An Implementation of 128 bit Blowfish Algorithm with Performance Improvement for High Speed Digital Processing Applications

Chandramohan Kumar[1], Prof. Ashish Raghuwanshi[2]
[1]M.Tech Scholar, Department of Electronics and Communication Engineering
[2]Assistant Professor, Department of Electronics and Communication Engineering
[1&2]IES College of Technology, Bhopal, India

Abstract : Cryptography is best known as a method for keeping the substance of a message mystery. The designed Blowfish as a general-purpose algorithm intended as an alternative to the aging DES and free of the problems and constraints associated with other algorithms. The focus of proposed research is to implement built in self test using verilog coding on xilinx 14.7software. The main motivation behind for proposed algorithm is to extend the existing blowfish and improve performance. Previously it is designed for the 64 bit encryption and decryption process. Presently it is designed for the 128 bit processing. Calculate parameters using standard formula and approach and compare from existing work. Proposed 128 blowfish achieve better results than existing 64 blowfish.

IndexTerms – Blowfish, Cryptography, DES, Xilinx, Encryption, Decryption, delay, power.

I. Introduction

The Blowfish is a symmetric-key block cipher, designed in 1993 by Bruce Schneier and included in many cipher suites and encryption products. Blowfish provides a good encryption rate in software and no effective cryptanalysis of it has been found to date. However, the Advanced Encryption Standard (AES) now receives more attention, and Schneier recommends Twofish for modern applications.

Schneier designed Blowfish as a general-purpose algorithm, intended as an alternative to the aging DES and free of the problems and constraints associated with other algorithms. At the time Blowfish was released, many other designs were proprietary, encumbered by patents or were commercial or government secrets. Schneier has stated that, "Blowfish is unpatented, and will remain so in all countries. The algorithm is hereby placed in the public domain, and can be freely used by anyone.

Blowfish has a 64-bit block size and a variable key length from 32 bits up to 448 bits.[3] It is a 16-round Feistel cipher and uses large key-dependent S-boxes. In structure it resembles CAST-128, which uses fixed S-boxes.

The Feistel structure of Blowfish

The adjacent diagram shows Blowfish's encryption routine. Each line represents 32 bits. There are five subkey-arrays: one 18-entry P-array (denoted as K in the diagram, to avoid confusion with the Plaintext) and four 256-entry S-boxes (S0, S1, S2 and S3).

Every round are consists of 4 actions:

Action 1	XOR the left half (L) of the data with the r th P-array entry
Action 2	Use the XORed data as input for Blowfish's F-function
Action 3	XOR the F-function's output with the right half (R) of the data
Action 4	Swap L and R

The F-function splits the 32-bit input into four eight-bit quarters, and uses the quarters as input to the S-boxes. The S-boxes accept 8-bit input and produce 32-bit output. The outputs are added modulo 232 and XORed to produce the final 32-bit output (see image in the upper right corner).[4]

After the 16th round, undo the last swap, and XOR L with K18 and R with K17 (output whitening).

Decryption is exactly the same as encryption, except that P1, P2, ..., P18 are used in the reverse order. This is not so obvious because xor is commutative and associative. A common misconception is to use inverse order of encryption as decryption algorithm (i.e. first XORing P17 and P18 to the ciphertext block, then using the P-entries in reverse order).

Blowfish's key schedule starts by initializing the P-array and S-boxes with values derived from the hexadecimal digits of pi, which contain no obvious pattern (see nothing up my sleeve number).

The secret key is then, byte by byte, cycling the key if necessary, XORed with all the P-entries in order. A 64-bit all-zero block is then encrypted with the algorithm as it stands. The resultant ciphertext replaces P1 and P2. The same ciphertext is then encrypted again with the new subkeys, and the new ciphertext replaces P3 and P4. This continues, replacing the entire P-array and all the S-box entries. In all, the Blowfish encryption algorithm will run 521 times to generate all the subkeys - about 4KB of data is processed.

II. LITERATURE OVERVIEW

S. B. Nalawade et al.,[1] Security is an important issue during communication and data transmission. There are many ways to provide security. One method to ensure security is the use of cryptographic algorithms such as DES, AES, RC5, Blowfish etc. Cryptography is a method used for encoding the data which may be hacked by the unauthorized person. In this work FPGA based design and implementation of Blowfish algorithm has been proposed.

H. Setiawan et al.,[2] Many Government institutions still applies manual dispositions so that there are still some obstacles including the speed of delivery, accessibility, search, and the absence of security against disposition so that the disposition is prone to damage. In this study an electronic secure disposition application will be developed in accordance with the Regulation of the Minister of State for Administrative Reform and Bureaucratic Reform number 6 of 2011 concerning Regulation of Electronic Service Manuscripts to overcome manual disposition problems in Government Institution. The application will apply the Blowfish algorithm as its encryption method. And digital signatures with SHA-512 hash functions and RSA digital signatures in the attached file.

M. A. Muin et al.,[3] Security level is an essential feature of a cryptographic algorithm. Performing two well known cryptographic algorithms may improve a possibility to gain higher security level. This research implements a composite cryptosystem, which consists of AES256 and Blowfish algorithms. In combining AES256 and Blowfish, two options are available. The first option executes the AES256 followed by Blowfish (AES256-Blowfish). The second option is performing Blowfish and followed by AES256 (Blowfish-AES256).

S. Vyakaranal et al.,[4] proposed work discusses different symmetric key cryptographic algorithms like DES, 3DES, AES and Blowfish by considering encryption time, decryption time, entropy, memory usage, throughput, avalanche effect and energy consumption by practical implementation using java. Practical implementation of algorithms has been highlighted in proposed work considering tradeoff performance in terms of cost of various parameters rather than mere theoretical concepts. Battery consumption and avalanche effect of algorithms has been discussed. It reveals that AES performs very well in overall performance analysis among considered algorithms.

S. Varshney et al.,[5] In this work a hardware architecture is proposed with inner-loop pipelining, loop unrolling for the amalgam of Blowfish and RC6. The used algorithm uses two random numbers “a” and “w” that helps in removing weak key attack and Known plaintext attack of Blowfish. Also the used algorithm uses one S-Box by overlapping process that eliminates the collision key attack of Blowfish. The used algorithm requires less cycles than blowfish and RC6.

I. A. Landge et al.,[6] The sensitive data is encrypted before transmission so that only authorized user can have access to such information. Hardware implementation of encryption algorithm is helpful in designing secured Embedded System. VHDL based Blowfish algorithm implementation and analysis is discussed in this work. The algorithm is implemented with different keys and timing required for encryption, decryption are presented.

T. K. Hazra et al.,[7] In this work we have proposed a new algorithm of encrypting and decrypting images and text files. The proposed method is implemented by combining the concepts of Diffie Hellman algorithm and Blowfish algorithm. In this new technique at first a computer user will encrypt a file using a secret key generated by blowfish algorithm. Then using Diffie-Hellman protocol a shared private key will be generated for two computer users who are trying to communicate over an insecure channel.

A. Chauhan et al.,[8] To attain the goals of security system, the encryption algorithms must definitely provide enough power with high security put in place within an acceptable speed restriction. Therefore, the performance analysis becomes very important to the prevailing encryption algorithms. This work proposes a novel parallel cryptographic algorithm, blending and changing from MD5 and Blowfish encryption schemes, which can upgrade security. A hybrid MD5-Blowfish cryptographic calculation is created to defeat the shortcoming from symmetric block cryptographic and hash function schemes.

A. Gaur, et al.,[9] In the field of networking, one of the emergent technologies is cloud computing whose popularity is increasing day by day. Various types of cloud services are provided by many software companies(such as Salesforces, Microsoft, Amazon, etc) to their users. The cloud employment is reduced as the encrypted data storage and time increases. As the encrypted data is not sliced or distributed form which increases its possibility of being hacked or attacked. In this research work, the hybrid cryptographic algorithm is used to enhance data security by using an encryption algorithm in the cloud and the results are analyzed on the basis of parameters like storage space and time (both encryption and decryption time). This work consists of the combination of Blowfish algorithm and MD5 hashing algorithm and comparison with EDS-AES cryptographic algorithm is shown.

R. Ahmad et al.,[10] It is well-known that advanced encryption standard (AES) algorithm is used for protection against various classes of wireless attacks in wireless communication standard such as WiFi, WiMAX, Zigbee and Bluetooth. However, the AES is a complex algorithm that consumes a larger design core, time, and power source. Hence, this work presents a development of an improved power-throughput Blowfish algorithm on Zynq-7000 field-programmable gate array (FPGA) as an alternative security algorithm. The proposed memory-based method is used to optimize the performance of Blowfish.

III. PROPOSED METHODOLOGY

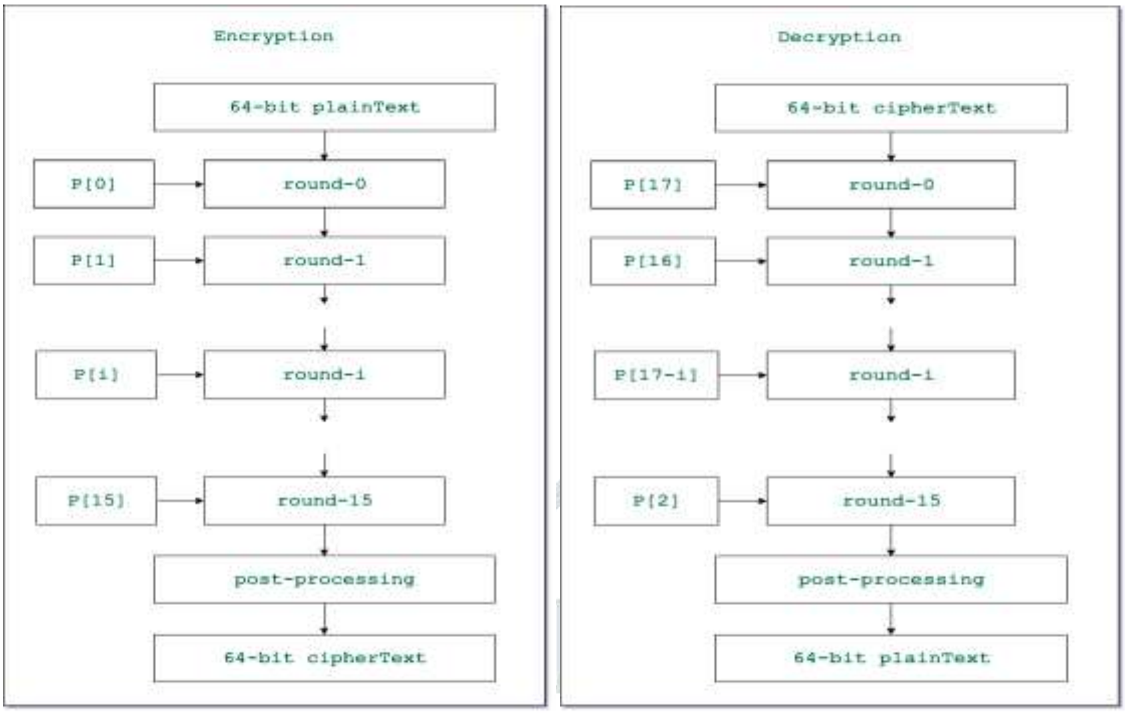

Figure 1: Flow Chart (a) Encryption (b) Decryption

Encryption: Generation of subkeys:

18 subkeys{P[0]…P[17]} are needed in both encryption aswell as decryption process and the same subkeys are used for both the processes.

These 18 subkeys are stored in a P-array with each array element being a 32-bit entry.

It is initialised with the digits of pi(?).

The hexadecimal representation of each of the subkeys is given by:

The resultant P-array holds 18 subkeys that is used during the entire encryption process

Decryption:

The Decryption function also consists of two parts:

Rounds: The decryption also consists of 16 rounds with each round(Ri)(as explained above) taking inputs the cipherText(C.T.) from previous round and corresponding subkey(P[17-i])(i.e for decryption the subkeys are used in reverse).

IV. IMPLEMENTATION RESULT

The implementation and simulation of the proposed algorithm is done over Xilinx 14. The behavioral modeling style and Isim simulator is adopted for simulation. RTL and synthesis results is also generated.

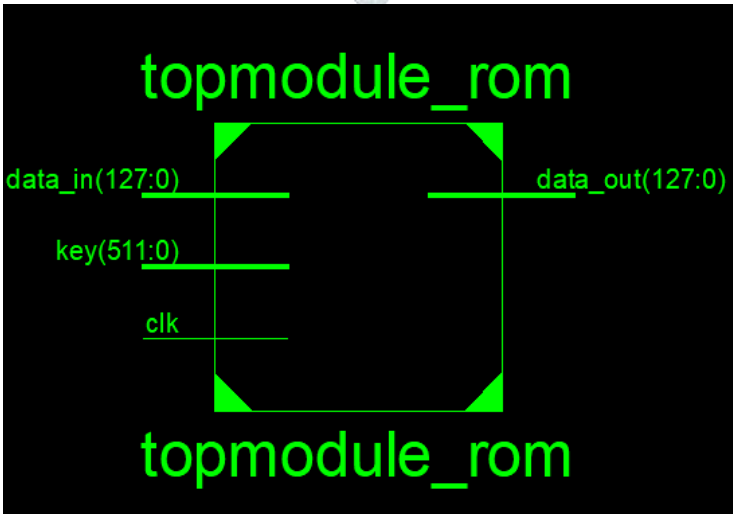

Figure 2: Top module of proposed 128 bit blowfish

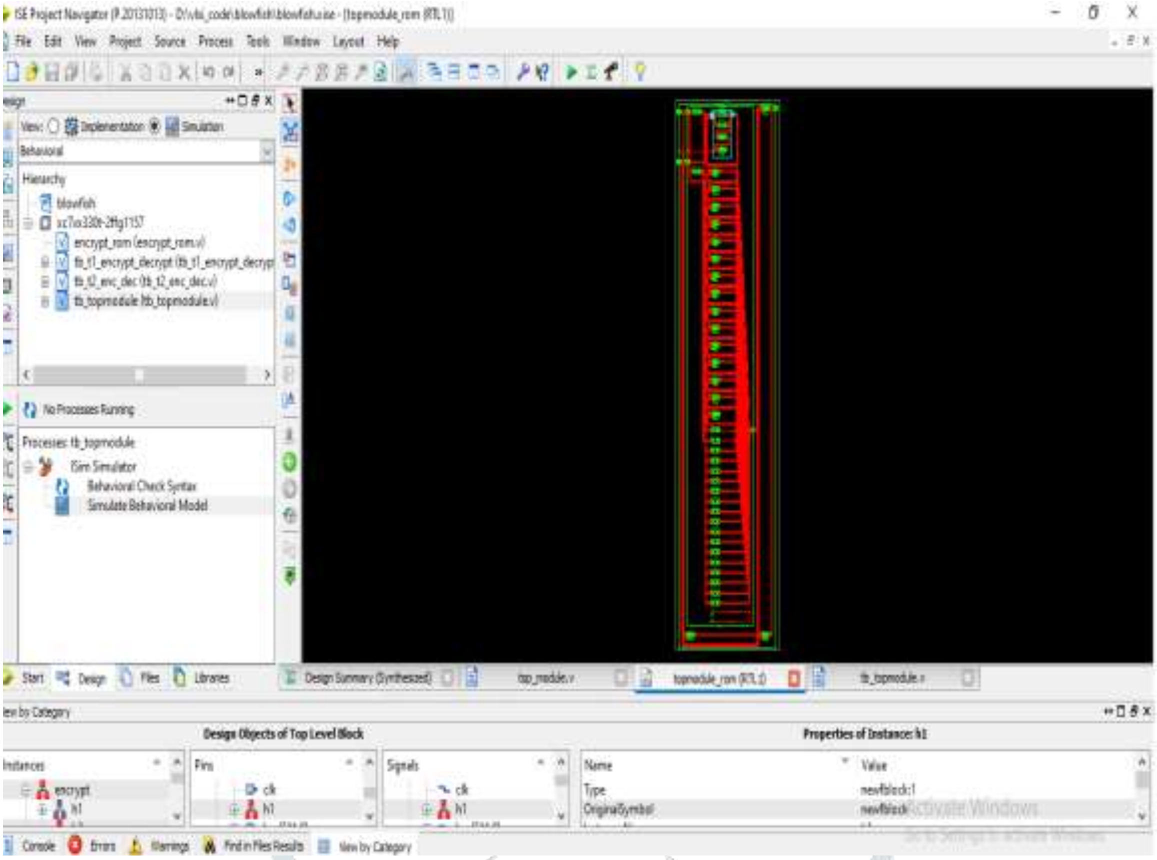
ISE Project Navigator (P.20131013) - D:\vlsi_code\blowfish\blowfish.xise - [topmodule_rom (RTL1)]
File Edit View Project Source Process Tools Window Layout Help
View: ○ Implementation ● Simulation
Behavioral
Hierarchy
blowfish
xc7vx330t-2ffg1157
encrypt_rom (encrypt_rom.v)
tb_t1_encrypt_decrypt (tb_t1_encrypt_decrypt
tb_t2_enc_dec (tb_t2_enc_dec.v)
tb_topmodule (tb_topmodule.v)
No Processes Running
Processes: tb_topmodule
ISim Simulator
Behavioral Check Syntax
Simulate Behavioral Model
Start Design Files Libraries
Design Summary (Synthesized) top_module.v topmodule_rom (RTL1) tb_topmodule.v
View by Category
Design Objects of Top Level Block
Properties of Instance: h1
Instances
encrypt
h1
Pins
clk
h1
Signals
clk
h1
Name Value
Type newfblock1
OriginalSymbol newfblock
Console Errors Warnings Find in Files Results View by Category

Figure 3: Complete RTL view

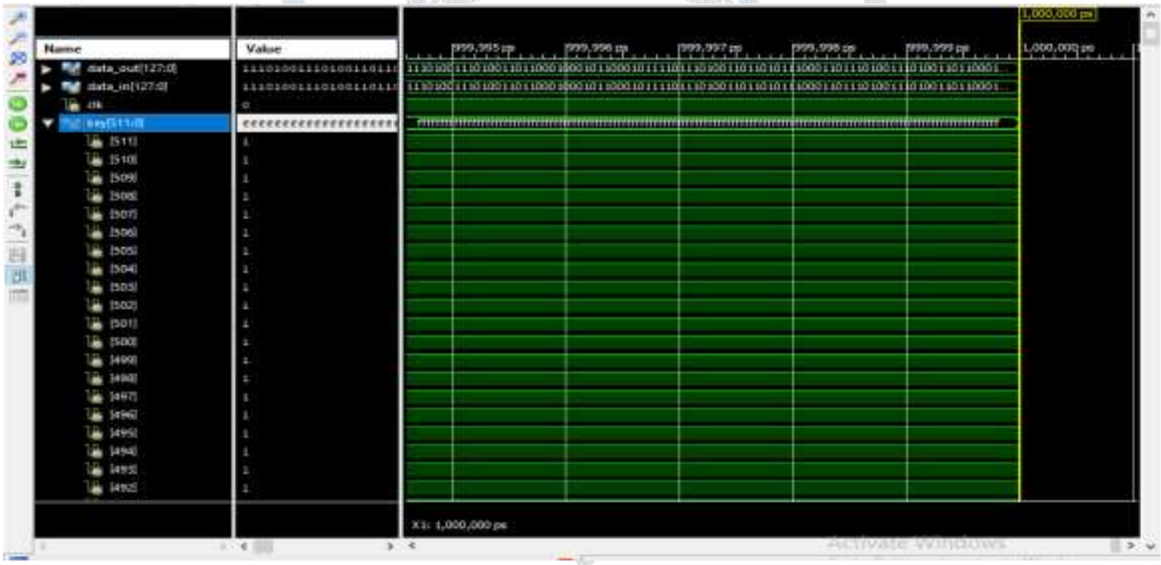

Figure 4: Results in test bench-4

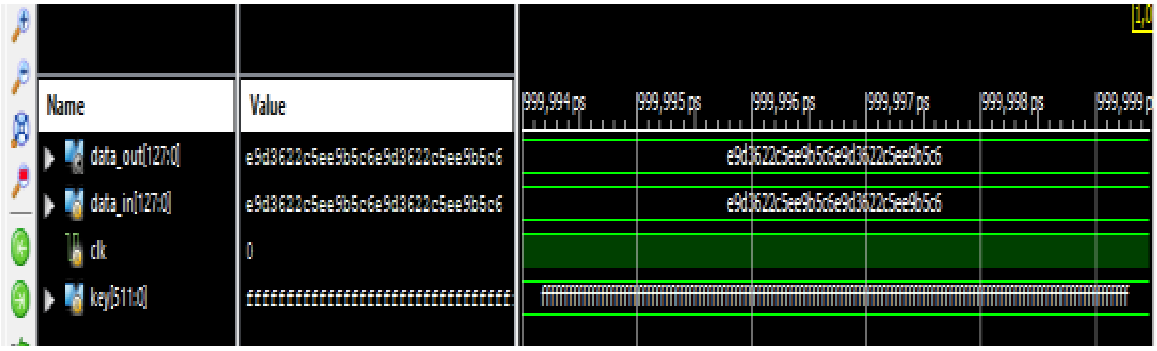
Name
Value
data_out[127:0]
data_in[127:0]
clk
key[511:0]
e9d3622c5ee9b5c6e9d3622c5ee9b5c6
e9d3622c5ee9b5c6e9d3622c5ee9b5c6
0
ffffffffffffffffffffffffffffffff
999,994 ps
999,995 ps
999,996 ps
999,997 ps
999,998 ps
999,999 ps
e9d3622c5ee9b5c6e9d3622c5ee9b5c6
e9d3622c5ee9b5c6e9d3622c5ee9b5c6

Figure 5: Results in test bench-5

Table 1: Simulation Parameters

Sr No	Parameter	Value
1	Area	12%
2	Power	0.45 mW
3	Latency	2.993

Table 2: Comparison chart of proposed work with previous work

Sr No.	Parameters	Existing work result	Proposed work result
1	Method	64 bit	128 bit
2	Area	12.5%	12%
3	Power	0.46 mW	0.40 mW
4	Latency	8 ns	2.993 ns

Therefore proposed work result is better than previous work so 128 bit blowfish approach is considerable and significant results is achieved.

V. CONCLUSION

This paper proposed the scheme to extend the Blowfish block cipher security. In proposed scheme, 128 bit Blowfish is implemented. So total no of rounds are altered by skipping few Blowfish rounds using round key. As a result, proposed scheme increase additional Blowfish cipher security against attack apart from minimum to maximum size of Blowfish key. In addition to that the proposed scheme also decreases encryption and decryption execution time of Blowfish cipher.

REFERENCES

1. S. B. Nalawade and D. H. Gawali, "Design and implementation of blowfish algorithm using reconfigurable platform," 2017 International Conference on Recent Innovations in Signal processing and Embedded Systems (RISE), Bhopal, 2017, pp. 479-484, doi: 10.1109/RISE.2017.8378204.
2. H. Setiawan and K. Rey Citra, "Design of Secure Electronic Disposition Applications by Applying Blowfish, SHA-512, and RSA Digital Signature Algorithms to Government Institution," 2018 International Seminar on Research of Information Technology and Intelligent Systems (ISRITI), Yogyakarta, Indonesia, 2018, pp. 168-173, doi: 10.1109/ISRITI.2018.8864280.
3. M. A. Muin, M. A. Muin, A. Setyanto, Sudarmawan and K. I. Santoso, "Performance Comparison Between AES256-Blowfish and Blowfish-AES256 Combinations," 2018 5th International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE), Semarang, 2018, pp. 137-141, doi: 10.1109/ICITACEE.2018.8576929.
4. S. Vyakaranal and S. Kengond, "Performance Analysis of Symmetric Key Cryptographic Algorithms," 2018 International Conference on Communication and Signal Processing (ICCSP), Chennai, 2018, pp. 0411-0415, doi: 10.1109/ICCSP.2018.8524373.
5. S. Varshney, T. Sudarshan and S. Khare, "Efficient Hardware Architecture for Amalgam of Blowfish and Rc6," 2017 International Conference on Current Trends in Computer, Electrical, Electronics and Communication (CTCEEC), Mysore, 2017, pp. 1126-1130, doi: 10.1109/CTCEEC.2017.8455189.
6. I. A. Landge and B. K. Mishra, "VHDL based BLOWFISH implementation for secured Embedded System design," 2017 Third International Conference on Advances in Electrical, Electronics, Information, Communication and Bio-Informatics (AEEICB), Chennai, 2017, pp. 497-501, doi: 10.1109/AEEICB.2017.7972363.
7. T. K. Hazra, A. Mahato, A. Mandal and A. K. Chakraborty, "A hybrid cryptosystem of image and text files using blowfish and Diffie-Hellman techniques," 2017 8th Annual Industrial Automation and Electromechanical Engineering Conference (IEMECON), Bangkok, 2017, pp. 137-141, doi: 10.1109/IEMECON.2017.8079577.
8. A. Chauhan and J. Gupta, "A novel technique of cloud security based on hybrid encryption by Blowfish and MD5," 2017 4th International Conference on Signal Processing, Computing and Control (ISPCC), Solan, 2017, pp. 349-355, doi: 10.1109/ISPCC.2017.8269702.
9. A. Gaur, A. Jain and A. Verma, "Analyzing storage and time delay by hybrid Blowfish-Md5 technique," 2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS), Chennai, 2017, pp. 2985-2990, doi: 10.1109/ICECDS.2017.8390003.

10. R. Ahmad[1], A. A. Manaf and W. Ismail, "Development of an improved power-throughput Blowfish algorithm on FPGA," 2016 IEEE 12th International Colloquium on Signal Processing & Its Applications (CSPA), Malacca City, 2016, pp. 237-241, doi: 10.1109/CSPA.2016.7515838.
11. V. C. Dongre and S. G. Shikalpure, "Ensuring privacy preservation in wireless networks against traffic analysis by employing network coding and Blowfish encryption," 2016 International Conference on Signal and Information Processing (IConSIP), Vishnupuri, 2016, pp. 1-5, doi: 10.1109/ICONSIP.2016.7857442.
12. S. S. Kondawar and D. H. Gawali, "Blowfish algorithm for patient health monitoring," 2016 International Conference on Inventive Computation Technologies (ICICT), Coimbatore, 2016, pp. 1-6, doi: 10.1109/INVENTIVE.2016.7830230.
13. N. Jayapandian, A. M. J. M. Zubair Rahman, R. B. Sangavee and R. Divya, "Improved cloud security trust on client side data encryption using HASBE and Blowfish," 2016 Online International Conference on Green Engineering and Technologies (IC-GET), Coimbatore, 2016, pp. 1-6, doi: 10.1109/GET.2016.7916767.
14. P. V. Maitri and A. Verma, "Secure file storage in cloud computing using hybrid cryptography algorithm," 2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET), Chennai, 2016, pp. 1635-1638, doi: 10.1109/WiSPNET.2016.7566416.
15. V. P. Bansal and S. Singh, "A hybrid data encryption technique using RSA and Blowfish for cloud computing on FPGAs," 2015 2nd International Conference on Recent Advances in Engineering & Computational Sciences (RAECS), Chandigarh, 2015, pp. 1-5, doi: 10.1109/RAECS.2015.7453367.