Computer Engineering and Intelligent Systems
ISSN 2222-1719 (Paper) ISSN 2222-2863 (Online)
Vol 2, No.3

An Authenticated Bit Shifting and Stuffing (BSS) Methodology for Data Security

B. Ravi Kumar[1], P.R.K.Murti[2], B. Hemanth Kumar[3]

[1,2] Department of Computer and Information Sciences, University of Hyderabad,

P.O. Central University, Gachibowli, Hyderabad 500 046, Andhra Pradesh, INDIA.

Email:[1]ravi_budithi@yahoo.com & [2]murti.poolla@gmail.com

[3] Department of IT, R.V.R.& J.C. College of Engineering ,

Guntur, Andhra Pradesh, INDIA.

Email: [3] bhkumar_2000@yahoo.com

Abstract

Providing security to the data means the third party cannot interpret the actual information. When providing authentication to the data then only authorized persons can interpret the data. One of the methodology to provide security is cryptography. But in previous paper we have proposed a methodology for the cryptography process is *BSS*. 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. To provide authentication a four bit dynamic key is generated for every four characters of the encrypted data and the key is also maintained in the data itself. While decryption the key is retrieved from the data and check whether the data is corrupted or not.

Key Words: Bit Shifting, Stuffing, Security, authentication.

1. Introduction

In our previous paper (Kumar 2011) we discussed how encryption and decryption process is done by using BSS methodology. In this paper we generate a dynamic key for data authentication and security. 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) (Wikipedia 2006). 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. The algorithm used to encrypt data is called a cipher, or cipher text which is representation of the original data in a difference form (Freeman 1998), 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 security of encrypted data depends on several factors like what algorithm is used, what is the key and how was the algorithm implemented in the product.

In section 2 we have discussed about the previous work of the proposed system, in section 3 we give our proposed system, in section 4 we have given the methodology and algorithms for generating dynamic key

encryption and decryption for the proposed system, in section 5 implementation results with discussions, section 6 conclusions and future work.

2. Previous Work

The system deals with security of data by using BSS encryption and decryption (Kumar 2011).

2.1 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

2.2 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

3. PROPOSED SYSTEM

In our proposed system, we have presented new algorithms named Bits Shifting and Stuffing (BSS) methodology(Kumar 2011). Commonly in encryption or decryption process some of the characters are inter changed by using some encryption and decryption algorithms(Beth 1989, IBM 1994, Lai 1990, Bruce Schneier 1994)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. Then for **Authentication** and Security we generating key, by using a random function generating a 4 bit polynomial. Take 4 characters from the encrypted file and by using this polynomial perform modulo- 2 division operation on these 4 characters. we will get a remainder of 3bits. In the second step of encryption these 4 characters, polynomial and the remainder are adjusted in 5 bytes. These five bytes are maintained separately in another file. Like for every 4 characters of first encrypted file after performing modulo-2 division operation, 5 bytes are maintained in the file.

4. Methodology

The system deals with security and authentication of data by using BSS encryption and decryption by using dynamic key. Figures 5 and 8 shows the architectural *Diagrams for authentication process.*

4.1 Encryption with Dynamic key

By using a random function generate a 4 bit Dynamic polynomial. Take 4 characters from the encrypted file and by using this polynomial perform modulo- 2 division operation on these 4 characters. You will get a remainder of 3bits. In the second step of encryption these 4 characters, polynomial and the remainder are adjusted in 5 bytes. These five bytes are maintained separately in another file (say encr2.). Like for every 4 characters of first encrypted file after performing modulo-2 division operation, 5 bytes are maintained in the file encr2. Figure 5 shows the architecture *for Encryption with Key and Authentication Process.* The original encrypted data by using BSS algorithm for as shown in figure 6. After generating 4 bit (p1,p2,p3,p4) dynamic key , perform modulo 2 division on these 4 characters and the remainder will be (r1,r2,r3). The key and remainder is embedded in the data itself as shown in figure 7.

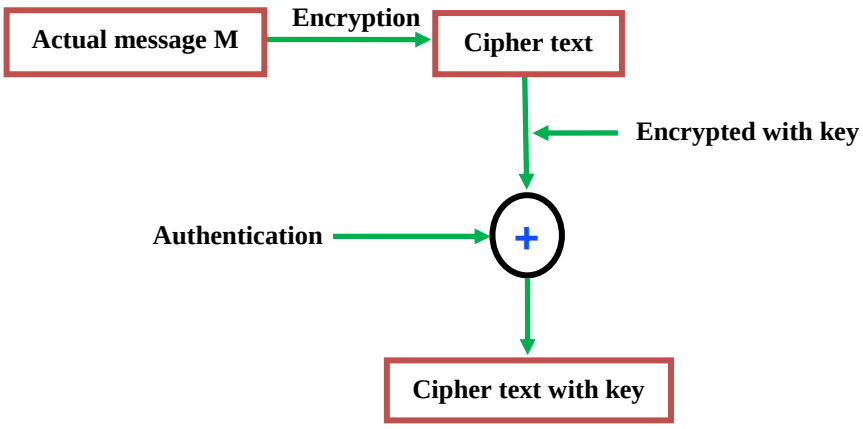


Figure 5: Architecture for Encryption with Key

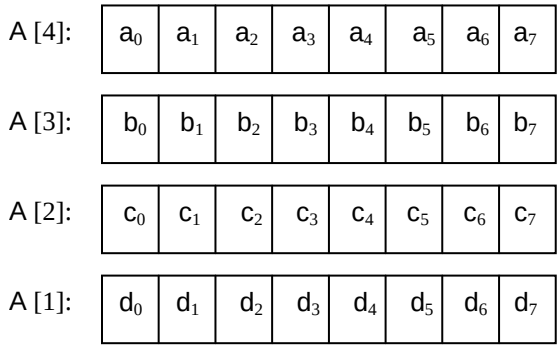


Figure 6: original encrypted message

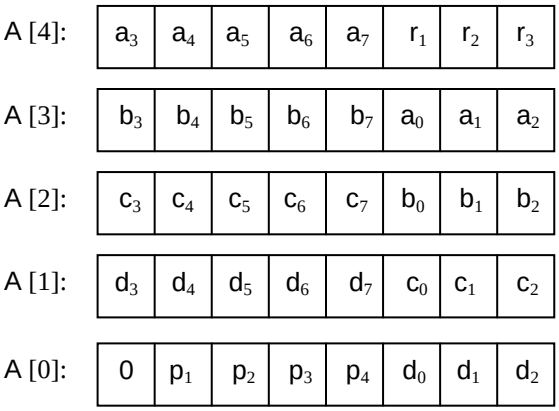


Figure 7:encrypted message with key

4.2 Dynamic Polynomial

In this algorithm we are using random() method which generates random number and we are using a number which is between 8 and 15 i.e 4bit number and this function returns this number known as the POLY.

Integer dyn_poly()

1. while TRUE
2. POLY ← Rand()%8 + 8 // (8 to 15)
3. If (Mod_2div (0x02 ,POLY, 0x00,2) && Mod_2div(0x03 ,POLY, 0x00,2)) then
4. return POLY

   End while
5. End.

4.3 Modulus 2 Division

POLY.PREPOS,A are characters and PSIZE Integer.

Mod_2div(POLY, A, PREPOS, PSIZE)

1. Let CH and MASK are characters

   MASK ← 0x07 (Hexadecimal number)
2. For I 1 to 8
   a) CH ← A >> (8-I)

      CH ←CH & 0x01 // (" & " AND operator)
   b) PREPOS ← PREPOS & MASK

      PREPOS ← PREPOS << 1

      PREPOS ← PREPOS | CH // (" | " OR operator)
   c) If PREPOS < PSIZE then

      PREPOS ← PREPOS ^ 0x00 //(" ^ " Exclusive OR)

      Else

      PREPOS ← PREPOS ^ POLY

d) Ene for loop

3. return PREPOS
4. End.

4.4 Authentication

Crc32() takes Poly and array of characters AR[ ] and returns CRC

1. CRC ← 0x00 (hexadecimal numbers)
2. For J is 0 to 4
3. CRC ← Mod _2div(POLY , AR[J] ,CRC)
4. Return CRC
5. End.

4.5 Encryption with key Process

It takes the encrypted data FILE as argument.

1. Let A[5] is the character array of 5 elements
   Take first 4 characters into A[4], A[3], A[2], A[1]
   Initially the characters as shown in figure 6.
2. Do the fallowing operations
   Let T and T1 are temporary variables
   a) T1 ← 0x00
   b) For J is 3 to 0
      T ←A[J + 1]
      A[J + 1] ← A[J + 1] << 3
      A[J + 1] ← A[J + 1] & 0xf8
      A[J + 1] ← A[J + 1] | T1
      T1 ← T >>5
      T1 ← T1 & 0x07
      End for.
      A[0] ← T1
      After these operations this array is as fallows.

A [4]:	a_3	a_4	a_5	a_6	a_7	0	0	0
A [3]:	b_3	b_4	b_5	b_6	b_7	a_0	a_1	a_2
A [2]:	c_3	c_4	c_5	c_6	c_7	b_0	b_1	b_2
A [1]:	d_3	d_4	d_5	d_6	d_7	c_0	c_1	c_2
A [0]:	0	0	0	0	0	d_0	d_1	d_2

3. POLY ← dyn_poly()
   POLY ← POLY << 3

POLY:	0	p_1	p_2	p_3	p_4	0	0	0

4. A[0] ← A[0] | POLY

A [0]:	0	p_1	p_2	p_3	p_4	d_0	d_1	d_2

5. CRC ← Crc32(POLY, A[ ])
   CRC ← CRC & 0x07

CRC:	0	0	0	0	0	r_1	r_2	r_3

6. A[4] ← A[4] | CRC

A [4]:	a_3	a_4	a_5	a_6	a_7	r_1	r_2	r_3

7. Write these 5 characters A[0], A[1], A[2], A[3], and A[4] to the file.
8. Take next 4 characters into the array A [ ] and continue steps from 2 to 8 until the end of the file.
9. End.

4.6 Decryption process

The architecture of the decryption process is as shown in figure 8. 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. Figure 3 shows data before decryption, and figure 4 shows the data after decryption.

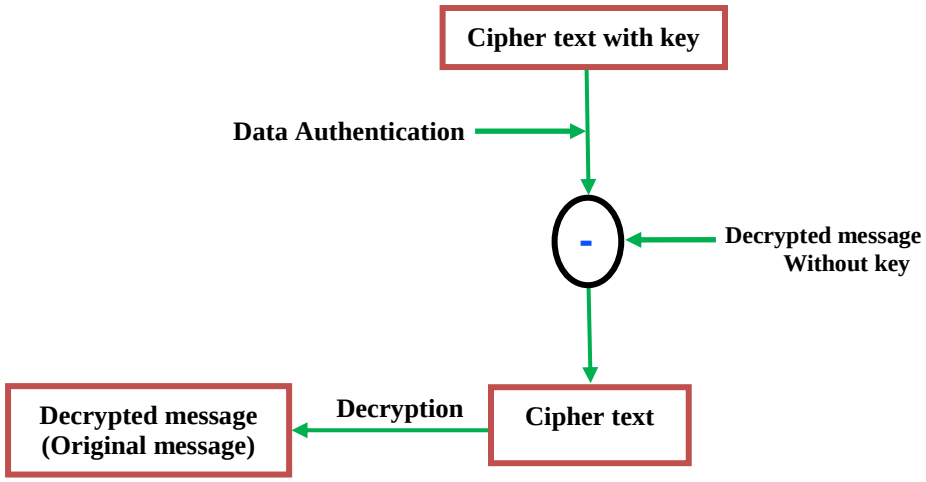


Figure.8: Architecture for extracting data from Decryption

4.7 Data Authentication

Take five characters from the retrieved data file (decypt2.) as shown in figure 9 and detect the key from the fifth character and by using this key perform modulo 2 division on these five characters. If the remainder is zero then there is no corruption in the data otherwise data is corrupted. If the data is not corrupted then remove key and remainder from the five characters and arrange the bits in 4 bytes to get the encrypted data as shown in figure 10. These 4 bytes are maintained separately in a special file say dect1. Like that for every 5 bytes from decypt1. after removing key and remainder the 4bytes are maintained in the file dect1.This is the encrypted file without key.

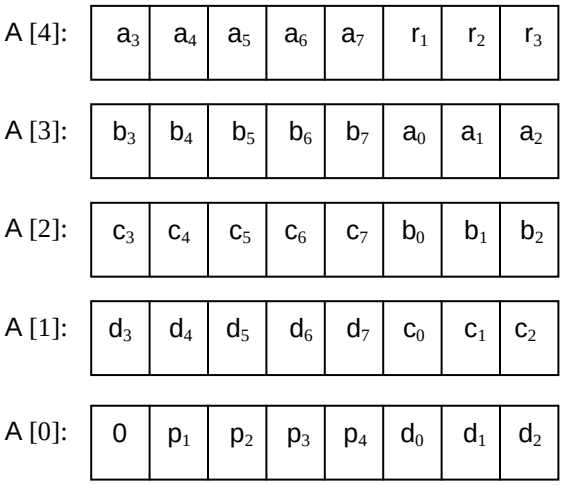


Figure 9: Encrypted message with key

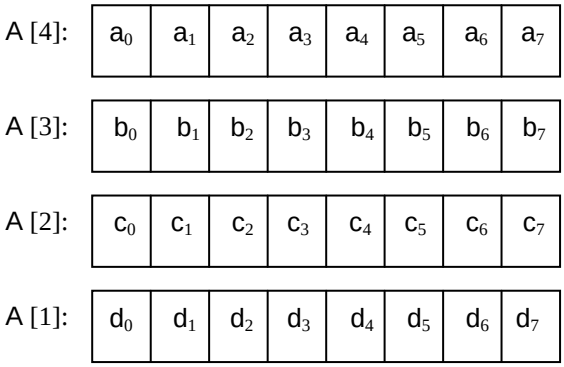


Figure 10: Authenticated encrypted message

4.7.1 Finding polynomial from the data and verifying crc by modulo 2 division**.**

1. Take first five characters into the array A[5] of elements A[0],A[1],A[2],A[3] and A[4]
2. Finding polynomial

   POLY ← A[0] >> 3

   POLY ← POLY & 0x0f
3. RMAINDER ← Crc 32(POLY, A[])
4. If REMAINDER is ZERO then
   a) verified successfully and no error in data.
   b) Take array B[4] , T
   c) For J is 3 to 0

      B[J] ← A[J + 1]

      B[J] ←B[J] >> 3

B[J] ←B[J] & 0x1f

T ← A[J]

T ← T << 5

T ← T & 0xe0

B[J] ← T | B[J]

End for loop

d) Write these 4 character array B[ ] to the file.

e) Repeat these steps until the end of the file.

5. Else

Data corrupted

End.

5. Implementation Results and Discussions

We have applied this BSS system on different sizes of data to encryption and decryption and data is authenticated and secure with key

5.1 Sample Test Data

5.1.1 Encryption

The size of the actual image here is 4,608 bytes and its dimensions is 160 × 120 the actual message size is 175 bytes after the encryption the size of the encrypted message is 156 bytes and the size encrypted with key is 195 bytes.

Actual message (232 bytes)

Encryption

Ôhå séú of tèeáctõaì máge èåe és 4¶0¸ bùôs
ánä éó diíåîions és1¶0 × 1° the áôõal íåságå óéå
és ±7 byteó ætår ôè eîcòùðion tèesize ïftèe
åîãùpôed íóóagå é 1µ6 âües aîä he
siúeånãryðôä wétè åy éó 15 âùtås

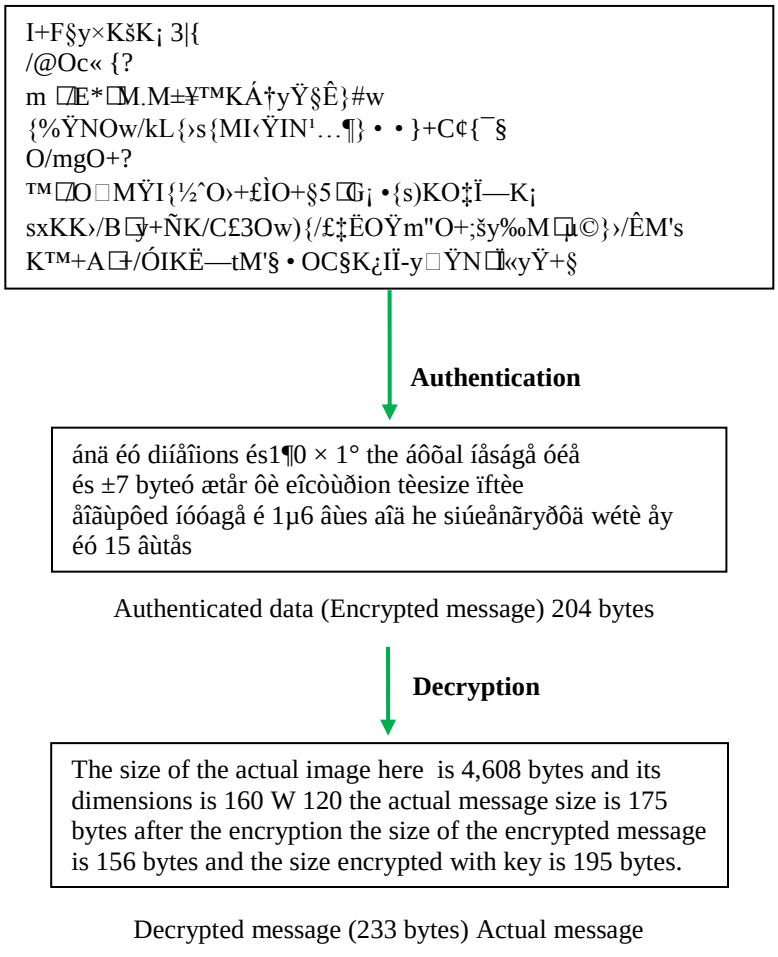
Encrypted message (204 bytes)

Authentication with key

I+F§y×KšK¡ 3|{
/@Oc« {?
m □Æ*□M.M±¥™KÁ†yŸ§Ê}#w
{%ŸNOw/kL{›s{MI‹ŸIN¹…¶} • • }+C¢{¯§
O/mgO+?
™□O□MŸI{½ˆO›+£ÌO+§5□Ğ¡ •{s)KO‡Ï—K¡
sxKK›/B□y+ÑK/C£3Ow){/£‡ËOŸm"O+;šy‰M□µ©}›/ÊM's
K™+A□+/ÓIKË—tM'§ • OC§K¿IÏ-y□ŸN□Ï«yŸ+§

5.1.2 Decryption

Encrypted message with key (255 bytes)



5.2 Analysis

After encryption the size of the encrypted data is reduced and after the dynamic key the size increases and then after decryption the size of the decrypted data is decreased, i.e original size of the actual data. The following table1 represents the variation of size of different data sets after encryption and decryption with key.

TABLE 1

SNO	Data size (bytes)	After encryption Size (bytes)	Encrypted data with key (bytes)	After decryption size (bytes)
1	3,250	2,844	3,555	3,250

2	3,254	2,848	3,560	3,254
3	9,741	8,524	10,655	9.741
4	9,746	8,528	10,660	9.746
5	9,747	8,532	10,665	9.750
6	29,161	25.516	31,895	29,161
7	29,165	25.520	31,900	29,165

Table1: Represents the variation of size of different data sets after encryption and decryption with key.

6. Conclusions

In this paper we presented an implementation of BSS encryption algorithm with dynamic key. The main objective was to evaluate the performance of this algorithm in terms of data size and authentication and security. The results showed that the BSS algorithm with dynamic key was very effective in complexity and security. In our future work with this methodology we are combining cryptograpy with setganograpy to achive data security and authentication.

References

Wikipedia, "*Encryption*", http://en.wikipedia.org/wiki/Encryption, modified on 13 December 2006.

Freeman, J. Neely, R. & Megalo, L.(1998). "*Developing Secure Systems: Issues and Solutions*". IEEE Journal of Computer And Communication, Vol. 89, PP. 36-45.

Kumar, B.R. Murti, P.R.K, Dr.,(July2011). "Data Encryption and Decryption process Using Bit Shifting and Stuffing (BSS) Methodology" International Journal on Computer Science and Engineering (IJCSE) Vol. 3 No. 7, pp. 2818-2827

Beth, T. & Gollmann, D.(1989). "*Algorithm Engineering for Public Key Algorithms*". IEEE Journal on Selected Areas in Communications; Vol. 7, No 4, PP. 458-466.

IBM. "*The Data Encryption Standard (DES) and its strength against attacks*". IBM Journal of Research and Development, Vol. 38, PP. 243-250. 1994

Lai, X. Massey, J. (1990) "A Proposal for a New Block Encryption Standard", Proceedings, EUROCRYPT '90.
Bruce Schneier,(April 1994). "The Blowfish encryption algorithm", Dr. Dobb's Journal of Software Tools, 19(4),p. 38, 40, 98, 99.

Kofahi, N.A., Turki Al-Somani, Khalid Al-Zamil (Dec.2003). "Performance evaluation of three encryption/decryption algorithms" 2005 IEEE International Symposium on Micro-NanoMechatronics and Human Science, Volume: 2, pp 790-793

Stallings, W.,(1999). "Cryptography and Network Security: Principles and Practice", 2nd Edition, pgs. 102-109, 128.

Wayne G. Barker,(1991). "Introduction to the analysis of the Data Encryption Standard (DES)", A cryptograph-ic series, Vol. 55, p. viii + 190, Aegean Park Press.

Wu, C.-P. Kuo, C.-C. J. (2005). "Design of integrated multimedia compression and encryption systems," IEEE Trans. Multimedia, vol. 7, no. 5, pp. 828-839.

Biham, E. & Shamir, A.(1993). Differential Cryptanalysis of the Data Encryption Standard, Springer-Verlag.

Matsui, M.(1994). "Linear Cryptanalysis Method for DES Cipher," Advances in Cryptology-CRYPTO '93 Proceedings, Springer-Verlag.