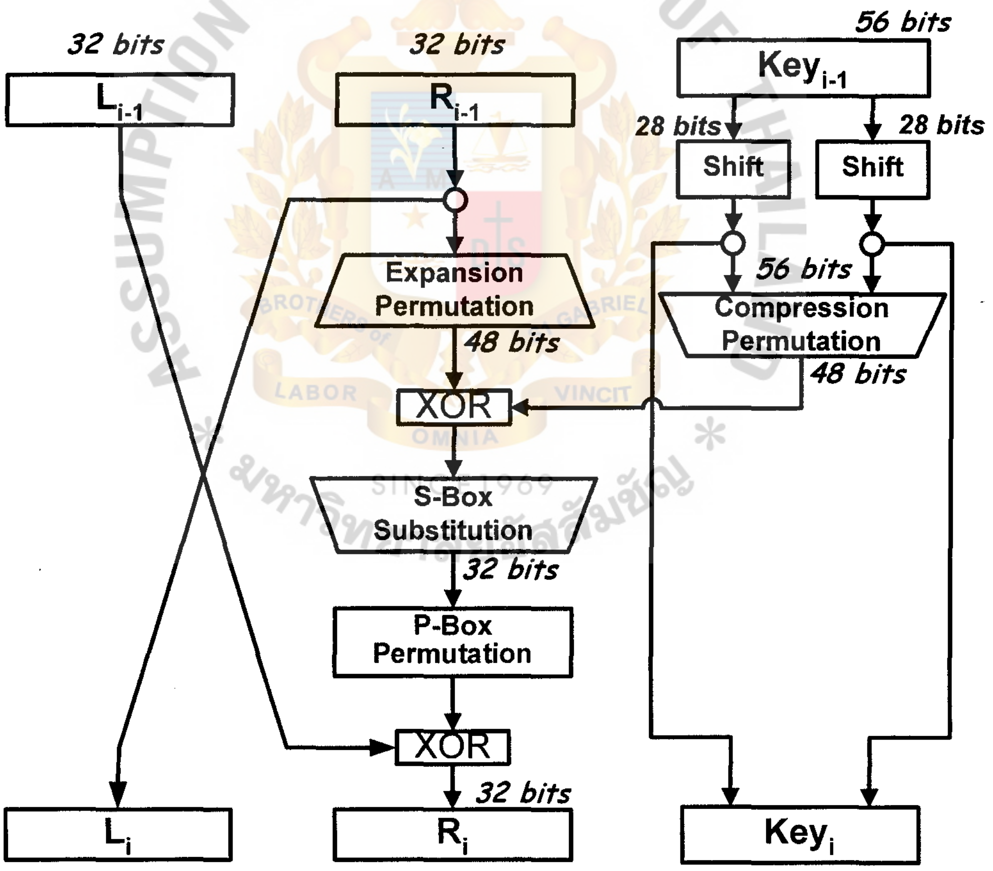


Figure 2.3 One round of DES algorithm

Table 2.1 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

Table 2.2 Key Permutation

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

Table 2.3 Number of Key Bits Shifted per Round.

Round	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Number	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Table 2.4 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

Table 2.5 Expansion Permutation

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

In S-box substitution, there are 48 bits after XORed (figure2.3) and sent through S-box that substituted with 8 different substitution boxes, or S-boxes for a 6-bit input and a 4-bit output. It means that the first and last bits combine to i^th row and the middle 4 bits combine to j^th column.

Table 2.6 S-Boxes

S-box 1:

14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	0	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	3	13

S-box 2:

15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

S-box 3:

10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

S-box 4:

7	13	10	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

S-box 5:

2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

S-box 6:

12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

S-box 7:

4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	6	5	0	15	14	2	3	12

S-box 8:

13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Table 2.7 P-Box Permutation

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

Table 2.8 Inverse of the Initial 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

To decrypt DES, there is the same algorithm as the encryption except it must be reverse order of keys. That is, if the encryption keys are in K_1, K_2, K_3, …, K_{16}

order, the decryption keys are in K_{16}, K_{15}, K_{14}, ..., K_1. It can be expressed by the equations [22]:

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

The key begins with $R_{16}L_{16}$ and the last key is L_0R_0.

Also the key shifted per round is circularly shifted right, then the number of positions shifted is 0, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1.

DES's attack is as following

1. *Brute-force attack* is to determine the key by trying 2^{56} possible keys until finding the correct key. In 1976 and 1977, Diffie and Hellman argued that a special-purpose DES-cracking parallel computer could recover the key in a day. In 1990, DES would be totally insecure.
2. *Differential Cryptanalysis* is introduced by Eli Biham and Adi Shamir in 1990. This attack is heavily dependent on the structure of the S-boxes. In the other hand, it can improve DES's resistance by increasing the number of rounds.
3. *Related-Key Cryptanalysis* is similar to differential cryptanalysis, but it examines the difference between keys. However, this attack is not at all pratical. The interesting points of this attack is:
   3.1 The attack is against DES's subkey-generation algorithm.
   3.2 It is independent of the number of algorithm's rounds.
   3.3 DES is impervious to it.
4. *Linear Cryptanalysis* is heavily dependent upon the structure of the S-boxes and the S-boxes in DES are also not optimized against this attack.

2.1.1.2 Triple DES

Triple DES is 3 times repeated of DES encryption and decryption operations [20,22]. The outline operation can be described by letting $E_K(I)$ and $D_K(I)$ as the DES encryption and decryption of I with DES key K respectively.

1. Triple DES encryption: beginning with a 64-bit block I is transformed into a 64-bit block O.

   $O = E_{K3}(D_{K2}(E_{K1}(I)))$

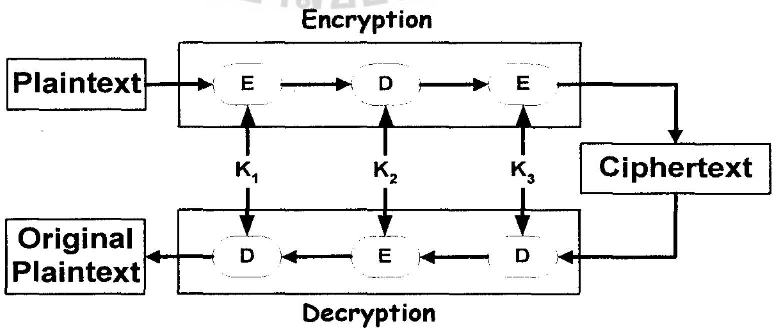
2. Triple DES decryption: beginning with a 64-bit block I is transformed into a 64-bit block O.

   $O = D_{K1}(E_{K2}(D_{K3}(I)))$

There are 3 options of the triple DES key (K_1, K_2, K_3)

1. Option 1: K_1, K_2 and K_3 are independent keys.
2. Option 2: K_1 and K_2 are independent keys with $K_3 = K_1$.
3. Option 3: $K_1 = K_2 = K_3$

In this thesis, option 2 is chosen and applied in the presented software for K_1 has the fixed value and K_2 is from user, that is, triple DES with 2 keys. Then, it has 2^{112} attempts for the brute-force attack.



with E : DES encryption and D : DES decryption.

Figure 2.4 Triple DES

2.1.1.3 Secure And Fast Encryption Routine with a Key of length 64 bits (SAFER K-64)

SAFER K-64 is known as a non-proprietary secret-key block-enciphering algorithm with 64-bit block length for both plaintext and ciphertext [20,23]. In 64 bits (8 bytes), only one byte can be used in the processes of encryption and decryption. The overall rounds (r) for encryption and decryption are 6 rounds (recommended) or larger values for greater security.

In each round, it is required two 8-byte subkeys and another one 8-byte subkey is used for the output transformation.

The SAFER K-64 encryption algorithm and the detail for one round is shown in figure 2.5 and figure 2.6 respectively.

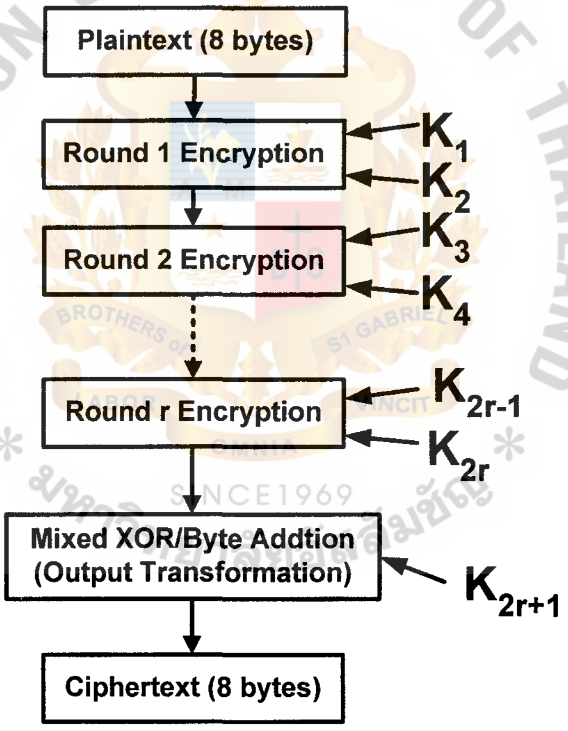


Figure2.5 SAFER K-64 encryption

In figure 2.6, the first step with the ith round is the Mixed XOR/Byte-Addition of the round input with the subkey K_{2i-1}. Then, the eight bytes pass through a nonlinear layer, that is, $45^{(\cdot)}$ and $\log_{45}$. For the $45^{(\cdot)}$ operation, if the byte input is integer j then the byte output is 45^j modulo 257 (except that this output is taken to be 0 if the modular result is 256, which occurs for j = 128). For the $\log_{45}$ operation, if the byte is the integer j then the byte output is $\log_{45}(j)$ (except that this output is taken to be 128 if the input is j=0). After that, the output of the eight nonlinear transformations is combined with subkey K_{2i} and then there are bytewise addition of bytes 1, 4, 5, 8 and bytewise XOR (modulo-2 sum) of bytes 2, 3, 6, 7. The output is then passed though the three-level "linear layer" of 2-PHT. That is, if the input bytes to a 2-PHT are (a_1, a_2), where a_1 is the more significant byte, then the two output bytes are (b_1, b_2) where

$$b_1 = 2a_1 + a_2$$

$$b_2 = a_1 + a_2$$

where there is the normal byte arithmetic, i.e., arithmetic modulo 256. The linear layer output constitutes the round output.

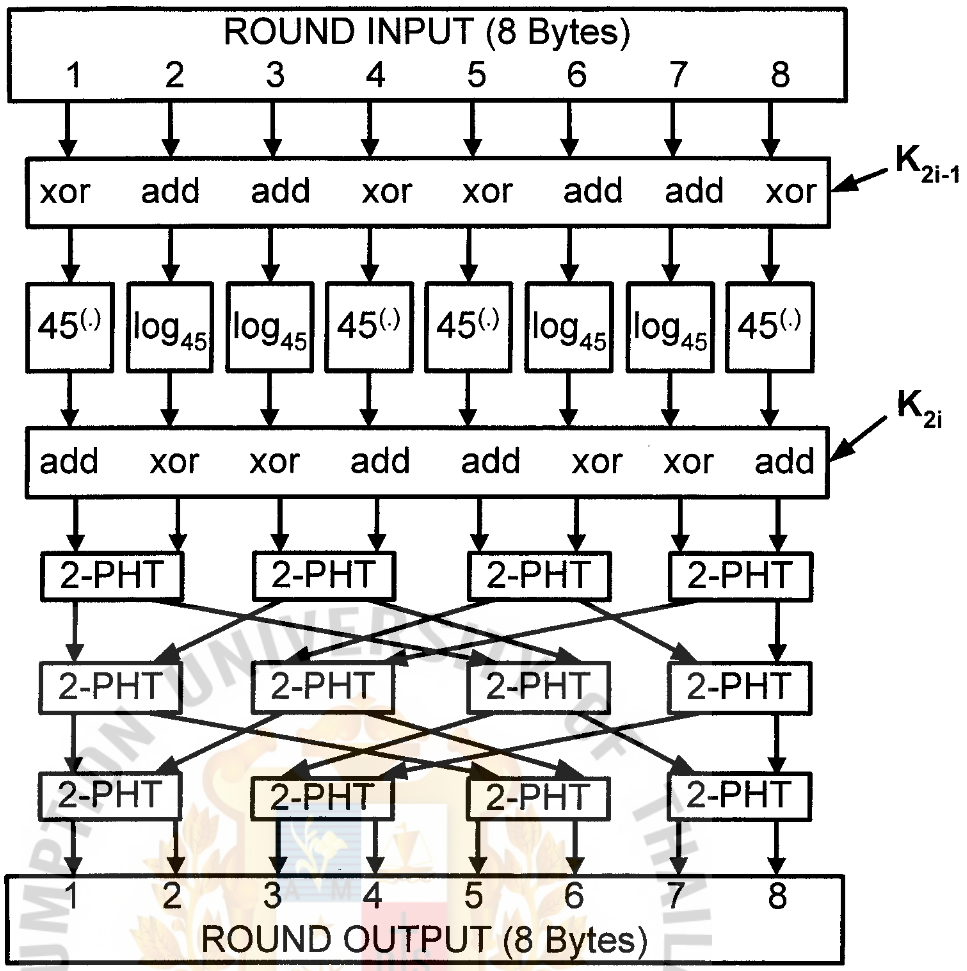

ROUND INPUT (8 Bytes)
1 2 3 4 5 6 7 8
xor add add xor xor add add xor
K_{2i-1}
$45^{(.)}$ $\log_{45}$ $\log_{45}$ $45^{(.)}$ $45^{(.)}$ $\log_{45}$ $\log_{45}$ $45^{(.)}$
K_{2i}
add xor xor add add xor xor add
2-PHT
1 2 3 4 5 6 7 8
ROUND OUTPUT (8 Bytes)

Figure2.6 One round SAFER K-64 encryption

The SAFER K-64 decryption algorithm and the detail for one round is shown in figure 2.7 and figure 2.8 respectively.

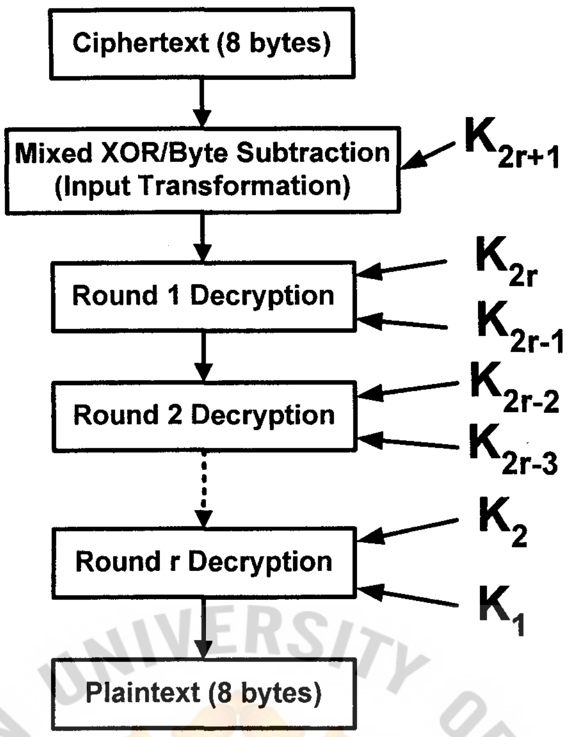


Figure 2.7 SAFER K-64 decryption

In figure 2.7, SAFER K-64 decryption has the r rounds. The first step, it begins with the input transformation of the Mixed XOR/Byte-Subtraction with K_{2r+1} subkey. There are bytewise subtraction (modulo-256 subtraction) of bytes 1, 4, 5, 8 and bytewise XOR (modulo-2 sum) of bytes 2, 3, 6, 7. But there are not simply reversing SAKER K-64 encryption.

In figure 2.8, the first step with the i^{th} round is the three-level inverse linear layer. That is, the inverse PHT (IPHT) is expressed by

$a_1 = b_1 - b_2$

$a_2 = -b_1 + 2b_2$

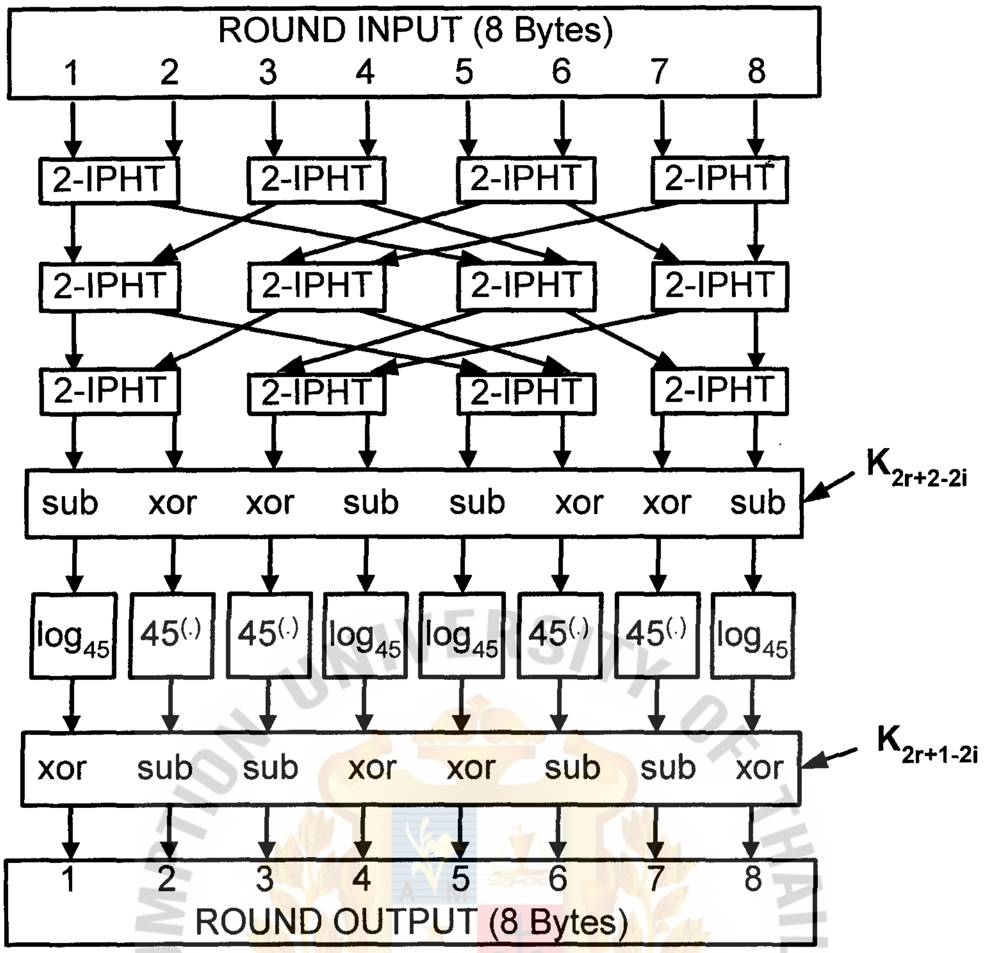


Figure 2.8 One round SAFER K-64 decryption

Then, those 8 bytes pass through the Mixed Byte-Subtraction/XOR of the inverse linear layer output with $K_{2r+2-2i}$ subkey and then inverse nonlinear layer. Finally with the ith decryption round, there are the Mixed XOR/Byte-Subtraction with $K_{2r+1-2i}$ subkey. There are bytewise XOR (modulo-2 sum) of bytes 1, 4, 5, 8 and bytewise subtraction (modulo-256 subtraction) of bytes 2, 3, 6, 7.

The procedure for generating subkeys $K_2, K_3, \ldots, K_{2r+1}$ from the user-selected subkey K1 is shown in figure 2.9. To ensure the round subkeys are individually random and no more than all-zero of one round subkey, the key biases $B_2, B_3, \ldots, B_{2r-1}$ are designed. If b[i,j] denotes the j-th byte of bias B_i, this byte can be expressed as the double exponential : b[i,j] = 45 ** [45 ** (9i+j) mod 257] mod 257 is also the key biases used in SAFER K-64.

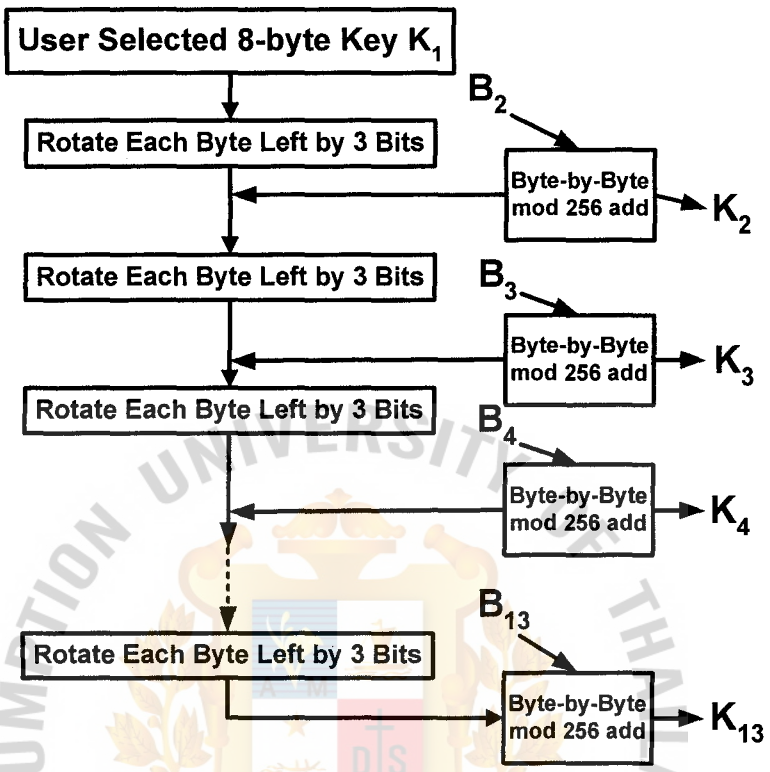

User Selected 8-byte Key K_1
Rotate Each Byte Left by 3 Bits
B_2
Byte-by-Byte mod 256 add
K_2
Rotate Each Byte Left by 3 Bits
B_3
Byte-by-Byte mod 256 add
K_3
Rotate Each Byte Left by 3 Bits
B_4
Byte-by-Byte mod 256 add
K_4
B_{13}
Rotate Each Byte Left by 3 Bits
Byte-by-Byte mod 256 add
K_{13}

Figure 2.9 Subkeys of SAFER K-64

Security of SAFER K-64

The 6-round SAFER K-64 is secure against differential cryptanalysis [23]. The key length for the brute-force attack is 2^{64} attempts. After only 3 rounds linear cryptanalysis is ineffective against this algorithm [20].

2.1.2 Public-Key Algorithms

Public-key algorithms, sometimes are called asymmetric algorithms [20]. There are two different keys: a public key that is available to everyone and a private key or secret key that is known only to oneself.

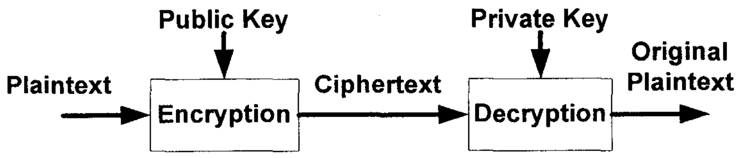


Figure 2.10 Public-Key Algorithm

2.1.2.1 Rivest, Shamir and Adleman (RSA)

RSA is one of the public-key algorithms and RSA are the initials of the tree creators, that is, Rivest, Shamir and Adleman [20].

In RSA encryption, the two large prime numbers of equal length (p and q must remain secret) is first chosen, that is, $n = pq$. Then the encryption key e is randomly chosen such that e and $(p-1)(q-1)$ are relatively prime. After that, the decryption key d is $d = e^{-1} \bmod ((p-1)(q-1))$ and also relatively prime with n. To encrypt a message m, first divide it into numerical blocks smaller than n (with binary data, choose the largest power of 2 less than n), that is, $c_i = m_i^e \bmod n$. To decrypt a message, take each encrypted block c_i and compute $m_i = c_i^d \bmod n$.

For example with the case of small values of prime numbers, let's A as a receiver and B as a sender. First , choose $p = 47$ and $q = 71$. Then, $n = pq = 3337$. $(p-1)(q-1)$=(47-1)(71-1) = 46 * 70 = 3220 Choose e from the relative prime of e and 3220 , $e = 79$ (at random). $d = 79^{-1} \bmod 3220 = 1019$ Discard the value of p and q. To encrypt the message, $m = 6882326879666683$, first divide m into small blocks (three-digit blocks), in which $m_1 = 688, m_2 = 232, m_3 = 687, m_4 = 966$, $m_5 = 668$ and $m_6 = 003$. For the first block ($m_1 = 688$) is encrypted as $c_1 = 688^{79} \bmod 3337 = 1570$. Repeatedly with $c_i = m_i^e \bmod n$ until $i = 6$, $c = 1570\ \ 2756\ \ 2091\ \ 2276\ \ 2423\ \ 158$.

Decrypting the message with $d = 1019$, so $1570^{1019} \bmod 3337 = 688 = m_1$. Repeatedly with $m_i = c_i^d \bmod n$ until $i = 6$, the original message is finally recovered.

Disadvantages of RSA is that RSA cannot encrypt only '0' or '1' since the ciphertext is the same result '0' and '1'. If we send the same message, it will get the same ciphertext. Then, it is easy to predict. The solution is that public key must be changed each time for sending the data.

Security of RSA depends on the difficulty for finding the value d from N and e since we can factorize the value N then we can find value p, q and also d. If the factorization is not so difficult, the value d can find easily and the original message can be recovered.

2.2 Neural Encryption Algorithm (NEA)

Neural Encryption Algorithm (NEA) is designed to be a symmetric-key encryption and secret-key block-enciphering algorithm with flexible block size of plaintext, flexible key length and multiple of layers. If block size for both plaintext and ciphertext is n bits, the NEA key (user key) are separated into 6 different keys with equally n-bit length in each except ESkey and DSkey have (n×n)-bit length i.e.

Selective key = ESkey, DSkey

Bias key = key11, key21

Replacing key = key12, key22

The total NEA key length is sum of ESkey , key11, key12, key21 and key22, that is (n×n) + (4×n) = n^2+4n bits.

From figure 2.12, the structure and operation of this algorithm is similar to neural network, that is,

1. NEA algorithm composes of 3 layers like partially connected feedforward multilayer perceptron i.e. input layer, hidden layer and output layer. More hidden layers are more secure. For simplicity, the proposed software is implemented only one hidden layer.
2. ESkey and DSkey are weights of the algorithm and their values are only 0 and 1, not real number like neural networks. ESkey is the modified inverse of DSkey and both are called selective keys since only the value of one bit is passed through or selected but the value of zero bit is not selected or not passed through. This modified inverse is the modified Gauss Jordan that uses only exclusive-or (xor) operation instead of plus and minus.

Example

Suppose

$$\text{ESkey} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

DSkey is the modified inverse (modified Gauss Jordan) of ESkey, then

$$\left[\begin{array}{c|c} \text{ESkey} & \text{Identity Matrix} \end{array}\right]$$

$$= \left[\begin{array}{cccc|cccc} 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{array}\right]$$

$$\left[\begin{array}{cccc|cccc} 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \end{array}\right] \begin{array}{l} \\ \text{row2} \oplus \text{row3} \\ \text{row1} \oplus \text{row3} \\ \text{row3} \oplus \text{row4} \end{array}$$

$$\left[\begin{array}{cccc|cccc} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \end{array}\right] \begin{array}{l} \text{row1} \oplus \text{row2} \\ \\ \\ \\ \end{array}$$

$$= \left[\begin{array}{c|c} \text{Identity Matrix} & \text{DSkey} \end{array}\right]$$

Finally,

$$\text{DSkey} = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

3. key11 and key21 are called bias key since those keys used to combine with n-bit block.
4. To separate the layers, nonlinear operation called Replacing-Box (R-Box) is inserted between hidden layer and output layer with key12 and key22 that why those keys are called replacing keys.
5. Replacing-box (R-Box) is inserted between hidden layer and output layer

The strength of NEA is key lengths and multiple hidden layers. More key lengths and more multiple hidden layers are more secure. Before going to NEA encryption, let's see how R-Box works:

Replacing-box (R-Box)

The R-Box is a 2-dimensional array or a square matrix with m×m size. The value m (R-Box size) depends on number of bit i in each sub-block i.e. $2^i = m$. For example,

- The 2-bit sub-block results in R-Box with 4×4 size. In this case, $i = 2 \rightarrow m = 2^2 = 4$.
- The 3-bit sub-block results in R-Box with 8×8 size. In this case, $i = 3 \rightarrow m = 2^3 = 8$.

Each element in R-Box with m×m size has the range value from 0 to m-1. For example,

- If m = 4, each element in R-Box is value form 0 to 3.
- If m = 8, each element in R-Box is value from 0 to 7.

Note that the sequence of element values in each row is not in the same order.

In this thesis, the n block is broken into (n / 2) sub-blocks with 2-bit input of R-Box. Then R-Box is a matrix with dimension of 4×4 and the element values are from 0 to 3.

The R-Box in this thesis is shown in Table 2.9. The output of R-Box has 2 bits which replacing with the values in R-Box. The 2-bit input of R-Box firstly converts into base 10 and that number is the column number of R-Box output. The row number of R-Box output is from the 2-bit replacing key after converting into base 10.

Table 2.9 R-Box

	0	*1*	*2*	*3*
0	0	2	3	1
1	2	0	1	3
2	1	3	0	2
3	3	1	2	0

Example:

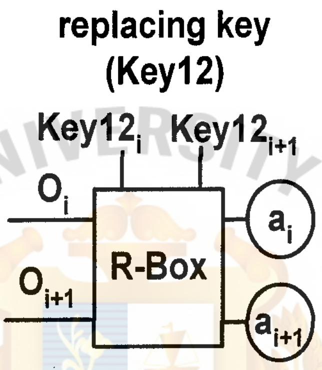


Figure 2.11 R-Box

Input of R-Box : O_i and O_{i+1} $\rightarrow$ Column number

Replacing key : $key12_i$ and $key12_{i+1}$ $\rightarrow$ Row number

If $[O_i \quad O_{i+1}]$ = $[1 \quad 0]$ $\rightarrow$ convert to base 10 = 2 $\rightarrow$ column 2

If $[key12_i \quad key12_{i+1}]$ = $[0 \quad 1]$ $\rightarrow$ convert to base 10 = 1 $\rightarrow$ row 1

Output of R-Box is the value in column 2 and row 1 from table 2.9 = 1 = 01_2

Therefore, $[a_i \quad a_{i+1}]$ = $[0 \quad 1]$ is the 2-bit output of R-Box.

The NEA Encryption is described in figure 2.12. There are 3 layers i.e. input layer, hidden layer and output layer. Hidden layer and output layer have the same operation with different keys. Hidden layer uses ESkey as selective key, key11 as bias key and key12 as replacing key. Output layer uses DSkey as selective key, key21 as bias key and key 22 as replacing key. First, the n-bit block of plaintext goes to the hidden layer with ESkey weight and the each bit of n-bit block combines with key11 and modulo 2. The result after modulo 2 breaks into 2-bit sub-blocks (n/2 sub-blocks). Each sub-block passes through R-Box with Key12. The output of R-Box with Key 12 is the n-bit output of hidden layer that composes of (n/2) 2-bit sub-blocks. With the same operation with different keys in output layer , the output is n-bit block of ciphertext.

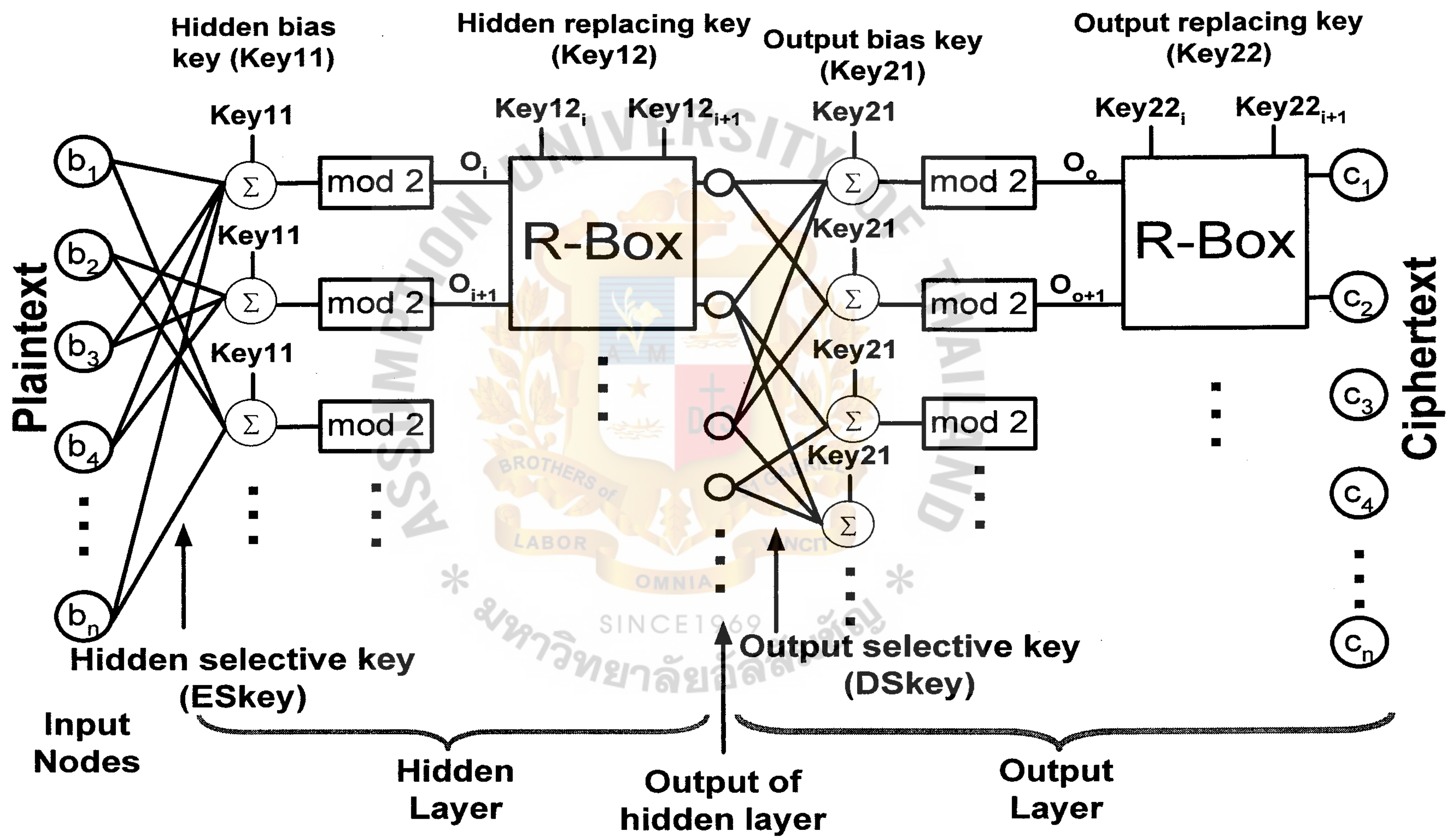


Figure 2.12 NEA encryption

The NEA decryption (see figure 2.13) is the reverse operation of the NEA encryption and also has 3 layers i.e. input layer, hidden layer and output layer. The operation in hidden layer and output layer has the same with different keys. ESkey as selective key, key22 as replacing key and key21 as bias key are used in hidden layer. DSkey as selective key, key12 as replacing key and key11 as bias key are used in output layer. First, the n-bit block of ciphertext goes to the hidden layer and passes through R-Box with key22. The (n/2) 2-bit sub-blocks result in the n-bit block, the output of R-Box. The n-bit block is weighted with ESkey and then each bit combines with key21 and modulo 2. The result after modulo 2 is the n-bit output of hidden layer. With same operation with different keys in output layer, the output is n-bit block of original plaintext.

To compare with other implemented encryption algorithms, there are 2 cases for NEA complexity comparison i.e.

1. 64-bit block for both plaintext and ciphertext (n = 64)

In this case, n is equal to 64. There are 64-bit block for both plaintext and ciphertext with 64-bit key length for key11, key12, key21 and key 22 including (64×64)-bit key length for ESkey and DSkey. The total key length are 64^2+ (4×64) = 4352 bits.

2. 8-bit block for both plaintext and ciphertext (n = 8)

In this case, n is equal to 8. There are 8-bit block for both plaintext and ciphertext with 8-bit key length for key11, key12, key21 and key 22 including (8×8)-bit key length for ESkey and DSkey. The total key length are 8^2+ (4×8) = 96 bits.

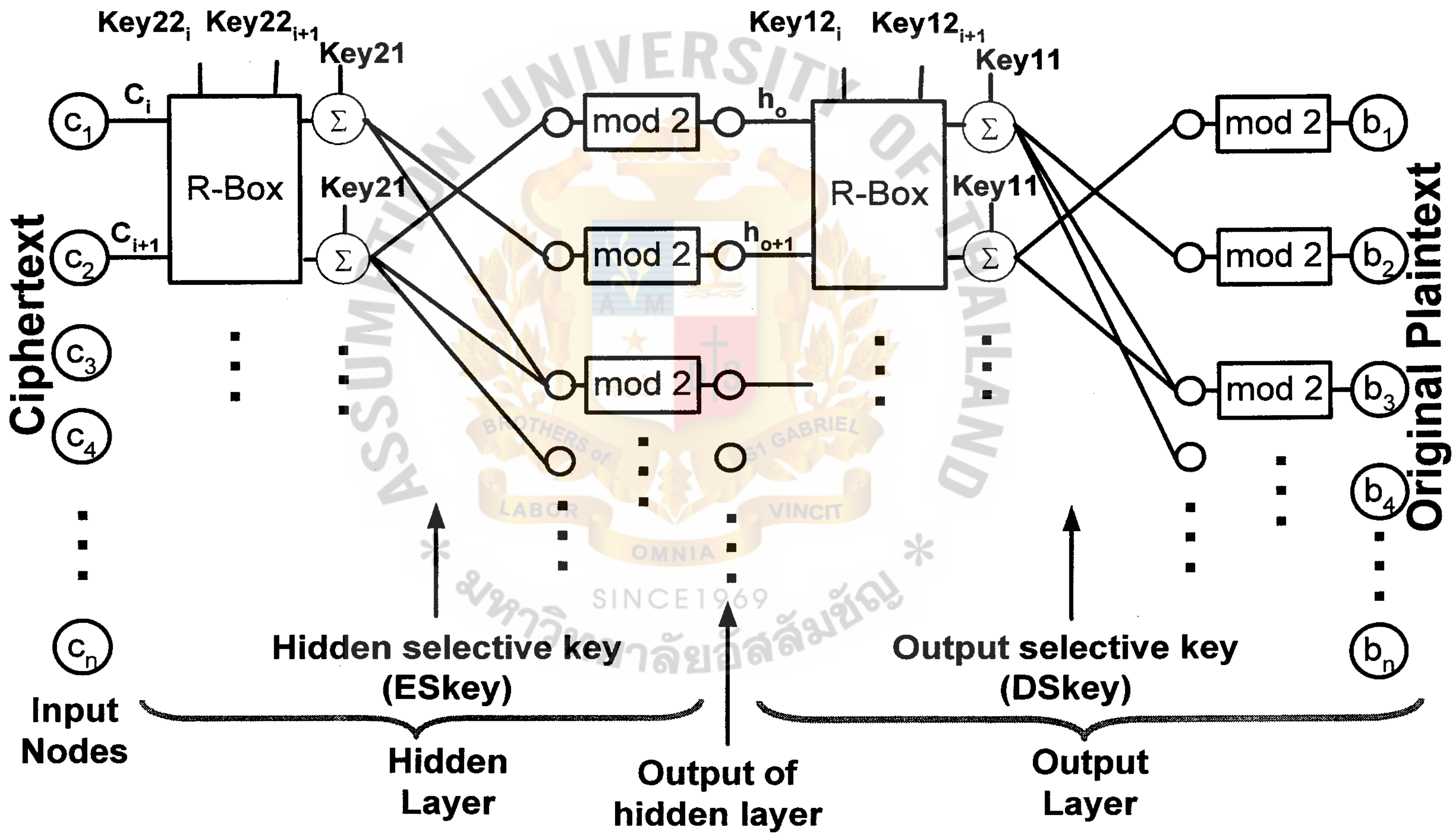


Figure 2.13 NEA decryption

For comparison, if the 1st case with n = 64 runs in MATLAB, it will take a bit more time than the case with n = 8. To against with brute force attack, the 1st case has 2^{4352} possible keys more than the (n= 8) case with 2^{96} possible keys, that is, the 1st case is hard for brute force attack than the latter case.

Brute force attack is by trying every possible key one by one and checking whether the resulting plaintext is meaningful.

Table 2.10 Comparison of all implemented encryptions against the brute-force attack

Encryptions	Key length (bits)	Possible keys
DES	56	2^{56}
Triple DES	56×2 = 112	2^{112}
SAFER K-64	64	2^{64}
NEA with 64 bit block of plaintext and ciphertext	4352	2^{4352}

With 64-bit block of plaintext and ciphertext in DES, triple DES, SAFER K-64 and NEA, the result from table2.10 shows that NEA is the strongest encryption algorithm against a brute-force attack.

2.3 Pulse Code Modulation (PCM)

PCM is one of analog-to-digital encoding techniques and is used by soundcard to convert analog sound to digital.

First, analog pulses are sampled to be PAM (Pulse Amplitude Modulation) with equal intervals [24]. Next, the PAM pluses are quantized. Quantization is a method of assigning integral values in a specific range to sampled instances. Each quantized value is translated into sign-and-magnitude binary equivalent and then formed the digital pulses. The PCM process is shown in figure 2.13.

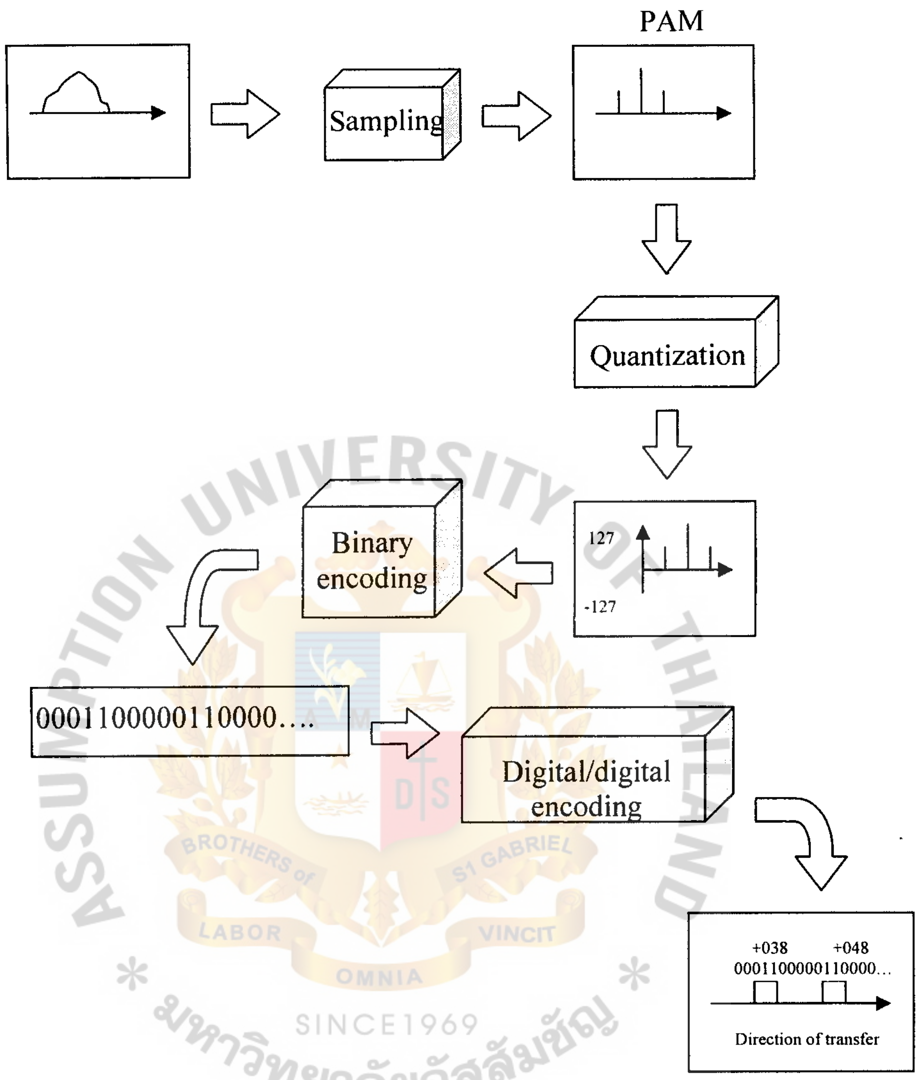


Figure 2.14 PCM process

2.4 Computer network

In this research, voice over TCP/IP testing is PC-to-PC chat. It requires at least 2 computers connected. It does not depend on topologies as well as network address class. If the computers can use the command "ping" in the command prompt to connect other computers, they can chat to each other by using this developed software. Users are unlimited at a time, but only the couple can chat to each other at a time. The software implementation and the testing are provided in chapter 3 and 4 respectively.

Chapter 3

Software Design and Development

The software development for testing voice over TCP/IP with DES, Triple DES, SAFER K-64, RSA, NEA and none of encryption is divided into 5 parts:

1. Development of voice procedure: To send (by microphone) / receive (by speaker) voice and set the soundcard parameters.
2. Development of encryption techniques: There are 5 encryption techniques in the software, that is, DES, triple DES, SAFER K-64, RSA and NEA.
3. Design of message protocols: To send messages in network.
4. Development of server: To authenticate the user name and password of all software users.
5. Development of client with DES, Triple DES, SAFER K-64, RSA, NEA and none of encryption : To compare the well-known encryption in both symmetric algorithms (DES, Triple DES and SAFER K-64) and public algorithm (RSA) with the proposed encryption (NEA).

All parts of the software have been developed with Borland Delphi Enterprise, version 6.0 under Microsoft Windows XP (given in Appendix A).

3.1 Development of voice procedure

In run-time library of Delphi 6.0, there is MMsystem unit for multimedia API interface unit and then it can be applied for soundcards. Waveform audio part in MMsystem unit is applied in the software. There are procedures for send/receive voice by microphone/speaker:

1. InitAudio is to define the format of voice and to be initial voice parameters used in the software. There are:

1.1 wFormatTag := Wave_Format_PCM; The waveform format structure is set to PCM.

1.2 nChannels := 1; The channel is set to one for voice mode.

1.3 nSamplesPerSec := 8000; The sampling rate is set to 8000 Hz.

1.4 nAvgBytesPerSec := 8000; The buffer estimation is set to 8000 b/s.

1.5 nBlockAligh := 1; The block size of data is set to 1.

1.6 wBitsPerSample := 8; The number of bits per sample for voice is set to 8 bits/sample.

1.7 WaveOutOpen is to test the assigned format that can form the voice waveform.

1.8 WaveInOpen is to test the assigned format that can record the voice waveform.

1.9 INBLOCKFISRTBUFFER = 5 is to form initially 5 blocks of buffer.

1.10 WaveInStart is to form the voice waveform.

2. CreateInBlock is to form and store the voice.

   2.1 bs is to assign the buffer size.

   2.2 GetMem is to allocate memory for voice packet.

   2.3 Header is to form the header of new input voice waveform.

   2.3.1 lpData is to point the stored memory.

   2.3.2 dwBufferLength is to define the buffer size.

   2.4 WaveInPrepareHeader is to test the new Header.

   2.5 WaveInAddBuffer is to store the voice packet in memory with lpData pointer.

3. MMInDone is when the sender talk with microphone and then soundcard digitizes with PCM to be the voice packet stored in Windows.

3.1 AudioInited is to check the initialization of voice.

3.2 WaveInUnPrepareHeader is to check the header from Windows that can work.

3.3 Size is to assign the buffer size.

3.4 GetMem is to allocate the memory.

3.5 pow2 is calculate the root mean square power of voice.

3.6 filter1 is set due to different quality of microphone.

4. CreateOutBlock is to form the voice packets and then send them to soundcard for speaker.

   4.1 AudioInited is to check the initialization of voice.

   4.2 Header is to form the header of new output voice waveform.

      4.2.1 lpData is to point the voice packet.

      4.2.2 dwBufferLength is to set the buffer size.

   4.3 WaveOutPrepareHeader to test the new output Header.

   4.4 WaveOutWrite is to sent voice packet to soundcard

5. MMOutDone is when voice packet is sent to speaker.

   5.1 Header is to form the header of voice from Windows.

   5.2 AudioInited is to check the initialization of voice.

   5.3 WaveOutUnPrepareHeader is to check the header from Windows that can work.

   5.4 FreeMem is to dispose of the header

   5.5 Dispose is to release memory of the header.

6. CloseAudio is to close audio device.

   6.1 AudioInited is to check the initialization of voice.

6.2 HWaveOut is to point the data in buffer.

6.3 HWaveln is to point the data in buffer.

3.2 Development of encryption techniques

There are 5 encryption techniques developed for testing and comparing voice/text chat over computer network.

3.2.1 DES

The main functions are developed for DES :

1. IpOut : is for initial permutation (IP)
2. Efunction : is for expansion permutation.
3. left_shift : is one round of circular left shift.
4. PC1 : is for Key Permutation
5. PC2 : is for compression permutation.
6. myxor : is for xor operation.
7. S : is for ith row and jth column
8. Sprocess : is for S-box substitution
9. Pprocess : is for P-box permutation.
10. inIpOut : is for final permutation (IP^{-1})
11. DES_enblock : is for DES encryption.
12. DES_deblock :is for DES decryption.
13. DES_encryp : is to encrypt the input message with DES encryption.
14. DES_decryp : is to decrypt the cipher message with DES decryption.

3.2.2.Triple DES

The main functions are developed for triple DES :

1. IpOut : is for initial permutation (IP).
2. Efunction : is for expansion permutation.
3. left_shift : is one round of circular left shift.
4. PC1 : is for Key Permutation
5. PC2 : is for compression permutation.
6. myxor : is for xor operation.
7. S : is for ith row and jth column
8. Sprocess : is for S-box substitution
9. Pprocess : is for P-box permutation.
10. inIpOut : is for final permutation (IP^{-1})
11. DES_enblock : is for DES encryption.
12. DES_deblock :is for DES decryption.
13. Swap_subkey1 : is the user key.
14. Swap_subkey2 : is the fixed key.
15. TDES_encryp : is to encrypt the input message with Triple DES encryption and 2 keys (user key and fixed key)
16. TDES_decryp : is to decrypt the cipher message with Triple DES decryption and 2 keys (user key and fixed key)

Triple DES is 3 times repeated of DES encryption and decryption operations

Functions from number 1 to number 12 is the same as DES, but triple DES is 3 times of DES operation and applied with 2 different keys. The first key is the user key and another key is the fixed key (set constant values).

3.2.3 SAFER K64

The main functions are developed for SAFER K-64 :

1. gen_key : to generate key from K_1 to K_{13}.
2. gen_table : to generate table of exponential and logarithm to the base 45.
3. mat : is for 2-PHT.
4. invmat : is for 2-IPHT
5. SaferK64_encryp : is for SAFER K-64 encryption
6. SaferK64_decryp : is for SAFER K-64 decryption.
7. K64_encryp : is to encrypt the input message with SAFER K-64 encryption.
8. K64_decryp : is to decrypt the cipher message with SAFER K-64 decryption.

3.2.4 RSA

The main functions are developed for RSA :

1. PartialFaction :to generate table of prime number values
2. FindFaction : to generate faction table for the d value.
3. FindE : to find e value (encryption exponent)
4. FindD : to find d value (decryption exponent)
5. Gentable, GenTable_en : to generate faction table of exponential base 2.
6. RSA_encryption : is for RSA encryption.
7. RSA_decryption : is for RSA decryption.
8. RSA_encrypt : is to encrypt the input message with RSA encryption
9. RSA_decrypt : is to decrypt the cipher message with RSA decryption.

3.2.5 NEA

The main functions are developed for NEA :

1. fix : is for non integer value towards to zero, e.g., 2.7 = 2, 4.1 = 4, -1.9 = -1
2. rem : is for remainder after division.
3. keygen_nea : is to generate NEA key.
4. findinv : is to find the inverse of array of matrix.
5. encrypt1 : is to find the output of hidden layer which is the first half of NEA encryption.
6. encrypt2 : is to find the output of output layer which is the second half of NEA encryption.
7. decrypt1 : is to find the output of hidden layer which is the first half of NEA decryption.
8. decrypt2 : is to find the output of output layer which is the second half of NEA decryption.
9. NEA_encrypt : is to encrypt the input message with NEA encryption
10. NEA_decrypt : is to decrypt the cipher message with NEA decryption

3.3 Design of message protocols

The message protocols are designed to send messages in network as followings:

1. @nme<name>@pas<password>@end : to send username-and-password login to server.

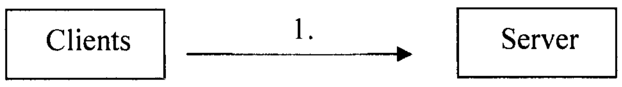


Figure 3.1 @nme message protocol

2. @ lst<onlinelist>@end : to send list of online users to client.

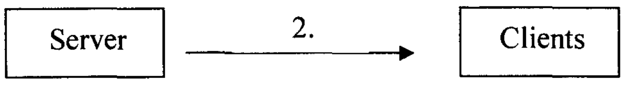


Figure 3.2 @lst message protocol

3.@req : to send the requested chat message.

3.1 @req<receiver>@typ<type>[@pas<password>]@end

3.2 @req<caller>@typ<type>[@pas<password>]@end

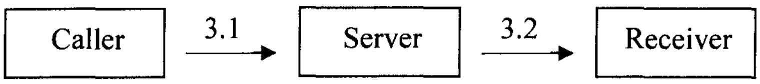


Figure 3.3 @req message protocol

where caller/receiver = user login name

type = encryption

password = encryption key

4.@acp : to send receiver's acceptance message.

4.1 @acp<caller>[@c<n>&<e>]@end

4.2 @acp<receiver>[@c<n>&<e>]@end

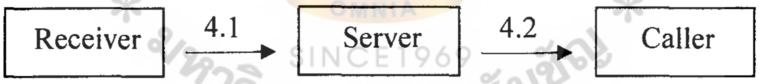


Figure 3.4 @acp message protocol

[@c<n>&<e>] is used only in RSA.

5.@rej : to send receiver's refusal message.

5.1 @rej<caller>@end

5.2 @rej<receiver>@end

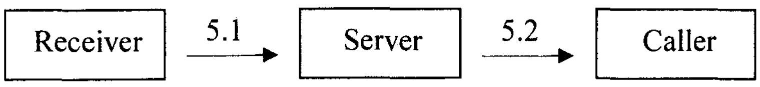


Figure 3.5 @rej message protocol

6. @pas<password>@end : to change login password.

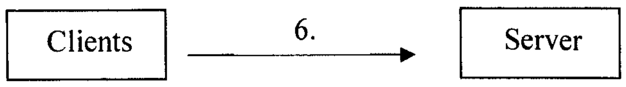


Figure 3.6 @pas message protocol

7. @ipc<ip>@end : to send caller ip to receiver by server.

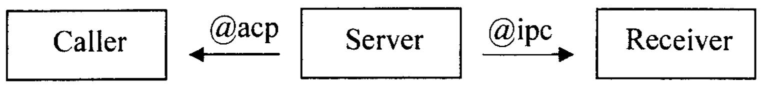


Figure 3.7 @ipc message protocol

8. @msg<data>#0: to send message (text chat) between caller and receiver, not voice chat.

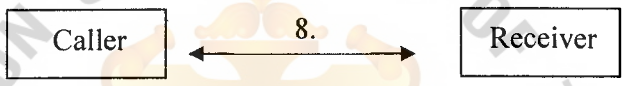


Figure 3.8 @msg message protocol

9. @2#0 : to inform successful receiver's voice chat and let caller start to record the time.

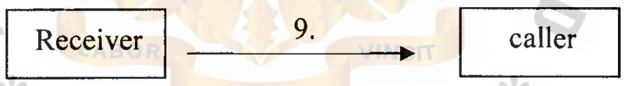


Figure 3.9 @2 message protocol

10. @1#0 : to inform successful caller's voice chat and let receiver start to record the time.

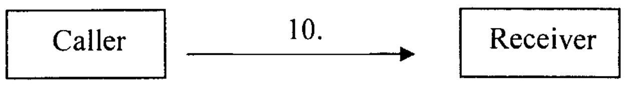


Figure 3.10 @1 message protocol

11. @can@end : when caller cancels his request chat.

12. @fin@end : to inform the end of voice/text chat.

13. @bus@end : to inform that user is now chatting to another person.

Note : <> : required and [] : option.

3.4 Development of server

Server is designed to check the user name-and-password login before using the software as well as send the requested message from caller to receiver. All process between server and client is encrypted with DES. Database Desktop version 7.0 is used to store all user name-and-password login (account.db) and is designed to enter password before open this database. Only the administration of software assigns the username-and-password login to software users. There are procedures used to connect with clients.:

1 serverAccept is to accept the user login.

2 Broadcast_list is to send all online uses list to all online users with the message protocol: @lst<onlinelist>@end

3 Answer_chat is to manage both request and answer of chat.

4 Req_chat is for request chat.

5 User_connect is when users connect to server.

6 serverClientRead is to receive and decrypt the data from network with DES encryption.

7 serverClientDisconnect is to disconnect if users assign the wrong login password.

8 serverClientError is to support all errors that are possible to occur during run-time.

9 FormCreate is to set the initial value.

10 end_data is to send data to users in network with DES encryption.

The request/answer chat of users is passed though the server described in figure 3.2 and as following:

1. To request chat, caller sends the requested message to server and then server send that message to receiver.
2. If receiver accepts the requested message, there is no connection to server and caller is able to send message (voice/text chat) directly to receiver.
3. If receiver refuses the caller contact, server sends the denied message to caller.

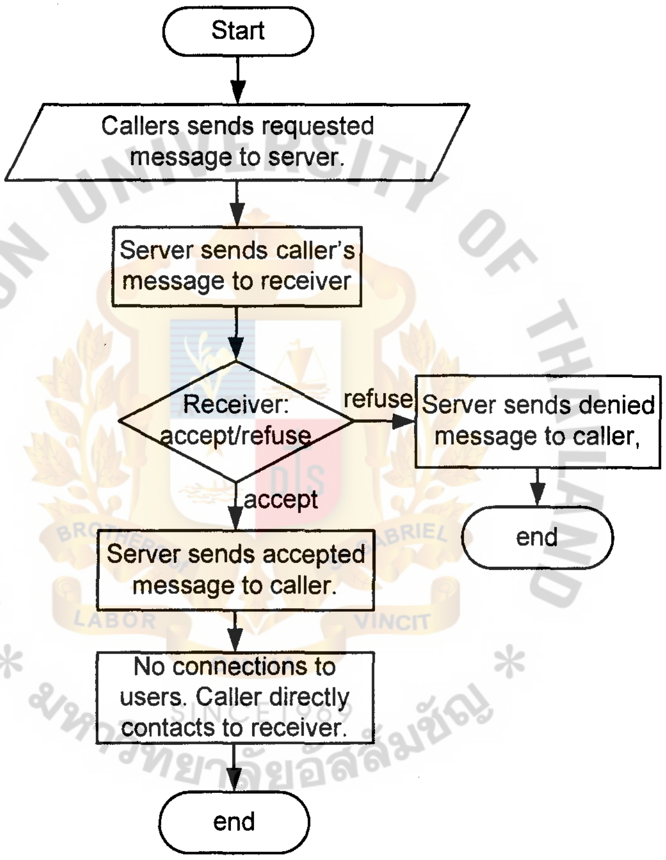

Start
Callers sends requested message to server.
Server sends caller's message to receiver
Receiver: accept/refuse
refuse
Server sends denied message to caller,
end
accept
Server sends accepted message to caller.
No connections to users. Caller directly contacts to receiver.
end

Figure 3.11 Relations between server and users for making the request call.

3.5 Development of client.

Client is designed to be voice/text chat over computer network with DES, Triple DES, SAFER K-64, RSA, NEA and none of encryption. There are 6 units to form client and their relation to make a request call is described in figure 3.3.

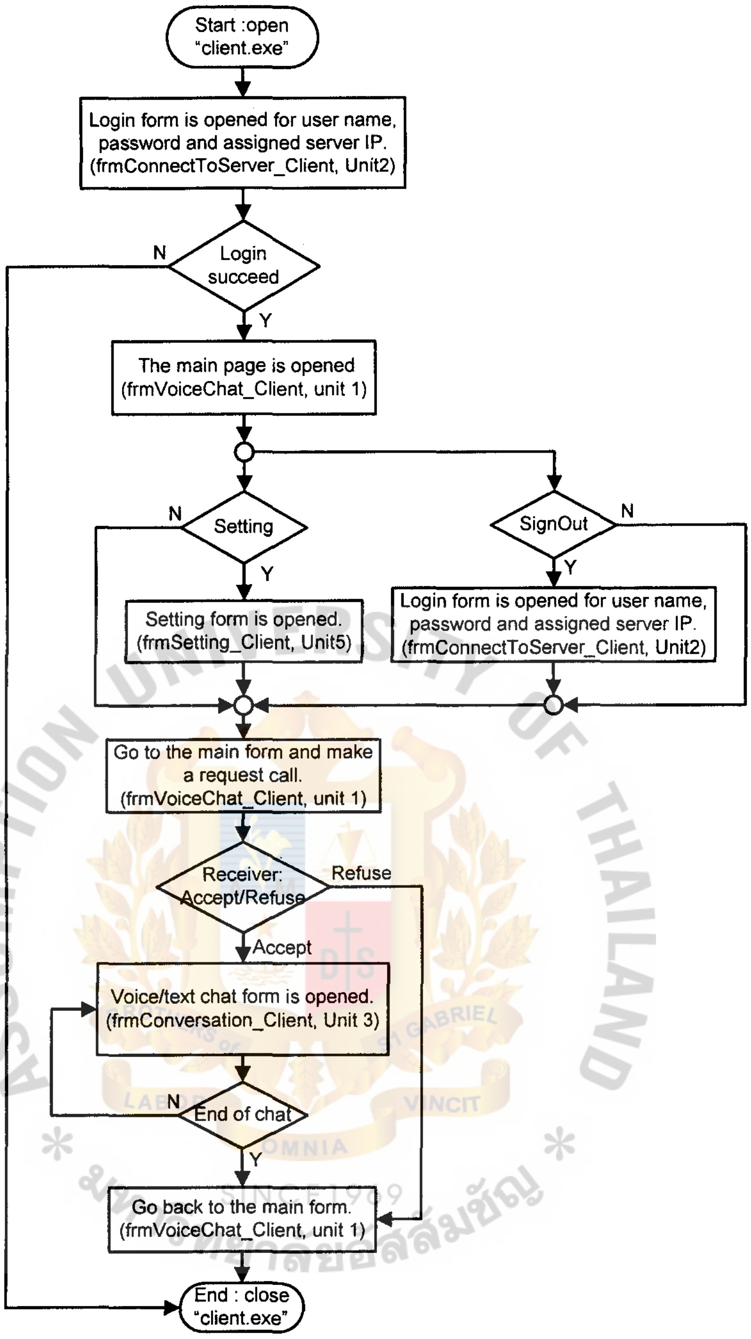

Start :open "client.exe"
Login form is opened for user name, password and assigned server IP. (frmConnectToServer_Client, Unit2)
Login succeed
N
Y
The main page is opened (frmVoiceChat_Client, unit 1)
Setting
N
Y
Setting form is opened. (frmSetting_Client, Unit5)
SignOut
N
Y
Login form is opened for user name, password and assigned server IP. (frmConnectToServer_Client, Unit2)
Go to the main form and make a request call. (frmVoiceChat_Client, unit 1)
Receiver: Accept/Refuse
Refuse
Accept
Voice/text chat form is opened. (frmConversation_Client, Unit 3)
End of chat
N
Y
Go back to the main form. (frmVoiceChat_Client, unit 1)
End : close "client.exe"

Figure 3.12 Relations between forms in clients for voice/text chat.

The development of client consists of 6 forms (units) as followings:

1. Unit 1: frmVoiceChat_Client is to store all values, to make the request call, to sign out, to call frmSetting_Client (unit 5) for changing the login password, setting the

encryption type and generate the key, and to login. This form uses unit 2, unit 3, unit 4 and unit 5. There are main procedures/functions/events in form1:

1.1 ClientConnect is when clients login to server.

1.2 ClientRead is to read all contact message between client and server with DES decryption.

1.3 FormCreate is to initial the parameter values.

1.4 bbtCallClick is to send the request message and key to server.

1.5 bbtSignOutClick is to logout the software.

1.6 bbtSettingClck is to call frmSetting_Client (unit 5).

1.7 ClientConnecting is to show the connecting status in mmoDisplay.

1.8 ClientDisConnect is logout the software and show the disconnect status in mmoDisplay.

1.9 ClientError is to show all possible errors.

1.10 btnSignInClick is to call frmConnectToServer_Client (unit 2).

1.11 FormClose is to close the connection.

1.12 Send_data is to send messages to server with DES encryption.

1.13 FormActivate is to call frmConnectToServer_Client (unit 2) before frmVoiceChat_Client (Unit 1) beginning.

1.14 All functions in NEA is in this form and the NEA key is generated from frmSetting_Client (unit 5).

1.15 All functions in DES are in this form to encrypt/decrypt messages between server and clients.

2. Unit 2 : frmConnectToServer_Client is to login the software. There are events in form 2:

2.1 btnConnectClick is to send user name, password login, port number, server ip address and call frmVoiceChat_Client (unit 1).

2.2 btnCancelClick is to close frmConnectToServer_Client (unit 2).

3. Unit 3 : frmConversation_Client is for voice/text chat after login and encryption selection. This form uses unit 1 and unit 5. There are main procedures/functions/events in form3:

   3.1 All functions/procedures of DES, triple DES, SAFER K-64 and RSA are in this form to encrypt/decrypt voice/text chat. For NEA, it is to call NEA functions in unit 1.

   3.2 All voice procedures in section 3.1 except MMInDone has process for encryption process.

   3.3 find_charindex is to convert character to ANSI (Windows character set).

   3.4 charindex is to convert ANSI to character.

   3.5 sbtConnectClick is to close connections.

   3.6 FormCreate is set the initial values.

   3.7 PostMessage is to show sound level in mmoShwSoundLevel.

   3.8 sbtSoundClick is to turn on the speaker.

   3.9 ServerSockect1ClientConnect is to enable the voice chat between users.

   3.10 ServerSocket1ClientDisconnet is to close the voice chat between users.

   3.11 ServerSocket1ClientError is to show all possible error messages

   3.12 ServerSocket1ClientRead is to read messages from network with selected decryption.

   3.13 edtSendTextKeyPress is to send text chat between client users with selected encryption.

3.14 FormClose is to inform frmVoiceChat_Client for this form termination and show the free status.

3.15 ServerSocket1Accept is to enable voice/text chat control object.

3.16 edtSoundFilterChange is to set the filter value.

3.17 sbtAdjustFilterClick is to set the assigned filter value from edtSoundFilterChange.

4. Unit 4: frmIncomingCall_Client is to inform the receiver for caller's request call with caller name, encryption type and key (encrypted with DES). This form uses unit 1 and unit 3. There are events in form4:

   4.1 btnAcceptClick is for accepting the caller's requested chat and send the acceptance message to frmVoiceChat_Client.

   4.2 btnCancelClick is for refusing the caller's requested chat and send the refusal message to frmVoiceChat_Client.

5. Unit 5 : frmSetting_Client is to change the password login, to choose the encryption and key. This form uses unit 1, unit3 and unit 6. There are 4 tabs:

   5.1 tbsPasswdSignIn for changing the password login

      5.1.1 btnOK_PasswdSignInTabClick is to send the new password login to frmVoiceChat_Client.

      5.1.2 btnCancel_PasswdSignInTabClick is to clear all edit control box in tbsPasswordSignIn tab.

      5.1.3 btnClose_PasswdSignInTabClick is to close frmSetting_Client.

   5.2 tbsChEncrypt for choosing the encryption.

      5.2.1 btnApply_ChEncryptTabClick is to send the users's selection of encryption to frmVoiceChat_Client.

      5.2.2 rdgChEncrypt_ChEncryptTabClick is to open the next tab after user chooses the encryption.

5.2.3 btnClose_ChEncryptTabClick is to close frmSetting_Client.

5.3 tbsEncryptKey for key generation of DES, triple DES, SAFER K-64 and NEA.

5.3.1 btnGen_EncryptKeyTabClick is to generate key up to user selection in DES, triple DES, SAFER K-64 and NEA as well as call functions for key generation and send this key to frmVoiceChat_Client.

5.3.2 btnCLose_EncryptKeyTabClick is to close frmSetting_Client.

5.3.3 edtKey_EncryptKeyTabChange is to count the character number in edtKey.

5.4 tbsRsaKey is for RSA key generation.

5.4.1 btnGenKey_RsaKeyTabClick is to generate RSA keys and then send those values to frmVoiceChat_Client.

5.4.2 btnShwPrime_RsaKeyTabClick is to call frmPrime_Client for showing prime numbers.

6. Unit 6: frmPrime_Client is only to show the prime numbers for RSA and is called by btnShwPrime_RsaKeyTabClick button in tbsRsaKey tab.

Chapter 4

Testing

The developed software for voice/text chat over computer network is used to test DES, triple DES, SAFER K-64, RSA, the proposed NEA and no encryption for comparison. There are 2 experiments for the testing:

1. The discontinuity of voice at several buffer size.
2. The security of voice/text chat over computer network.

4.1 The discontinuity of voice at several buffer size

This experiment is to see whether the buffer size affects the continuity of voice at 64 bytes, 128 bytes, 256 bytes, 512 bytes, 1024 bytes, 2048 bytes, 4096 bytes and 8192 bytes. The music from sound recorder is used for voice testing. The 50 questionnaires are used for testing by students and teachers at Saint Dominic School. The criterias/scales are as following :

5	: Very Good	=	Voice quality is very good with no noise and no shiver
4	: Good	=	Voice quality is very good with noise and shiver sometimes.
3	: Fair	=	Voice quality is good with no noise and no shiver
2	: Satisfactory	=	Voice quality is good with noise and shiver sometimes.
1	: Poor	=	Voice quality is fair with noise and shiver.

The results are shown in Table 4.1, Table 4.2, Figure 4.1 and Figure 4.2. Table 4.1 is the average values and Table 4.2 is the standard deviation of all 50 questionnaires. Figure 4.1 and Figure 4.2 shows the average result of voice quality and length of voice discontinuity, respectively.

Table 4.1 Effect of buffer size for voice discontinuity with average result

Encryption / Buffer size (Bytes)	DES			Triple DES			SAFER K-64			RSA			NEA			None		
	Voice quality	Times	Seconds	Voice quality	Times	Seconds	Voice quality	Times	Seconds	Voice quality	Times	Seconds	Voice quality	Times	Seconds	Voice quality	Times	Seconds
64	1	0.42	1.2	1	0.72	1.6	1.66	0.9	1.6	1.9	0.8	1.4	0.65	0.75	1.5	2	0.7	0.7
128	1.94	0.74	2.52	1.74	0.66	2.34	1	0.82	2.7	1.7	0.8	1.55	3.55	0.75	1.5	3.65	0.6	1.5
256	2.98	0.61	3.44	3.56	0.8	3.4	4.52	0.88	3.7	4.3	0.8	2.95	5	0.75	2.3	5	0.75	2.3
512	4.14	1.85	3.42	4.16	1.98	3.16	4.62	1.65	3.8	4.5	0.95	3.2	5	0.75	2.1	5	0.7	2.15
1024	3.46	1.82	4.06	2.9	3.04	5.36	4.4	1.75	4.65	4.45	0.85	5.85	5	0.8	3.1	5	1.95	3.65
2048	0.68	3.44	5.68	3.58	4.22	5.7	2.98	3.25	7.6	3.55	3.5	5.3	3.65	3.5	5.6	3.7	3.5	5.3
4096	0.5	2.66	9.52	1.82	3.14	9.74	3.42	4.25	9.4	3.75	4.7	5.8	3.7	1.95	7.1	3.55	3.05	7.6
8192	0.54	2.32	11.8	1.82	1.52	14.52	3.06	2.65	13.15	3.55	4.15	7.7	4.35	2.6	9.63	4	1.9	11.95

Note: Number of discontinuity (times) = Times
Length of discontinuity (seconds) = Seconds

Figure 4.1 Effect of buffer size for voice quality with average result

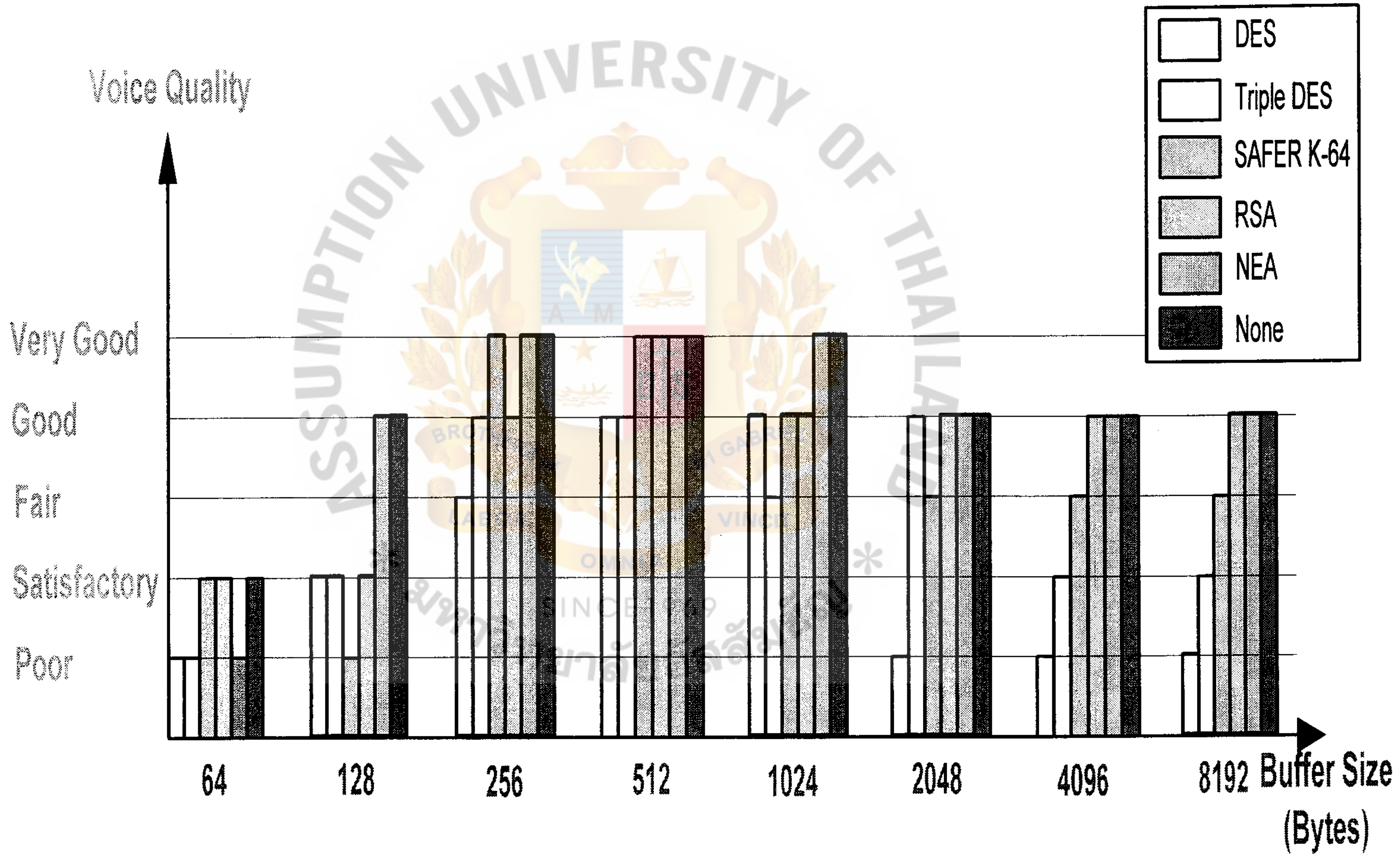


Figure 4.2 Effect of buffer size for voice discontinuity with average result

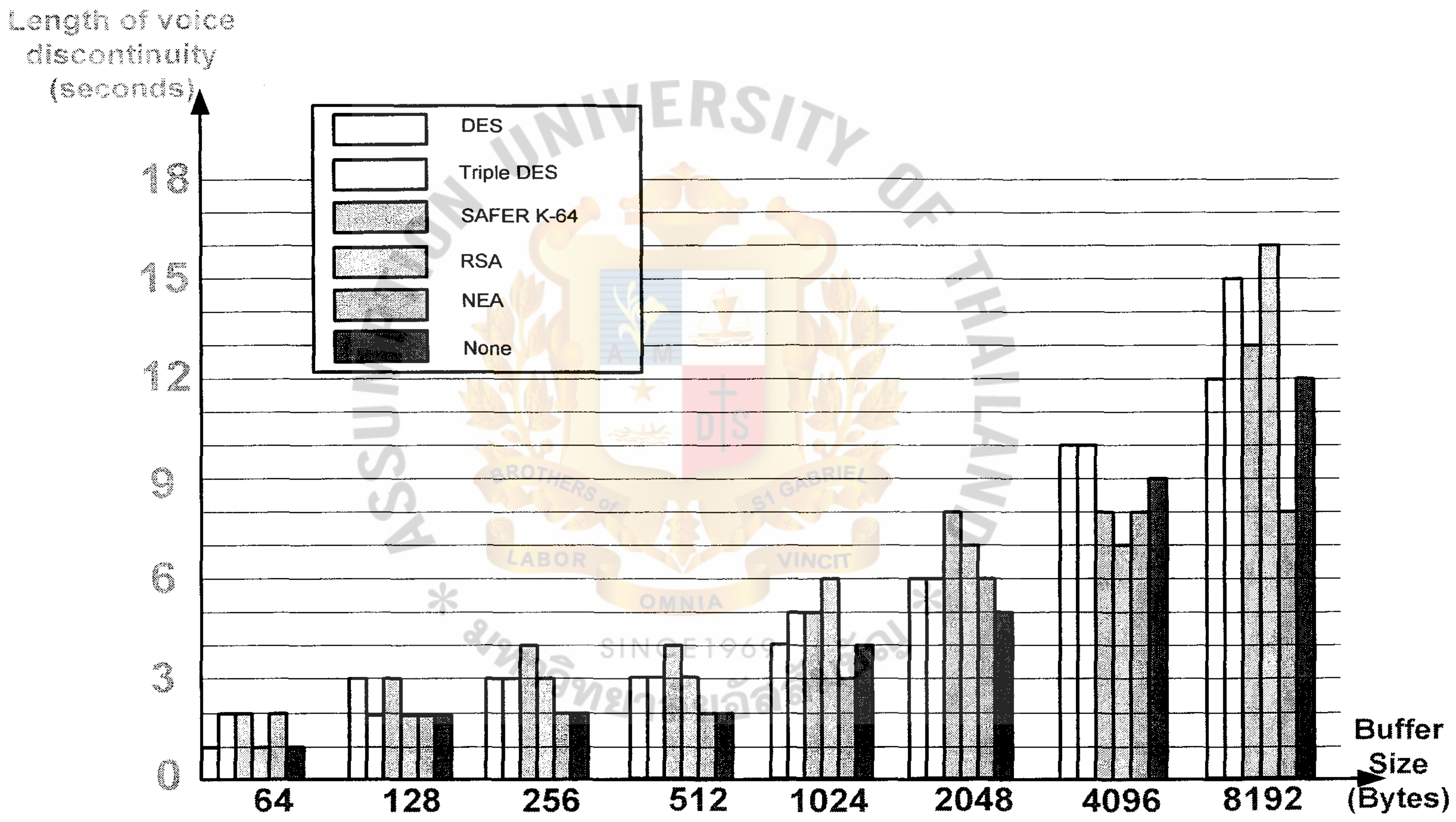


Table 4.2 Effect of buffer size for voice discontinuity with standard deviation result

Encryption / Buffer size (Bytes)	DES			Triple DES			SAFER K-64			RSA			NEA			None		
	Voice quality	Times	Seconds	Voice quality	Times	Seconds	Voice quality	Times	Seconds	Voice quality	Times	Seconds	Voice quality	Times	Seconds	Voice quality	Times	Seconds
64	0	0.78	0.81	0	0.45	0.61	0.48	0.31	0.68	0.72	0.41	0.88	0.57	0.44	0.76	0	0.57	0.57
128	0.24	0.44	0.65	0.49	0.47	0.72	0	0.38	0.47	0.47	0.41	0.60	0.38	0.44	0.78	0.51	0.50	0.76
256	0.14	1.44	0.61	0.73	0.40	0.61	0.50	0.57	1.42	0.57	0.41	0.89	0	0.44	1.08	0	0.55	0.78
512	0.64	0.34	0.61	0.79	0.89	0.89	0.49	0.59	0.89	0.51	0.39	0.76	0	0.41	0.78	0	0.47	0.86
1024	0.61	0.44	0.96	0.81	0.81	0.61	0.64	0.44	1.14	0.51	0.36	0.63	0	0.41	0.25	0	0.22	0.59
2048	0.48	0.67	0.86	0.54	0.82	0.31	0.55	0.72	0.44	0.76	0.43	0.42	0.39	0.63	0.79	0.75	0.76	0.59
4096	0.63	0.48	0.98	0.56	0.81	0.79	0.61	0.71	1.5	0.64	1.13	0.48	0.63	0.39	0.42	0.51	0.97	0.74
8192	0.52	0.65	0.44	0.44	0.50	0.93	0.71	0.49	0.79	0.60	0.74	0.95	0.39	0.50	0.45	0	0.85	0.82

Note: Number of discontinuity (times) = Times
Length of discontinuity (seconds) = Seconds

The results from 50 questionnaires show that the buffer size causes the voice discontinuity. A big buffer size causes much longer duration of voice discontinuity than a small buffer size, that is, a big buffer size has to collect packets until its limit but the small buffer size is collected fewer packets than a big buffer size then the duration of voice discontinuity is shorter. At 512-byte buffer size, the voice quality with encryptions and no encryption is almost very good. The voice quality with NEA is the best of all implemented encryptions. The turbulent noise depends on the way to develop software in each algorithm while the voice discontinuity depends on the buffer size. Finally, NEA is the best for voice over computer network when comparing with other implemented encryptions.

4.2 The security of voice/text chat over computer network

The security of voice/text chat in the developed program is designed that only two people can chat at a time, then the third person cannot use the program for listening the conversation. However, the hacker can detect packets flowing in network. In this experiment, the packet sniffer (Sniffer Pro version 4.70.04) is used to detect the data flow in network and it also has sniffer voice add-on module for decoding and analyzing voice over IP (VoIP) protocols. The text chat can be tested by using sniffer pro but voice chat cannot be tested by using sniffer voice since the sniffer voice provides only in specific VoIP protocols i.e. H.323, RTP, RTCP, SIP, Cisco's SCCP version3, MGCP and SAP/SDP, not TCP/IP used in developing voice/text chat over TCP/IP (the presented software).

To test the security of voice chat, the program is modified, that is, no server for authenticating user name-and-password login and multiuser can chat at a time.

There is the third person who wants to chat but he doesn't know encryption type to decrypt voice during conversation of couple people. The results for mismatch encryption during conversation are that the outcome voice is "Sue Sue"; therefore he doesn't know the conversation.

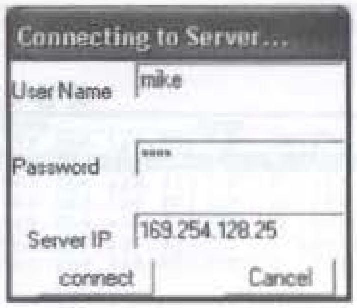
In testing the security of text chat with the developed software, the sniffer pro is then set to filter TCP/IP. There are 2 communicating sessions in the software i.e. between server and user and between user (caller) and user (receiver). Then, the test has 2 parts:

4.2.1 Between server and user.

The testing software is designed so that user must firstly login the software by passing through server with DES encryption for protect his user name-and-password login. In this test,

server's IP address	=	169.254.128.25 with 255.255.0.0 subnet mask
user's IP address	=	169.254.128.3 with 255.255.0.0 subnet mask.
user name	=	mike
password	=	1234

User logins the software (figure 4.3) and then server receives the login message (figure 4.4). The sniffer is used to detect user name and password, but the result is ciphertext due to DES encryption shown in figure 4.5. "mike", "1234" and "169.254.128.25" is detected by sniffer shown in figure 4.5 (1), (2) and (3) respectively.

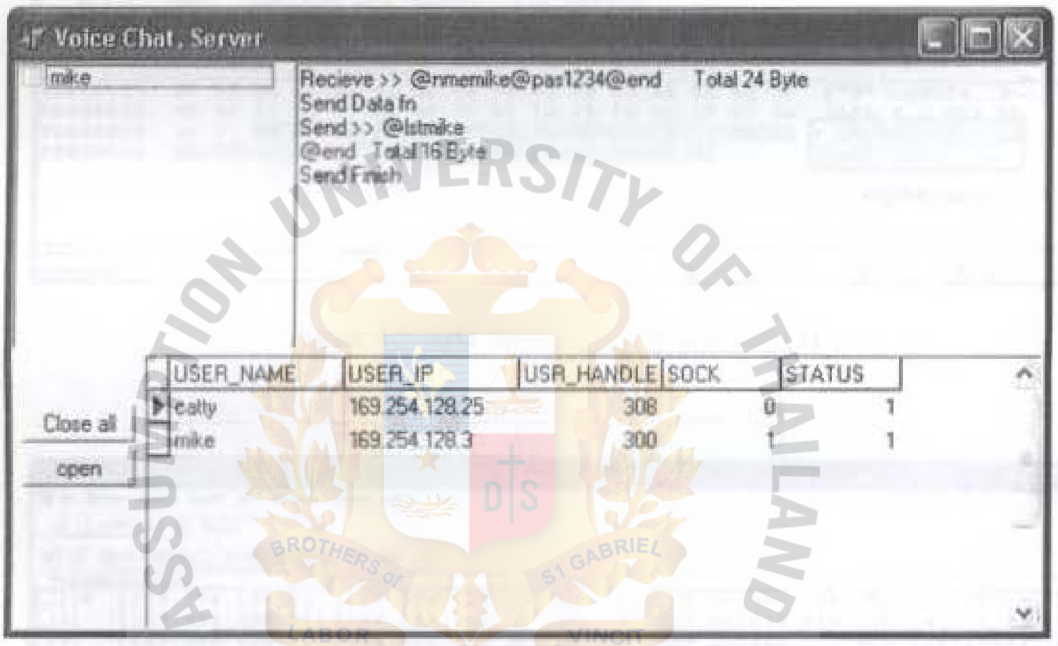


User Name: mike

Password: ****

Server IP: 169.254.128.25

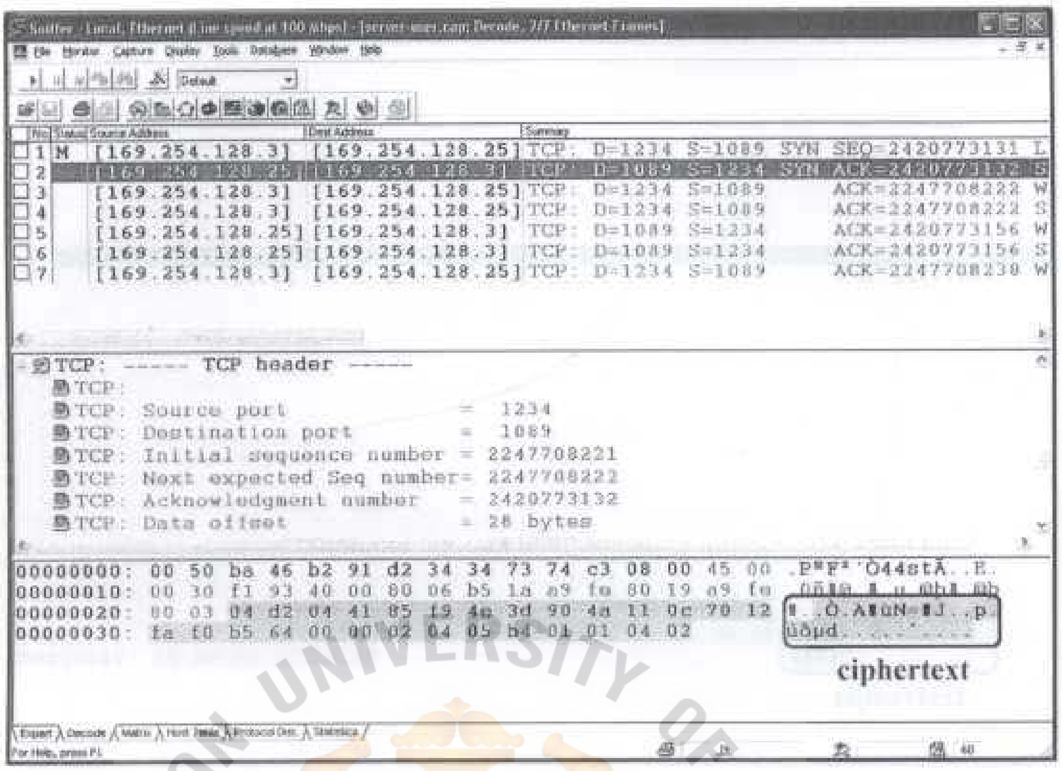
Figure 4.3 User logins the software.



```
Recieve >> @nmemike@pas1234@end     Total 24 Byte
Send Data fn
Send >> @lstmike
@end  Total 16 Byte
Send Finish
```

USER_NAME	USER_IP	USR_HANDLE	SOCK	STATUS
catty	169.254.128.25	308	0	1
mike	169.254.128.3	300	1	1

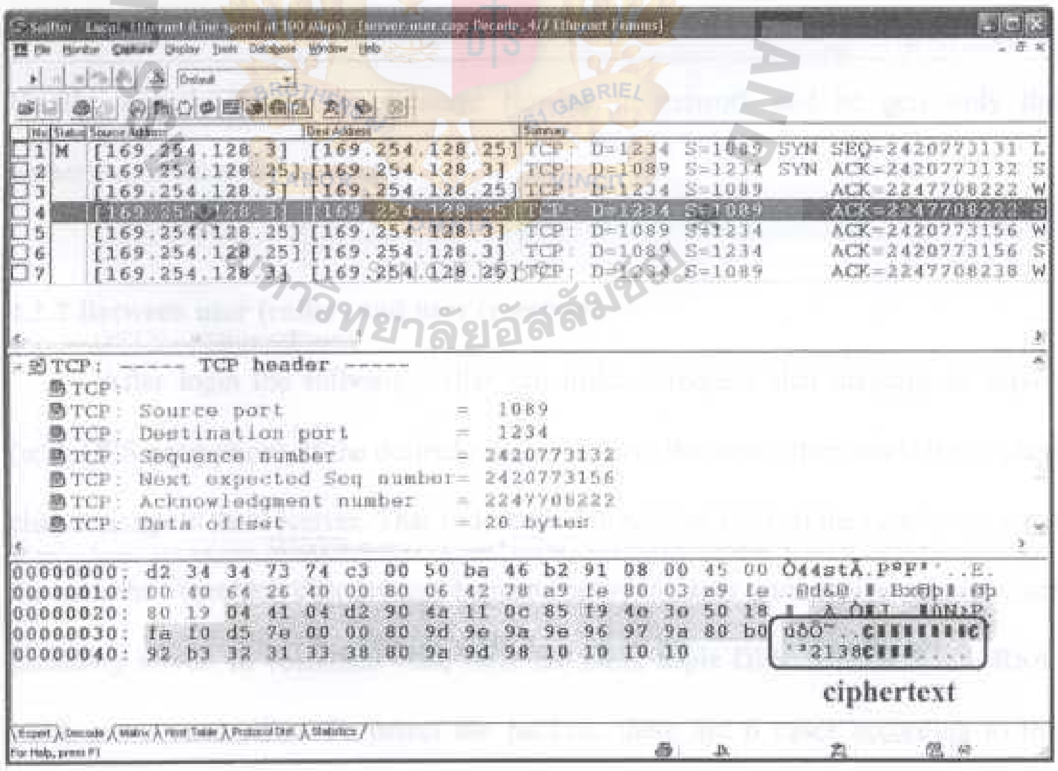
Figure 4.4 Server receives user login message.



```
Sniffer - Local, Ethernet (Line speed at 100 Mbps) - [server-user.cap: Decode, 2/7 Ethernet Frames]

No Status Source Address      Dest Address          Summary
 1  M    [169.254.128.3]   [169.254.128.25] TCP: D=1234 S=1089 SYN SEQ=2420773131 L
 2       [169.254.128.25]  [169.254.128.3]  TCP: D=1089 S=1234 SYN ACK=2420773132 S
 3       [169.254.128.3]   [169.254.128.25] TCP: D=1234 S=1089     ACK=2247708222 W
 4       [169.254.128.3]   [169.254.128.25] TCP: D=1234 S=1089     ACK=2247708222 S
 5       [169.254.128.25]  [169.254.128.3]  TCP: D=1089 S=1234     ACK=2420773156 W
 6       [169.254.128.25]  [169.254.128.3]  TCP: D=1089 S=1234     ACK=2420773156 S
 7       [169.254.128.3]   [169.254.128.25] TCP: D=1234 S=1089     ACK=2247708238 W

TCP: ----- TCP header -----
  TCP:
  TCP: Source port               =  1234
  TCP: Destination port          =  1089
  TCP: Initial sequence number   = 2247708221
  TCP: Next expected Seq number= 2247708222
  TCP: Acknowledgment number     = 2420773132
  TCP: Data offset               = 28 bytes

00000000: 00 50 ba 46 b2 91 d2 34 34 73 74 c3 08 00 45 00  .P°F²'Ò44stÃ..E.
00000010: 00 30 f1 93 40 00 80 06 b5 1a a9 fe 80 19 a9 fe  .0ñ.@...µ.©þ..©þ
00000020: 00 03 04 d2 04 41 85 f9 4e 3d 90 4a 11 0c 70 12  ...Ò.A.ùN=.J..p.
00000030: fa f0 b5 64 00 00 02 04 05 b4 01 01 04 02        úðµd..........
```

ciphertext

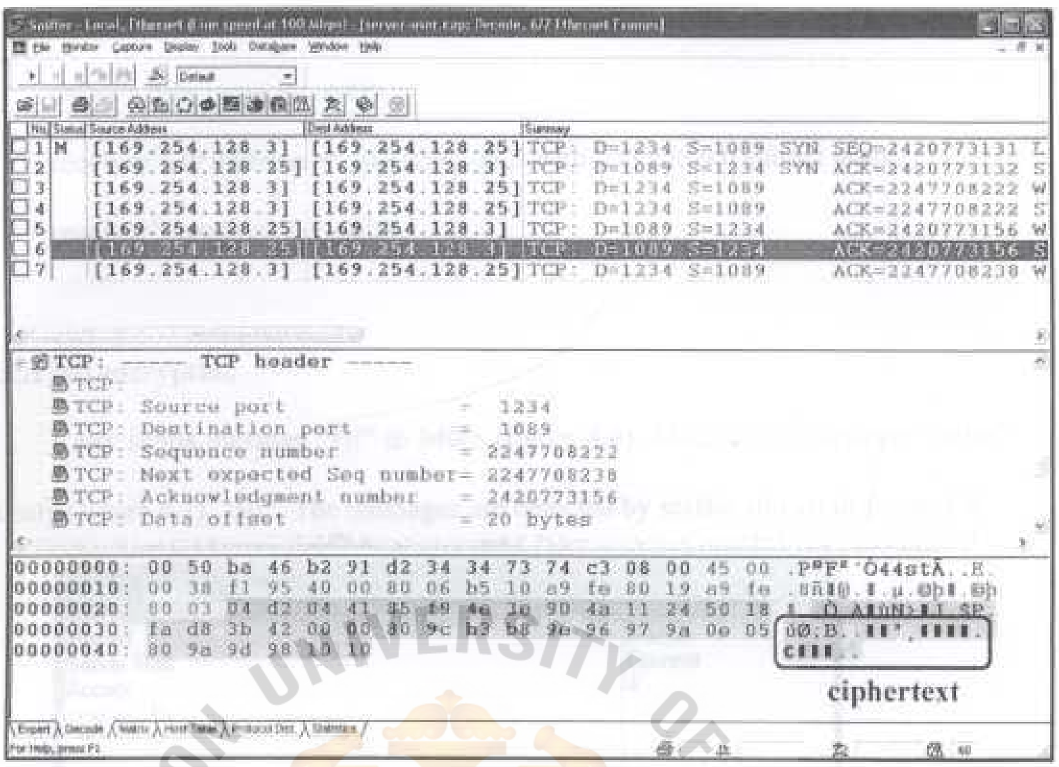
Figure 4.5 data between server and user (1)



```
Sniffer - Local, Ethernet (Line speed at 100 Mbps) - [server-user.cap: Decode, 4/7 Ethernet Frames]

No Status Source Address      Dest Address          Summary
 1  M    [169.254.128.3]   [169.254.128.25] TCP: D=1234 S=1089 SYN SEQ=2420773131 L
 2       [169.254.128.25]  [169.254.128.3]  TCP: D=1089 S=1234 SYN ACK=2420773132 S
 3       [169.254.128.3]   [169.254.128.25] TCP: D=1234 S=1089     ACK=2247708222 W
 4       [169.254.128.3]   [169.254.128.25] TCP: D=1234 S=1089     ACK=2247708222 S
 5       [169.254.128.25]  [169.254.128.3]  TCP: D=1089 S=1234     ACK=2420773156 W
 6       [169.254.128.25]  [169.254.128.3]  TCP: D=1089 S=1234     ACK=2420773156 S
 7       [169.254.128.3]   [169.254.128.25] TCP: D=1234 S=1089     ACK=2247708238 W

TCP: ----- TCP header -----
  TCP:
  TCP: Source port               =  1089
  TCP: Destination port          =  1234
  TCP: Sequence number           = 2420773132
  TCP: Next expected Seq number= 2420773156
  TCP: Acknowledgment number     = 2247708222
  TCP: Data offset               = 20 bytes

00000000: d2 34 34 73 74 c3 00 50 ba 46 b2 91 08 00 45 00  Ò44stÃ.P°F²'..E.
00000010: 00 40 64 26 40 00 80 06 42 78 a9 fe 80 03 a9 fe  .@d&@...Bx©þ..©þ
00000020: 80 19 04 41 04 d2 90 4a 11 0c 85 f9 4e 3e 50 18  ...A.Ò.J...ùN>P.
00000030: fa f0 d5 7e 00 00 80 9d 9e 9a 9e 96 97 9a 80 b0  úðÕ~............
00000040: 92 b3 32 31 33 38 80 9a 9d 98 10 10 10 10        '³2138........
```

ciphertext

Figure 4.5 data between server and user (2)



```
No Status Source Address      Dest Address       Summary
1  M   [169.254.128.3]  [169.254.128.25] TCP: D=1234 S=1089 SYN SEQ=2420773131 L
2      [169.254.128.25] [169.254.128.3]  TCP: D=1089 S=1234 SYN ACK=2420773132 S
3      [169.254.128.3]  [169.254.128.25] TCP: D=1234 S=1089     ACK=2247708222 W
4      [169.254.128.3]  [169.254.128.25] TCP: D=1234 S=1089     ACK=2247708222 S
5      [169.254.128.25] [169.254.128.3]  TCP: D=1089 S=1234     ACK=2420773156 W
6      [169.254.128.25] [169.254.128.3]  TCP: D=1089 S=1234     ACK=2420773156 S
7      [169.254.128.3]  [169.254.128.25] TCP: D=1234 S=1089     ACK=2247708238 W
```

```
TCP: ----- TCP header -----
  TCP:
  TCP: Source port              =  1234
  TCP: Destination port         =  1089
  TCP: Sequence number          = 2247708222
  TCP: Next expected Seq number= 2247708238
  TCP: Acknowledgment number    = 2420773156
  TCP: Data offset              = 20 bytes
```

```
00000000: 00 50 ba 46 b2 91 d2 34 34 73 74 c3 08 00 45 00  .PºF² Ò44stÃ..E.
00000010: 00 38 f1 95 40 00 80 06 b5 10 a9 fe 80 19 a9 fe  .8ñ■@.■.µ.©þ■.©þ
00000020: 80 03 04 d2 04 41 85 f9 4e 3e 90 4a 11 24 50 18  ■..Ò.A■ùN>■J.$P.
00000030: fa d8 3b 42 00 00 80 9c b3 b8 9e 96 97 9a 0e 05  úØ;B..■■³¸■■■■..
00000040: 80 9a 9d 98 10 10                                ■■■■..
```

Figure 4.5 data between server and user (3)

From the sniffer pro, the results show the ciphertext of 'mike' in Figure 4.5 (1), the ciphertexts of '1234' in Figure 4.5 (2) and '169.254.128.25' in Figure 4.5 (3). Finally, nobody knows the message flowing in network and he gets only the ciphertext or unknown messages.

4.2.2 Between user (caller) and user (receiver)

After login the software, caller can make a request chat message to server (with DES encryption). If the desired receiver is free, the server then sends the request chat message to that receiver. That is described in section 4.3.1. If the receiver accepts the call, the voice/text chat process happens between caller and receiver directly, not including server. In voice/text chat, there are DES, triple DES, SAFER K-64, RSA, NEA or no encryption. To detect the packets, there are 6 cases according to the developed encryptions with caller's IP address = 169.254.128.25 with 255.255.0.0 subnet mask.

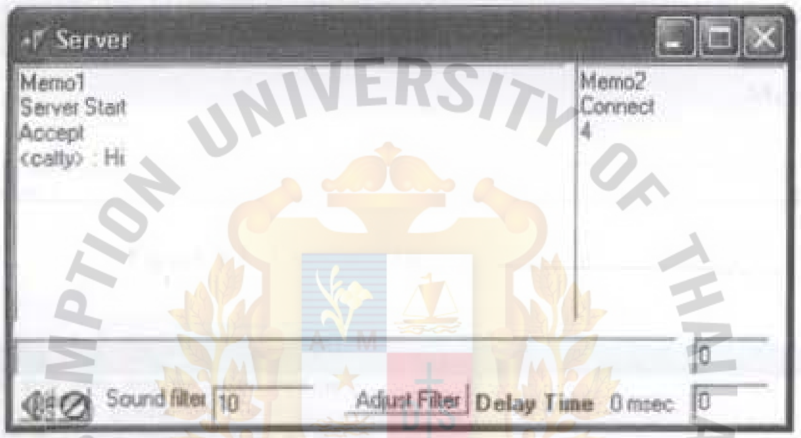
caller's user name = catty

receiver's IP address = 169.254.128.3 with 255.255.0.0 subnet mask.

receiver's user name = mike

4.2.2.1 No encryption

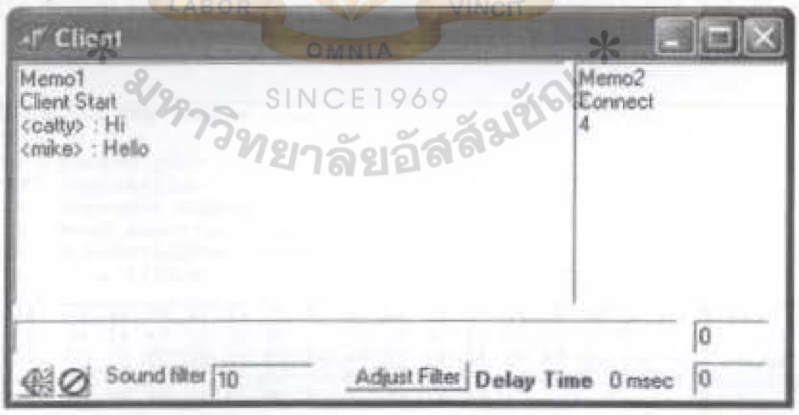
Catty sends message "Hi" to Mike (figure 4.6). Mike sends message "Hello" to Catty (figure 4.7). "Hi". The messages are detected by sniffer shown in figure 4.8.


Server

Memo1
Server Start
Accept
<catty> : Hi

Memo2
Connect
4

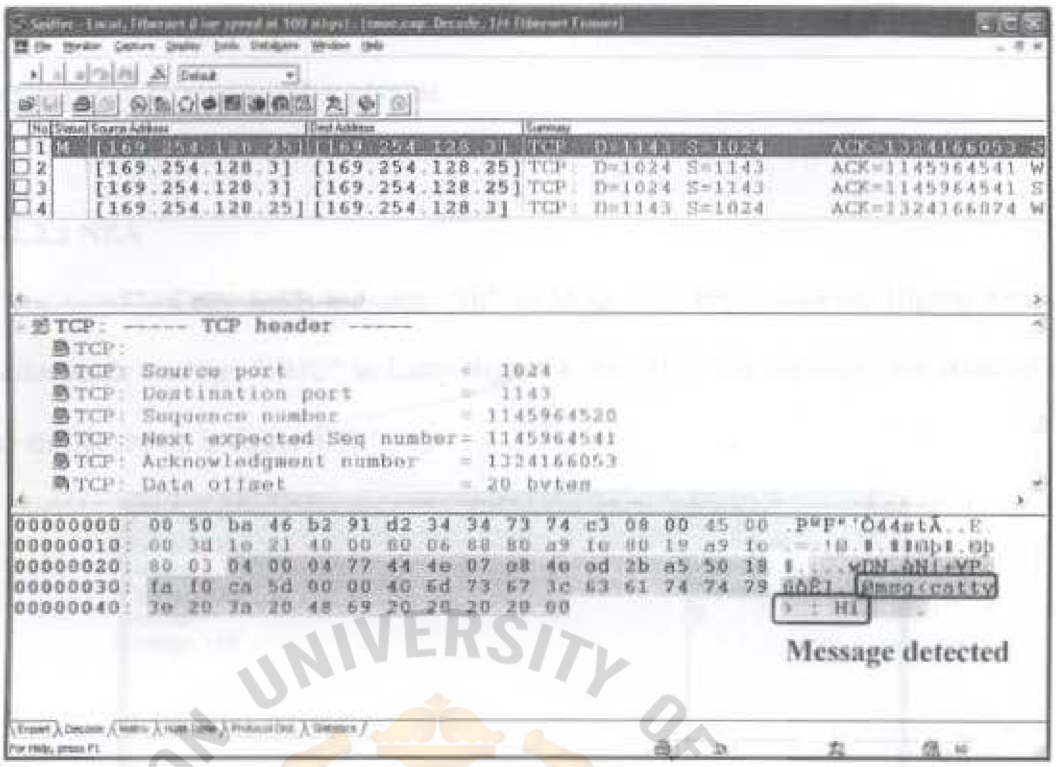
Figure 4.6 Catty sends message "Hi" to Mike with none of encryption


Client

Memo1
Client Start
<catty> : Hi
<mike> : Hello

Memo2
Connect
4

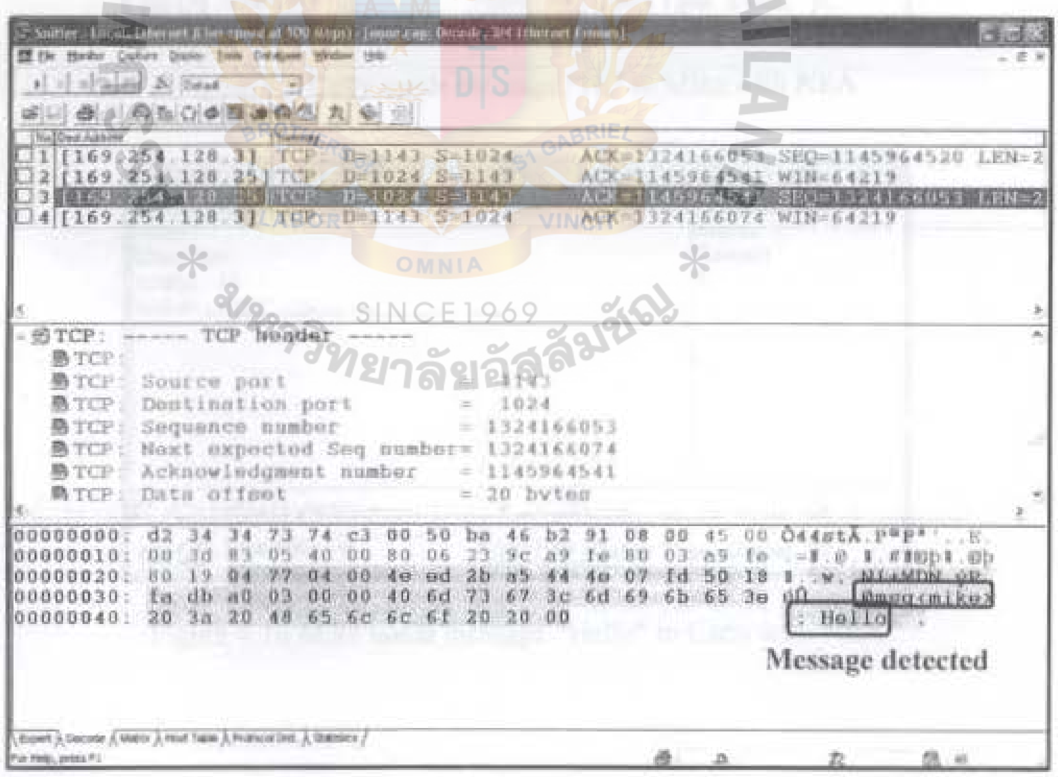
Figure 4.7 Mike sends message "Hello" to Catty with none of encryption



```
No Status Source Address      Dest Address        Summary
1  M     [169.254.128.25] [169.254.128.3]  TCP: D=1143 S=1024     ACK=1324166053 S
2        [169.254.128.3]  [169.254.128.25] TCP: D=1024 S=1143     ACK=1145964541 W
3        [169.254.128.3]  [169.254.128.25] TCP: D=1024 S=1143     ACK=1145964541 S
4        [169.254.128.25] [169.254.128.3]  TCP: D=1143 S=1024     ACK=1324166074 W

TCP: ----- TCP header -----
   TCP:
   TCP: Source port              =  1024
   TCP: Destination port         =  1143
   TCP: Sequence number          = 1145964520
   TCP: Next expected Seq number= 1145964541
   TCP: Acknowledgment number    = 1324166053
   TCP: Data offset              = 20 bytes

00000000: 00 50 ba 46 b2 91 d2 34 34 73 74 c3 08 00 45 00  .P°F²'Ò44stÃ..E.
00000010: 00 3d 1e 21 40 00 80 06 88 80 a9 fe 80 19 a9 fe  .=.!@.€.ˆ€©þ€.©þ
00000020: 80 03 04 00 04 77 44 4e 07 e8 4e ed 2b a5 50 18  €....wDN.èNí+¥P.
00000030: fa f0 ca 5d 00 00 40 6d 73 67 3c 63 61 74 74 79  úðÊ]..@msg<catty
00000040: 3e 20 3a 20 48 69 20 20 20 20 00                 > : Hi    .
```

Message detected

Figure 4.8 The result messages with none of encryption (1)



```
No Dest Address      Summary
1 [169.254.128.3]  TCP: D=1143 S=1024     ACK=1324166053 SEQ=1145964520 LEN=2
2 [169.254.128.25] TCP: D=1024 S=1143     ACK=1145964541 WIN=64219
3 [169.254.128.25] TCP: D=1024 S=1143     ACK=1145964541 SEQ=1324166053 LEN=2
4 [169.254.128.3]  TCP: D=1143 S=1024     ACK=1324166074 WIN=64219

TCP: ----- TCP header -----
   TCP:
   TCP: Source port              =  1143
   TCP: Destination port         =  1024
   TCP: Sequence number          = 1324166053
   TCP: Next expected Seq number= 1324166074
   TCP: Acknowledgment number    = 1145964541
   TCP: Data offset              = 20 bytes

00000000: d2 34 34 73 74 c3 00 50 ba 46 b2 91 08 00 45 00  Ò44stÃ.P°F²'..E.
00000010: 00 3d 83 05 40 00 80 06 23 9c a9 fe 80 03 a9 fe  .=ƒ.@.€.#œ©þ€.©þ
00000020: 80 19 04 77 04 00 4e ed 2b a5 44 4e 07 fd 50 18  €..w..Ní+¥DN.ýP.
00000030: fa db a0 03 00 00 40 6d 73 67 3c 6d 69 6b 65 3e  úÛ ...@msg<mike>
00000040: 20 3a 20 48 65 6c 6c 6f 20 20 00                  : Hello  .
```

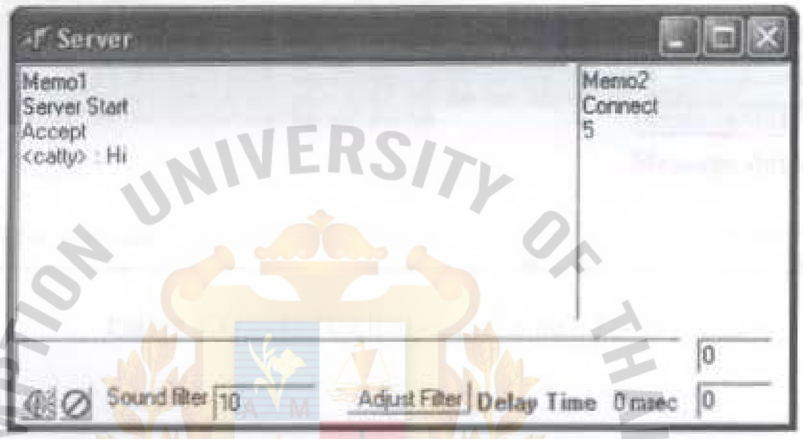
Message detected

Figure 4.8 The result messages with none of encryption (2)

From Figure 4.8, the sniffer pro detects message with no encryption then the message can be read by other persons.

4.2.2.2 NEA

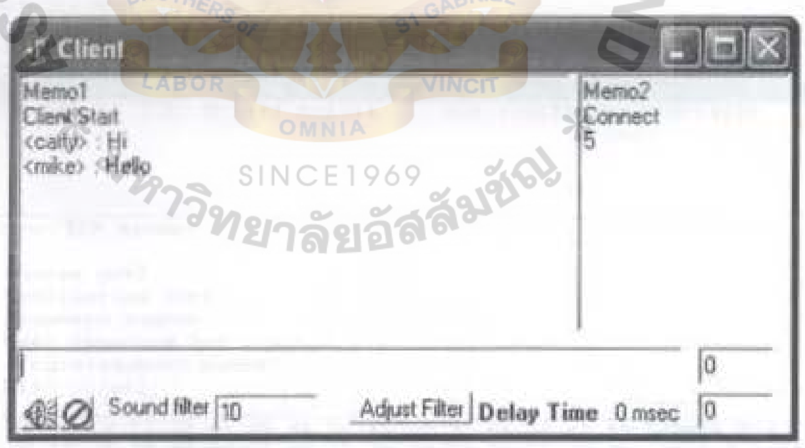
Catty sends message "Hi" to Mike with key: password (figure 4.9). Mike sends message "Hello" to Catty (figure 4.10). "Hi". The messages are detected by sniffer shown in figure 4.11.



```
Memo1
Server Start
Accept
<catty> : Hi
```

```
Memo2
Connect
5
```

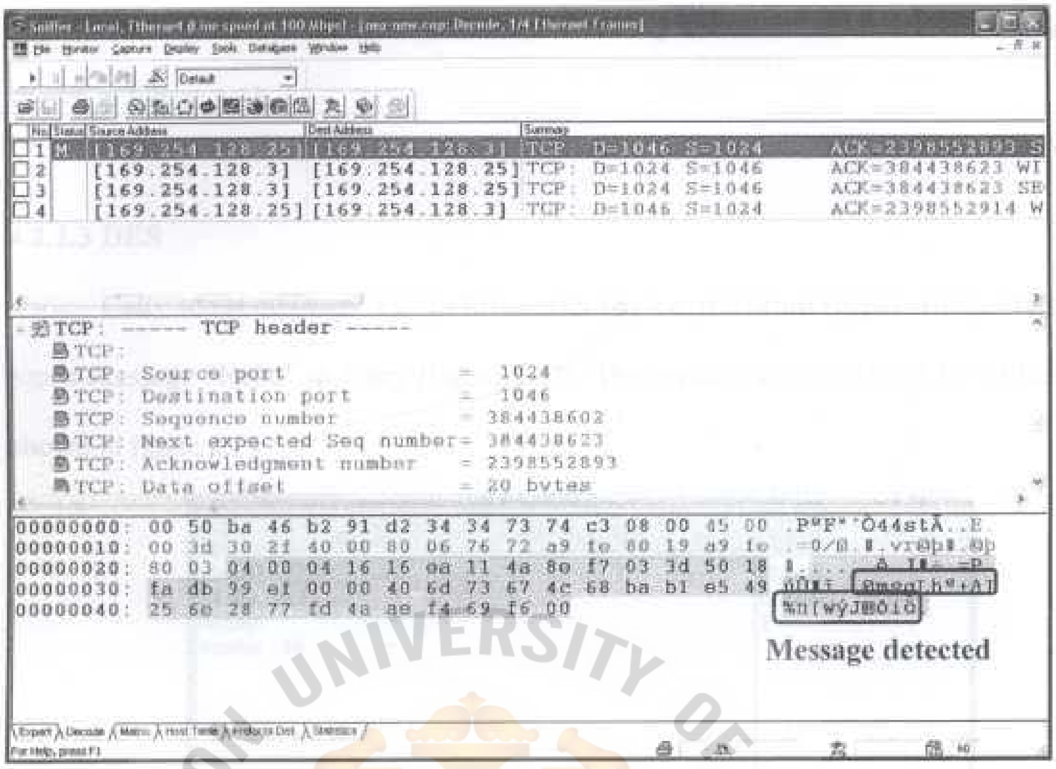
Figure 4.9 Catty sends message "Hi" to Mike with NEA



```
Memo1
Client Start
<catty> : Hi
<mike> : Hello
```

```
Memo2
Connect
5
```

Figure 4.10 Mike sends message "Hello" to Catty with NEA

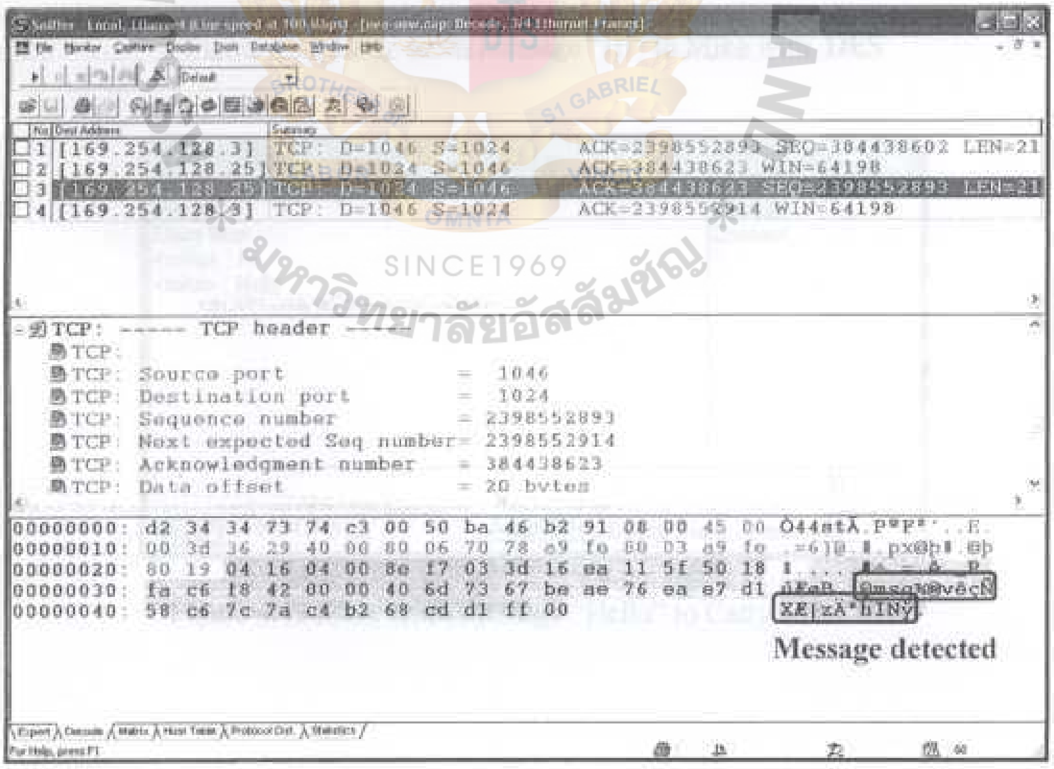


```
No Status Source Address     Dest Address         Summary
1  M.  [169.254.128.25] [169.254.128.3]  TCP: D=1046 S=1024    ACK=2398552893 S
2      [169.254.128.3]  [169.254.128.25] TCP: D=1024 S=1046    ACK=384438623 WI
3      [169.254.128.3]  [169.254.128.25] TCP: D=1024 S=1046    ACK=384438623 SE
4      [169.254.128.25] [169.254.128.3]  TCP: D=1046 S=1024    ACK=2398552914 W
```

```
-TCP: ----- TCP header -----
 TCP:
 TCP: Source port              = 1024
 TCP: Destination port         = 1046
 TCP: Sequence number          = 384438602
 TCP: Next expected Seq number= 384438623
 TCP: Acknowledgment number    = 2398552893
 TCP: Data offset              = 20 bytes
```

```
00000000: 00 50 ba 46 b2 91 d2 34 34 73 74 c3 08 00 45 00 .PºF²'Ò44stÃ..E.
00000010: 00 3d 30 2f 40 00 80 06 76 72 a9 fe 80 19 a9 fe .=0/@.€.vr©þ€.©þ
00000020: 80 03 04 00 04 16 16 ea 11 4a 8e f7 03 3d 50 18 €......ê.J.÷.=P.
00000030: fa db 99 e1 00 00 40 6d 73 67 4c 68 ba b1 e5 49 úÛ™á..@msgLhº±åI
00000040: 25 6e 28 77 fd 4a ae f4 69 f6 00                %n(wýJ®ôiö.
```

Figure 4.11 The result messages with NEA (1)



```
No Dest Address       Summary
1  [169.254.128.3]  TCP: D=1046 S=1024    ACK=2398552893 SEQ=384438602 LEN=21
2  [169.254.128.25] TCP: D=1024 S=1046    ACK=384438623 WIN=64198
3  [169.254.128.25] TCP: D=1024 S=1046    ACK=384438623 SEQ=2398552893 LEN=21
4  [169.254.128.3]  TCP: D=1046 S=1024    ACK=2398552914 WIN=64198
```

```
-TCP: ----- TCP header -----
 TCP:
 TCP: Source port              = 1046
 TCP: Destination port         = 1024
 TCP: Sequence number          = 2398552893
 TCP: Next expected Seq number= 2398552914
 TCP: Acknowledgment number    = 384438623
 TCP: Data offset              = 20 bytes
```

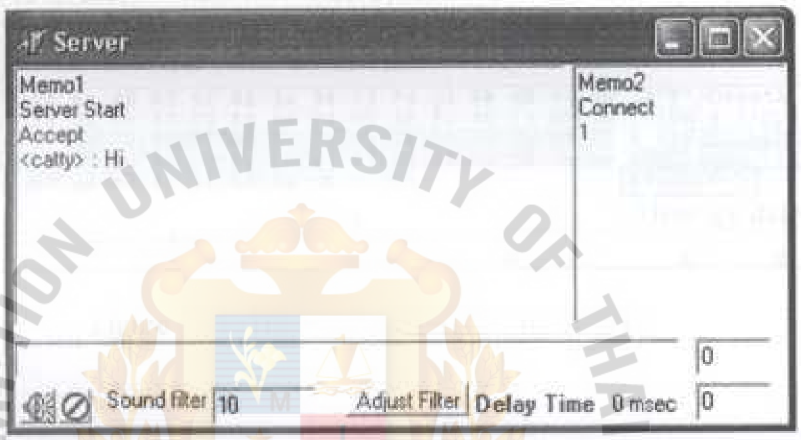
```
00000000: d2 34 34 73 74 c3 00 50 ba 46 b2 91 08 00 45 00 Ò44stÃ.PºF²'..E.
00000010: 00 3d 36 29 40 00 80 06 70 78 a9 fe 80 03 a9 fe .=6)@.€.px©þ€.©þ
00000020: 80 19 04 16 04 00 8e f7 03 3d 16 ea 11 5f 50 18 €......÷.=.ê._P.
00000030: fa c6 18 42 00 00 40 6d 73 67 be ae 76 ea e7 d1 úÆ.B..@msg¾®vêçÑ
00000040: 58 c6 7c 7a c4 b2 68 cd d1 ff 00                XÆ|zÄ²hÍÑÿ.
```

Figure 4.11 The result messages with NEA (2)

With NEA, the sniffer pro detects the ciphertext or unknown messages shown in Figure 4.11.

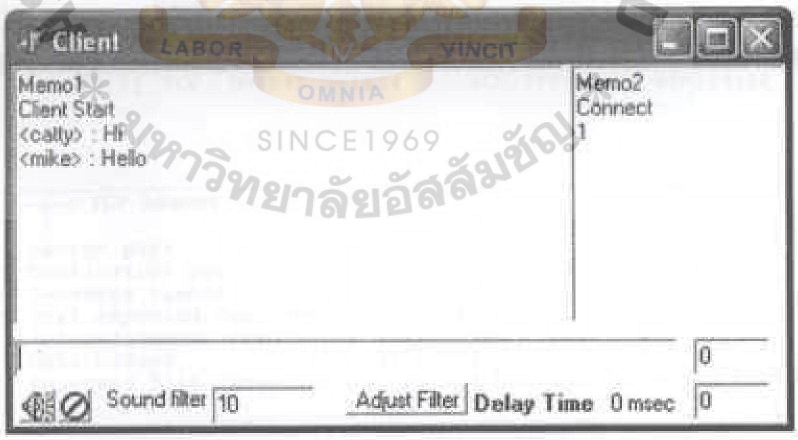
4.2.2.3 DES

Catty sends message "Hi" to Mike with key i.e. password (figure 4.12). Mike sends message "Hello" to Catty (figure 4.13). The messages are detected by sniffer shown in figure 4.14.



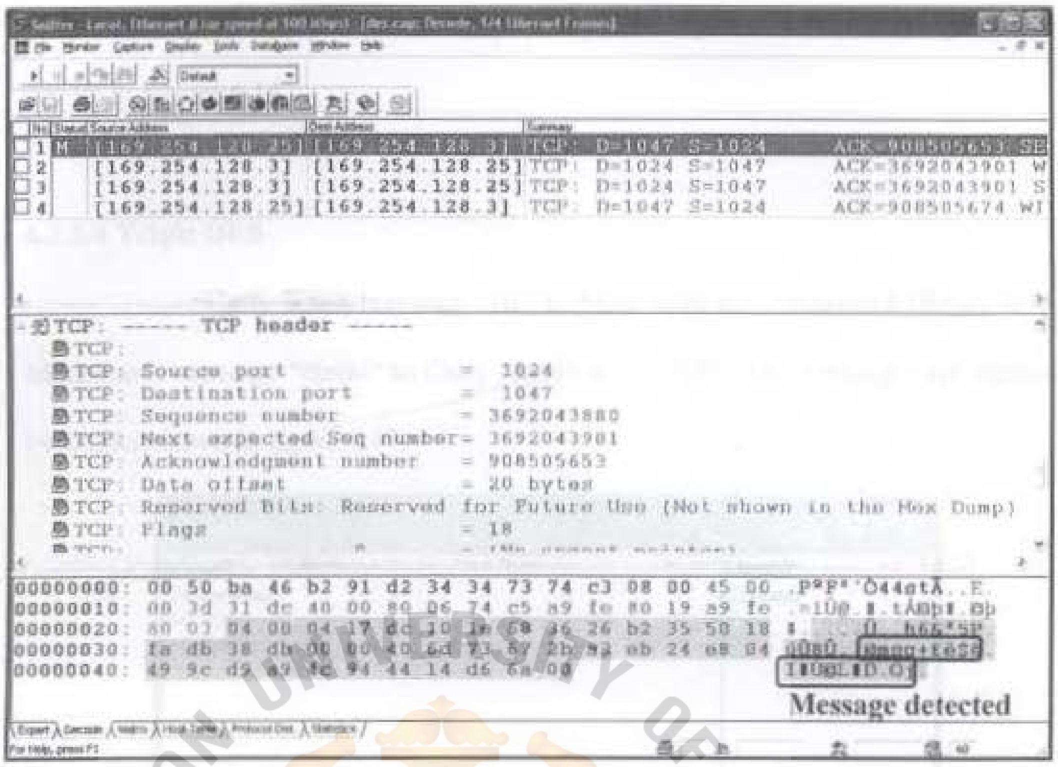
```
Memo1
Server Start
Accept
<catty> : Hi
```

Figure 4.12 Catty sends message "Hi" to Mike with DES



```
Memo1
Client Start
<catty> : Hi
<mike> : Hello
```

Figure 4.13 Mike sends message "Hello" to Catty with DES



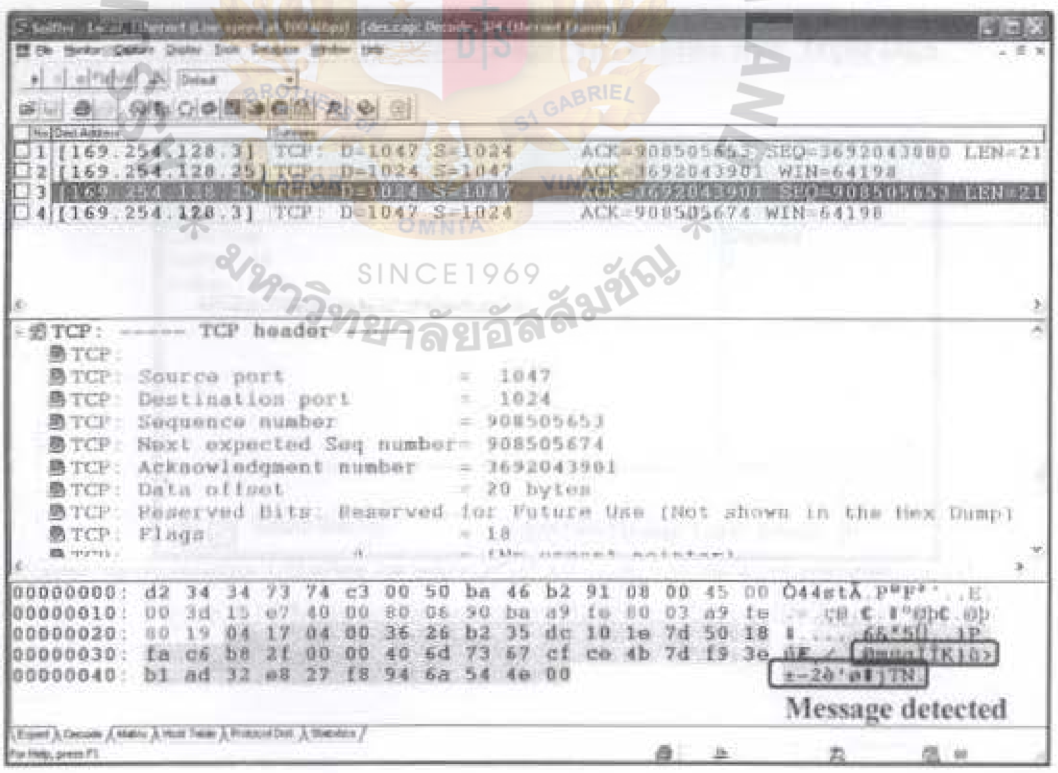
```
No Status Source Address       Dest Address          Summary
1  M  [169.254.128.25] [169.254.128.3]  TCP: D=1047 S=1024    ACK=908505653 SE
2     [169.254.128.3]  [169.254.128.25] TCP: D=1024 S=1047    ACK=3692043901 W
3     [169.254.128.3]  [169.254.128.25] TCP: D=1024 S=1047    ACK=3692043901 S
4     [169.254.128.25] [169.254.128.3]  TCP: D=1047 S=1024    ACK=908505674 WI
```

```
TCP: ----- TCP header -----
TCP:
TCP: Source port                = 1024
TCP: Destination port           = 1047
TCP: Sequence number            = 3692043880
TCP: Next expected Seq number= 3692043901
TCP: Acknowledgment number      = 908505653
TCP: Data offset                = 20 bytes
TCP: Reserved Bits: Reserved for Future Use (Not shown in the Hex Dump)
TCP: Flags                      = 18
```

```
00000000: 00 50 ba 46 b2 91 d2 34 34 73 74 c3 08 00 45 00  .P°F²'Ò44stÃ..E.
00000010: 00 3d 31 de 40 00 80 06 74 c5 a9 fe 80 19 a9 fe  .=1Þ@.€.tÅ©þ€.©þ
00000020: 80 03 04 00 04 17 dc 10 1e 68 36 26 b2 35 50 18  €.....Ü..h6&²5P.
00000030: fa db 38 db 00 00 40 6d 73 67 2b 92 eb 24 e8 04  úÛ8Û..@msg+'ë$è.
00000040: 49 9c d9 a9 fc 94 44 14 d6 6a 00                 I.Ù©ü.D.Öj.
```

Message detected

Figure 4.14 The result messages with DES (1)



```
No Dest Address       Summary
1 [169.254.128.3]  TCP: D=1047 S=1024    ACK=908505653 SEQ=3692043880 LEN=21
2 [169.254.128.25] TCP: D=1024 S=1047    ACK=3692043901 WIN=64198
3 [169.254.128.25] TCP: D=1024 S=1047    ACK=3692043901 SEQ=908505653 LEN=21
4 [169.254.128.3]  TCP: D=1047 S=1024    ACK=908505674 WIN=64198
```

```
TCP: ----- TCP header -----
TCP:
TCP: Source port                = 1047
TCP: Destination port           = 1024
TCP: Sequence number            = 908505653
TCP: Next expected Seq number= 908505674
TCP: Acknowledgment number      = 3692043901
TCP: Data offset                = 20 bytes
TCP: Reserved Bits: Reserved for Future Use (Not shown in the Hex Dump)
TCP: Flags                      = 18
```

```
00000000: d2 34 34 73 74 c3 00 50 ba 46 b2 91 08 00 45 00  Ò44stÃ.P°F²'..E.
00000010: 00 3d 15 e7 40 00 80 06 90 ba a9 fe 80 03 a9 fe  .=.ç@.€..º©þ€.©þ
00000020: 80 19 04 17 04 00 36 26 b2 35 dc 10 1e 7d 50 18  €.....6&²5Ü..}P.
00000030: fa c6 b8 2f 00 00 40 6d 73 67 cf ce 4b 7d f9 3e  úÆ¸/..@msgÏÎK}ù>
00000040: b1 ad 32 e8 27 f8 94 6a 54 4e 00                 ±-2è'ø.jTN.
```

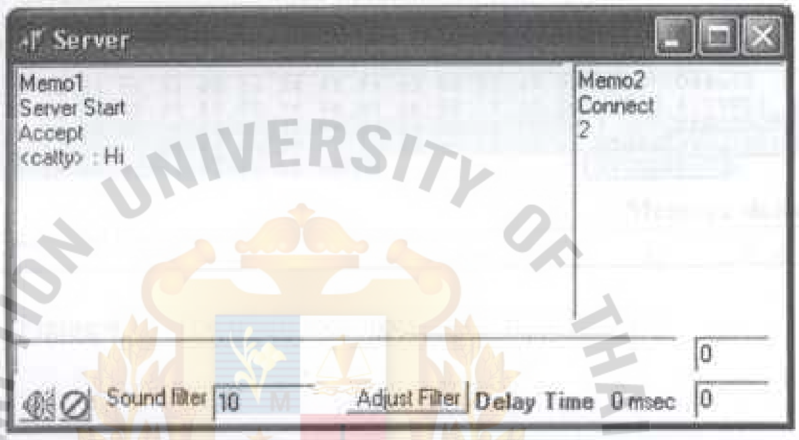
Message detected

Figure 4.14 The result messages with DES (2)

Similarly to NEA, the sniffer pro detects the ciphertext or unknown messages shown in Figure 4.14.

4.2.2.4 Triple DES

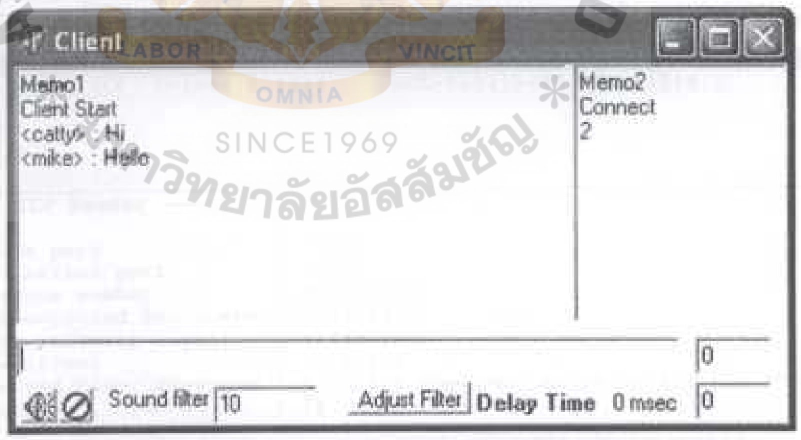
Catty sends message "Hi" to Mike with key: password (figure 4.15). Mike sends message "Hello" to Catty (figure 4.16). "Hi". The messages are detected by sniffer shown in figure 4.17.



```
Memo1
Server Start
Accept
<catty> : Hi
```

```
Memo2
Connect
2
```

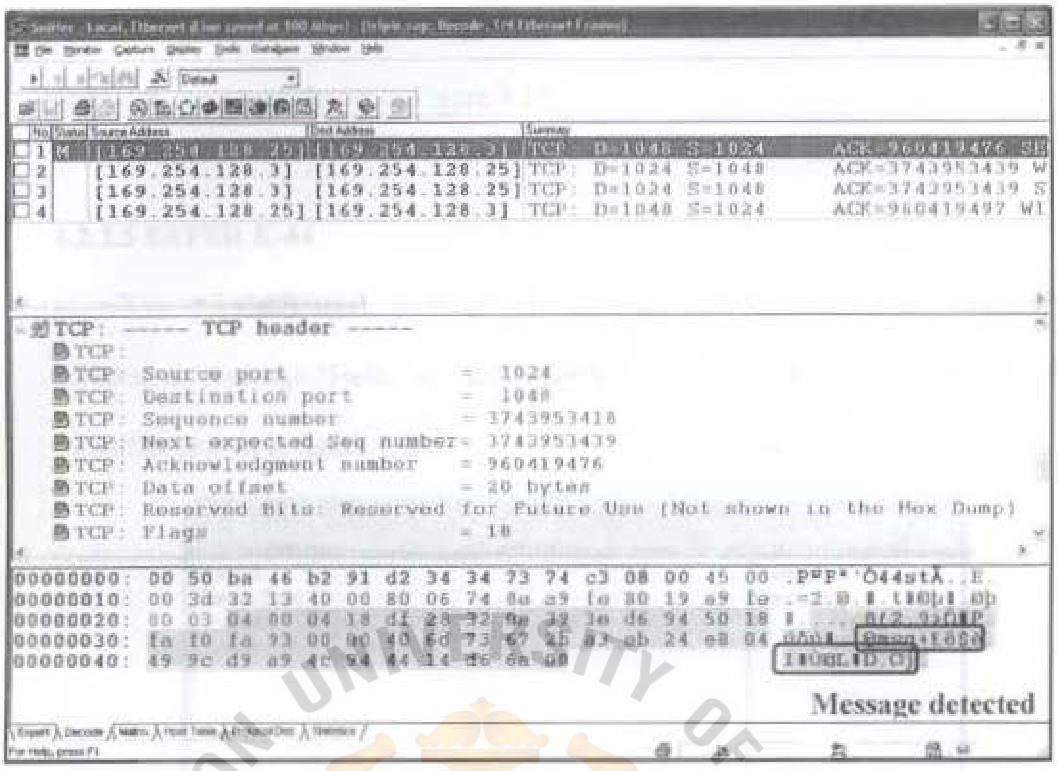
Figure 4.15 Catty sends message "Hi" to Mike with Triple DES



```
Memo1
Client Start
<catty> : Hi
<mike> : Hello
```

```
Memo2
Connect
2
```

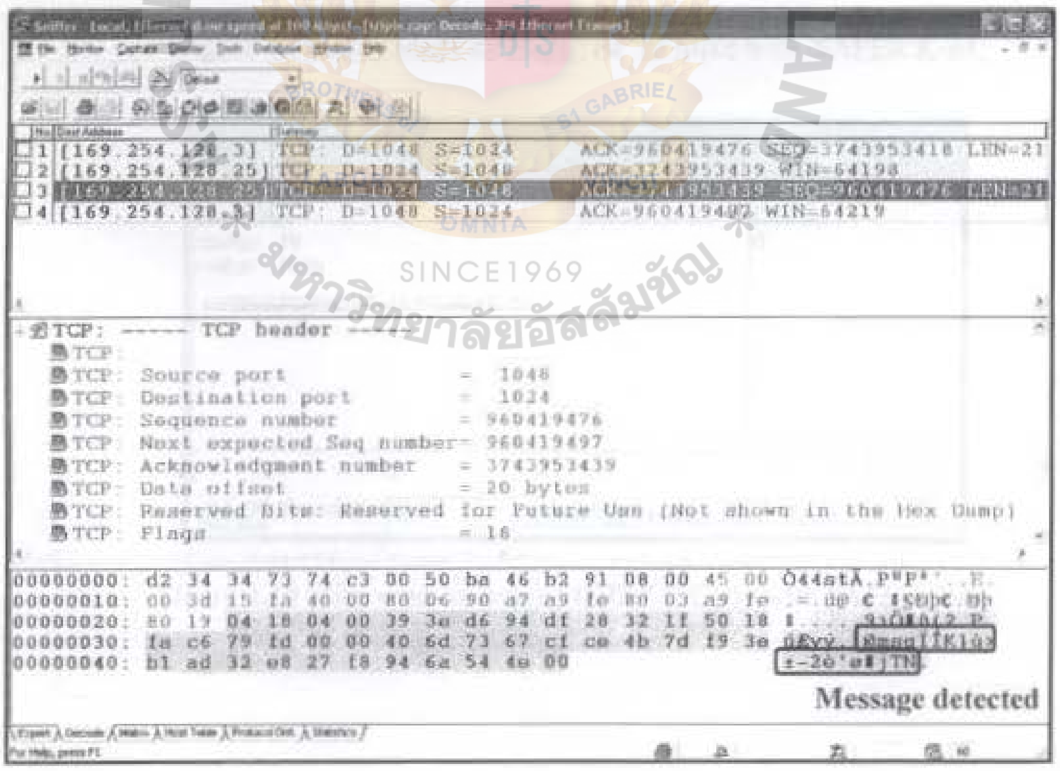
Figure 4.16 Mike sends message "Hello" to Catty with Triple DES



```
No Status Source Address       Dest Address          Summary
 1  M     [169.254.128.25]     [169.254.128.3]       TCP: D=1048 S=1024        ACK=960419476 SE
 2        [169.254.128.3]      [169.254.128.25]      TCP: D=1024 S=1048        ACK=3743953439 W
 3        [169.254.128.3]      [169.254.128.25]      TCP: D=1024 S=1048        ACK=3743953439 S
 4        [169.254.128.25]     [169.254.128.3]       TCP: D=1048 S=1024        ACK=960419497 WI

TCP: ----- TCP header -----
    TCP:
    TCP: Source port             = 1024
    TCP: Destination port        = 1048
    TCP: Sequence number         = 3743953418
    TCP: Next expected Seq number= 3743953439
    TCP: Acknowledgment number   = 960419476
    TCP: Data offset             = 20 bytes
    TCP: Reserved Bits: Reserved for Future Use (Not shown in the Hex Dump)
    TCP: Flags                   = 18

00000000: 00 50 ba 46 b2 91 d2 34 34 73 74 c3 08 00 45 00 .P°F²'Ò44stÃ..E.
00000010: 00 3d 32 13 40 00 80 06 74 8e a9 fe 80 19 a9 fe .=2.@.€.t©þ€.©þ
00000020: 80 03 04 00 04 18 df 28 32 0a 39 3e d6 94 50 18 €....ß(2.9>Ö.P.
00000030: fa f0 fa 93 00 00 40 6d 73 67 2b a3 eb 24 e8 04 úðú...@msg+£ë$è.
00000040: 49 9c d9 a9 4c 94 44 14 d6 6a 00                I.Ù©L.D.Öj.
```

Message detected

Figure 4.17 The result messages with Triple DES (1)



```
No Dest Address          Summary
 1 [169.254.128.3]       TCP: D=1048 S=1024        ACK=960419476 SEQ=3743953418 LEN=21
 2 [169.254.128.25]      TCP: D=1024 S=1048        ACK=3743953439 WIN=64198
 3 [169.254.128.25]      TCP: D=1024 S=1048        ACK=3743953439 SEQ=960419476 LEN=21
 4 [169.254.128.3]       TCP: D=1048 S=1024        ACK=960419497 WIN=64219

TCP: ----- TCP header -----
    TCP:
    TCP: Source port             = 1048
    TCP: Destination port        = 1024
    TCP: Sequence number         = 960419476
    TCP: Next expected Seq number= 960419497
    TCP: Acknowledgment number   = 3743953439
    TCP: Data offset             = 20 bytes
    TCP: Reserved Bits: Reserved for Future Use (Not shown in the Hex Dump)
    TCP: Flags                   = 16

00000000: d2 34 34 73 74 c3 00 50 ba 46 b2 91 08 00 45 00 Ò44stÃ.P°F²'..E.
00000010: 00 3d 15 fa 40 00 80 06 90 a7 a9 fe 80 03 a9 fe .=.ú@.€..§©þ€.©þ
00000020: 80 19 04 18 04 00 39 3e d6 94 df 28 32 1f 50 18 €.....9>Ö.ß(2.P.
00000030: fa c6 79 fd 00 00 40 6d 73 67 c1 ce 4b 7d f9 3e úÆýý..@msgÁÎK}ù>
00000040: b1 ad 32 e8 27 f8 94 6a 54 4e 00                ±-2è'ø.jTN.
```

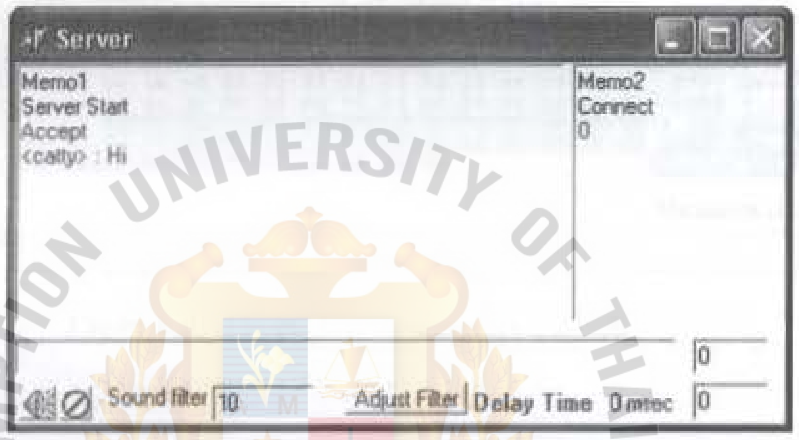
Message detected

Figure 4.17 The result messages with Triple DES (2)

Similarly to other encryptions, the sniffer pro detects the ciphertext or unknown messages shown in Figure 4.17.

4.2.2.5 SAFER K-64

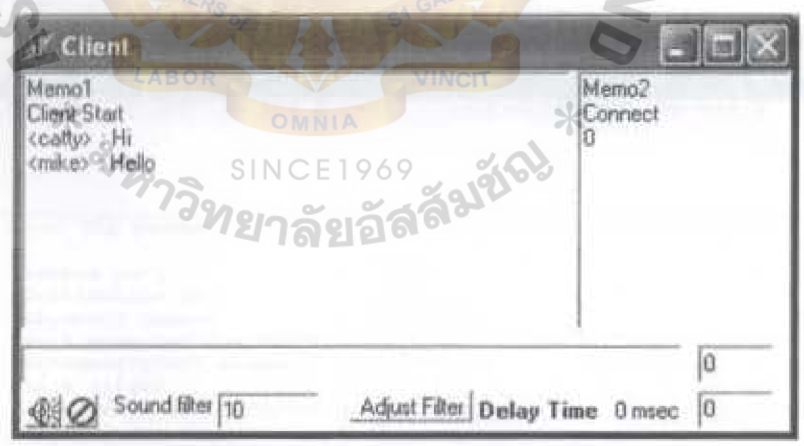
Catty sends message "Hi" to Mike with key: password (figure 4.18). Mike sends message "Hello" to Catty (figure 4.19). "Hi". The messages are detected by sniffer shown in figure 4.20.



Memo1
Server Start
Accept
<catty> : Hi

Connect
0

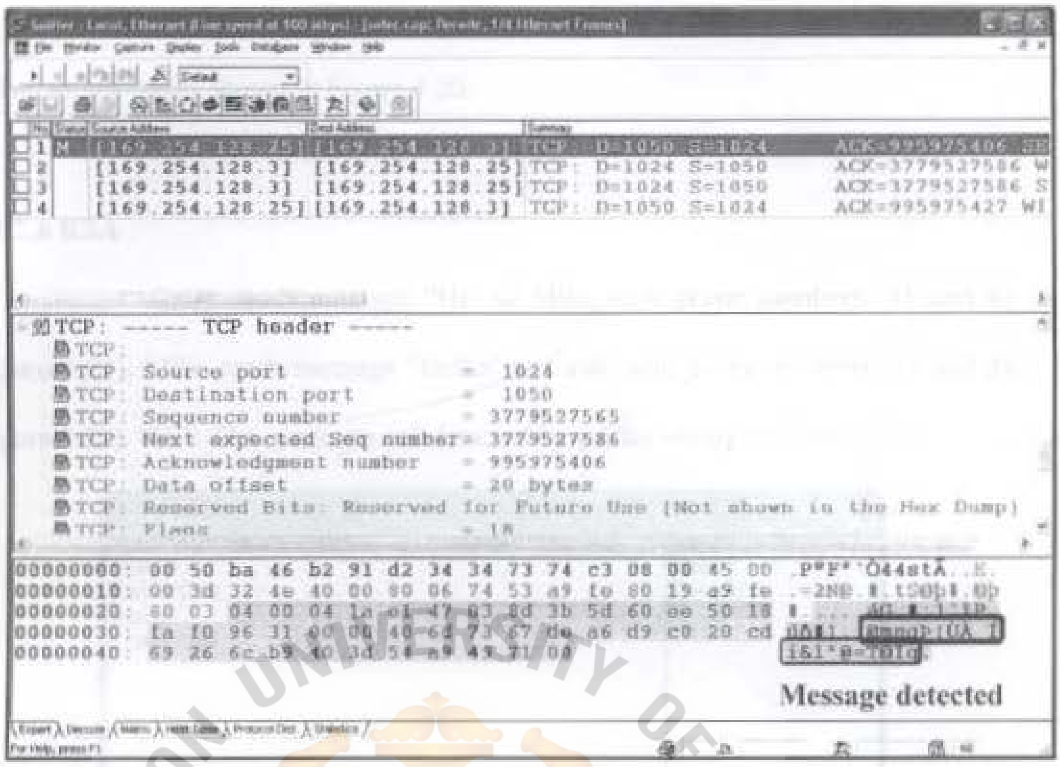
Figure 4.18 Catty sends message "Hi" to Mike with SAFER K-64



Memo1
Client Start
<catty> : Hi
<mike> : Hello

Connect
0

Figure 4.19 Mike sends message "Hello" to Catty with SAFER K-64



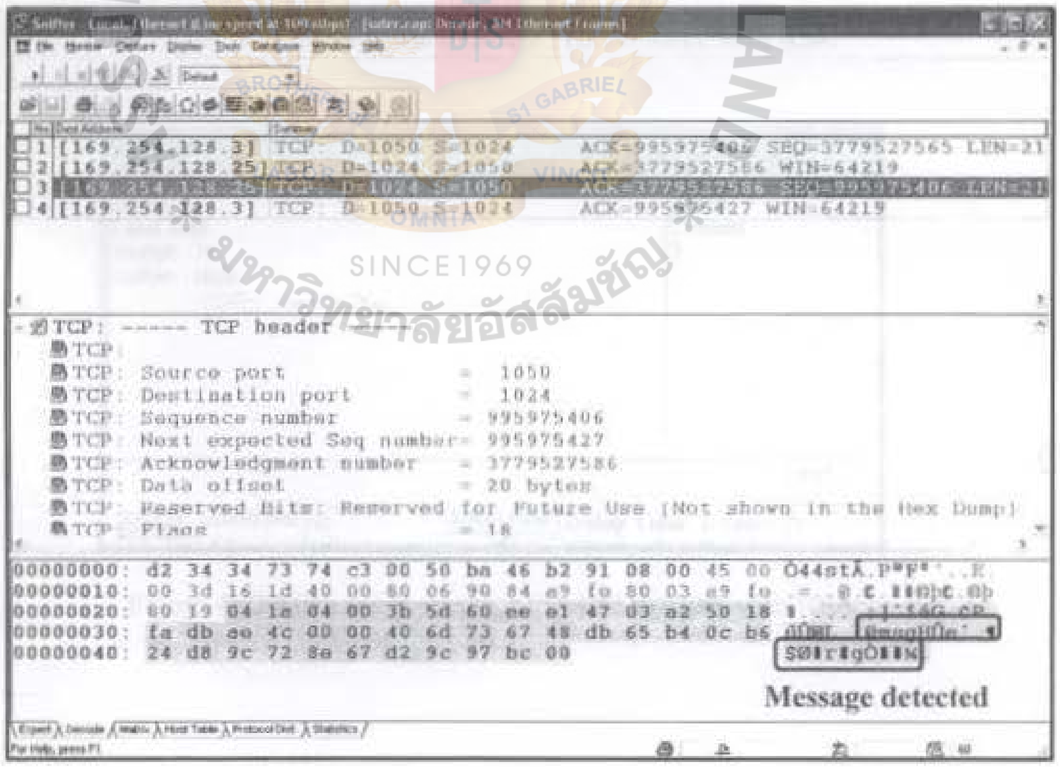
```
 1 M [169.254.128.25] [169.254.128.3]  TCP: D=1050 S=1024   ACK=995975406 SE
 2   [169.254.128.3]  [169.254.128.25] TCP: D=1024 S=1050   ACK=3779527586 W
 3   [169.254.128.3]  [169.254.128.25] TCP: D=1024 S=1050   ACK=3779527586 S
 4   [169.254.128.25] [169.254.128.3]  TCP: D=1050 S=1024   ACK=995975427 WI
```

```
TCP: ----- TCP header -----
  TCP:
  TCP: Source port             =  1024
  TCP: Destination port        =  1050
  TCP: Sequence number         = 3779527565
  TCP: Next expected Seq number= 3779527586
  TCP: Acknowledgment number   = 995975406
  TCP: Data offset             = 20 bytes
  TCP: Reserved Bits: Reserved for Future Use (Not shown in the Hex Dump)
  TCP: Flags                   = 18
```

```
00000000: 00 50 ba 46 b2 91 d2 34 34 73 74 c3 08 00 45 00
00000010: 00 3d 32 4e 40 00 80 06 74 53 a9 fe 80 19 a9 fe
00000020: 80 03 04 00 04 1a e1 47 03 8d 3b 5d 60 ee 50 18
00000030: fa f0 96 31 00 00 40 6d 73 67 de a6 d9 c0 20 cd
00000040: 69 26 6c b9 40 3d 54 a9 49 71 00
```

Message detected

Figure 4.20 The result messages with SAFER K-64 (1)



```
 1 [169.254.128.3]  TCP: D=1050 S=1024   ACK=995975406 SEQ=3779527565 LEN=21
 2 [169.254.128.25] TCP: D=1024 S=1050   ACK=3779527586 WIN=64219
 3 [169.254.128.25] TCP: D=1024 S=1050   ACK=3779527586 SEQ=995975406 LEN=21
 4 [169.254.128.3]  TCP: D=1050 S=1024   ACK=995975427 WIN=64219
```

```
TCP: ----- TCP header -----
  TCP:
  TCP: Source port             =  1050
  TCP: Destination port        =  1024
  TCP: Sequence number         = 995975406
  TCP: Next expected Seq number= 995975427
  TCP: Acknowledgment number   = 3779527586
  TCP: Data offset             = 20 bytes
  TCP: Reserved Bits: Reserved for Future Use (Not shown in the Hex Dump)
  TCP: Flags                   = 18
```

```
00000000: d2 34 34 73 74 c3 00 50 ba 46 b2 91 08 00 45 00
00000010: 00 3d 16 1d 40 00 80 06 90 84 a9 fe 80 03 a9 fe
00000020: 80 19 04 1a 04 00 3b 5d 60 ee e1 47 03 a2 50 18
00000030: fa db ee 4c 00 00 40 6d 73 67 48 db 65 b4 0c b6
00000040: 24 d8 9c 72 8e 67 d2 9c 97 bc 00
```

Message detected

Figure 4.20 The result messages with SAFER K-64 (2)

Similarly to other encryptions, the sniffer pro detects the ciphertext or unknown messages shown in Figure 4.20.

4.2.2.6 RSA

Catty sends message "Hi" to Mike with prime numbers: 71 and 43 (figure 4.21). Mike sends message "Hello" to Catty with prime numbers: 17 and 59 (figure 4.22). "Hi". The messages are detected by sniffer shown in figure 4.23.

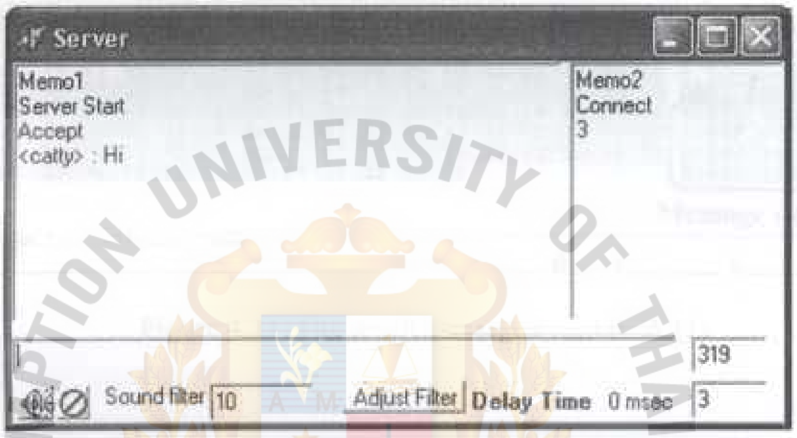


Figure 4.21 Catty sends message "Hi" to Mike with RSA

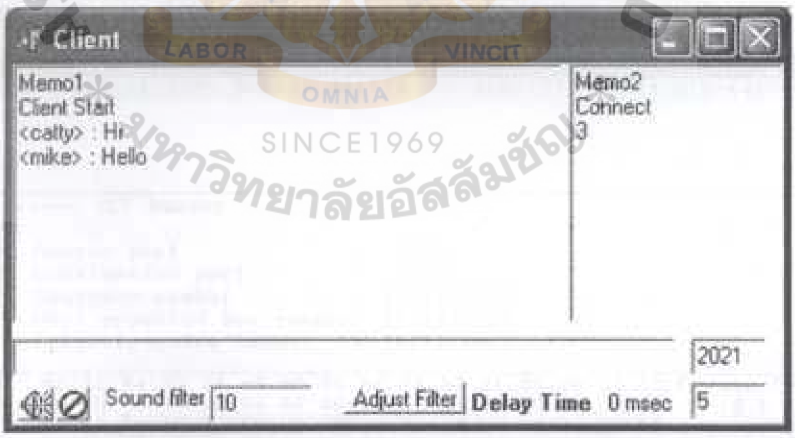
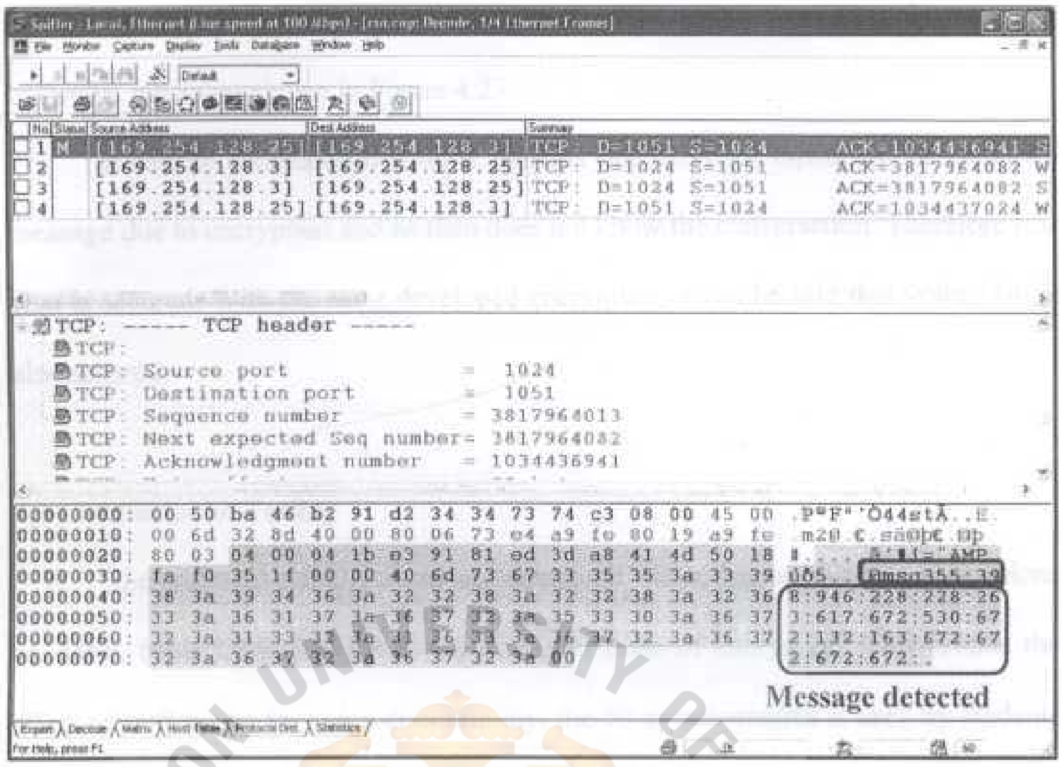
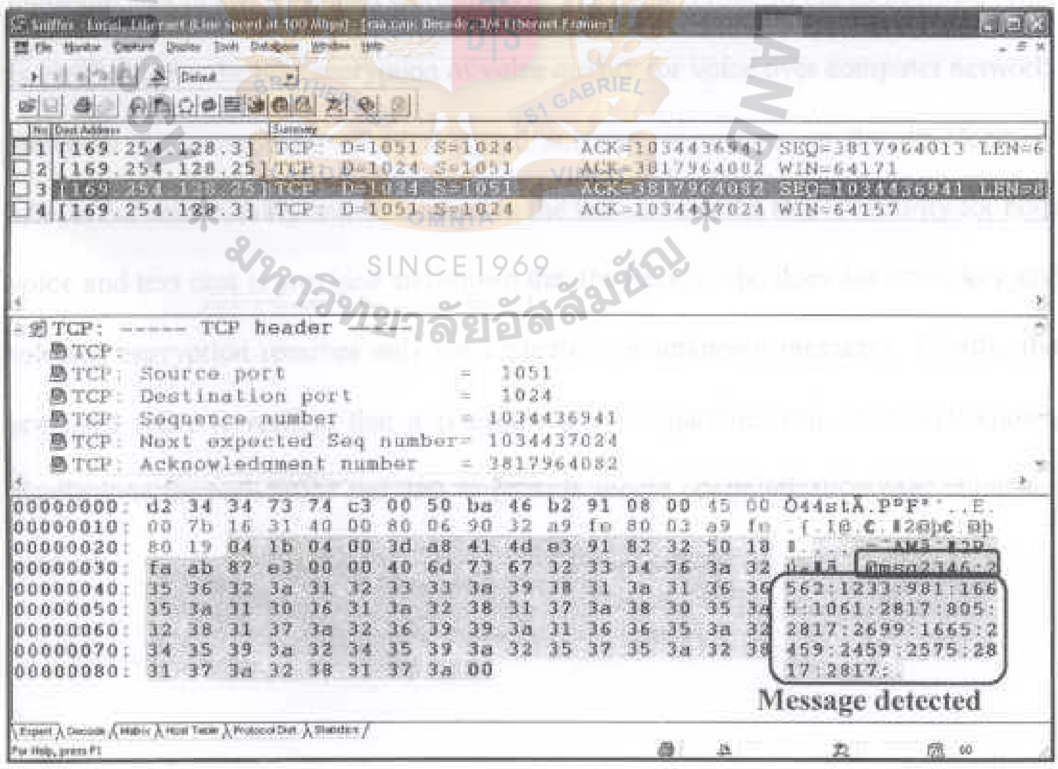


Figure 4.22 Mike sends message "Hello" to Catty with RSA



```
Sniffer - Local, Ethernet (Line speed at 100 Mbps) - [rsa.cap: Decode, 1/4 Ethernet Frames]

No  Status  Source Address        Dest Address          Summary
1   M       [169.254.128.25]      [169.254.128.3]       TCP: D=1051 S=1024      ACK=1034436941 S
2           [169.254.128.3]       [169.254.128.25]      TCP: D=1024 S=1051      ACK=3817964082 W
3           [169.254.128.3]       [169.254.128.25]      TCP: D=1024 S=1051      ACK=3817964082 S
4           [169.254.128.25]      [169.254.128.3]       TCP: D=1051 S=1024      ACK=1034437024 W

TCP: ----- TCP header -----
   TCP:
   TCP: Source port              =  1024
   TCP: Destination port         =  1051
   TCP: Sequence number          = 3817964013
   TCP: Next expected Seq number= 3817964082
   TCP: Acknowledgment number    = 1034436941

00000000: 00 50 ba 46 b2 91 d2 34 34 73 74 c3 08 00 45 00  .PºF²'Ò44stÃ..E.
00000010: 00 6d 32 8d 40 00 80 06 73 e4 a9 fe 80 19 a9 fe  .m2.@.€.sä©þ€.©þ
00000020: 80 03 04 00 04 1b e3 91 81 ed 3d a8 41 4d 50 18  €.....ã'.í=¨AMP.
00000030: fa f0 35 1f 00 00 40 6d 73 67 33 35 35 3a 33 39  úð5...@msg355:39
00000040: 38 3a 39 34 36 3a 32 32 38 3a 32 32 38 3a 32 36  8:946:228:228:26
00000050: 33 3a 36 31 37 3a 36 37 32 3a 35 33 30 3a 36 37  3:617:672:530:67
00000060: 32 3a 31 33 32 3a 31 36 33 3a 36 37 32 3a 36 37  2:132:163:672:67
00000070: 32 3a 36 37 32 3a 36 37 32 3a 00                 2:672:672:.
```

Figure 4.23 The result messages with RSA (1)



```
Sniffer - Local, Ethernet (Line speed at 100 Mbps) - [rsa.cap: Decode, 3/4 Ethernet Frames]

No  Dest Address          Summary
1   [169.254.128.3]       TCP: D=1051 S=1024      ACK=1034436941 SEQ=3817964013 LEN=6
2   [169.254.128.25]      TCP: D=1024 S=1051      ACK=3817964082 WIN=64171
3   [169.254.128.25]      TCP: D=1024 S=1051      ACK=3817964082 SEQ=1034436941 LEN=8
4   [169.254.128.3]       TCP: D=1051 S=1024      ACK=1034437024 WIN=64157

TCP: ----- TCP header -----
   TCP:
   TCP: Source port              =  1051
   TCP: Destination port         =  1024
   TCP: Sequence number          = 1034436941
   TCP: Next expected Seq number= 1034437024
   TCP: Acknowledgment number    = 3817964082

00000000: d2 34 34 73 74 c3 00 50 ba 46 b2 91 08 00 45 00  Ò44stÃ.PºF²'..E.
00000010: 00 7b 16 31 40 00 80 06 90 32 a9 fe 80 03 a9 fe  .{.1@.€..2©þ€.©þ
00000020: 80 19 04 1b 04 00 3d a8 41 4d e3 91 82 32 50 18  €.....=¨AMã'‚2P.
00000030: fa ab 87 e3 00 00 40 6d 73 67 32 33 34 36 3a 32  ú«‡ã..@msg2346:2
00000040: 35 36 32 3a 31 32 33 33 3a 39 38 31 3a 31 36 36  562:1233:981:166
00000050: 35 3a 31 30 36 31 3a 32 38 31 37 3a 38 30 35 3a  5:1061:2817:805:
00000060: 32 38 31 37 3a 32 36 39 39 3a 31 36 36 35 3a 32  2817:2699:1665:2
00000070: 34 35 39 3a 32 34 35 39 3a 32 35 37 35 3a 32 38  459:2459:2575:28
00000080: 31 37 3a 32 38 31 37 3a 00                       17:2817:.
```

Figure 4.23 The result messages with RSA (2)

Similarly to other encryptions, the sniffer pro detects the ciphertext or unknown messages shown in Figure 4.23.

It can be concluded that other persons (hackers) see ciphertext or unknown message due to encryption and he then does not know the conversation. Therefore text chat is secured. With the same developed encryption, it can be said that voice chat is also secured.

4.3 Conclusion of Testing

The proposed NEA is tested and compared with other well-known encryptions i.e. DES, triple DES, SAFER K-64, RSA and none of encryption. To evaluate the effect of buffer size for voice discontinuity, the 50 questionnaires is used by students and teachers at Saint Dominic School. The results are shown that the big buffer size causes longer duration of voice discontinuity than the small buffer size and the optimum size in this presented software is at 512 bytes. From 50 questionnaires, NEA is verified to be the best encryption of voice quality for voice over computer network. However the complexity of all designed software algorithms in the development affects the voice quality and also noise. In the last section, the test of security for both voice and text chat is provided and shown that the hacker who does not know key and selected encryption receives only the ciphertext or unknown messages. Finally, the proposed NEA is verified that it is usable and comparable with other well-known encryptions for both voice and data to provide secure communication over computer network.

Chapter 5

Conclusion

5.1 Thesis summary

With the structure of feed forward multilayer neural network, the proposed encryption is called Neural Encryption Algorithm (NEA), but its weights are only 0 and 1, not real values like neural network. NEA is a symmetric-key encryption and secret-key block-enciphering algorithm with flexible block size of plaintext, flexible key length and can be multiple of hidden layers. To compare with Data Encryption Standard (DES), Triple DES, Secure And Fast Encryption Routine with a Key of length 64 bits (SAFER K-64), and Rivest Shamir and Adleman (RSA); NEA has 64-bit block size of plaintext, 64-bit key length in key11, key12, key21, key22 and (64×64)-bit in ESkey and DSkey. Then, the strength of NEA is key length and multiple of hidden layers. The software development is used for comparison of DES, Triple DES, SAFER K-64, RSA, NEA and no encryption for voice/text chat over computer network by using TCP/IP.

The communication sessions in software implementation have 2 parts:

1. Between user (caller) and server:
    1.1 Users login to server by sending their user name and password with DES encryption.
    1.2 Users (callers) send requested call to users (receivers) with selected encryption and key passing through server with DES encryption. Then their keys remain secret. In the same way, users (receivers) accept or refuse the call passing through server with DES encryption. If the answer is the accepted message, server informs caller and caller then chats directly to receiver (by

changing the port number). If the answer is the refused message, server informs caller and communication is then terminated.

2. Between user (caller) and (receiver)

After caller receives accepted message from server, caller is able to chat to receiver directly. Encryption and key are selected by caller and sent with the requested message. The encryption between caller and receiver then depends on caller's selection.

The developed software provides voice/text chat with TCP/IP over computer network under Windows operating system. The software implementation is developed with Borland Delphi Enterprise version 6.0.

To test the voice quality and discontinuity, 50 questionnaires is used to evaluate. The results are summarized as following :

1. The big buffer size causes the long duration of voice discontinuity.
2. The optimum buffer size for voice over computer network is at 512 bytes.
3. NEA is verified to be the best encryption for voice over computer network with resulting in the best voice quality.

Finally, NEA is verified that it is usable with the best encryption to provide secure voice/data communication over computer network by comparing with other well known encryptions.

5.2 Recommendation for future works

1. In this research, the proposed NEA is applied to only voice and data over computer network. It is further applied to image or other desired secure communications.
2. In the developed software, there are only one couple people for conversation at a time. It can be further developed for multiple chat in one conversation.
3. With voice/text chat in software implementation, it can be tested with other encryption techniques.
4. The concept of NEA is able to be further designed for more strengthen.
5. NEA can be further modified to implement in both software and hardware.
6. NEA algorithm is the initiative for further design and development of encryption technique.
7. The software implementation for encryption techniques can be modified for less computation and then for better results.

Appendix

Questionnaire

แบบสอบถาม

เรื่องการใช้ โปรแกรมการคุยและส่งข้อความผ่านเครือข่ายคอมพิวเตอร์ (Voice/Text chat program)

โปรแกรมการคุยและส่งข้อความผ่านเครือข่าย (Voice/Text chat program) ได้ถูกออกแบบมาเพื่อทดสอบและใช้เปรียบเทียบการเข้ารหัส (encryption) ของ Neural Encryption Algorithm (NEA) รวมทั้ง Data Encryption Standard (DES), triple DES, Secure And Fast Encryption Routine with a Key of length 64 bits (SAFER K-64), Rivest Shamir and Adleman (RSA) และแบบที่ไม่มีการเข้ารหัส ซึ่งโปรแกรมนี้ได้ถูกพัฒนาโดย นางสาว ทับทิม สงวนวงษ์ทอง ขอความอนุเคราะห์จากท่านตอบแบบสอบถามต่อไปนี้ จักขอบพระคุณอย่างสูง

แบบสอบถามมี 3 ตอน คือ

ตอนที่ 1 ข้อมูลทั่วไปของผู้ตอบแบบสอบถาม

ตอนที่ 2 การทดสอบเปรียบเทียบคุณภาพและการขาดช่วงของเสียง

ตอนที่ 3 ความคิดเห็น

ตอนที่ 1 ข้อมูลทั่วไปของผู้ตอบแบบสอบถาม

คำชี้แจง โปรดเขียนเครื่องหมาย √ ลงใน [ ] หน้าข้อความที่ตรงกับตัวท่านมากที่สุด

1. เพศ : [ ] ชาย [ ] หญิง

2. อายุ :

[ ] 15 – 20 ปี [ ] 21 – 25 ปี [ ] 26 – 30 ปี [ ] 31 – 35 ปี

[ ] 36 – 40 ปี [ ] 41 – 45 ปี [ ] 46 – 50 ปี [ ] 51 – 55 ปี

3. วุฒิทางการศึกษา :

[ ] ต่ำกว่าปริญญาตรี [ ] ปริญญาตรี [ ] ปริญญาโท [ ] อื่น ๆ โปรดระบุ ....

4. อาชีพ

[ ] นักเรียน/นักศึกษา [ ] พนักงานบริษัท [ ] ข้าราชการ/พนักงานรัฐวิสาหกิจ

[ ] อื่น ๆ โปรดระบุ ....

5. ท่านคิดว่าการเข้ารหัสข้อมูลสำหรับการสื่อสารผ่านเครือข่ายคอมพิวเตอร์ สำคัญหรือไม่

[ ] ใช่ [ ] ไม่, กรุณาให้เหตุผล : ....

ตอนที่ 2 การทดสอบเปรียบเทียบคุณภาพและการขาดช่วงของเสียงที่อัตราการชักตัวอย่าง (sampling rate) 8 kHz โดยการเข้ารหัสด้วย DES, triple DES, SAFER K-64, RSA, NEA และแบบไม่มีการเข้ารหัส ณ ขนาดบัฟเฟอร์ 64 bytes, 128 bytes, 256 bytes, 512 bytes, 1024 bytes, 2048 bytes, 4096 bytes และ 8192 bytes

คำชี้แจง โปรดใส่ระดับของคุณภาพเสียง (หมายเลข) ตามที่กำหนดไว้ต่อไปนี้ นับจำนวนการขาดช่วงของเสียง (ครั้ง) และ ความยาวในการขาดของเสียง โดยการฟังจากเทป ติดต่อกัน 35 วินาที ตามความเป็นจริง

5	=	ระดับดีมาก	หมายความว่า	คุณภาพเสียงดีมาก ไม่มีเสียงรบกวน (noise)และสั่น
4	=	ระดับดี	หมายความว่า	คุณภาพเสียงดีมาก มีเสียงรบกวนหรือสั่นเป็นบางครั้ง
3	=	ระดับปานกลาง	หมายความว่า	คุณภาพเสียงดี ไม่มีเสียงรบกวน และสั่น
2	=	ระดับพอใช้	หมายความว่า	คุณภาพเสียงดี มีเสียงรบกวนหรือสั่นเป็นบางครั้ง
1	=	ระดับปรับปรุง	หมายความว่า	คุณภาพเสียงไม่ดี มีเสียงรบกวนหรือสั่น

ขนาดบัฟเฟอร์ (Bytes)	DES			Triple DES			SAFER K-64			RSA			NEA			None		
	คุณภาพเสียง	จำนวนขาดช่วง (ครั้ง)	ความยาวการขาด (วินาที)	คุณภาพเสียง	จำนวนขาดช่วง (ครั้ง)	ความยาวการขาด (วินาที)	คุณภาพเสียง	จำนวนขาดช่วง (ครั้ง)	ความยาวการขาด (วินาที)	คุณภาพเสียง	จำนวนขาดช่วง (ครั้ง)	ความยาวการขาด (วินาที)	คุณภาพเสียง	จำนวนขาดช่วง (ครั้ง)	ความยาวการขาด (วินาที)	คุณภาพเสียง	จำนวนขาดช่วง (ครั้ง)	ความยาวการขาด (วินาที)
64																		
128																		
256																		
512																		
1024																		
2048																		
4096																		
8192																		

ตอนที่ 3 ความคิดเห็น

คำชี้แจง โปรดเขียนเครื่องหมาย √ ลงใน [ ] หน้าข้อความและเติมข้อความที่ตรงกับความเป็นจริง

1. ขนาดบัฟเฟอร์มีผลกระทบต่อการขาดช่วงของเสียงหรือไม่ [ ] มี [ ] ไม่มี

2. ขนาดบัฟเฟอร์ใหญ่ทำให้มีการขาดช่วงของเสียงมากใช่หรือไม่ [ ] ใช่ [ ] ไม่ใช่

3. ขนาดบัฟเฟอร์ที่ทำให้การรับส่งเสียงมีคุณภาพดีที่สุด.......................... Bytes

4. การเข้ารหัสแบบไหนที่คุณภาพเสียงส่วนใหญ่อยู่ในระดับดีมาก ....

References

[1] T. Schaffer, A. Glaser, S. Rao, and P. Franzon, "A Flip-Chip Implementation of the Data Encryption Standard (DES)," Multi-Chip Module Conference, MCMC '97., IEEE, pp. 13 – 17, 1997.

[2] A. Schubert, V. Meyer, and W. Anheier, "Reusable Cryptographic VLSI Core Based on the SAFER K-128 Algorithm with 251.8 Mbit/s Throughput," Signal Proceesing Systems, SIPS 98, IEEE Workshop, pp. 437 – 446, 1998.

[3] P. Hamalainen, M. Hannikainen, T. Hamalainen, and J. Saarinen, "Configurable Hardware Implementation of Triple-DES Encryption Algorithm for Wireless Local Area Network," Acoustics, Speech, and Signal Processing, IEEE International Conference, vol. 2, pp. 1221 – 1224, 2001.

[4] A. Selby, and C. Mitchell, "Algorithms for Software Implementation of RSA," Computers and Digital Techniques, IEE Proceedings E, vol. 136, Pt. E, no. 3 , pp. 166 – 170, May 1989.

[5] A.. Royo, J. Moran, and J. Carlos Lopez, "Design and Implementation of a Coprocessor for Cryptography Applications," European Design and Test Conference, ED&TC 97 proceedings, pp. 213 – 217 , 1997.

[6] T. Korkishko, and A. Melnyk, "Cryptographic Processor Architectures for DES Algorithm," Africon, IEEE, vol. 1, pp. 175 – 180, 1999.

[7] S. S. Raghuram, and C. Chakrabarti, "A Programmable Processor for Cryptography," Circuits and Systems, proceedings, ISCAS 2000, IEEE International Symposium, vol. 5, pp. 685 – 688, May 2000.

[8] N. Hadjina, and P. Thompson, "Data Security on Ethernet LANs," Electrotechnical Conference, MELECON 2000, 10th Mediterranean, vol. 1, pp. 23 – 26, 1998.

[9] L. Wu, C. Weaver, and T. Austin, "CryptoManiac: A Fast Flexible Architecture for Secure Communication," Computer Architecture, proceedings 28th Annual International Symposium, pp. 110 – 119, 2001.

[10] C. E. Davis, "The Livermore Secure-Voice Radio System," Security Technology, Crime Countermeasures proceedings, IEEE, International Carnahan Conference, pp. 139-144, 1988.

[11] S. Sridharan, E. Dawson, and B. Goldburg, "Speech Encryption in the transform domain," Electronics Letters, vol. 26 no. 10, pp. 655 – 657, May 1990.

[12] S. Sridharan, E. Dawson, and B. M. Goldburg, "Speech Encryption Using Discrete Orthogonal Transforms," Acoustics, Speech, and Signal Processing, ICASSP-90, 1990 IEEE International Conference, vol. 3, pp. 1647 – 1650, 1990.

[13] S. Sridharan, E. Dawson, and B. Goldburg, "Fast Fourier transform based speech encryption system," Communications, Speech and Vision, IEE Proceedings-I, vol. 138, no. 3, pp. 215 - 223, June 1991.

[14] C. Duraiappan, and Y. Zheng, "Improving Speech Security and Authentication in Mobile Communications," Australian Research Council, Ref no. A49232172, 1991.

[15] H. Schulzrinne, "Voice Communication Across the Internet," Naval Research, contract: N00014-90-J-1293, July 1992.

[16] G. Troulllnos, "A Software Based Approach to Secure Voice Applications," ICECS'96, Proceedings of the Third IEEE International Conference, vol.1, pp.176 – 182, 1996.

[17] C.-P. Wu, and C.-C. Jay Kuo, "Fast Encryption Methods for Audiovisual Data Confidentiality," SPIE Proceedings, vol. 4209, pp. 284 – 295, Nov. 2000.

[18] S. Etemadi Borujeni, "Speech Encryption Based on Fast Fourier Transform Permutation," Electronics, Circuits and Systems, ICECS 2000, 7th IEEE International Conference, vol. 1, pp. 290 – 293, 2000.

[19] http://intania.kku.ac.th/projects/2002/COE2002-05/main.html , "Software Development for Data Encryption by Voice," Khonkaen University, 2002.

[20] B. Schneier, "Applied Cryptography," 2nd ed., John Wiley & Sons, 1996.

[21] James F. Kurose, and Keith W. Ross, “Computer networking: a top-down approach featuring the Internet,” Addison Wesley Longman, 2001.

[22] “Data Encryption Standard (DES),” Federal Information Processing Standards (FIPS) Publication 46-3, National Institute of Standards and Technology (NIST), USA, 1999.

[23] James L. Massey, “SAFER K-64: A Byte-Oriented Block-Ciphering Algorithm,” Lecture Notes in Computer Science No. 809, New York: Springer, 1994.

[24] B. Forouzan, “Introduction to Data Communications And Network,” McGraw-Hill, 1998.