A FAST ENCRYPTING ALGORITHM

Ritu Agarwal, Dhiraj Dafouti, Vishal Bhargava, Nikhil Maheshwari, Shobha Tyagi
Department of Information Technology
Delhi Technological University
Delhi, India
ritu.jeea@gmail.com, ddafouti100@gmail.com, vishalbharg@gmail.com, nikhimaheshwari123@gmail.com, tyagishobha2@gmail.com

Abstract— **The principal goal guiding the design of any encryption algorithm must be security against unauthorized attacks. However, for all practical applications, performance and speed are also important concerns. A data encryption algorithm would not be of much use if it is only secure enough, but slow in performance and speed. In this paper, the two of the popular secret key encryption algorithms, i.e., cast-128 and the Blowfish have been merged and taken the best features of both the algorithm their performance is evaluated by encrypting input files of varying contents and sizes, on different Hardware platforms. The algorithms have been implemented in a uniform language, using their standard specifications, also a fair comparison of execution speeds with the other cryptographic algorithms are presented. The performance results have been summarized and a conclusion has been presented. Based on the results, it has been concluded that this new fast encrypting algorithm is the best performing algorithm among the algorithms in case of execution time of encryption and decryption.**

Keywords- ***Encryption Algorithms , Performance Analysis , Blowfish, Cast-128, Execution time, DES, AES.***

I. INTRODUCTION

As the importance and the value of exchanged data over the internet or other media types are increasing, the search for the best solution to offer the necessary protection against the data thieves' attacks along with providing the services under timely manner is one of the most active subject in the security related communities. This paper tries to present a fast encrypting algorithm through merging the two cryptographic algorithm taking the best features of both the algorithm and shown fair comparison between the most common and used algorithms in the data encryption field. Since our main concern here is the execution speed of the algorithm under different setting, the presented comparison takes into consideration the behavior and the performance of the algorithm when different data loads are used. Section 2 will give an introduction of the algorithm blowfish and cast-128 and its advantages and disadvantages. Section 3 is the description of the fast encrypting algorithm. Section 4 is the result under different size of data and comparison .Section 5 is the limitation of this algorithm and section 6 is the conclusion.

II. INTRODUCTION OF ALGORITHMS

This fast encrypting algorithm is a simple classical Feistel network[19] with 16 rounds and operating on 64 bit blocks of plaintext to produce 64 bit blocks of ciphertext. It uses key ranging from 32 bits to 448bits. It uses good features of Blowfish[1] and CAST-128[6] algorithms.

The good features that are taken from CAST-128[6] algorithm include:

1. Round dependent function operation.
2. Use of Circular shift operation in each round.

The good features that are taken from Blowfish[1] algorithm include:

1. Varying key length of up to 448 bits.
2.Key dependent substitution box(S-box) entries.
3. Key expansion procedure.

In order to understand the design of Fast encrypting algorithm it is necessary to know about Blowfish and CAST-128 algorithms.

Blowfish[1] is a variable-length key block cipher network, iterating a simple encryption function 16 times .The block size is 64 bits, and the key can be any length up to 448 bits.

CAST-128[6] is a design procedure for symmetric encryption algorithm which has the structure of a classical Feistel network with 16 rounds and operating on 64 bit blocks of plaintext to produce 64 bit blocks of ciphertext. It uses key of length 128 bits.

A. Design Requirements

This section summarizes the different design aspects:

1. Global structure and Number of rounds

Global structure: Our approach follows Feistel network consists of dividing the input into two halves, and applying a non-linear function only to the right half. The result is added into the left half , and subsequently left and right half are swapped. Ciphers following this approach are called Feistel ciphers (Figure 1). The output of one nonlinear function is input directly to the next one, which increases the propagation of local changes.

Number of rounds: Most block ciphers obtain their strength from repeating a number of identical rounds. In the key paper of Luby and Rackoff [21] it is shown that a 3-round Feistel network can provide a provable secure construction of a pseudo-random permutation from a pseudo-random function.

2. Non-linearity

A nonlinear component is essential to every strong cryptographic primitive. The goal of the designer is to build a 'large' nonlinear primitive from smaller ones. The design approaches differ in the choice of the basic
nonlinear component. A straightforward way to implement simple nonlinear functions is to use S-boxes.

3. Diffusion

In order to restrict the complexity of the implementation, nonlinear operations can only be applied to small parts of the block. Several techniques are used to spread local changes. One way to achieve this is to add the output of several S-boxes, as is done for Blowfish and CAST.

4. Key schedule

The key schedule is an important component of a block cipher; it computes the round keys from the external key.
Other design strategies for this algorithm include

- 64 bit data input.
- An input key ranging from 32 bits to 448 bits.
- The operations used here are addition, subtraction, XOR, left circular rotation and right circular rotation. The rotation is based on the last 5 bit values of sub key. The shift used here is variable length.
- Pre computable sub keys: These pre computed keys perform faster operation.
- This algorithm consists of 16 iterations.

B. *Building Blocks*

Key and Sub keys : This algorithm makes use of a key that ranges from 32 bits to 448 bits (1 to 14 32bit words). That key is used to generate 18, 32 bit sub keys. The sub keys are stored in P –Array:

P1,P2,………P18.

S-boxes (Substitution box): There are 4 S-boxes, each with 256 entries and each entry is of 32-bit length. S-box works as a multiplexer. The input to the S-box is 8-
bit and the output is 32-bit.

S1.0,S1.1,…………..S1.255
S2.0,S2.1,…………..S2.255
S3.0,S3.1,…………….S3.255
S4.0,S4.1,…………….S4.255

Operations: It uses 4 primitive operations. Addition, Subtraction, Bitwise exclusive OR and Left circular rotation(<<<).

III. Algorithm Description

This fast encrypting algorithm is a variable-length key, 64-bit block cipher. The algorithm consists of two parts: a key expansion part and a data- encryption part. Key expansion converts a key of at most 448 bits into several sub key arrays totaling 4168 bytes. Fast encrypting algorithm is a Feistel network consisting of 16 rounds.

The input is a 64-bit plaintext and is denoted by x.
Divide x into two 32-bit halves: xL, xR

```
For i = 1 to 16:
xL = xL XOR Pi
xL = xL <<< ( Last 5bit value of Pi)
xR = F(xL) XOR xR
Swap xL and xRNext i
Swap xL and xR (Undo the last swap.)
xR = xR XOR P17
xL = xL XOR P18
Recombine xL and xR
```

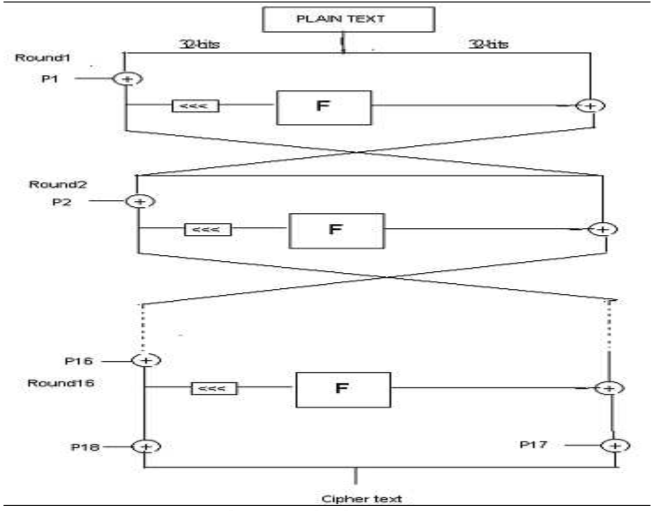

Fig 1: Fiestel network showing encryption

Divide xL into four eight-bit quarters: a, b, c, and d. The function F is defined as follows.

Function F ( ):

Rounds : 1,4,7,10,13,16
F=((S1[Ia] XOR S2[Ib]) – S3[Ic]) + S4[Id];

Rounds : 2,5,8,11,14
F=((S1[Ia] - S2[Ib]) + S3[Ic]) XOR S4[Id];

Rounds : 3,6,9,12,15
F=((S1[Ia] + S2[Ib]) XOR S3[Ic]) - S4[Id];

Decryption is exactly the same as encryption, except that P1, P2,..., P18 are used in the reverse order.

A. Subkeys Generation

Initialize first the P-array and then the four S-boxes, in order, with a fixed string. This string consists of the hexadecimal digits of pi (less the initial 3).
For example:
P1 = 0x243f6a88
P2 = 0x85a308d3
P3 = 0x13198a2e
P4 = 0x03707344

XOR P1 with the first 32 bits of the key, XOR P2 with the second 32-bits of the key, and so on for all bits of the key (possibly up to P14). Repeatedly cycle through the key bits until the entire P-array has been XOR ed with key bits. (For every short key, there is at least one equivalent longer key; for example, if A is a 64-bit key, then AA, AAA, etc., are equivalent keys.)
Encrypt 64bit block of all-zeros using the current P and S array, replace P1 and P2 with the output of the encryption. Encrypt the output of step (3) using the current P and S array and replace P3 and P4 with the resulting cipher text. Continue the process to update all elements of P and then in order all elements of S using at each, the output of the continuously changing this fast encrypting algorithm.

B. Decryption

Decryption (Figure 2) works in the reverse order of the encryption beginning with the ciphertext as input. The sub-keys are used in reverse order. So the decryption of this algorithm is as follows:

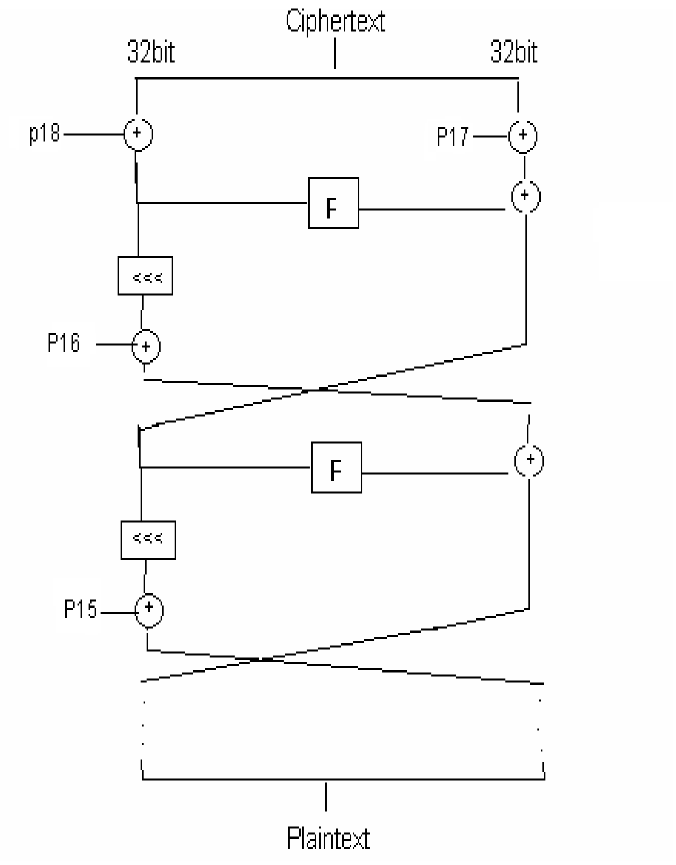

Figure 2: Fiestal network of Decryption.

C. Function F in Fast Encrypting Algorithm

The function F (Figure 3) in the is defined as follows.

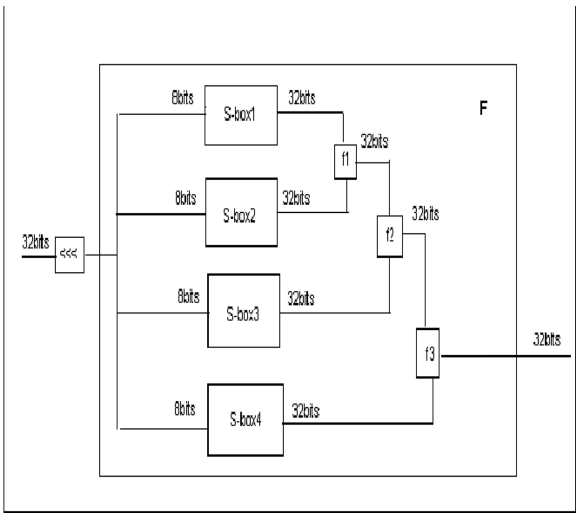

Figure 3: Function F in fast encrypting algo.

Rounds : 1,4,7,10,13,16
F=(S1[Ia] XOR S2[Ib]) + (S3[Ic]) - S4[Id]);

Rounds : 2,5,8,11,14
F=(S1[Ia] - S2[Ib]) XOR (S3[Ic]) + S4[Id]);

Rounds : 3,6,9,12,15
F=(S1[Ia] + S2[Ib]) - (S3[Ic]) XOR S4[Id]);

Where f1,f2, and f3 in the above diagram are one of the +,-, XOR operations

D. Modified F in Fast Encrypting Algorithm

The modified function F (Figure 4) in the Blow-CASTFishis defined as follows.

Rounds : 1,4,7,10,13,16
F=(S1[Ia] XOR S2[Ib]) + (S3[Ic]) - S4[Id]);

Rounds : 2,5,8,11,14
F=(S1[Ia] - S2[Ib]) XOR (S3[Ic]) + S4[Id]);

Rounds : 3,6,9,12,15
F=(S1[Ia] + S2[Ib]) - (S3[Ic]) XOR S4[Id]);
Where f1,f2, and f3 in the above diagram are one of the +,-, XOR operations
This modification shows parallel evaluation of different operations within the function in order to reduce the execution time. It reduces number of computation levels from three to two. This is repeated in all the 16 iterations.

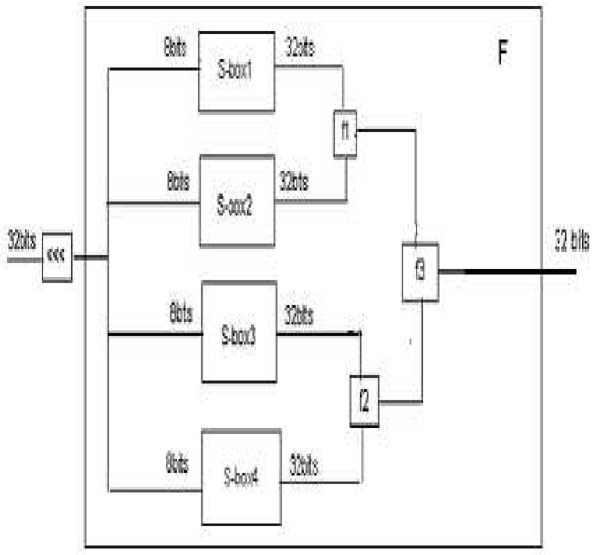

Figure 4: Modified fast encrypting algo

IV. RESULTS

To give more prospective about the performance of the proposed algorithm , this section discuss the result . The execution results of secret key algorithm in ECB (electronic code book) modes are presented in table 1. P-II 266 MHz machine and P-4 2.4 GHZ machine respectively are used for the execution of the code . The algorithms are implemented in a uniform language (C), using their standard specification.
An obvious way to compare the performance of these algorithms is to take average of execution times for each algorithm, and rank them accordingly.

Input Size (bytes)	**DES**	**3DES**	**BF**	**FAST ENCRYPTIN ALGORITHM**
20,527	**24**	**42**	**29**	**19**
36,002	**68**	**223**	**54**	**35**
45,911	**77**	**158**	**74**	**46**
59,852	**124**	**202**	**65**	**58**
69,545	**83**	**243**	**143**	**67**
137,325	**187**	**461**	**123**	**136**
158,959	**190**	**543**	**224**	**158**
166,364	**198**	**569**	**155**	**162**
191,383	**227**	**655**	**178**	**176**
232,398	**276**	**799**	**260**	**219**
Average Time	**134**	**383**	**228**	**108**
Bytes/sec	**835**	**292**	**1010**	**1,036**

Table 1 Comparative execution times (in seconds) of encryption algorithms.
It is apparent from Table 1 that four symmetric key algorithm are in the following order, as regards their performance

1-FAST ENCRYPTING ALGO.(FASTEST)
2-BLOWFISH
3-DES
4-TRIPLE DES(SLOWEST).

V. Limitations

Every project will have one or the other limitations. The limitation of this algorithm include

- It is not suitable for applications where the secret key changes frequently.
- It is not appropriate for applications with limited memory such as smart cards. The data to be encrypted should be plain text only**.**

VI. Areas of Applications

A standard encryption algorithm must be suitable for many different applications :

Bulk encryption: The algorithm should be efficient in encrypting data files or a continuous data stream.

Random bit generation: The algorithm should be efficient in producing single random bits. Internet based applications (network security).

File Encryption Utility: Blowfish suits file encryption utility because the key does not change. The entire file is encrypted with the same key.

Packet encryption: The algorithm should be efficient in encrypting packet-sized data. (An ATM packet has a 48-byte data field.) It should be implementable in an application where successive packets may be encrypted or decrypted with different keys for different packets.

Hashing: The algorithm should be efficient in being converted to a one-way hash function.

VII. Conclusion

It is a simple, fast, variably secure algorithm as it uses variable key length of 32 to 448 bits and hence meets basic design criteria of any block cipher, i.e., simple, fast and secure. The execution time of this algorithm lies in-between that of Blowfish and CAST- 128, but offers higher security compared to both. After modification to its function, the time of execution reduces by more than 16%, which is almost equal to execution time of Blowfish, which is lesser than execution time of CAST-128. This modification does not make this algorithm vulnerable as security depends to a greater extent upon S-boxes and its values, key scheduling, and round dependent functions. This modification is done so as to perform parallel evaluation of some operations of function, without changing the basic operations. The only drawback of the algorithm is that, it does not suit for applications where key is changed very often.

References

[1] B. Schneier, “Description o f a New Variable-Length Key, 64-Bit Block Cipher (Blowfish)”, *Fast Software Encryption*, Cambridge Security Workshop proceedings (December 1993), Springer-Verlag, 1994, pp. 191-204.

[2] B. Schneier, Applied Cryptography: Protocols, Algorithms, and Source Code in C, 2nd ed., John Wiley & Sons, 1995.

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

[4] B. Schneier, J. Kelsey, “Unbalanced Feistel networks and block cipher design,” *Fast Software Encryption (FSE’96), LNCS 1039,* D. Gollmann, Ed., Springer-Verlag, 1996, pp. 121–144.

[5] S. Vaudenay, “On the weak keys of Blowfish,” *Fast Software Encryption (FSE’96), LNCS 1039,* D. Gollmann,

[6] Adams, C. The CAST-128 Encryption Algorithm. RFC 2144, May 1997.

[7] Anne-Canteaut(Editor)“Ongoing Research Areas in Symmetric Cryptography” ECRYPT, 2006.

[8] S. Vaudenay, “On the weak keys of Blowfish,” *FastSoftware Encryption (FSE’96), LNCS 1039,* D. Gollmann,

[9] Lausanne, Statistical Cryptanalysis of Block Ciphers,Doctoral Thesis, EPFL, 2005. IJCSNS International Journal of Computer Science and Network Security, VOL.8 No.4, April 2008290

[10] Orr Dunkelman, Techniques for Cryptanalysis of Block Ciphers, Doctoral Thesis, Haifa, 2006

[11] L. Knudsen, "Block Ciphers: A Survey", State of the Artin Applied Cryptography: Course on Computer Securityand Industrial Cryptography (Lecture Notes in Computer Science no. 1528), Springer-Verlag, pp. 18-48, 1998.

[12] C.M. Adams, “Simple and effective key scheduling for symmetric ciphers,” *Proceedings of SAC’94, workshop onSelected Areas in Cryptography,* pp. 129–133.

[13] C.M. Adams, “Constructing symmetric ciphers using theCAST design procedure,” *Designs, Codes, and Cryptography*, Vol. 12, No. 3, November 1997, pp. 71–104.

[14] C.M. Adams, S.E. Tavares, “The structured design ofcryptographically good Sboxes,” *Journal of Cryptology,* Vol. 3, No. 1, 1990, pp. 27–42.

[15] C.M. Adams, S.E. Tavares, “Designing S-boxes for ciphers resistant to differential cryptanalysis,” *Proceedings of the 3rd Symposium on State and Progress of Research in Cryptography,* W. Wolfowicz, Ed., Fondazione Ugo Bordoni, 1993, pp. 181–190.

[16] E. Biham, A. Shamir, Differential Cryptanalysis of the Data Encryption Standard, Springer-Verlag, 1993.

[17] D. Chaum, J.-H. Evertse, “Cryptanalysis of DES with a reduced number of rounds sequences of linear factors in block ciphers,” *Advances in Cryptology, Proceedings Crypto’85, LNCS 218,* H.C. Williams, Ed., Springer-Verlag, 1985, pp. 192–211.

[18] R.L. Rivest, “The RC5 encryption algorithm,” *Fast Software Encryption (FSE’94), LNCS 1008,* B. Preneel, Ed., Springer-Verlag, 1995, pp. 86–96.

[19] B. Schneier, J. Kelsey, “Unbalanced Feistel networks and block cipher design,” *Fast Software Encryption (FSE’96), LNCS 1039,* D. Gollmann, Ed., Springer-Verlag, 1996, pp. 121–144.

[20] S. Vaudenay, “On the weak keys of Blowfish,” *FastSoftware Encryption (FSE’96), LNCS 1039,* D. Gollmann, Ed., Springer-Verlag, 1996, pp. 27–32.

[21] M. Luby, C. Rackoff, “How to construct pseudorandom permutations from pseudorandom functions,” *SIAM Journal on Computing,* Vol 17, No. 2, April 1988, pp. 373–386. IE, India.