ISSN 2349-6193
Impact Factor (PIF): 2.243

IJESMR
International Journal OF Engineering Sciences & Management Research

BLOWFISH ALGORITHM

Ms Neha Khatri – Valmik
Dept. of Comp. Science & Engg., People's Education Societys college of Engg, Aurangabad, India

ABSTRACT

Modern day systems require data security more than it was required in the later. Blowfish algorithm is a derived Feistel net-work block cipher that has a 64 bit block size and it also contains a variable key size that can get up to 448 bits long. Needless to say, the Blowfish algorithm is not a patented algorithm and it can be used freely by any person under any situation. Blowfish algorithm is a symmetric block cipher that can be used as a drop-in replacement for DES (Data Encryption Standard) or IDEA (International Data Encryption Al-gorithm). It takes a variable-length key, makes it ideal for both domestic and export-able use. Blowfish was designed in 1993 by Bruce Schneier as a free & fast alternative to existing encryption algorithms. Blowfish is a Feistel network block cipher with a 64 bit block size and a variable key size up to 448 bits long. The Blowfish algorithm is unencumbered by patents and is free to use for any one is any situation.

Keywords: algo-rithm, blowfish, cryptography encryption, security

INTRODUCTION

Decryption algorithms come in two flavors, sym-metric and public key. A symmetric algorithm, such as Blowfish, uses the same key for encryption and decryption. Like password, you have to keep the key secret from everyone except the sender and receiver of the message. A public key encryption algorithm makes use of two keys, one for encryption and another for decryption. The key that is used for encryption, i.e. the "public key" need not be kept secret. The sender of the message uses public key to encrypt their message and the recipient uses their secret decryption key, or "private key", to read it. In a sense, the public key "locks" the message, and the private key "unlocks" it. Once encrypted with the public key, except the holder of the private key nobody can decrypt the message. RSA is one of the popular public key encryption algorithms.

The Blowfish algorithm Blowfish is symmetric encryption algorithms that it uses the same secret key to both encrypt and decrypt messages. Blowfish is also a block cipher; it divides a message up into fixed length blocks during en-crypt ion and decryption. The block length for Blow-fish is 64 bits; messages that aren't a multiple of eight bytes in size must be padded.

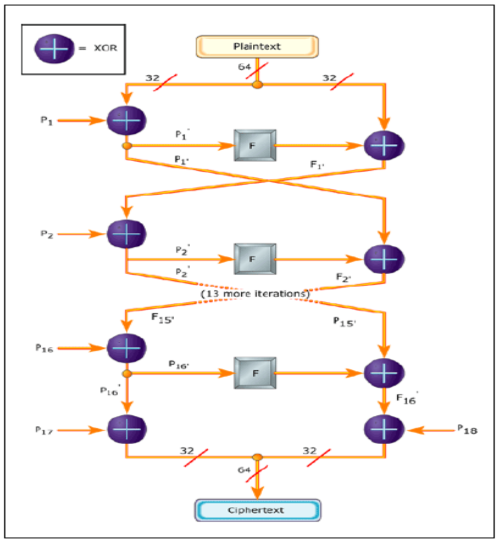


Figure 1: ***Blowfish algorithm***

The Blowfish algorithm introductorily includes addition, table lookup and XOR. The table includes four S-boxes and a P-array. Blowfish is a cipher based on Feistel, and the design of the F-function used amounts to a simplification of the principles used in DES to provide the same security with greater speed and efficiency in software. The algorithm is compact and can run in less than 5K of memory.

Blowfish consists of two parts: key-expansion and data encryption. During the key expansion stage, the inputted key is converted into several sub key arrays total 4168 bytes. There is the P array, which is eighteen 32-bit boxes, and the S-boxes, which are four 32-bit arrays with 256 entries each.

After the string initialization, the first 32 bits of the key are XORed with P1 (the first 32-bit box in the P-array). The second 32 bits of the key are XORed with P2, and so on, until all 448, or fewer, key bits have been XORed Cycle through the key bits by returning to the beginning of the key, until the entire P-array has been XORed with the key. Encrypt the all zero string using the Blowfish algorithm, using the modified P-array above, to get a 64 bit block. Replace P1 with the first 32 bits of output, and P2 with the second 32 bits of output (from the 64 bit block). Use the 64 bit output as input back into the Blowfish cipher, to get a new 64 bit block. Replace the next values in the P-array with the block. Repeat for all the values in the P-array and all the S boxes in order.

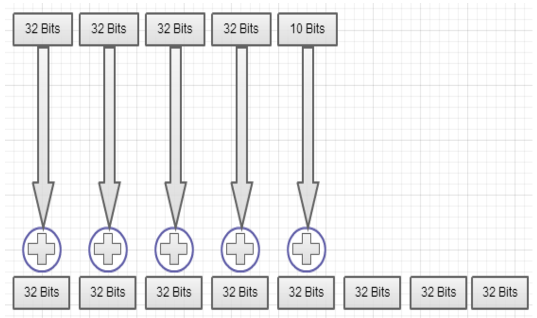


Figure 2. ***XORing bits once the key has been traversed through once***

II. LITERATURE SURVEY

Block Cipher - An encryption scheme that "the clear text is broken up into blocks of fixed length, and encrypted one block at a time".

Usually, a block cipher encrypts a block of clear text into a block of cipher text of the same length. In this case, a block cipher can be viewed as a simple substitute cipher with character size equal to the block size.

ECB Operation Mode - Blocks of clear text are encrypted independently. ECB stands for Electronic Code Book. Main properties of this mode:

- Identical clear text blocks are encrypted to identical cipher text blocks.
- Re-ordering clear text blocks results in re-ordering cipher text blocks.
- An encryption error affects only the block where it occurs.

CBC Operation Mode - The previous cipher text block is XORed with the clear text block before applying the encryption mapping. Main properties of this mode:

- An encryption error affects only the block where is occurs and one next block.

Product Cipher - An encryption scheme that "uses multiple ciphers in which the cipher text of one cipher is used as the clear text of the next cipher". Usually, substitution ciphers and transposition ciphers are used alternatively to construct a product cipher.

Iterated Block Cipher - A block cipher that "iterates a fixed number of times of another block cipher, called round function, with a different key, called round key, for each iteration".

Cryptography is a well known and widely used technique which deals with protecting the information by encoding or transformation of data into an unreadable format . The original text is converted into a scramble equivalent text called cipher text and this process is called as “Encryption” and the reverse is called “Decryption”.

There are two types of cryptographic schemes available on the basis of key.

• Symmetric key Cryptography:

The cryptographic scheme which uses a single common key for enciphering and deciphering the message.

• Asymmetric or Public Key Cryptography:

This type of cryptographic scheme uses two keys for encryption and decryption called Public key and Private Keys.

• Stream Ciphers: It is a symmetric key cipher where stream of plaintext are mixed with a random cipher bit stream (key stream), typically by any logical operation. In this case of stream cipher one byte is usually encrypted at a particular time.

• Block Ciphers: It is also known as symmetric key cipher which operates over a fixed-length group of bits. It usually takes particular bit block of plaintext as input, and produces a corresponding n-bit output block of cipher text.

For our research work, we adopted modified - blowfish algorithm one of our research works which falls under symmetric block cipher category [2].

PROPOSED SYSTEM:

This Software tool involves Cryptographic enciphering and deciphering along with File Splitting and Merging mechanisms. In this approach a file which has secret data is sliced into desired number of pieces upon user's specification and then the cryptographic encryption phase is carried out. In order to achieve more security we can adopt more than one cryptographic scheme which definitely ensures nil suspicion and more security. In this paper, we differentiate the cryptographic scheme by providing different key for each encryption of sliced files; provided the key should be given correctly at the time of decryption to avoid erroneous results. We are using modified Blowfish algorithm for Encryption and Decryption of data which serves as a better solution both in terms of performance and as well as security. This enhancement in security and performance is sustainably justified in our previous work [2].

In the file joining phase, En-Ciphered files thus obtained from the different En- Ciphering techniques are merged and hence transmitted to reception side as a single file which makes the file infeasible to breach and suspicion less to get to know that varying crypto schemes are adopted. And hence the data security is maximized. At the receiver's end, We once again splits the files and decrypts it using the same algorithm and then joins all split files together to retrieve the original message. When coming to cryptographic perspective, in order to enhance the performance of the Blowfish Algorithm it is proposed to modify the F-Function by adopting the concept of multithreading.

3.1 Modified F-Function:

Function F plays an important role in the algorithm, and we decided to modify function F. Original function F is defined as follows. [3]

F(X) = ((S1 + S2 mod 232) XOR S3) + S4 mod 232

Instead, we modified the F-Function by replacing 2 addition operations as XOR Operations . Thus the modified F-

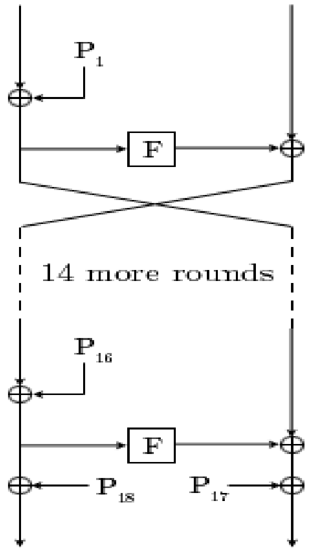
Function is written as,

F(X) = ((S1 XOR S2 mod 232) + (S3 XOR S4mod 232))

This modification leads to the simultaneous execution of two XOR operations. In the case of original F-function which executes in sequential order and it requires 32 Addition operations and 16 XOR operations. But in the case of our modified F-function it requires the same 48 gate operations (32-XOR,16-addition) but time taken to execute these 48 operations will be reduced because of multithreading [2].

We executed 32 XOR operations in parallel order using threads and hence time taken to complete 16 gate operations will be equal to the time taken to complete 32 XOR operations since we are running it in parallel environment [4]

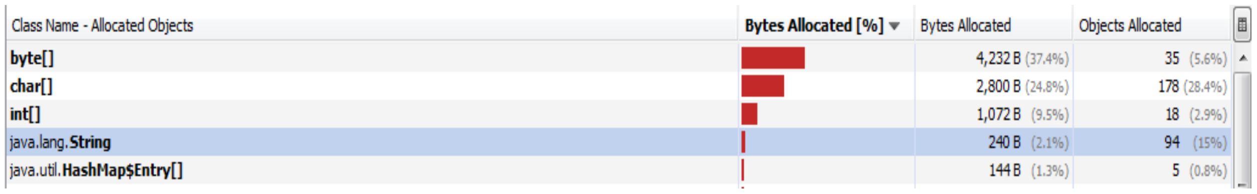
Fig 3: Existing F-Function



```
ALGORITHM STEPS
Divide X into two 64 -bit halves XL and XR
For i=1 to 32:
$XL = XL \oplus Pi$
$XR = F(XL) \oplus XR$
Swap XL and XR
End for
Swap XL and XR (Undo the last swap.)
$XR = XR \oplus P17$
$XL = XL \oplus P1$
Recombine XL and
Output X (64-bit data block: cipher text)
```

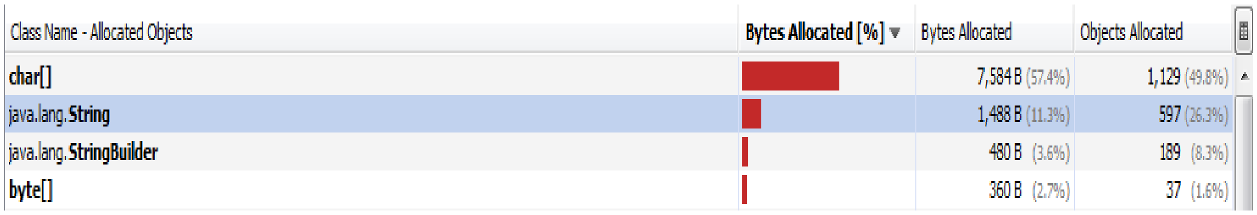
For decryption, the same process is applied, ex-cept that the sub-keys Pi must be supplied in re-verse order. The nature of the Feistel network ensures that every half is swapped for the next round (except, here, for the last two sub-keys P17 and P18).

SIMULATION & RESULTS



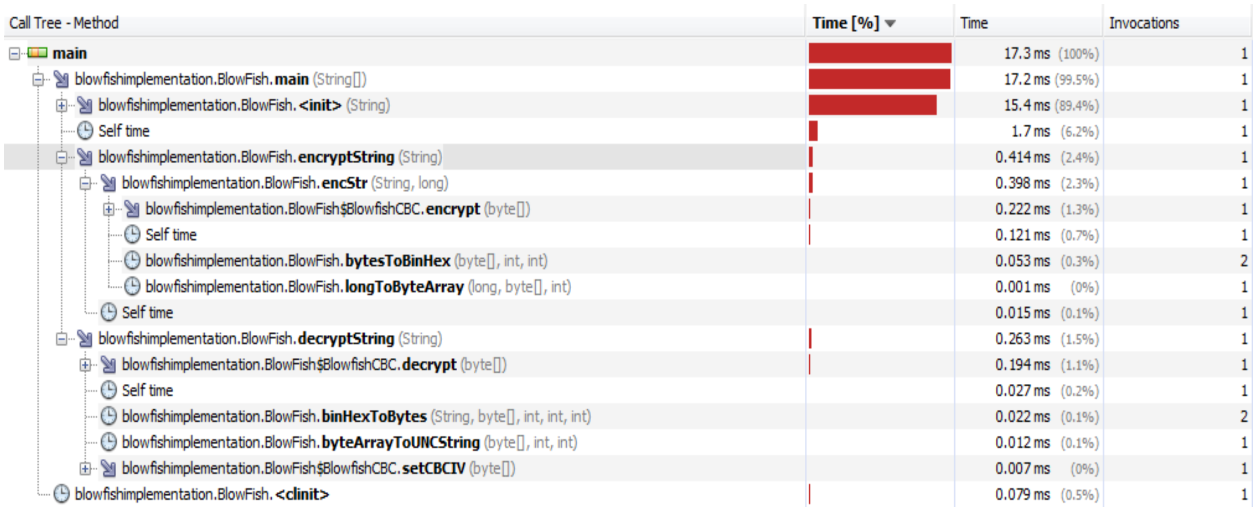
Class Name - Allocated Objects	Bytes Allocated [%]	Bytes Allocated	Objects Allocated
byte[]	🟥	4,232 B (37.4%)	35 (5.6%)
char[]	🟥	2,800 B (24.8%)	178 (28.4%)
int[]	🟥	1,072 B (9.5%)	18 (2.9%)
java.lang.String	🟥	240 B (2.1%)	94 (15%)
java.util.HashMap$Entry[]	🟥	144 B (1.3%)	5 (0.8%)

Fig 5: String Memory Allocation

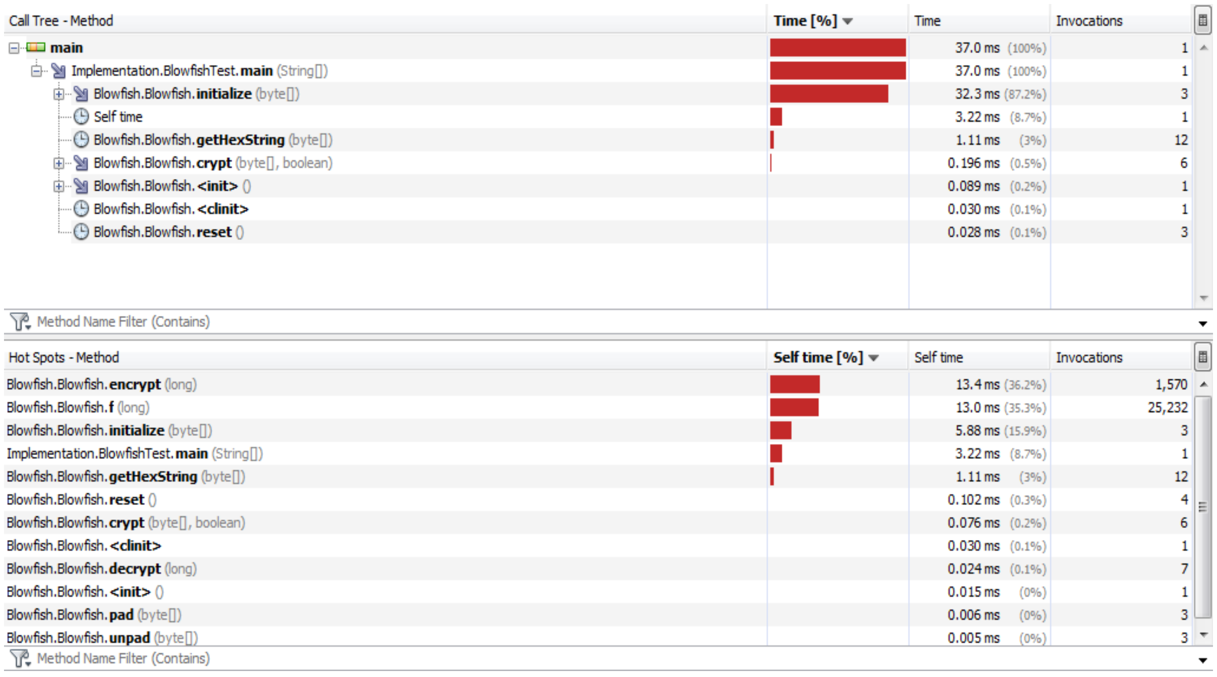


Class Name - Allocated Objects	Bytes Allocated [%]	Bytes Allocated	Objects Allocated
char[]	🟥	7,584 B (57.4%)	1,129 (49.8%)
java.lang.String	🟥	1,488 B (11.3%)	597 (26.3%)
java.lang.StringBuilder	🟥	480 B (3.6%)	189 (8.3%)
byte[]	🟥	360 B (2.7%)	37 (1.6%)

Fig 6: String Memory Allocation



Call Tree - Method	Time [%]	Time	Invocations
main	🟥	17.3 ms (100%)	1
blowfishimplementation.BlowFish.main (String[])	🟥	17.2 ms (99.5%)	1
blowfishimplementation.BlowFish.<init> (String)	🟥	15.4 ms (89.4%)	1
Self time	🟥	1.7 ms (6.2%)	1
blowfishimplementation.BlowFish.encryptString (String)	🟥	0.414 ms (2.4%)	1
blowfishimplementation.BlowFish.encStr (String, long)	🟥	0.398 ms (2.3%)	1
blowfishimplementation.BlowFish$BlowfishCBC.encrypt (byte[])	🟥	0.222 ms (1.3%)	1
Self time	🟥	0.121 ms (0.7%)	1
blowfishimplementation.BlowFish.bytesToBinHex (byte[], int, int)		0.053 ms (0.3%)	2
blowfishimplementation.BlowFish.longToByteArray (long, byte[], int)		0.001 ms (0%)	1
Self time		0.015 ms (0.1%)	1
blowfishimplementation.BlowFish.decryptString (String)	🟥	0.263 ms (1.5%)	1
blowfishimplementation.BlowFish$BlowfishCBC.decrypt (byte[])	🟥	0.194 ms (1.1%)	1
Self time		0.027 ms (0.2%)	1
blowfishimplementation.BlowFish.binHexToBytes (String, byte[], int, int, int)		0.022 ms (0.1%)	2
blowfishimplementation.BlowFish.byteArrayToUNCString (byte[], int, int)		0.012 ms (0.1%)	1
blowfishimplementation.BlowFish$BlowfishCBC.setCBCIV (byte[])		0.007 ms (0%)	1
blowfishimplementation.BlowFish.<clinit>	🟥	0.079 ms (0.5%)	1



Call Tree - Method	Time [%]	Time	Invocations
main	🟥	37.0 ms (100%)	1
Implementation.BlowfishTest.main (String[])	🟥	37.0 ms (100%)	1
Blowfish.Blowfish.initialize (byte[])	🟥	32.3 ms (87.2%)	3
Self time	🟥	3.22 ms (8.7%)	1
Blowfish.Blowfish.getHexString (byte[])	🟥	1.11 ms (3%)	12
Blowfish.Blowfish.crypt (byte[], boolean)	🟥	0.196 ms (0.5%)	6
Blowfish.Blowfish.<init> ()		0.089 ms (0.2%)	1
Blowfish.Blowfish.<clinit>		0.030 ms (0.1%)	1
Blowfish.Blowfish.reset ()		0.028 ms (0.1%)	3

Method Name Filter (Contains)

Hot Spots - Method	Self time [%]	Self time	Invocations
Blowfish.Blowfish.encrypt (long)	🟥	13.4 ms (36.2%)	1,570
Blowfish.Blowfish.f (long)	🟥	13.0 ms (35.3%)	25,232
Blowfish.Blowfish.initialize (byte[])	🟥	5.88 ms (15.9%)	3
Implementation.BlowfishTest.main (String[])	🟥	3.22 ms (8.7%)	1
Blowfish.Blowfish.getHexString (byte[])	🟥	1.11 ms (3%)	12
Blowfish.Blowfish.reset ()		0.102 ms (0.3%)	4
Blowfish.Blowfish.crypt (byte[], boolean)		0.076 ms (0.2%)	6
Blowfish.Blowfish.<clinit>		0.030 ms (0.1%)	1
Blowfish.Blowfish.decrypt (long)		0.024 ms (0.1%)	7
Blowfish.Blowfish.<init> ()		0.015 ms (0%)	1
Blowfish.Blowfish.pad (byte[])		0.006 ms (0%)	3
Blowfish.Blowfish.unpad (byte[])		0.005 ms (0%)	3

Method Name Filter (Contains)

Fig 8: Blowfish Decryption & Decryption

Thus it is experimentally proved that the execution time of modified blowfish algorithm is 13.5% lesser than the original algorithm.

IJESMR
International Journal OF Engineering Sciences & Management Research

SIGNIFICANT FEATURES

This proposal has several merits to be appreciated.

Encryption Function F(x) recombining and re iterating modules:

1. Its simpler design & easy in implementing it.
2. The Design and working is fashioned in such a way that, it is infeasible to breach.
3. Next, On considering the Encryption & Decryption Phase,

a. The execution time of Blowfish algorithm is approximately reduced up to 13.5% on comparing with the original Blowfish Algorithm.

b. Although, we used 2-XOR gates and 1- ADDER but the original F-function uses 2-ADDERs and 1-XOR gate and there is no abrupt change in the execution time or clock cycles required for execution.
This is because all fundamental logical operations like AND,OR,XOR takes more or less equal time when running under any programming languages since those languages are logically driven.

c. It's quite hard for the eavesdroppers to realize that the F-function is modified and hence probability of attack is less on comparing with the original Blowfish algorithm.

CONCLUSION

This paper will satisfy our foremost aim of providing a system which is "infeasible to get breached". It also provides a high end data security when transmitting over any insecure medium. Intruders will not have any idea about
our modification both in terms of algorithm as well as in our design, so breaching this system is highly impossible. We are sure that this software tool is unique of its kind and it can also be tuned in terms of higher performance and security in near future by adding or replacing cryptographic part because of its modularity in design. That is,
it has a good performance without compromising the security and the modified F-function also enhances the performance by reducing the clock cycles upto 33% and reduces the execution time upto 14%

ACKNOWLEDGMENT

First and foremost, I would like to thank my guide, Prof. Vivek Khirsagar, for his guidance and support. I will forever remain grateful for the constant support and guidance extended by guide, in making this paper. Through our many discussions, he helped me to form and solidify ideas. The invaluable discussions I had with him, the penetrating questions he has put to me and the constant motivation, has all led to the development of this project.

REFERENCES

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

[2]. Manikandan Ganesan, Krishnan Ganesan, "A Novel Approach to the Performance and Security Enhancement Using Blowfish Algorithm", *International journal of Advanced Research in Computer Science, 2011.*

[3]. Kishnamurthy G.N, Dr.V.Ramaswamy and Mrs.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. pp. 240-244.

[4]. William Stallings, Cryptography and Network Security, 3rd Ed, Wiley, 1995.

[5]. B. Schneier, "Description of 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.*

[6]. Dr.V.Ramaswamy, Kishnamurthy.G.N, Mrs. 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.

[7]. 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- Verla*g, pp. 18-48, 1998.
[8]. Encryption Technology White paper, http://security.resist.ca/crypt.htm.
[9]. Bruce Schneier, http://www.schneier.com/paper-blowfishfse html security.

ISSN 2349-6193
Impact Factor (PIF): 2.243