Data Encryption and Decryption process Using Bit Shifting and Stuffing (BSS) Methodology

B. Ravi Kumar[1], Dr.P.R.K.Murti[2]
[1, 2] Department of Computer and Information Sciences,
University of Hyderabad,
P.O. Central University, Gachibowli, Hyderabad- 500046, India.
Email:[1]ravi_budithi@yahoo.com & [2]murti.poolla@gmail.com

***Abstract*—Commonly in encryption or decryption process some of the characters are inter changed by using some encryption and decryption algorithms (like DES, IDEA) with key. But in *Bit Shifting and Stuffing (BSS)* system to represent a printable character it needs only seven bits as per its ASCII value. In computer system to represent a printable character it requires one byte, i.e. 8 bits. So a printable character occupies 7 bits and the last bit value is 0 which is not useful for the character. In BSS method we are *stuffing* a new bit in the place of unused bit which is *shifting* from another printable character. So in this *BSS* methodology after encryption, for every eight bytes of plain text it will generate seven bytes cipher text and in decryption, for every seven bytes of cipher text it will reproduce eight bytes of plain text.**

Keywords- Encryption, Decryption, Bit Shifting and Stuffing

I. INTRODUCTION

Data transmitted over the Internet passes through many servers and/or routers and there are many opportunities for third parties to intercept that transmission. Preventing interception is impossible; instead, the data must be made unreadable (encrypted) during transmission, with a way for the intended recipient to be able to transform the received transmission back to its readable form (decryption process) [1]. Encryption is a mechanism by which a message is transformed so that only the sender and recipient can see. When a message is *encrypted*, that means that it is transformed into a form when the data is passed through some substitute technique, shifting technique, table references or mathematical operations. All those processes generate a different form of that data and that is not readable; the encrypted form often looks like random characters or gibberish. When a message is *decrypted*, it is returned to its original readable form. Encryption can provide strong security for data to give sensitive data the highest level of security. As a general term, cryptography is used in order to keep crucial or secret information from unauthorized access. Encryption, a cryptographic implementation, is the conversion of data into a seemingly incomprehensible mixture of characters that, when viewed, cannot be read as simple text. Simple text is defined as standard written text, such as this document. The algorithm used to encrypt data is called a cipher, or cipher text which is representation of the original data in a difference form [2], while unencrypted data is called plaintext. Decryption is the process of converting encrypted data (ciphertext) back into its original form (plaintext), so it can be understood. The goal of encryption is to make data unintelligible to unauthorized readers and extremely difficult to decipher when attacked. The security of encrypted data depends on several factors like what algorithm is used, what is the key size and how was the algorithm implemented in the product.

A. DES

The DES algorithm is a popular algorithm that has been used by the U.S. Government as the standard encryption algorithm, adopted in 1977, DES is based on a conventional or secret key system in which, the sender and the receiver use a single key to encrypt and decrypt the data. The sender uses the key to convert the data to scrambled format according to a complex mathematical algorithm, and only users with the correct key can successfully decrypt the data. The DES is an example of a block cipher, which operates on blocks of 64 bits at a time, with an input key of 64 bits. The key length of 64 bits in it is effectively reduced to 56 bits [3, 4] are

used as a key, while the remaining eight are used to check for errors. The DES algorithm will encrypt data in the same amount of space used by the original data.

The user selects which one of more than 72 quadrillion transformation functions are to be used by selecting a 56-bit key. The theory behind the security of DES has been that, short of trying all 72 quadrillion combinations, there is no way to "break" the algorithm. DES has many variants (Triple DES, DESX).

B. IDEA

International Data Encryption Algorithm (IDEA) came into picture in 1990[6]. This Algorithm offers very good performance (twice as fast as DES) and high security. It is often considered as the quickest and most secure algorithm available to the public today. IDEA uses a 128 bit key which is double the key size of DES. Thus, making it highly immune to attacks. The code is public, but commercial use is subject to license. The strength of IDEA lies in it's modulo multiplication operations.

In our proposed BSS methodology after encryption, for every eight bytes of plain text it will generate seven bytes cipher text and in decryption, for every seven bytes of cipher text it will reproduce eight bytes of plain text.

In section 2 we have discussed about the proposed system methodology, in section 3 we have given the algorithms for encryption and decryption for the proposed system, in section 4 implementation results with discussions, section 5 conclusions and future work.

II. METHODOLOGY

The system deals with security of data by using BSS encryption and decryption.

A. Encryption process

In this process every eight bytes of plain text becomes seven bytes of cipher text. So another advantage of this method is when it encrypts it reduces the size of the data.

In this process let us consider $I_1, I_2, I_3, I_4, I_5, I_6, I_7$ and I_8 represents 8 printable characters of plain text and the values in the boxes represents the byte equivalent values of each character. i.e. a_1 a_2 a_3 a_4 a_5 a_6 a_7 _ represents 7bits of character I_1 and their value may be either 0 or 1. Similarly remaining character bits are represented in boxes as shown in figure 1. In this process the last character I_8 bits $h_1, h_2, h_3, h_4, h_5, h_6, h_7$ are shifted and stuffed in to the characters $I_7, I_6, I_5, I_4, I_3, I_2, I_1$ respectively as shown in figure 2.

I_1:	0	a_1	a_2	a_3	a_4	a_5	a_6	a_7
I_2:	0	b_1	b_2	b_3	b_4	b_5	b_6	b_7
I_3:	0	c_1	c_2	c_3	c_4	c_5	c_6	c_7
I_4:	0	d_1	d_2	d_3	d_4	d_5	d_6	d_7
I_5:	0	e_1	e_2	e_3	e_4	e_5	e_6	e_7
I_6:	0	f_1	f_2	f_3	f_4	f_5	f_6	f_7
I_7:	0	g_1	g_2	g_3	g_4	g_5	g_6	g_7
I_8:	0	h_1	h_2	h_3	h_4	h_5	h_6	h_7

Figure 1: Before Encryption

I_1:	h_7	a_1	a_2	a_3	a_4	a_5	a_6	a_7
I_2:	h_6	b_1	b_2	b_3	b_4	b_5	b_6	b_7
I_3:	h_5	c_1	c_2	c_3	c_4	c_5	c_6	c_7
I_4:	h_4	d_1	d_2	d_3	d_4	d_5	d_6	d_7
I_5:	h_3	e_1	e_2	e_3	e_4	e_5	e_6	e_7
I_6:	h_2	f_1	f_2	f_3	f_4	f_5	f_6	f_7
I_7:	h_1	g_1	g_2	g_3	g_4	g_5	g_6	g_7

Figure 2: After Encryption

For example the eight characters are a, b, c, d, e, f, g and h. i.e. $I_1 = A$, $I_2 = B$, $I_3 = C$, $I_4 = D$, $I_5 = E$, $I_6 = F$, $I_7 = G$, and $I_8 = H$.

The equivalent byte values of these characters before encryption are as follows:

I_1 ASCII value 65 and its bits: 01000001, i.e. $a_1 = 1$, $a_2 = 0$, $a_3 = 0$, $a_4 = 0$, $a_5 = 0$, $a_6 = 0$, $a_7 = 1$.
I_2 ASCII value 66 and its bits: 01000010, i.e. $b_1 = 1$, $b_2 = 0$, $b_3 = 0$, $b_4 = 0$, $b_5 = 0$, $b_6 = 1$, $b_7 = 0$.
I_3 ASCII value 67 and its bits: 01000011, i.e. $c_1 = 1$, $c_2 = 0$, $c_3 = 0$, $c_4 = 0$, $c_5 = 0$, $c_6 = 1$, $c_7 = 1$.
I_4 ASCII value 68 and its bits: 01000100, i.e. $d_1 = 1$, $d_2 = 0$, $d_3 = 0$, $d_4 = 0$, $d_5 = 1$, $d_6 = 0$, $d_7 = 0$.
I_5 ASCII value 69 and its bits: 01000101, i.e. $e_1 = 1$, $e_2 = 0$, $e_3 = 0$, $e_4 = 0$, $e_5 = 1$, $e_6 = 0$, $e_7 = 1$.
I_6 ASCII value 70 and its bits: 01000110, i.e. $f_1 = 1$, $f_2 = 0$, $f_3 = 0$, $f_4 = 0$, $f_5 = 1$, $f_6 = 1$, $f_7 = 0$.
I_7 ASCII value 71 and its bits: 01000111, i.e. $g_1 = 1$, $g_2 = 0$, $g_3 = 0$, $g_4 = 0$, $g_5 = 1$, $g_6 = 1$, $g_7 = 1$.
I_8 ASCII value 72 and its bits: 01001000, i.e. $h_1 = 1$, $h_2 = 0$, $h_3 = 0$, $h_4 = 1$, $h_5 = 0$, $h_6 = 0$, $h_7 = 0$.

After encryption by using BSS method the equivalent byte values of these characters are as fallows.

I_1 bits: 01000001, i.e. $h_7 = 0$, $a_1 = 1$, $a_2 = 0$, $a_3 = 0$, $a_4 = 0$, $a_5 = 0$, $a_6 = 0$, $a_7 = 1$.
I_2 bits: 01000010, i.e. $h_6 = 0$, $b_1 = 1$, $b_2 = 0$, $b_3 = 0$, $b_4 = 0$, $b_5 = 0$, $b_6 = 1$, $b_7 = 0$.
I_3 bits: 01000011, i.e. $h_5 = 0$, $c_1 = 1$, $c_2 = 0$, $c_3 = 0$, $c_4 = 0$, $c_5 = 0$, $c_6 = 1$, $c_7 = 1$.
I_4 bits: 11000100, i.e. $h_4 = 1$, $d_1 = 1$, $d_2 = 0$, $d_3 = 0$, $d_4 = 0$, $d_5 = 1$, $d_6 = 0$, $d_7 = 0$.
I_5 bits: 01000101, i.e. $h_3 = 0$, $e_1 = 1$, $e_2 = 0$, $e_3 = 0$, $e_4 = 0$, $e_5 = 1$, $e_6 = 0$, $e_7 = 1$.
I_6 bits: 01000110, i.e. $h_2 = 0$, $f_1 = 1$, $f_2 = 0$, $f_3 = 0$, $f_4 = 0$, $f_5 = 1$, $f_6 = 1$, $f_7 = 0$.
I_7 bits: 11000111, i.e. $h_1 = 1$, $g_1 = 1$, $g_2 = 0$, $g_3 = 0$, $g_4 = 0$, $g_5 = 1$, $g_6 = 1$, $g_7 = 1$.

B. Decryption process

In decryption process every seven bytes of cipher text produces eight characters of plain text. So after decryption process the decrypted data will automatically get its original size. The fallowing figure 3 shows data before decryption, and figure 4 shows the data after decryption.

I_1:	h_7	a_1	a_2	a_3	a_4	a_5	a_6	a_7
I_2:	h_6	b_1	b_2	b_3	b_4	b_5	b_6	b_7
I_3:	h_5	c_1	c_2	c_3	c_4	c_5	c_6	c_7
I_4:	h_4	d_1	d_2	d_3	d_4	d_5	d_6	d_7
I_5:	h_3	e_1	e_2	e_3	e_4	e_5	e_6	e_7
I_6:	h_2	f_1	f_2	f_3	f_4	f_5	f_6	f_7
I_7:	h_1	g_1	g_2	g_3	g_4	g_5	g_6	g_7

Figure 3: Before Decryption

I_1:	0	a_1	a_2	a_3	a_4	a_5	a_6	a_7
I_2:	0	b_1	b_2	b_3	b_4	b_5	b_6	b_7
I_3:	0	c_1	c_2	c_3	c_4	c_5	c_6	c_7
I_4:	0	d_1	d_2	d_3	d_4	d_5	d_6	d_7
I_5:	0	e_1	e_2	e_3	e_4	e_5	e_6	e_7
I_6:	0	f_1	f_2	f_3	f_4	f_5	f_6	f_7
I_7:	0	g_1	g_2	g_3	g_4	g_5	g_6	g_7
I_8:	0	h_1	h_2	h_3	h_4	h_5	h_6	h_7

Figure 4: After Decryption

III. BSS ALGORITHMS

A. Encryption Algorithm:

Step 1: Take first 8 characters from the given text file (say "file.txt") into I_1, I_2, I_3, I_4, I_5, I_6, I_7, I_8 . Let the characters be as shown in figure 1 in binary format.

Step 2: If the number of characters less than 8 then write all the characters from text file to the encrypted file (Say "file.cmp").

Step 3: Else

Let T, T_2 be the characters.

(a) T ← 128 i.e. T = 10000000 (in binary form)
T_2 ← 0 i.e. T_2 = 00000000 (in binary form)

(b) Take 7^{th} bit from I_8, into "T" by doing the fallowing shifting and bit wise operations .

$T_2 = I_8 << 1$ (left shift)
$T = T \& T_2$ (bit wise "And" operation)

T:

h_1	0	0	0	0	0	0	0

(c) Place this 7^{th} bit of I_8, into the 8^{th} bit position of I_7, as fallows

$I_7 = T \mid I_7$ (bit wise "Or" operation)

I_7:

h_1	g_1	g_2	g_3	g_4	g_5	g_6	g_7

Step 4:

(a) T= 128, T_2 = 0;

(b) Take the 6^{th} bit from I_8 into "T" by doing the fallowing shifting and bit wise operations .

$T_2 = I_8 << 2$ (left shift)
$T = T \& T_2$ (bit wise "And" operation)

T:

h_2	0	0	0	0	0	0	0

(c) Place this 6^{th} bit of I_8, into the 8^{th} bit position of I_6, as fallows

$I_6 = T \mid I_6$ (bit wise "Or" operation)

I_6 :

h_2	f_1	f_2	f_3	f_4	f_5	f_6	f_7

Step 5:

(a) T = 128, T_2 = 0;

(b) Take the 5^{th} bit from I_8 into "T" by doing the fallowing shifting and bit wise operations .

$T_2 = I_8 << 3$ (left shift)
$T = T \& T_2$ (bit wise "And" operation)

T:

h_3	0	0	0	0	0	0	0

(c) Place this 5^{th} bit of I_8, into the 8^{th} bit position of I_5, as fallows

$I_5 = T \mid I_5$ (bit wise "Or" operation)

I_5 :

h_3	e_1	e_2	e_3	e_4	e_5	e_6	e_7

Step 6:

(a) $T = 128$, $T_2 = 0$;
(b) Take the 4^{th} bit from I_8 into "T" by doing the fallowing shifting and bit wise operations.

$T_2 = I_8 << 4$ (left shift)
$T = T \& T_2$ (bit wise "And" operation)

T :

h_4	0	0	0	0	0	0	0

(c) Place this 4^{th} bit of I_8, into the 8^{th} bit position of I_4, as fallows

$I_4 = T \mid I_4$ (bit wise "Or" operation)

I_4:

h_4	d_1	d_2	d_3	d_4	d_5	d_6	d_7

Step 7:

(a) $T = 128$, $T_2 = 0$;
(b) Take the 3^{rd} bit from I_8 into "T" by doing the fallowing shifting and bit wise operations .

$T_2 = I_8 << 5$ (left shift)
$T = T \& T_2$ (bit wise "And" operation)

T:

h_5	0	0	0	0	0	0	0

(c) Place this 3^{rd} bit of I_8, into the 8^{th} bit position of I_3, as fallows

$I_3 = T \mid I_3$ (bit wise "Or" operation)

I_3:

h_5	c_1	c_2	c_3	c_4	c_5	c_6	c_7

Step 8:

(a) $T = 128$, $T_2 = 0$;
(b) Take the 2^{nd} bit from I_8 into "T" by doing the fallowing shifting and bit wise operations .

$T_2 = I_8 << 6$ (left shift)
$T = T \& T_2$ (bit wise "And" operation)

T:

h_6	0	0	0	0	0	0	0

(c) Place this 2^{nd} bit of I_8, into the 8^{th} bit position of I_2, as fallows

$I_2 = T \mid I_2$ (bit wise "Or" operation)

I_2:

h_6	b_1	b_2	b_3	b_4	b_5	b_6	b_7

Step 9

(a) $T = 128, T_2 = 0$;
(b) Take the 1st bit from I_8 into "T" by doing the fallowing shifting and bit wise operations .

$T_2 = I_8 << 7$ (left shift)
$T = T \& T_2$ (bit wise "And" operation)

T:

h_7	0	0	0	0	0	0	0

(c) Place this 1st bit of I_8, into the 8th bit position of I_1, as fallows

$I_1 = T \mid I_1$ (bit wise "Or" operation)

I_1:

h_7	a_1	a_2	a_3	a_4	a_5	a_6	a_7

Step 10:

Write these 7 characters $I_1, I_2, I_3, I4, I_5, I_6$ and I_7 to the encrypted file say "file.cmp"

After process these 7 characters are as shown in fig 2.

B. Decryption algorithm

Step 1:

Take first 7 characters from the given encrypted file (say "file.cmp") into $I_1, I_2, I_3, I_4, I_5, I_6, I_7$.
Let the binary format of these characters are as shown in figure 3.

Step 2:

If the number of characters less than 7 then write all these characters from encrypted file to the decrypted text file (say "file.txt").

Step 3:

Else
Let $T, T_2, D_1, D_2, D_3, D_4, D_5, D_6, D_7, D_8$ are the characters.

(a) $T \leftarrow 127$ i.e. T = 01111111 (in binary form)
$T_2 \leftarrow 1$ i.e. T_2 = 00000001 (in binary form)

(b) Replace 1 to 7 bits of D_1 with 1 to 7 bits of I_1 respectively and 8th bit of D_1 with 0 with the following operations.

$D_1 \leftarrow I_1 \& T$ (bitwise "And" operation)

D_1:

0	a_1	a_2	a_3	a_4	a_5	a_6	a_7

(c) Replace 1st bit of I_1 with 8th bit of I_1 and the remaining bits with '0' with the following operations.

$I_1 \leftarrow (I_1 >> 7) \& T_2$

I_1:

0	0	0	0	0	0	0	h_7

Step 4:

(a) $T \leftarrow 127$ i.e. T = 01111111 (in binary form)
$T_2 \leftarrow 2$ i.e. T_2=00000010 (in binary form)

(b) Replace 1 to 7 bits of D_2 with 1 to 7 bits of I_2 respectively and 8^{th} bit of D_2 with 0 with the following operations.

$D \leftarrow I_2 \& T$ (bitwise "And" operation)

D_2:	0	b_1	b_2	b_3	b_4	b_5	b_6	b_7

(c) Replace 2^{nd} bit of I_2 with 8^{th} bit of I_2 and the remaining bits with '0' with the following operations.

$I_2 \leftarrow (I_2 >> 6) \& T_2$

I_2:	0	0	0	0	0	0	h_6	0

Step 5:

(a) $T \leftarrow 127$ i.e. T = 01111111 (in binary form)
$T_2 \leftarrow 4$ i.e. T_2=00000100(in binary form)

(b) Replace 1 to 7 bits of D_3 with 1 to 7 bits of I_3 respectively and 8^{th} bit of D_3 with 0 with the following operations.

$D_3 \leftarrow I_3 \& T$ (bitwise "And" operation)

D_3:	0	c_1	c_2	c_3	c_4	c_5	c_6	c_7

(c) Replace 3^{rd} bit of I_3 with 8^{th} bit of I_3 and the remaining bits with '0' with the following operations.

$I_3 \leftarrow (I_3 >> 5) \& T_2$

I_3:	0	0	0	0	0	h_5	0	0

Step 6:

(a) $T \leftarrow 127$ i.e. T = 01111111 (in binary form)
$T_2 \leftarrow 8$ i.e. T_2 = 00001000 (in binary form)

(b) Replace 1 to 7 bits of D_4 with 1 to 7 bits of I_4 respectively and 8^{th} bit of D_4 with 0 with the following operations.

$D_4 \leftarrow I_4 \& T$ (bitwise "And" operation)

D:	0	d_1	d_2	d_3	d_4	d_5	d_6	d_7

(c) Replace 4^{th} bit of I_4 with 8^{th} bit of I_4 and the remaining bits with '0' with the following operations.

$I_4 \leftarrow (I_4 >> 4) \& T_2$

I_4:	0	0	0	0	h_4	0	0	0

Step 7

(a) $T \leftarrow 127$ i.e. $T = 01111111$ (in binary form)
$T_2 \leftarrow 16$ i.e. $T_2 = 00010000$ (in binary form)

(b) Replace 1 to 7 bits of D_5 with 1 to 7 bits of I_5 respectively and 8^{th} bit of D_5 with 0 with the following operations.

$D_5 \leftarrow I_5 \& T$ (bitwise "And" operation)

D_5:

0	e_1	e_2	e_3	e_4	e_5	e_6	e_7

(c) Replace 5^{th} bit of I_5 with 8^{th} bit of I_5 and the remaining bits with '0' with the following operations.

$I_5 \leftarrow (I_5 >> 3) \& T_2$

I_5:

0	0	0	h_3	0	0	0	0

Step 8:

(a) $T \leftarrow 127$ i.e. $T = 01111111$ (in binary form)
$T_2 \leftarrow 32$ i.e. $T_2 = 00100000$ (in binary form)

(b) Replace 1 to 7 bits of D_6 with 1 to 7 bits of I_6 respectively and 8^{th} bit of D_6 with 0 with the following operations.

$D_6 \leftarrow I_6 \& T$ (bitwise "And" operation)

D_6:

0	f_1	f_2	f_3	f_4	f_5	f_6	f_7

(c) Replace 6^{th} bit of I_6 with 8^{th} bit of I_6 and the remaining bits with '0' with the following operations.

$I_6 \leftarrow (I_6 >> 2) \& T_2$

I_6:

0	0	h_2	0	0	0	0	0

Step 9:

(a) $T \leftarrow 127$ i.e. $T = 01111111$ (in binary form)
$T_2 \leftarrow 64$ i.e. $T_2 = 01000000$ (in binary form)

(b) Replace 1 to 7 bits of D_7 with 1 to 7 bits of I_7 respectively and 8^{th} bit of D_7 with 0 with the following operations.

$D_7 \leftarrow I_6 \& T$ (bitwise "And" operation)

D_7:

0	g_1	g_2	g_3	g_4	g_5	g_6	g_7

(c) Replace 7^{th} bit of I_7 with 8^{th} bit of I_7 and the remaining bits with '0' with the following operations.

$I_7 \leftarrow (I_7 >> 1) \& T_2$

I_7:

0	h_1	0	0	0	0	0	0

Step 10:

Replace the 1 to 7 bits of D_8 with the bits by doing the fallowing operation

$D_8 \leftarrow I_1 \mid I_2 \mid I_3 \mid I_4 \mid I_5 \mid I_6 \mid I_7$ (bitwise "OR" operations)

D_8:

0	h_1	h_2	h_3	h_4	h_5	h_6	h_7

Step 11:

Write D_1, D_2, D_3, D_4, D_5, D_6, D_7 and D_8 characters to decrypted file (" file.txt ").

Step 12:

Take next 7 characters into I_1, I_2, I_3, I_4, I_5, I_6, I_7 and repeat the Steps 2 to 11 until the end of the encrypted file.

Step 13: END.

IV. IMPLEMENTATION RESULTS AND DISCUSSIONS.

We have applied this BSS system on different sizes of data to encryption and decryption

A. Sample test data

- *Encryption*

Actual message

Steganography is the art of science that communication is being happening through audio, video and still images

Encryption

Óôågaîïòáphù é tèe áò oæ sãénce
Šôát cïíínéãátéï éó båég èápðåing
Šôòoõgè ádiï, öéåï aîä ôill éíges

Encrypted message

- *Decryption*

Encrypted message

Óôågaîïòáphù é tèe áò oæ sãénce
Šôát cïíínéãátéï éó båég èápðåing
Šôòoõgè ádiï, öéåï aîä ôill éíges

Decryption

Steganography is the art of science that communication is being happening through audio, video and still images

Actual message

B. Analysis

After encryption the size of the encrypted data is reduced and after decryption the size of the decrypted data is increased, i.e original size of the actual data. The following table1 represents the variation of size of different data sets after encryption and decryption.

TABLE I

S. No	***Data Size (bytes)***	***After Encryption Size (bytes)***	***After Decryption Size (bytes)***
1	3,250	2,844	3,250
2	3,254	2,848	3,254
3	9,741	8,524	9.741
4	9,746	8,528	9.746
5	9,747	8,532	9.750
6	29,161	25.516	29,161
7	29,165	25.520	29,165

V. CONCLUSION

In this paper we presented an implementation of BSS encryption algorithm. The main objective was to evaluate the performance of this algorithm in terms of data size. The results showed that the BSS algorithm was very effective in complexity and security. We extending this to our future work, we are going to provide a secret key for authentication of the encrypted data and decrypted data.

REFERENCES

[1] Wikipedia, "*Encryption*", http://en.wikipedia.org/wiki/Encryption, modified on 13 December 2006.
[2] J. Freeman, R. Neely, and L. Megalo "*Developing Secure Systems: Issues and Solutions*". IEEE Journal of Computer and Communication, Vol. 89, PP. 36-45. 1998
[3] Wayne G. Barker, "Introduction to the analysis of the Data Encryption Standard (DES)", A cryptographic Series, Vol. 55, p. viii + 190, Aegean Park Press, 1991.
[4] N. A Kofahi, Turki Al-Somani, Khalid Al-Zamil. "Performance evaluation of three encryption/decryption algorithms" 2005 IEEE International Symposium on Micro-NanoMechatronics and Human Science, Publication Date: 30-30 Dec. 2003. Volume: 2, pp 790-793
[5] W. Stallings, "Cryptography and Network Security: Principles and Practice", 2nd Edition, pgs. 102-109, 128, 1999.
[6] X. Lai, J. Massey, "A Proposal for a New Block Encryption Standard", Proceedings, EUROCRYPT '90, 1990.
[7] Diffie, Whitfield & Hellman, Martin E, (1976) . New Directions In Cryptography, IEEE TRANSACTIONS ON INFORMATION THEORY, available from: http://www- ee.stanford.edu/~hellman/publications/24.pdf
[8] C.-P. Wu, C.-C. J. Kuo, "Design of integrated multimedia compression and encryption systems," IEEE Trans. Multimedia, vol. 7, no. 5, pp. 828-839, 2005.
[9] M . J. Weiner, "Efficient DES Key Search," Advances in Cryptology—CRYPTO '93 Proceedings, Springer- Verlag, in preparation.
[10] E. Biham and A. Shamir, Differential Cryptanalysis of the Data Encryption Standard, Springer-Verlag, 1993.
[11] M. Matsui, "Linear Cryptanalysis Method for DES Cipher," Advances in Cryptology-CRYPTO '93Proceedings, Springer- Verlag, 1994, in preparation