**Available online @ www.actasatech.com**

*acta* **SATECH 3(2): 150 – 157 (2010)**

# Modeling of secure instant messaging system in GSM

**O. J. Anyaebu, Dike N.** and ***Oduali I.**
*Department of Electrical/Electronic Engineering, University of Port Harcourt, Nigeria*

*** Corresponding author:** <ihomeus@yahoo.com>

**ABSTRACT**
*With the rapid development in global communication networks, the threat of security and in particular that of Global System for Mobile Communication (GSM) is real and highly dangerous. Instant messaging has become so common on GSM for real time communication. This paper focuses on Modeling of Secure Instant Messaging System in GSM using Data Encryption Standard. A cipher text is obtained by the Initial Permutation (IP). If the times of the iteration are large enough, the randomness of the encryption and decryption function is so large that attackers can not break this cryptosystem. The packet data – conversation (audio), text, graphics, color, and video messages – delivered may be very important and confidential. This report discusses how to protect data from interception by authentication and encryption.*

**Keywords***: Instant messaging, security, encryption, algorithm, permutation, iteration*

## INTRODUCTION

Instant Messaging (IM) systems have the ability to fundamentally change the way one communicate and do business, many of today's implementations pose security challenges. Most IM systems presently in use were designed with scalability rather than security in mind. Virtually all freeware IM programs lack encryption capabilities and must have features that bypass traditional corporate firewalls, making it difficult for administrators to control instant messaging usage inside an organization (Persson, 2007). Many of these systems have insecure password management and are vulnerable to account spoofing and denial-of-service (DoS) attacks. Instant messaging has blossomed to become a staple mode of communication for tens of millions of Internet users. Popular systems such as America Online's Instant Messenger, Microsoft's MSN Messenger, and Internet Relay Chat (IRC) have changed the way one communicate with friends, acquaintances, and business colleagues (Hindocha, 2002). Once limited to desktops, popular instant messaging systems are finding their way onto Hand-held devices and cell phones, allowing users to chat from virtually anywhere. Instant Messenger (IM) systems, while not supported by many Information Technology (IT) departments, are quickly gaining popularity with knowledgeable workers who find these systems faster and more convenient than email or telephone (Mannan *et al.*, 2005). Also IM works with real time.

### Methodology

The model adopted for this project is Data Encryption Standard (DES) using the Conventional Encryption method which is outlined as follows:

- Conventional Encryption Principles and Conventional Encryption algorithms
- Cipher Block Modes of Operation and Location of Encryption Devices
- Key Distribution

Encryption involves plaintext, encryption algorithm, secret key, cipher text and decryption of algorithm. The security of data depends on the secrecy of the key and not the secrecy of the algorithm (Johnson, 2002).

**Conventional Encryption Principles and Conventional Encryption algorithms**

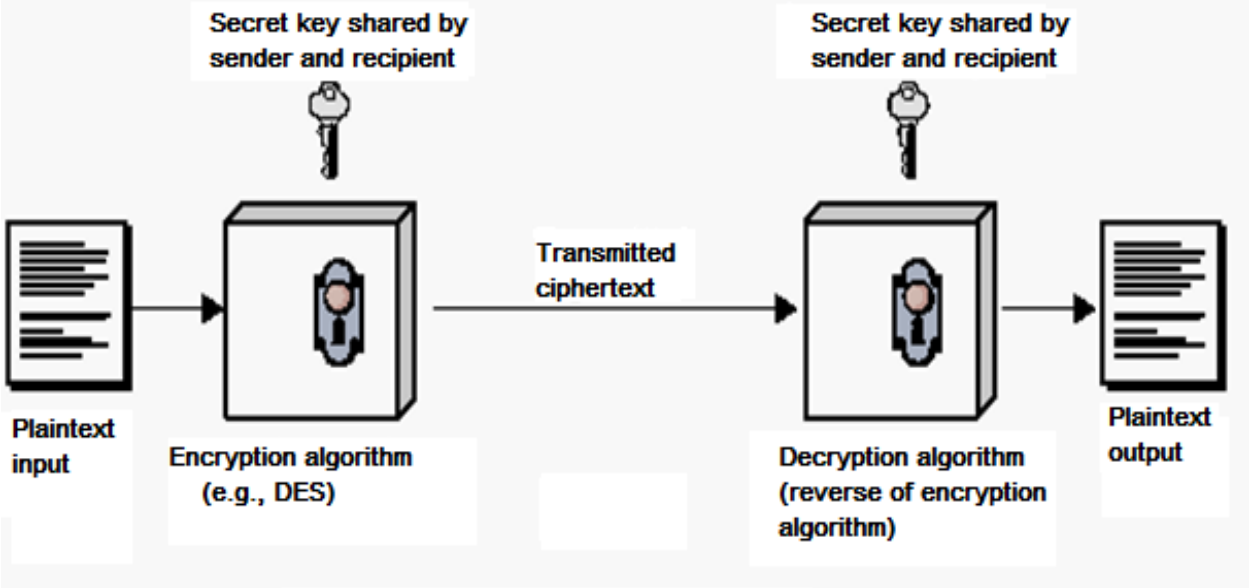


**Figure 1: Model of Conventional Encryption**

Figure1 illustrates the simple Conventional encryption algorithm of DES. A plaintext is sent as an input and is encrypted using the secret key of the sender. The message is then transmitted via the network as cipher text to the recipient. At the receipt of the message, the recipient decrypts the message with the same secret key shared by the sender and the recipient. The plaintext output is then displayed.

The algorithm is designed to cipher and decipher blocks of data consisting of 64 bits under control of a 64-bit key. Deciphering must be accomplished by using the same key as for ciphering, but with the schedule of addressing the key bits altered so that the deciphering process is the reverse of the enciphering process. A block to be ciphered is subjected to an initial permutation **IP**, then to a complex key-dependent computation and finally to a permutation which is the inverse of the initial permutation $\mathbf{IP}^{-1}$. The key-dependent computation can be simply defined in terms of a function ***f***, called the cipher function, and a function **KS**, called the key schedule (Engelfriet, 2005). A description of the computation is given first, along with details as to how the algorithm is used for encipherment. Next, the use of the algorithm for decipherment is described

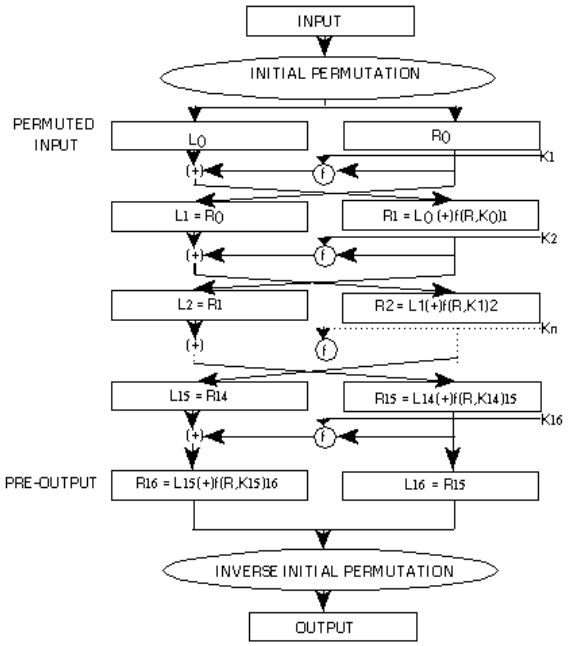


Figure 2: Conventional Encryption algorithms

The following notation is convenient: Given two blocks **L** and **R** of bits, **LR** denotes the block consisting of the bits of **L** followed by the bits of **R**. Since concatenation is associative, $B_1B_2...B_8$, for example, denotes the block consisting of the bits of $B_1$ followed by the bits of $B_2$...followed by the bits of $B_8$ initial permutation (split into left/right halves)
Let **K** be the hexadecimal key K=12AB34C5E2B761AC.This gives binary key setting 1 = 0001, 2 = 0010,...C = 1100, and grouping together every eight bits, of which the last one in each group known as parity bit will be unused.
From the original 64-bit key **K** = 00010010 10101011 00110100 11000101 11100010 10110111 01100001 10101100

**CIPHER BLOCK MODES OF OPERATION AND LOCATION OF ENCRYPTION DEVICES KEY DISTRIBUTION**

**Step 1: Create 16 sub keys, each of which is 48-bits long.**

First key permutation PC_1_perm

C sub key bits

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,

D sub key bits
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

Taking the 56-bit permutation excluding every 8 bit which is the parity bit: 8, 16, 24, 32, 40, 48, 56 and 64.

K+ = 1011101 1010110 0011110 0100010 0001001 1101011 0010000 0100101

Next, split this key into left and right halves, $C_0$ and $D_0$, where each half has 28 bits.

$C_0$=1011101 1010110 0011110 0100010
$D_0$=0001001 1101011 0010000 0100101

$C_1$=0111011010110001111001000101
$D_1$=0010011101011001000001001010

$C_2$=1110110101100011110010001010
$D_2$=0100111010110010000010010100
.
.
.
$C_{16}$=1011101101011000111100100010
$D_{16}$=0001001110101100100000100101

With $C_0$ and $D_0$ defined, we now create sixteen blocks $C_n$ and $D_n$, 1<=$n$<=16. Each pair of blocks $C_n$ and $D_n$ is formed from the previous pair $C_{n-1}$ and $D_{n-1}$, respectively, for $n$ = 1, 2,..., 16, using the following schedule of "left shifts" of the previous block. To do a left shift, move each bit one place to the left, except for the first bit, which is cycled to the end of the block.

**Table 1: Iteration Number**

| Number | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Left Shift | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 1 |

From table 1 above, $C_2$ and $D_2$ are obtained from $C_1$ and $D_1$, respectively, by one left shifts, and $C_{16}$ and $D_{16}$ are obtained from $C_{15}$ and $D_{15}$, respectively, by one left shift. In all cases, by a single left shift is meant a rotation of the bits one place to the left.

The code derived previously is used to form the keys $K_n$, for 1<=$n$<=16, by applying the following permutation table to each of the concatenated pairs $C_nD_n$. (Orlin, 2001). Each pair has 56 bits, but **PC-2** only uses 48 of these.

**Table 2 Permute Code 2**

| | | | | | |
|---|---|---|---|---|---|
| 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 |

Therefore, the first bit of $K_n$ is the 14th bit of $C_nD_n$, the second bit the 17th, and so on, ending with the 48th bit of $K_n$ being the 32th bit of $C_nD_n$.

The next step is to apply the permutation **PC-2**, then key $K_n$ becomes

$K_1$ = 011000 110100 011110 110001 101001 010000 100011

The other keys obtained are

$K_2$ = 011011 101111 000001 101001 010100 110010 011010 000000

.
.
.
.

$K_{16}$ =010011 100001 111101 011110 100100 001100 001100 010101

Considering the message itself

**Step 2: Encode each 64-bit block of data.**

There is an *initial permutation* **IP** of the 64 bits of the message data **O**. This rearranges the bits according to the following table, where the entries in the table show the new arrangement of the bits from their initial order.

DES numbers bits starting at 1 using the Initial permutation

Table 3: Initial Permutation

**IP**

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

Applying the initial permutation to the block of text **O=123456789ABCDEF**,

**O** = 0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111

**IP** = 1100 1100 0000 0000 1100 1100 1111 1111 1111 0000 1010 1010 1111 0000 1010 1010

Here the 58th bit of **O** is "1", which becomes the first bit of **IP**. The 50th bit of **O** is "1", which becomes the second bit of **IP**. The 7th bit of **O** is "0", which becomes the last bit of **IP**.

Next divide the permuted block **IP** into a left half $L_0$ of 32 bits, and a right half $R_0$ of 32 bits.

From **IP**,

$L_0$ = 1100 1100 0000 0000 1100 1100 1111 1111
$R_0$ = 1111 0000 1010 1010 1111 0000 1010 1010

16 iterations, for $1<=n<=16$, using a function $f$ which operates on two blocks--a data block of 32 bits and a key $K_n$ of 48 bits--to produce a block of 32 bits. **Let + denote XOR addition, (bit-by-bit addition modulo 2)**. Then for **n** going from 1 to 16 we calculate

$$L_n = R_{n-1}$$
$$R_n = L_{n-1} + f(R_{n-1},K_n)$$

This results in a final block, for $n$ = 16, of $L_{16}R_{16}$. For the right 32 bits in the current step, we XOR the left 32 bits of the previous step with the calculation $f$.

For example $n$ = 1, then

$K_1$ = 000110 110000 001011 101111 111111 000111 000001 110010
$L_1 = R_0$ = 1111 0000 1010 1010 1111 0000 1010 1010
$R_1 = L_0 + f(R_0,K_1)$

It remains to explain how the function $f$ works. To calculate $f$, first expand each block $R_{n-1}$ from 32 bits to 48 bits. This is done by using a selection table that repeats some of the bits in $R_{n-1}$ . The use of this selection table the function **E (**Fadia, 1993). Thus $E(R_{n-1})$ has a 32 bit input block, and a 48 bit output block.

Let **E** be such that the 48 bits of its output, written as 8 blocks of 6 bits each, are obtained by selecting the

Table 4: **E BIT-SELECTION**

| | | | | | |
|---|---|---|---|---|---|
| 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 |

Thus the first three bits of $\mathbf{E}(R_{n-1})$ are the bits in positions 32, 1 and 2 of $R_{n-1}$ while the last 2 bits of $\mathbf{E}(R_{n-1})$ are the bits in positions 32 and 1.
**Hence** calculate $\mathbf{E}(R_0)$ from $R_0$ as follows:

$R_0$ = 1111 0000 1010 1010 1111 0000 1010 1010
$\mathbf{E}(R_0)$ = 011110 100001 010101 010101 011110 100001 010101 010101

Next in the ***f*** calculation, the output $\mathbf{E}(R_{n-1})$ XORed with the key $K_n$:

$$K_n + \mathbf{E}(R_{n-1}).$$

For $K_1$ , $\mathbf{E}(R_0)$,

$K_1$ = 000110 110000 001011 101111 111111 000111 000001 110010
$\mathbf{E}(R_0)$ = 011110 100001 010101 010101 011110 100001 010101 010101
$K_1+\mathbf{E}(R_0)$ = 011000 010001 011110 111010 100001 100110 010100 100111.

The function ***f*** is not yet calculated but expanded $R_{n-1}$ from 32 bits to 48 bits, using the selection table, and XORed the result with the key $K_n$. Now using 48 bits, or eight groups of six bits to address tables called "**S boxes**". Each group of six bits will give us an address in a different **S** box. Located at that address will be a 4 bit number. This 4 bit number will replace the original 6 bits. The net result is that the eight groups of 6 bits are transformed into eight groups of 4 bits (the 4-bit outputs from the **S** boxes) for 32 bits total.

Write the previous result, which is 48 bits, in the form:

bits in its inputs in order according to the following table:

$$K_n + \mathbf{E}(R_{n-1}) = B_1B_2B_3B_4B_5B_6B_7B_8,$$

where each $B_i$ is a group of six bits.

**RESULT AND DISCUSSION**

The result of the expansion permutation is XOR-ed with the subkey, and then goes through the S-boxes.

There are 8 S-boxes, each of which takes a 6-bit input an spits out a 4-bit output.

$$S_1(B_1)S_2(B_2)S_3(B_3)S_4(B_4)S_5(B_5)S_6(B_6)S_7(B_7)S_8(B_8)$$

where $S_i(B_i)$ refers to the output of the ***i***-th **S** box.

To repeat, each of the functions ***S1, S2,..., S8***, takes a 6-bit block as input and yields a 4-bit block as output. The table to determine $S_1$ is shown and explained below:

**Table 5 S-Boxes**

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **S1** | | | | | | | | | | | | | | | |
| **14** | **4** | **13** | **1** | **2** | **15** | **11** | **8** | **3** | **10** | **6** | **12** | **5** | **9** | **0** | **7** |
| **0** | **15** | **7** | **4** | **14** | **2** | **13** | **1** | **10** | **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** | **6** | **13** |

If $S_1$ is the function defined in this table and $B$ is a block of 6 bits, then $S_1(B)$ is determined as follows: The first and last bits of $B$ represent in base 2 a number in the decimal range 0 to 3 (or binary 00 to 11). Let that number be ***i***. The middle 4 bits of $B$ represent in base 2 a number in the decimal range 0 to 15 (binary 0000 to 1111). Let that number be ***j***. Look up in the table the number in the ***i***-th row and ***j***-th column. It is a number in the range 0 to 15 and is uniquely represented by a 4 bit block. That block is the output $S_1(B)$ of $S_1$ for the input $B$. For example, for input block $B$ = 011011 the first bit is "0" and the last bit "1" giving 01 as the row. This is row 1. The middle four bits are "1101". This is the binary equivalent of decimal 13, so the column is column number 13. In row 1, column 13 appears 5. This determines the output; 5 is binary 0101, so that the output is 0101. Hence $S_1$(011011) = 0101.

The tables (Tropical, 2007) defining the functions $S_1,...,S_8$ are the following:

S-Boxes

S2

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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 |

.
.
.
.

S8

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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 |

For the first round, we obtain as the output of the eight **S** boxes:

$K_1 + \mathbf{E}(R_0)$ = 011000 010001 011110 111010 100001 100110 010100 100111.

$S_1(B_1)S_2(B_2)S_3(B_3)S_4(B_4)S_5(B_5)S_6(B_6)S_7(B_7)S_8(B_8)$ = 0101 1100 1000 0010 1011 0101 1001 0111

The final stage in the calculation of $f$ is to do a permutation **P** of the **S**-box output to obtain the final value of $f$:

$f = \mathbf{P}(S_1(B_1)S_2(B_2)...S_8(B_8))$

The permutation **P** is defined in the following table. **P** yields a 32-bit output from a 32-bit input by permuting the bits of the input block.

Table 6 **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 |

From the output of the eight **S** boxes:

$S_1(B_1)S_2(B_2)S_3(B_3)S_4(B_4)S_5(B_5)S_6(B_6)S_7(B_7)S_8(B_8)$ = 0101 1100 1000 0010 1011 0101 1001 0111

Which results to

$f$ = 0010 0011 0100 1010 1010 1001 1011 1011

$R_1 = L_0 + f(R_0 , K_1)$

= 1100 1100 0000 0000 1100 1100 1111 1111
+ 0010 0011 0100 1010 1010 1001 1011 1011
= 1110 1111 0100 1010 0110 0101 0100 0100

In the next round, $L_2 = R_1$, To calculate $R_2 = L_1 + f(R_1, K_2)$, and so on for 16 rounds. At the end of the sixteenth round blocks $L_{16}$ and $R_{16}$ are obtained. Then the order of the two blocks are reversed into the 64-bit block

$R_{16}L_{16}$

and apply a final permutation $\mathbf{IP}^{-1}$ as defined by the following table:

**Encrypted Communication using DES**

In the following image, the ongoing conversation between ihuoma1 and ihuoma2 are encrypted. However, Data Encryption Standard (DES) method was adopted and installed on the computer. In chat client receiving end the messages are decrypted to plaintext using the same key.

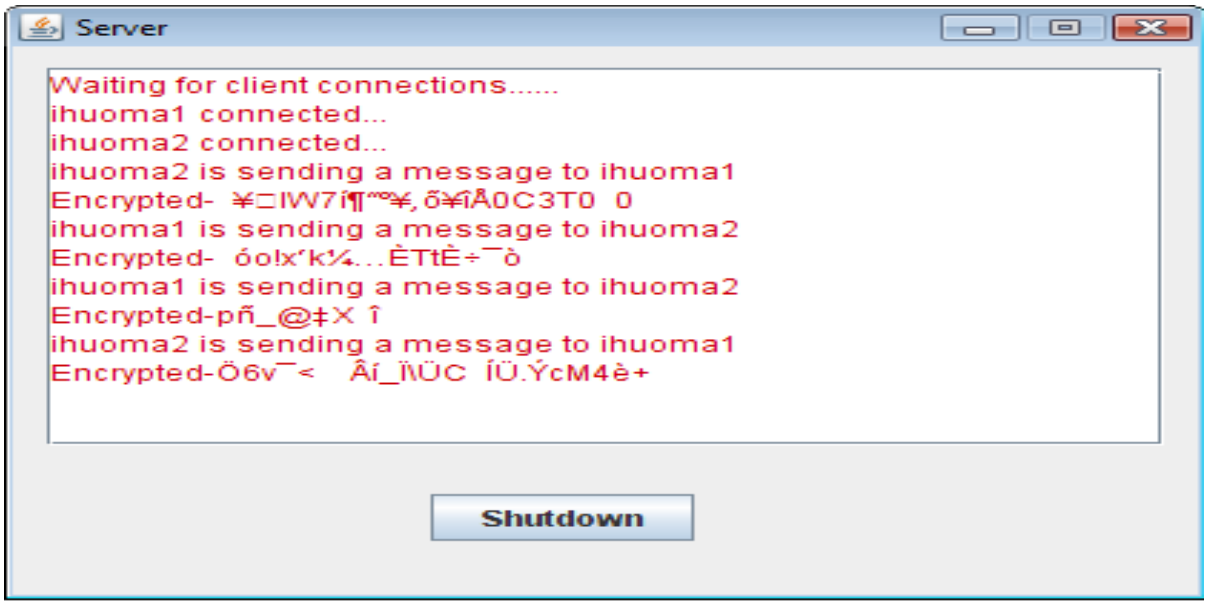


Fig 3 Display of encrypted data via network

Table 7 Inverse Permutation ($\mathbf{IP^{-1}}$)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 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 |

That is, the output of the algorithm has bit 40 of the pre-output block as its first bit, bit 8 as its second bit, and so on, until bit 25 of the pre-output block is the last bit of the output.

16 blocks are processed using the method defined previously; it results to 16th round,

$\boldsymbol{L_{16}}$ = 0100 0011 0100 0010 0011 0010 0011 0100
$\boldsymbol{R_{16}}$ = 0000 1010 0100 1100 1101 1001 1001 0101

The order of these two blocks reversed before applying the final permutation to

$\boldsymbol{R_{16}L_{16}}$ = 00001010 01001100 11011001 10010101 01000011 01000010 00110010 00110100

$\boldsymbol{IP^{-1}}$ = 10000101 11101000 00010011 01010100 00001111 00001010 10110100 00000101

which in hexadecimal format is

85E813540F0AB405.

This is the encrypted form of **O** = 0123456789ABCDEF: namely, **C** = 85E813540F0AB405.

Decryption is simply the inverse of encryption, following the same steps as above, but reversing the order in which the sub keys are applied.

**Conclusion**

In summary Communications security is increasing in importance as a result of the use of electronic communications in more and more business activities. Cryptography is the only practical means to provide security services in many applications. Cryptographic algorithms may be classified as either symmetric, if the same key is shared by the sender and receiver, or asymmetric, if they use different keys. Symmetric algorithms have been dominated by the Data Encryption Standard. The larger the block sizes and key size the greater the security of data, and also the number of rotations or rounds offer increase in security.

The greater complexity in sub key generation algorithm will lead to greater difficulty of cryptanalysis.

**References**

Engelfriet A. 2005 "The DES encryption Algorithm". Available at http://www. Encryption @ iusmentis.com[2009, July 5]

Fadia A 1993 "Algorithms.". Federal Information Processing Standards Publication 46-2

Hindocha, N., (2003). Instant insecurity: Security issues of instant messaging. Available at: http://www.securityfocus.com/infocus/. [2009, July 18]

Mohammad Mannan, van Oorschot P.C. (2005) "Secure Public Instant Messaging" School of Computer Science, Carleton University Ottawa, Ontario.

Orlin G. J., 2001 "The DES Algorithm Illustrated". *Laissez Faire City Times*, Vol 2, No. 28.

Persson S., (2007) "Mobile Peer-to-peer Applications in cellular Networks",

Tropical Software, 2007 "DES Encryption" webmaster@tropsoft.com. [2009, May 15]