International Journal of Computer Applications (0975 - 8887)
Volume 181 - No.25, November 2018

An Alternate Key Scheduling Algorithm for Blowfish and its Performance Analysis

Rekha C.
BNM institute of Technology
Dept. of Computer Science and Engg
Bangalore

Krishnamurthy G. N.
BNM institute of Technology
Dept. of Computer Science and Engg
Bangalore

Dilip Kumar S. M.
UV College of Engg
Dept. of Computer Science and Engg
Bangalore

ABSTRACT

Information security depends on the strength of cryptographic algorithm and key. The generated keys must be so secure that there is no better way to break it. Design of such good key scheduling algorithm is crucial part in symmetric cryptosystem, which is used to create a number of subkeys, used in encryption/decryption process in block cipher. In this paper a methodology is proposed for generating keys using an alternate key scheduling algorithm of blowfish. However, Blowfish has some demerits including complex key scheduling algorithm, high computational cost and static substitution of S-box. Therefore such demerits are taken care to improve the performance of key generation algorithm of blowfish algorithm. This alternating key scheduling algorithm decreases the computational cost and uses the dynamic substitution of S-box. The effectiveness of the proposed scheme is verified by performing security analysis and also metrics evaluation.

Keywords

Blowfish, Key Scheduling Algorithm, S-box

1. INTRODUCTION

Cryptography plays a very vital role in keeping the message confidential while sending from one end and should be received only by the intended receiver at the other end. The security of the cryptosystem is depended on the strength of the algorithm and key. The key must be so secure that there is no better way to break it. The cryptosystem uses a primary key to generate subkeys by using a good key scheduling algorithm which is a crucial part of symmetric cryptosystem. There are many cryptosystems which are developed and used to secure the information.
The blowfish algorithm [9], [11], [3], is an efficient 16 round Feistel network which consists of 64 bit block size, a variable key length from 32 to 448 bits and key dependent S-box. Blowfish algorithm has two parts data encryption/decryption and key expansion. Before data encryption/decryption it requires pre-computation of P-box and S-box which is a key expansion procedure.

1.1 Key expansion of Blowfish

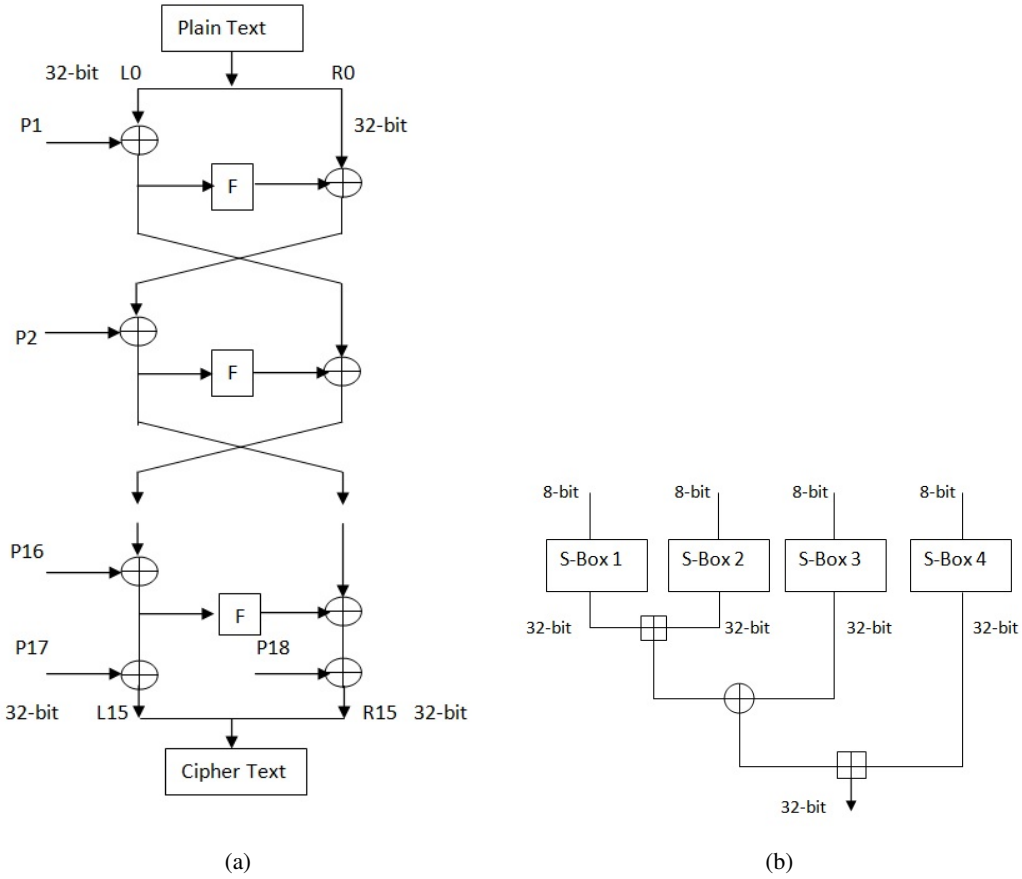
In Key expansion procedure, a key is converted into several subkeys [9]. In order to generate these subkeys, the algorithm uses 18, 32-bit P-array and 256 entries for each of the 4 key dependent S-boxes. The steps involved in generating subkeys are :

(1) 18 P-array and 4 S-boxes are initialized with hexadecimal value of II.
(2) XOR 18 P-arrays with key like P1 is XOR with first 32-bit of key, P2 is XOR with second 32-bit of key and so on for all remaining P-arrays are XOR with all values of key.
(3) After initialization of P-array with key, the first two values of P-array (P0 and P1) are encrypted with blowfish algorithm (in Fig. 1a, as L0 and R0). Replace the P0 and P1 with the output of blowfish algorithm (L17 and R17). Next two P-values (P2 and P3 as L0 and R0) are passed to encryption function and the output is stored in P2 and P3.
(4) continue the process, replacing all 18 P-values and four S-boxes in order with the output of the blowfish algorithm which changes continuously.

The subkeys are generated using key exapnsion procedure with blowfish algorithm is shown in Fig 1a. The key expansion procedure of blowfish algorithm is very complex, because generation of subkeys occurs via 16 round blowfish algorithm using
$L_i = L_{i-1}$
$R_i = L_{i-1} \bigoplus F(R_{i-1}, K_{i-1})$
in order to generate two values of P-array, where 64 bit is divided into two 32 bit halves L and R, the value of L_i and R_i of the i^{th} round are determined from the outputs of the previous round, that is L_{i-1} and R_{i-1} , K_i is i^{th} subkey and F is function (as shown in Fig. 2) used for substitution. From Fig. 1b, the F-function takes input as 32-bit value, which splits into four 8-bit quarters say a, b, c, and d, and are used as input to the S-boxes. The output from the S-boxes are then added and XORed to get 32-bit value as output.
$F(X_L) = (((S_{1,a} + S_{2,b}) \bigoplus S_{3,c}) + S_{4,c})$
The F-function uses static S-box entry like $S[1, a], S[2, b], S[3, c]$ and $S[4.d]$. From this we can easily find out that $'a'$ value is taken from first S-box, $'b'$ value taken from second S-box, $'c'$ value taken from third S-box and $'d'$ value taken from fourth S-box.

2. MOTIVATION

Many researchers have implemented blowfish algorithm either through software or hardware. These hardware and software implementations are based on the ease of use, ease of upgrade, speed, portability, and flexibility. The scope of this work is the design of



(a) (b)

Fig. 1: (a) Block Diagram of Blowfish algorithm. (b) Block Diagram of F-function in Blowfish algorithm.

key scheduling algorithm, which is complex and high computational cost, in order to reduce the computational cost of blowfish algorithm.
The main motivation of this work is to design and implementation of an alternate key scheduling algorithm to overcome the demerits of blowfish algorithm. In this paper, a simple key scheduling algorithm to overcome the initialization and computational cost as well as static substitution of S-box is designed and implemented.

3. RELATED WORK

The blowfish algorithm is fast and useful cryptosystem. many implementation has been carried out either through software or hardware. But very few implementation has been proposed to implement key scheduling algorithm for blowfish algorithm.
B. Schneier [9], proposed a new secret key block cipher, blowfish, which is a Feistel network with block size 64 bit and a variable key of size 32-bit to 448-bit. the algorithm implemented with a complex initialization and large data caches of 32-bit microprocessor.
P. Karthigaikumar et.al [7], implemented a novel VLSI architecture of blowfish algorithm based on partial pipelined structure. In this paper the author has used two different techniques of modified Feistel function, first is iterative method and, second is partially pipelined technique. A four stage pipelined architecture is used along with two iterations increases the throughput of the algorithm at the cost of increasing the area when compared to two stage pipelining with eight iterations which will reduce the area occupied with less throughput.
B. Schneier and D.Whiting [2], discusses general optimization principles of designing the algorithms, and performance analyzes of RC4, SEAL, RC5, Blowfish, and Khufu/Khafre on the Intel Pentium with respect to those principles.
Y.-K.LaiandY.-C.Shu [6], presented, a novel VLSI architecture of the BLOWFISH,which is based on the loop-folding technique combined with secure modes (ECB, CBC2, CFB2 and OFB2) of operation. The architecture uses a prototype chip to implement by using 0.35 /spl mu/ CMOS technology.
T.S.B.,Sudarshan et.al [12], presented DRIL architecture, which is a four tier architecture involving both software and hardware designs, for implementing blowfish algorithm using architectural features like inner loop pipelining and loop folding with dynamic reconfiguration.

4. PROPOSED KEY SCHEDULING ALGORITHM OF BLOWFISH

In this section,the key scheduling algorithm for blowfish is presented. The blowfish algorithm takes master key K and performs a

key scheduling algorithm through encryption procedure to generate 18 P-arrays of 32-bit each. Before generating subkeys, P-arrays should be initialized with Pi value. Then the key scheduling algorithm uses 9 iterations with 16 rounds of encryption function in order to generate 18 P- values, which takes high computational cost for initialization and generation of subkeys. In this paper, we have presented a modified key scheduling algorithm for blowfish which reduces the computational cost as well as dynamic S-box substitution in F-function. The algorithm takes input as 16-byte key as K0K1K2K3K4K5K6K7K8K9K10K11K12K13K14K15K16, which will be treated as $4 * 4$ matrix as:

$$Key = \begin{vmatrix} K0 & K1 & K2 & K3 \\ K4 & K5 & K6 & K7 \\ K8 & K9 & K10 & K11 \\ K12 & K13 & K14 & K15 \end{vmatrix}$$

Using this matrix all 18 p-values will be generated. All elements of first row $K0K1K2K3$ of the above matrix will be XORed with all elements of second row $K4K5K6K7$ of the matrix to generated 32 bit value which is stored in tr variable.
Like,
$tr(0) = [K0 \bigoplus K4K1 \bigoplus K5K2 \bigoplus K6K3 \bigoplus K7]$,
similarly,

$tr(1) = [K4 \bigoplus K8K5 \bigoplus K9K6 \bigoplus K10K7 \bigoplus K11]$,
$tr(2) = [K8 \bigoplus K12K9 \bigoplus K13K10 \bigoplus K14K11 \bigoplus K15]$,
$tr(3) = [K12 \bigoplus K0K13 \bigoplus K1K14 \bigoplus K2K14 \bigoplus K3]$.
Each value of tr i.e. $tr(0), tr(1), tr(2)$,and $tr(3)$ are passed to the function $F - key$, which generates 4 32-bit values $'Ro, R1, R2, R3'$. Same procedure is applied to column to generate $tc(0), tc(1), tc(2)$, and $tc(3)$, and pass each tc value to function $F - key$ to generate 4 32-bit values $'Q0, Q1, Q2, Q3'$. The P-array value is generated by
$T_i = R_i \bigoplus Q_i$
$P = T \bigoplus (diagonalelementfromT)$
Finally initialize $K0K1K2K3$ by $T(0)$, $K5K6K7K8$ = $T(1)$, $K8K9K10K11$ = $T(2)$, $K12K13K14K15$ = $T(3)$. Repeat this procedure until all 18 P-array values are generated.
The flowchart of the proposed key scheduling algorithm and the F-key function is given in Fig. 2a and Fig. 2b and the algorithm for proposed key scheduling algorithm and F-key function is given in section IV A.

4.1 proposed algorithm

Algorithm 1 F-key Function

1: **procedure** F-KEY FUNCTION(X)
2: The F-Key function takes input $'X'$ as 32-bit and produces 32-bit value.
3: a = X & $0x000000ff$; b = X & $0x000000f0$;
4: c = X & $0x0000ff00$; d = X & $0x0000f000$;
5: e = X & $0x00ff0000$; f = X & $0x00f00000$;
6: g = X & $0xff000000$; h = X & $0xf0000000$;
7: $y = S[b \mod 4][a] + S[d \mod 4][c]$;
8: $y = y \bigoplus S[f \mod 4][e]$;
9: $y = y + S[h \mod 4][g]$;
10: returns $'y'$ value.
11: Repeat step 1 to step 8, to get all 18 values of P-array.

Algorithm 2 Key Scheduling algorithm

1: **procedure** KEY SCHEDULING ALGORITHM
2: **for** i = 0 to N + 2 **do**
3: **for** j = 0 to 3 **do**
4: **for** k = 0 to 3 **do**
5: each elements of first row is XORed with each elements of second row tr = $(key[4 * j + k] \bigoplus key[((4 * j) + (k + 4)) \mod 16])$
$temp1(j) = F_k ey(c, tr)$
6: **for** j = 0 to 3 **do**
7: **for** k = 0 to 3 **do**
8: each elements of first row is XORed with each elements of second row
9: tc = $(key[4 * k + j] \bigoplus key[((4 * k) + (j + 1)) \mod 16])$
10: $temp2[j]$ = $F_k ey(c, tc)$
11: **for** j = 0 to 3 **do**
12: $t3[j]$ = $t3[j] \bigoplus (temp1[j] \,\&\, temp2[j])$
13: $P - array$ = $t3 \bigoplus (diagonalelementfromt3)$
14: Initialize $K0K1K2K3 = t3(1)$, $K5K6K7K8 = t3(2)$, $K8K9K10K11 = t3(3)$, $K12K13K14K15 = P - array$.
15: Repeat step 1 to step 8, to get all 18 values of P-array.

The aim of the this work is to provide better security, minimizes the time taken for initialization, computational cost and use of dynamic S-boxes. The proposed algorithm includes :

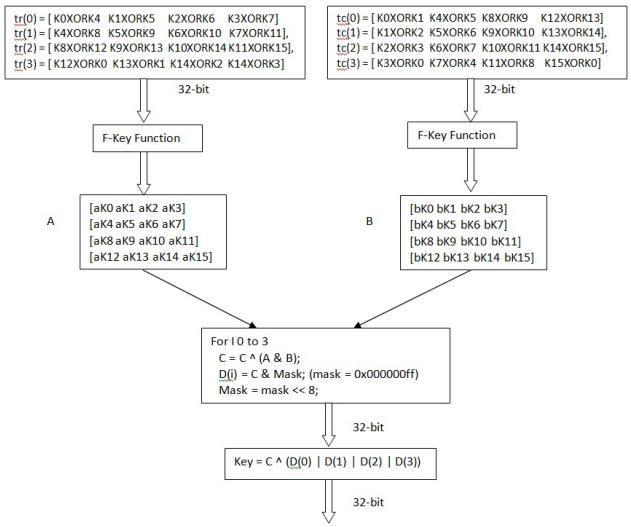
(1) No need of initialization of P-array with Π value, directly uses the Master K value for generating P-array.
(2) Reduction of computational cost, where original algorithm uses 16 round and 9 iterations in order to generate P-array value.
(3) Use of dynamic S-boxes in F-function.

5. PERFORMANCE ANALYSIS

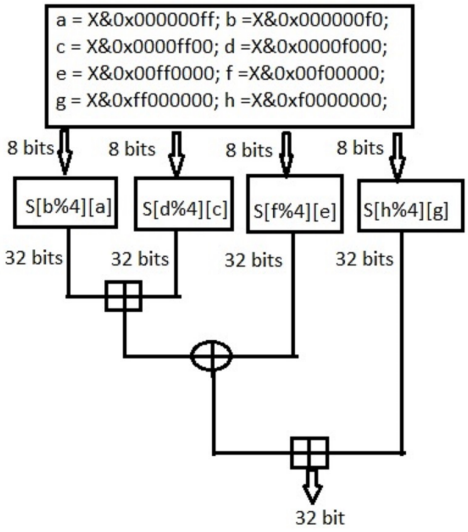
The proposed algorithm is implemented and successfully verified by considering various aspects of security like, time complexity, encryption quality, key sensitivity test, avalanche effect and statistical analysis is carried out by using a sample image by test on histogram of encrypted images and correlation coefficient of horizontally adjacent pixels in an encrypted image. The time taken by Blowfish algorithm is as shown in Fig. 4a where as the modified blowfish approach takes the time as shown in Fig 4b.
A BUTFLY.bmp image is considered to encrypt the image using both original blowfish and modified blowfish algorithm with the same key. Fig. 3a shows the original image BUTFLY.bmp and Fig. 3b and Fig. 3c shows the result of encryption and decryption images of BUTFLY.bmp image by applying original blowfish algorithm. Fig. 3d and Fig. 3e shows the result of encryption and decryption images of BUTFLY.bmp image by applying modified blowfish algorithm.

5.1 Encryption Quality Analysis

The most important factor, the visual inspection, is used to examining the effectiveness of encrypted technique used in hiding features. But this factor is not enough to evaluating the amount of information hidden, we need some evaluation metrics to estimate the degree of encryption quality. The mathematical formula for calculating encryption quality test [4], [8], [1], where it is considered the average number of changes to each grey level L, is given as:



(a)

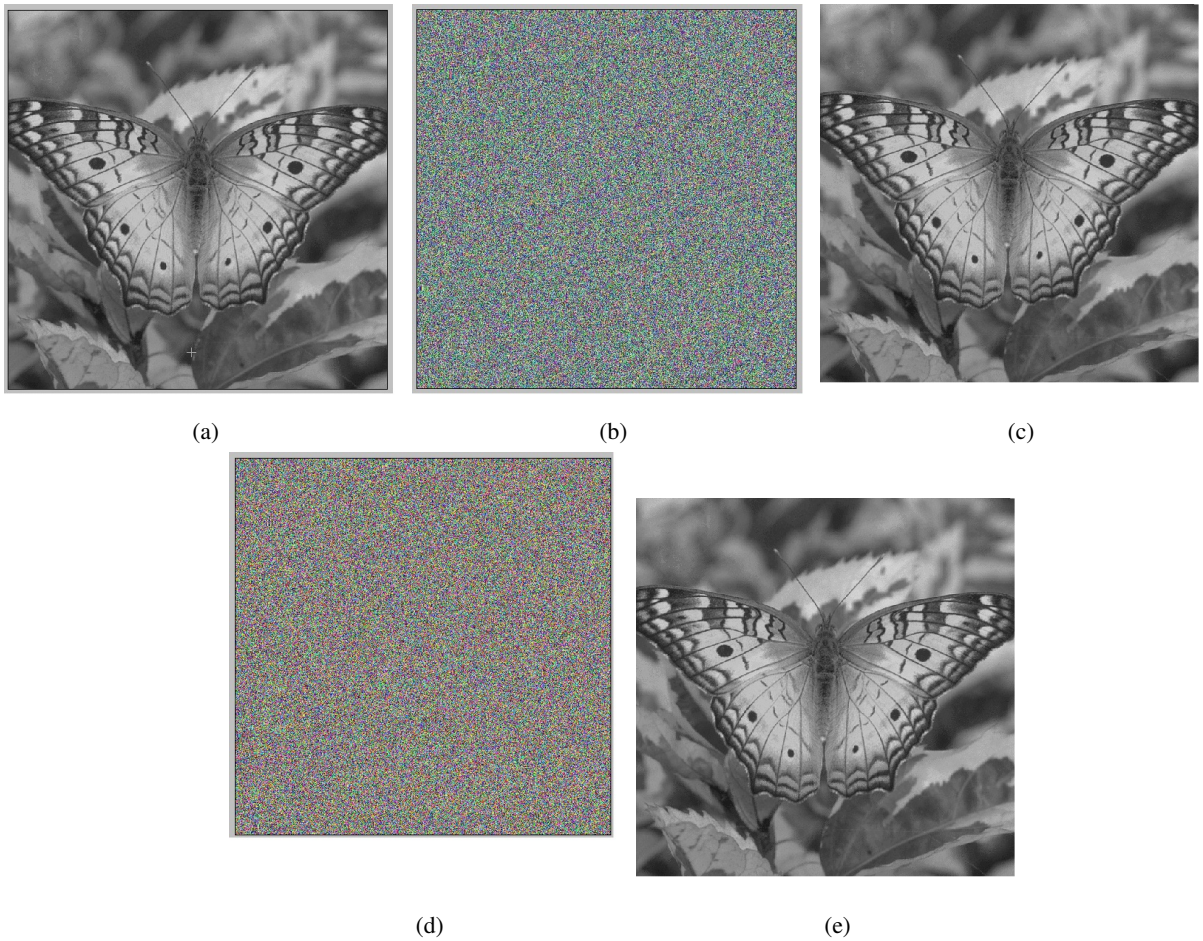


(b)

Fig. 2: (a) Flowchart of the modified Blowfish algorithm. (b) Block diagram of F-function of the modified Blowfish algorithm.

$$EQ = \frac{\sum_{L}^{255}[H_L(F^{/})] - H_L(F)}{256}$$

where, F and F' is represented as original image and encrypted image of size $M * N$ pixels with L grey levels. At position (x, y), the grey levels of the F and F',$(0 \leq x \leq M-1, 0 \leq y \leq N-1)$ is represented as $F(x, y)$ and $F'(x, y) \in L$ ranging from $0 \ldots 255$. And $HL(F)$ and $HL(F')$ is represents number of occurrences of each grey levels in original as well as encrypted image. Encryption Quality test is calculated using both original and modified algorithm for the BUTFLY image. The result obtained as given below in table 1.



(a) (b) (c)

(d) (e)

Fig. 3: (a) Original image BUTFLY.bmp, (b) and (c) Encryption and Decryption of am image BUTFLY.BMP using original Blowfish algorithm, (d) and (e) Encryption and Decryption of an image BUTFLY.BMP image using modified Blowfish algorithm.

(a) (b)

Fig. 4: (a) Time required to execute original key scheduling algorithm, (b) Time required to execute modified key scheduling algorithm.

5.2 Key Sensitivity Test

Key sensitive test [4], [8], [5] has been carried out as

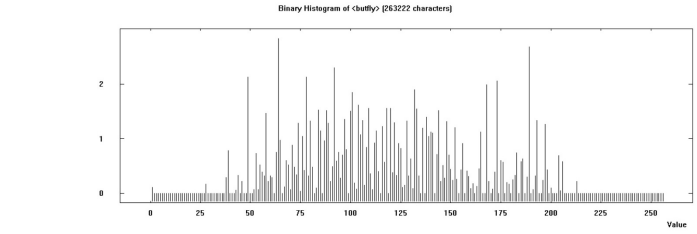
(1) Encrypt an image BUTFLY.bmp by applying the 16-character, 128-bit key, say key1, using original and modified algorithm.

(2) From key1, we need to change any randomly selected one bit. Then, from this modified key, say key2, encrypt the same image by applying the original algorithm as well as modified algorithm.
Ex : "ADF278565E262AD1F5DEC94A0BF25B27", from this key1 we have randomly selected "F" as shown in bold and changed to "B" as "ADF278565E262AD1F5DEC94A0BB25B27" which is key2.

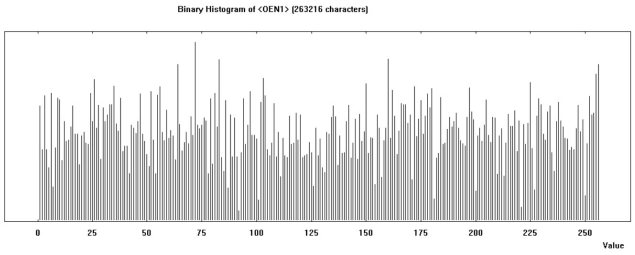
Below table 2 shows, comparison of both ciphered images which are encrypted by applying these two different keys by using original blowfish and modified blowfish algorithms using the same image.

5.3 Avalanche effect test

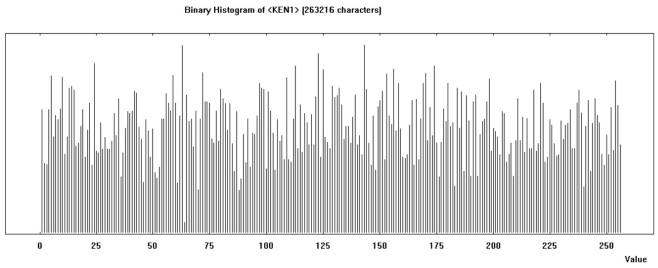
The avalanche effect [9], [10], [1], [5], means if there is a change in one bit in the plain text or key then there will be number of bits changes in the cipher text. In the case of block ciphers, such a small change in either the key or the plaintext drastic changes occur in the ciphertext. The avalanche effect is determined by counting number of times original Blowfish algorithm gives better avalanche, number of times modified blowfish algorithm gives better avalanche and the number of times both algorithm gives same avalanche for different rounds. Table 3 gives avalanche effect due to change in one bit of key.



(a)



(b)



(c)

Fig. 5: (a) Histogram of the Original Image BUTFLY.BMP, (b) Histogram of the encrypted image using original algorithm, and (c) Histogram of the encrypted image using modified algorithm.

Table 1. : Comparison Of Key Sensitivity Of Original And Modified Blowfish algorithm For Different Rounds.

Number of Rounds	Original Blowfish	Modified Blowfish
2	96.0681	99.5572
4	98.8352	99.4492
8	99.5072	99.4834
12	99.4413	99.4676
14	99.3991	99.4887
16	99.4657	99.4729

Key Sensitivity Test.

5.4 Statistical Analysis- Histogram

Statistical analysis using images by test on the histogram of encrypted images and correlation of horizontally adjacent pixels in an encrypted image is considered.
The histogram analysis is generated for original image BUTFLY.BMP and for encrypted images of BUTFLY.BMP using both original and modified algorithm. Fig.5a shows the histogram for original image and Fig. 5b and Fig. 5c shows the histogram for corresponding encrypted image obtained using original and modified blowfish algorithm respectively. From these figures we can see that the histogram of the encrypted images is fairly uniform and is significantly different from that of the original image.

5.5 Statistical Analysis- Correlation Coefficient

The correlation coefficient is determined between horizontally adjacent pixels [8], in an image. The procedure for determining the correlation of horizontal adjacent pixels [4], [8], [1], [5], in an image (BUTFLY) is as follows

(1) Select N pairs of horizontally adjacent pixels from an original image and their encrypted image.

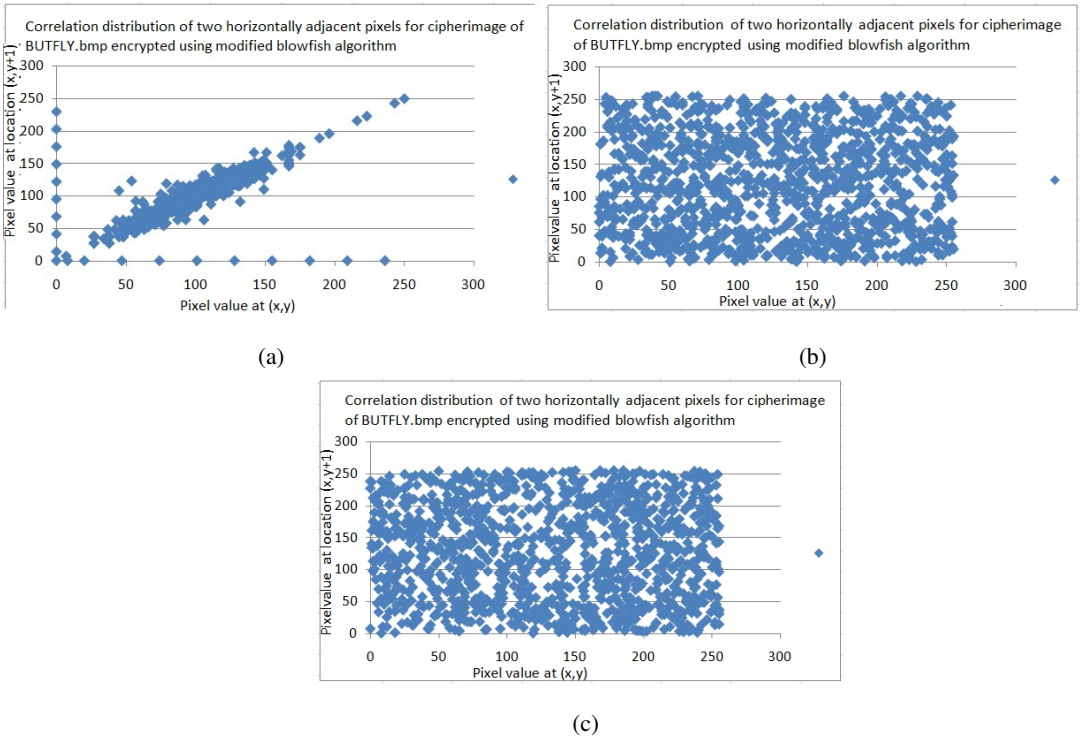


Fig. 6: (a) Correlation Distribution of two Horizontally Adjacent Pixels for Original Image BUTFLY.BMP, (b) Correlation Distribution of two Horizontally Adjacent Pixels for encrypted image using original algorithm, and (c) Correlation Distribution of two Horizontally Adjacent Pixels for encrypted image using modified algorithm.

Table 2. : Comparison Of Avalanche Effect for different rounds Of Original And Modified Blowfish algorithm For one bit change in key.

Number of Rounds	Original Blowfish	Modified Blowfish	Both algorithms
2	9064	17912	3024
4	13219	12884	3897
8	12960	13144	3896
12	13025	13037	3938
14	13058	13091	3851
16	12892	13256	3852

Avalanche Effect Test.

(2) To test the correlation between two horizontally adjacent pixels we have randomly selected 1200 pixels and pixels adjacent to them from original (Butterfly.bmp) and their encrypted images.

(3) Using r_xy formulae to find correlation coefficient, where x and y represent grey scale values of horizontally adjacent pixels in an image.

$$r_xy = \frac{cov(x,y)}{\sqrt{D(x)}\sqrt{D(y)}} \quad (1)$$

where $D(X)$ and $D(Y)$ represents the variance of x and y values and $COV(X,Y)$ is covariance of x and y and is given by

$$COV(x,y) = \frac{1}{N}\sum_{i=0}^{N}(x_i - E(x))(y_i - E(y)) \quad (2)$$

Where N represents number of randomly selected horizontal adjacent pixels, $E(X)$ and $E(Y)$ represents the mean values of x and y values. This test is carried out for about randomly selected 1200 horizontally adjacent pixels from the original image BUTFLY.BMP and encrypted images. Then using above equations correlation coefficient will be computed and is as shown in below Fig. 6a, Fig. 6b, and Fig. 6c. The correlation coefficient of original image is 0.022408 and for cipher image which is encrypted by blowfish algorithm is 0.022897 and for modified algorithm is 0.247323. In cases of both original and modified algorithm the correlation coefficients for plain image with that of cipher images are far apart.

6. CONCLUSION

A new approach has been considered and implemented to generate the P-array entries of blowfish using an alternate key scheduling algorithm for blowfish. The Design philosophy behind the proposed technique is to reduce the time spent on initialization and

Table 3. : Comparison Of Encryption Qualities Of Original And Modified Blowfish algorithm For Different Rounds.

Number of Rounds	Original Blowfish	Modified Blowfish
2	156.765	185.164
4	180.171	183.914
8	182.812	183.875
12	184.507	185.460
14	182.453	183.851
16	183.203	182.109

Encryption Quality Test.

creating P-array entries and use of dynamic S-box, which yields an algorithm that is easier to implement and achieves better security properties. The analysis has been carried out by performing security analysis through avalanche effect, key sensitivity, and statistical analysis. It is observed in all the analysis made throughout that the modified key scheduling algorithm has better performance than the original blowfish algorithm.

7. REFERENCES

[1] Jawad Ahmad and Fawad Ahmed. Efficiency analysis and security evaluation of image encryption schemes. *International Journal of Video and Image Processing and Network Security IJVIPNS-IJENS*, 12(4):8, 2007.

[2] D Whiting B. Schneier. Fast software encryption: Designing encryption algorithms for optimal software speed on the intel pentium processor. Lecter notes in Computer Science, 1997.

[3] B.Schneier. The blowfish encryption algorithm. *In Dr Dobbs Journal*, pages 38–40, 1994.

[4] Hamdy M. Kalash Hossam El din H. Ahmed and Osama S. Farag Allah. Encryption quality analysis of rc5 block cipher algorithm for digital images. *Journal of Optical Engineering*, 45, 2006.

[5] Dr. V Ramaswamy Krishnamurthy G N. Encryption quality analysis and security evaluation of blow-castfish using digital images. Communicated to International Journal of Computational Science, 2008.

[6] Yeong-Kang Lai and Yu-Chuan Shu. A novel vlsi architecture for a variable-length key, 64-bit blowfish block cipher. *IEEE workshop on signal processing systems. SiPS 99. design and implementation*, 1999. doi:10.1109/sips.

[7] K. Baskaran P. Karthigakumar. Partially pipelined vlsi implementation of blowfish encryption/decryption algorithm. *International Journal of Image and Graphics*, 10(3):327–341, 2010.

[8] K. Baskaran P. Karthigakumar. Partially pipelined vlsi implementation of blowfish encryption/decryption algorithm. *International Journal of Image and Graphics*, 10, 2010.

[9] B. Schneier. Description o f a new variable-length key,64-bit block cipher (blowfish). Fast Software Encryption,Cambridge Security Workshop proceedings (December 1993), December 4 1994. Springer-Verlag.

[10] B. Schneier. Applied cryptography: Protocols, algorithms, and source code in c. John Wiley and Sons, 1995.

[11] W. Stallings. *Cryptography and Network Security: Principles and Practices*. 2nd ed., Prentice Hall, 1999.

[12] Rahil Abbas; Vijayalakshmi S Sudarshan, TSB; Mir. Dril a flexible architecture for blowfish encryption using dynamic reconfiguration replication inner-loop pipelining loop folding techniques. *Springer*, page 625 639, 2005. Advances in Computer Systems Architecture.