Blow-CAST-Fish: A New 64-bit Block Cipher

Krishnamurthy G.N[†], Dr. V. Ramaswamy[†], Leela G.H[†] and Ashalatha M.E[†]

[†]Bapuji Institute of Engineering and Technology, Davangere-577004, Karnataka, India

Summary: This paper attempts to develop a simple, stronger and safer cryptographic algorithm which would not only be a secure one, but also reduces total time taken for encryption and decryption. The result of such an attempt is "Blow-CAST-Fish", a new secret-key block cipher that uses good features of CAST-128 and Blowfish algorithms. An effort is made to enhance performance of the resulting algorithm by parallel evaluation of some operations within the round function. In order to show the reduction in execution time, VHDL implementation is used and tested to show percentage improvement in the performance of the modified Blow-CAST-Fish.

1. Introduction

Blow-CAST-Fish 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 448 bits. 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 Blow-CAST-Fish 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.

2. 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 Blow-CAST-Fish 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 subkey. The shift used here is variable length.
- Precomputable subkeys: These precomputed keys perform faster operation.
- This algorithm consists of 16 iterations.

3. Building Blocks

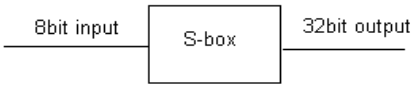
Key and Subkeys : Blow-CAST-Fish makes use of a key that ranges from 32 bits to 448 bits (1 to 14 32bit words). That key is used to generate 18, 32 bit subkeys.

The subkeys 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: Blow-CAST-Fish uses 4 primitive operations. Addition, Subtraction, Bitwise exclusive OR and Left circular rotation(<<<).

4. Algorithm Description

Blow-CAST-Fish 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 subkey arrays totaling 4168 bytes.

Blow-CAST-Fish 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 xR
Next i
Swap xL and xR (Undo the last swap.)
xR = xR XOR P17
xL = xL XOR P18
Recombine xL and xR

4.1 Encryption

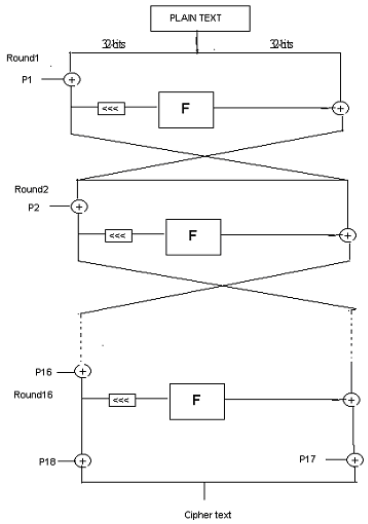


Fig 1: Fiestel network showing encryption

Function F ( ):
Divide xL into four eight-bit quarters: a, b, c, and d
The funcion F is 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]) + 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.

4.2 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 XORed 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 ciphrtext.

Continue the process to update all elements of P and then in order all elements of S using at each, the output of the continuesly changing Blow-CAST-Fish algorithm.

4.3 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 Blow-CAST-Fish algorithm is as follows:

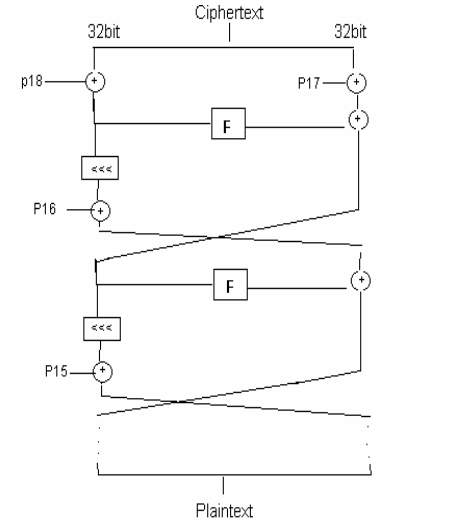


Figure 2: Blow-CAST-Fish Decryption

4.4 Function F in Blowfish

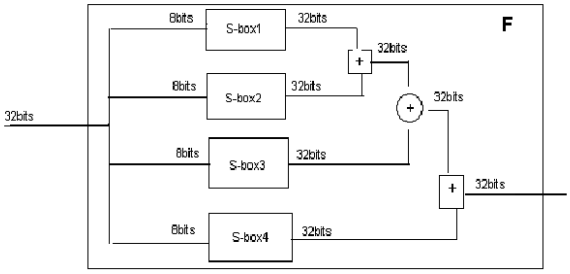


Figure 3: Function F in the existing Blowfish.

Function F (Figure 3) [1] of Blowfish is implemented as follows:-

Divide xL into four eight-bit quarters: a, b, c, and d

$F(xL) = ((S1,a + S2,b \bmod 2^{32}) \text{ XOR } S3,c) + S4,d \bmod 2^{32}$

4.5 Function F in CAST-128

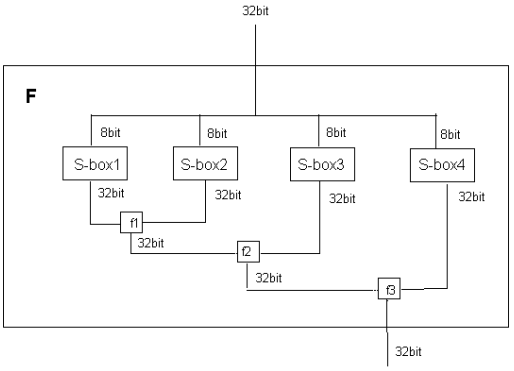


Figure 4: Function F in the existing CAST-128

The funcion F (figure 4) [6] in the CAST-128 is 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]) + S3[Ic]) XOR S4[Id];

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

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

4.6 Function F in Blow-CAST-Fish

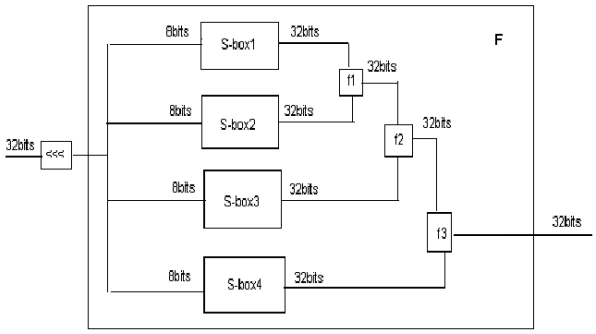


Figure 5: Function F in Blow-CAST-Fish

The funcion F (Figure 5) in the Blow-CAST-Fish is 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]) + S3[Ic]) XOR S4[Id]

Rounds : 3,6,9,12,15

F=((S1[Ia] + S2[Ib]) XOR S3[Ic]) - S4[Id]

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

4.7 Modification to the function F in Blow-CAST-Fish to reduce execution time

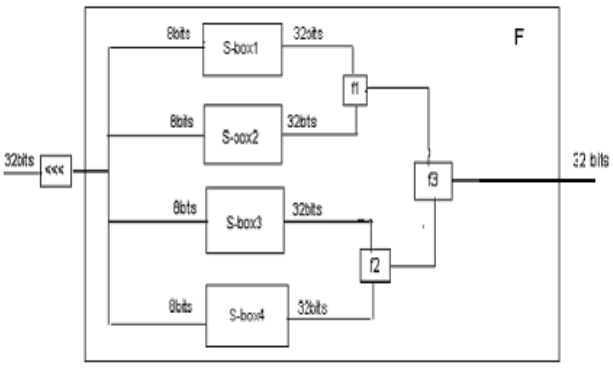


Figure 6: Modified Function F in Blow-CAST-Fish

The modified funcion F (Figure 6) in the Blow-CAST-Fish is 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]);

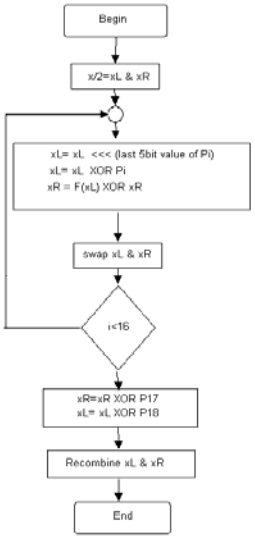
Rounds : 3,6,9,12,15
F=(S1[Ia] + S2[Ib]) - (S3[Ic]) XOR S4[Id]);

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 during the function evaluation.

4.8 Flowchart for the Algorithm

Following diagram (Figure 7) shows Flowchart for encryption operation of Blow-CAST-Fish. The 64-bit input is denoted with x, P-array is denoted with a Pi (where i is the iteration), left 32bit data with xL and right 32bit data with xR.



5. Hardware Implementation using VHDL

The improvements suggested to the algorithm stated above cannot be seen using software implementation. So, we have demonstrated this improvement using VHDL

implementation to show enhancement of performance of the modified algorithms in terms of total delays for encryption and decryption.

5.1 About VHDL

The VHSIC Hardware Description Language is an industry standard language used to describe hardware.

The following are some of the building blocks of VHDL.

1. Entity: All designs are expressed in terms of entities. An entity is the most basic building block in a design. It specifies the name of the entity, ports of the entity, and entity-related information. All designs are created using one or more entities.
2. Architecture: All entities that can be simulated have an architecture description. The architecture describes the behavior of the entity.
3. Configuration: A configuration statement is used to bind a component instance to an entity-architecture pair.
4. Package: A package is a collection of commonly used data types and subprograms used like a parts list for a design.

Using these building blocks we have implemented the original and modified algorithms to demonstrate the improvement in time, as shown in the waveforms at the end.

6. Implementation steps and Testing

The Blow-CAST-Fish algorithm is block cipher consisting of a standard Fiestel network with 16 rounds, plus some initial and final encryption functions. The main feature that sets it off from other similar algorithms is that the non-linear substitution boxes (S-boxes) are key-dependent. Because of this, there is an expensive initialization step required on every key change, requiring the equivalent of 520 encryptions to initialize the S-boxes.

In any implementation, there are several separate circuit pieces that can be identified. First is the encryption core that implements the actual Fiestel network. Second is the function F(xL) that cipher relies on for each round of the Fiestel network. Third is a generated array of sub-keys, called the p-array, which is also used by cipher in each round. Fourth are the four key-dependent s-boxes that are read by the F(xL) function also in each round. Fifth would be any control logic necessary to initialize the p-array and s-boxes.

6.1 Inputs and Outputs

Below is a description of the inputs and outputs and how they are used.

6.1.1 Inputs

clk : 1 - The input clock signal.

key : The key length is 448bits. If less than a 448 bits key is desired, this signal must be padded up to 448 bits.

Input data : The input data is 64bit. In encryption mode, this is the plaintext. In decryption mode, this is the ciphertext. This is only read when the **ready** signal is asserted.

new_data : 1 - This signal should be asserted when new data is presented on the **data_in** port. This signal is ignored when the circuit is not asserting **ready**. This should not be asserted at the same time as **new_key**.

new_key : 1 - This signal should be asserted when a new **key** or **key_size** is presented. This signal is ignored when the circuit is not asserting **ready**. This should not be asserted at the same time as **new_data**.

encrypt : 1bit - This signal toggles between encryption and decryption operation. **1** means encrypt, **0** means decrypt. This signal is only read the same cycle that **new_data** is asserted and read.

6.1.2 Outputs

data_out : The output data is 64bit. In encryption mode, this is the ciphertext. In decryption mode, this is the plaintext.

6.2 Initialization

To properly initialize the circuit, **new_key** and **new_data** must be low before the first rising edge of the clock. After the first clock cycle, the circuit will assert **ready** and it can then be used. If this constraint is not met, the circuit will still initialize, but can take up to 21106 clock cycles before **ready** is asserted. However, any time that **ready** is asserted after the first clock cycle, the circuit is usable.

6.3 Timing

The following timing is given as a reference. Generally, in interfacing with this implementation, it is not necessary to keep track of any timing or have any external counters--it is sufficient to assert the proper signals, then wait for the ready signal to come back on before performing the next operation.

Time from when **new_key** is asserted and read to when **ready** is asserted again (i.e. time to initialize the algorithm with a new key): **21106** clock cycles.

7. Limitations

Every project will have one or the other limitations. The limitation of our Blow-CAST-Fish include

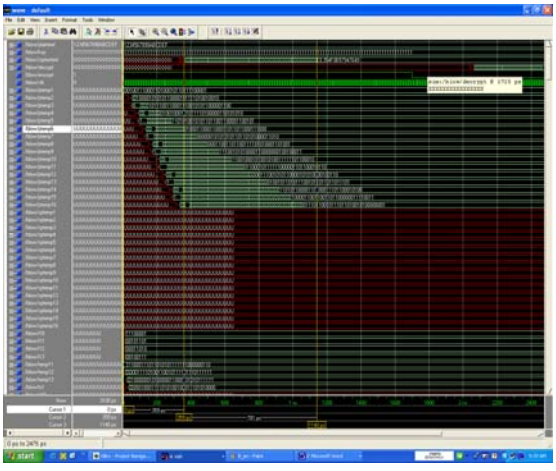
- Blow-CAST-Fish is not suitable for applications where the secret key changes frequently.
- Blow-CAST-Fish is not appropriate for applications with limited memory such as smart cards.

The data to be encrypted should be plain text only**.**

8. Result Analysis

The results shown below in the form of waveforms, give a clear picture of time taken by Blowfish, CAST-128, and Blow-CAST-Fish algorithms.

8.1 Blowfish Encryption

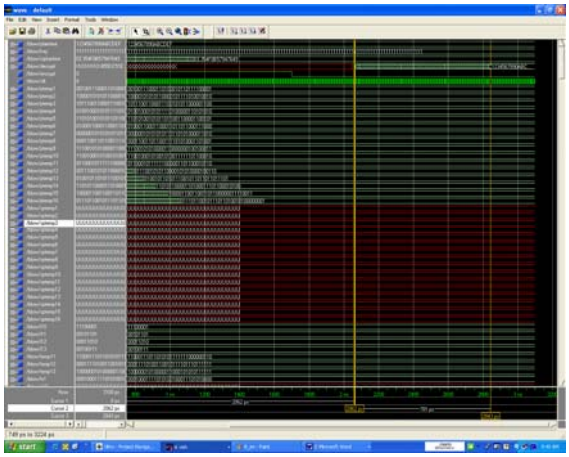


The figure above shows Encryption operation using Blowfish algorithm. The starting and end points of the operation are shown by two yellow lines. From the diagram we can observe the following.

Time at which Encryption operation initiated = 359 ps.
Time at which Encryption operation completed = 1140 ps.

So, time taken for Encryption = 1140 – 359 = 781 ps.

8.2 Blowfish Decryption

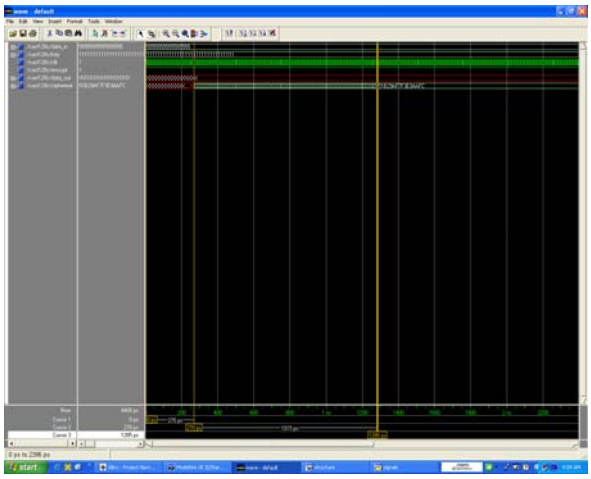


The figure above shows Decryption operation using Blowfish algorithm. The starting and end points of the operation are shown by two yellow lines. From the diagram we can observe the following.

Time at which Decryption operation initiated = 2062 ps.
Time at which Decryption operation completed = 2843 ps.

So, time taken for Encryption = 2843 – 2062 = 781 ps.

8.3 CAST-128 Encryption

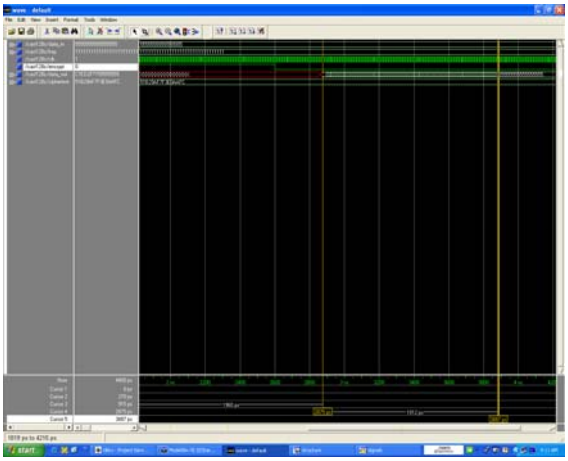


The figure above shows Encryption operation using CAST-128 algorithm. The starting and end points of the operation are shown by two yellow lines. From the diagram we can observe the following.

Time at which Encryption operation initiated = 270 ps.
Time at which Encryption operation completed = 1285 ps.

So, time taken for Encryption = 1285 – 270 = 1015 ps.

8.4 CAST-128 Decryption

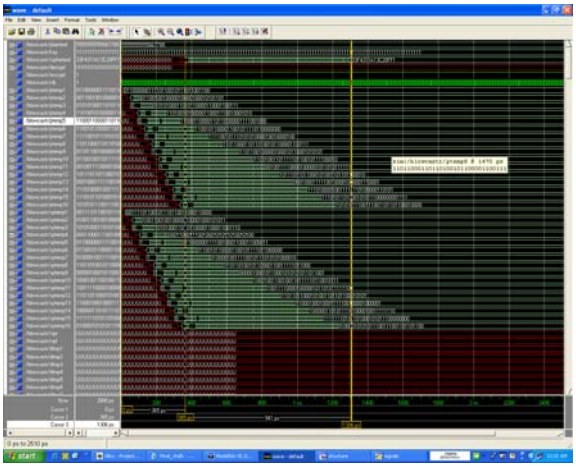


The figure above shows Decryption operation using Blowfish algorithm. The starting and end points of the operation are shown by two yellow lines. From the diagram we can observe the following.

Time at which Decryption operation initiated = 2072 ps.
Time at which Decryption operation completed = 3087 ps.

So, time taken for Encryption = 3087 – 2072 = 1015 ps.

8.5 Original Blow-CAST-Fish Encryption

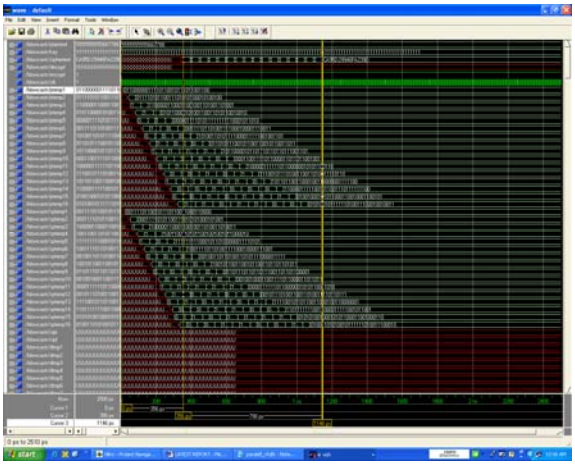


The figure above shows Encryption operation using Blow-CAST-Fish algorithm. The starting and end points of the operation are shown by two yellow lines. From the diagram we can observe the following.

Time at which Encryption operation initiated = 1306 ps.
Time at which Encryption operation completed = 365 ps.

So, time taken for Encryption = 1306 – 365 = 941 ps.

8.6 Modified Blow-CAST-Fish Encryption



The figure above shows Encryption operation using Blow-CAST-Fish algorithm. The starting and end points of the operation are shown by two yellow lines. From the diagram we can observe the following.

Time at which Encryption operation initiated = 1146 ps.
Time at which Encryption operation completed = 356 ps.

So, time taken for Encryption = 1146 – 356 = 790 ps.

9. Time Considerations

9.1 Original Blow-CAST-Fish

Original Data	Encrypted Data	Delay (in ps)	Decrypted Data	Delay (in ps)
55555555667788	33F4311A13C20FF1	941	55555555667788	941

Total Time Taken = 941+ 941 = 1882 ps

9.2 Modified Blow-CAST-Fish

Original Data	Encrypted Data	Delay (in ps)	Decrypted Data	Delay (in ps)
55555555667788	CA95D29946FA239D	790	55555555667788	790

Total Time Taken = 790 + 790 = 1580 ps

Difference = Existing – Modified = 1882 – 1580 = 302 ps

Percentage difference = (302/1882)*100 = 16%

10. 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.

11. Linear and Differential cryptanalysis

Till now, there is no attack on full 16 round Blowfish and CAST-128 algorithms which is practically feasible. There is a differential attack[9,10] against five round Blowfish in V. Rijmen's Ph.D. thesis. Vaudenay[20] in his paper "On the weak keys of Blowfish", has shown it is possible to blowfish key when key used is weak. But only a minute fraction of the keys will be weak. When the good features of Blowfish and CAST-128 are combined, the result is more secure Blow-CAST-Fish, and is open for cryptanalysis. Blow-CAST-Fish uses key dependent S-boxes, round dependent function and circular shift based on subkey value. This gives a strong shield against linear and differential cryptanalysis. Since S-boxes are unknown and are key dependent the situation becomes still worst for the cryptanalyst.

12. Conclusion

Blow-CAST-Fish 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 Blow-CAST-Fish 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 Blow-CAST-Fish 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] Dr.V.Ramaswamy, Kishnamurthy.G.N, Leela.G.H, Ashalatha M.E, "Performance enhancement of Blowfish and CAST-128 algorithms and Security analysis of improved Blowfish algorithm using Avalanche effect", International journal of Computer Science and Network Security, Vol. 8, No. 3, pp. 244-250.

[5] Kishnamurthy G.N, Dr.V.Ramaswamy and Leela.G.H, "Performance Enhancement of Blowfish algorithm by modifying its function" Proceedings of International Conference on Computers, Information, System Sciences and Engineering 2006, University of Bridgeport, Bridgeport, CT, USA., Springer, pp. 240-244

[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] Dr.V.Ramaswamy, Kishnamurthy.G.N, Leela.G.H, Ashalatha M.E, "Performance enhancement of CAST –128 Algorithm by modifying its function" Proceedings of International Conference on Computers, Information, System Sciences and Engineering 2007, University of Bridgeport, Bridgeport, CT, USA, Springer, to be published.

[9] Lausanne, Statistical Cryptanalysis of Block Ciphers, Doctoral Thesis, EPFL, 2005.

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

[11] L. Knudsen, "Block Ciphers: A Survey", State of the Art in Applied Cryptography: Course on Computer Security and 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 on Selected Areas in Cryptography*, pp. 129–133.

[13] C.M. Adams, "Constructing symmetric ciphers using the CAST 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 of cryptographically 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," *Fast Software 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.

Krishnamurthy G N obtained his B.E. degree in Electronics & Communication Engineering from Kuvempu University in 1996 and M.Tech. degree in Computer Science & Engineering from Visveswaraya technological University, India in 2000. He is presently pursuing his Ph.D. from Visveswaraya Technological University, India under the guidance of Dr. V ramaswamy. He has published papers in national and international conferences, journals in the area of Cryptography. After working as a lecturer (from 1997) he has been promoted to Assistant Professor (from 2005), in the Department of Information Science & Engineering, Bapuji Institute of Engineering & Technology, Davangere, affiliated to Visveswaraya Technological University, Belgaum, India. His area of interest includes Design and analysis of Block ciphers, He is a life member of ISTE, India.

Dr. V Ramaswamy obtained his Ph.D. degree from Madras University, in 1982, He is working as Professor and Head in the Department of Information Science and Engineering. He has more the 25 years of teaching experience including his four years of service in Malaysia. He is guiding many research scholars and has published many papers in national and international conference and in many international journals. He has visited many universities in USA and Malaysia.

Leela G.H. received the B.E. degree in Electronics & Communication Engineering from Kuvempu University in 1994 and M.E. degree in Digital Electronics from Karnataka University in 1998. After working as a lecturer (from 1994) she has been a Assistant Professor (from 2007), in the Department of Electronics & Communication Engineering, Bapuji Institute of Engineering & Technology, Davangere, affiliated to Visveswaraya Technological University, Belgaum. Her area of interest includes VLSI Design, FPGA Design and Cryptography. She is a member of ISTE, India.

M.E.Ashalatha received the B.E. degree in Electronics & Communication Engineering from Mysore University in 1987 and M.Tech degree in Industrial Electronics from Mysore University in 1990. After working as a lecturer (from 1990) and as Assistant Professor (from 1995), she has been a Professor (from 2007) in the Department of Electronics & Communication Engineering, Bapuji Institute of Engineering & Technology, Davangere, affiliated to Visveswaraya Technological University, Belgaum. Her area of interest includes VLSI Design, Cryptography and Embedded System Design. She is a member of ISTE and IE, India.