National College of Ireland

Modified Blowfish Algorithm to Enhance its Performance and Security

MSc Internship
Cyber-security

Ashokkumar Kothandan
Student ID: x19138857

School of Computing
National College of Ireland

Supervisor: Ross Spelman

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Ashokkumar Kothandan
Student ID:	x19138857
Programme:	Cyber-security
Year:	2020
Module:	MSc Internship
Supervisor:	Ross Spelman
Submission Due Date:	17/08/2020
Project Title:	Modified Blowfish Algorithm to Enhance its Performance and Security
Word Count:	6015
Page Count:	17

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

I agree to an electronic copy of my thesis being made publicly available on NORMA the National College of Ireland's Institutional Repository for consultation.

Signature:	
Date:	16th August 2020

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Modified Blowfish Algorithm to Enhance its Performance and Security

Ashokkumar Kothandan
x19138857

Abstract

Blowfish Algorithm is a 64-bit symmetric block cipher algorithm with 16 Feistel iterations constructed in the year 1993, which is still preferred to be one of the best algorithms. The motive of this paper is to enhance the performance and security of the Blowfish Algorithm by converting it to a 128-bit block cipher and designing a new S-Box function to generate 64-bit data from 8-bit input. On evaluating the performance of the modified block cipher algorithm based on metrics like time taken and memory allocation. The average time taken for the key expansion, encryption, and decryption by the modified algorithm was 0.8ms, 0.2ms, 0.2ms, respectively. The modified algorithm used double the size of the memory allocation required for the original algorithm.

1 Introduction

The rapid growth of data transmission over the internet, especially industrial data transmission, became a critical issue in the last two decades. Various sectors like finance, banking, require a secure way of data transmission between two mediums. Tampering or leakage of data by unauthorized persons commonly called as hackers has increased to a numerous extent. To overcome these security threats, experts have come up with many solutions while transmitting data through the internet. Encryption of data before broadcasting through the "public" channel is to be one of the best approaches in the security phase.

According to Katz and Lindell, "Modern Cryptography is the study of mathematical techniques for securing digital information, systems, and distributed computations against adversarial attacks" [1]. Cryptography provides confidentiality (maintaining data secrecy), integrity (protecting it from any unauthorized person), availability (authorized identity or authentication), and non-repudiation (assuring data not being changed when received) [2]. Encryption algorithm mainly categorized into two categories; the symmetric-key algorithm uses the same key of fixed length for both encryption and decryption, whereas the asymmetric-key algorithm uses a public key for encrypting and private key for decrypting data [1]. Apart from these two algorithms, another type of encryption method is the Hashing algorithm. Hashing the message or data is irreversible, the original text cannot be retrieved but compared with another hashed data. The symmetric-key algorithm includes stream ciphers like Vigenère Cipher and block cipher like AES, DES, Blowfish. Because of the same key used in symmetric-key encryption, a key distribution takes place before the data transmission happens. Bitcoin uses the

same type of asymmetric encryption methodology for the money transaction using public internet securely.

The blowfish algorithm is a 64-bit symmetric block cipher algorithm with 16 Feistel rounds designed by Bruce Schneier in 1993.[1] Blowfish algorithm is a widely accepted encryption scheme and the most recommended block cipher much faster than DES(Data Encryption Standard) and IDEA(International Data Encryption Algorithm). Block Cipher Algorithm is an unpatented, royalty-free algorithm, having a variable key length of 32-bits to 448-bits.[1] Comparing the other algorithms evaluated by Patil *et al.*,, Blowfish Algorithm is superior in security, execution time, and other parameters like avalanche effect, throughput [3].

Though being the most recommended encryption technique, the blowfish algorithm is outdated because of the 64bit data encryption. There is a lot of updated encryptions like AES, DES, Twofish, Threefish algorithm with 128, 192, 256-bit encryption. Another major defect of the blowfish algorithm is the weak keys expansion. According to Vaudenay, differential cryptanalysis on the blowfish algorithm is possible with a piece of information from the F function or by reducing the number of rounds [4]. The substitution-box have weakness based on collision. Weak keys can decrease the significance of complexity of the attackers around eight round when the Function is known [4].

In this paper, author will be addressing on "How to enhance the performance of the blowfish cipher by modifying the algorithm without affecting the basic functionality of the cipher?". This research work attempts to enhance the performance and security of the blowfish algorithm by converting the 64-bit block size encryption to 128-bit block size encryption with the altered S-Box calculation Function by maintaining the original structure of the algorithm. Finally, there will be an evaluation of the primary and modified algorithms based on parameters like time and memory consumption.

2 Related Work

Cryptography is art or study of defending the data leakage by encoding or encrypting the data using mathematical algorithms like block cipher, Vigenere cipher, permutation cipher, and Stream cipher. The block cipher algorithm operates on blocks of a fixed length of bits with a constant transformation using a symmetric key [5].

2.1 Blowfish Algorithm

Blowfish is a 16-round Feistel symmetric-key block cipher algorithm with a block size of 64-bits and the key-length varying from 32 to 448-bits. Block cipher has a complex initialization phase before the encryption takes place [6]. Key-expansion plays a vital role in the initialization of the Blowfish Algorithm. The initialization of the Blowfish algorithm includes instantiation of the Permutation Array(P-Array) and Substitution Boxes(S-Box) with the key-expansion process. Blowfish uses one P-Array of 18, 32-bit keys, and 4 S-Boxes of 256 elements. [7] The values of P-Array and S-Box elements are initialized with the hexadecimal value of pi(0x3.243f6a8885...). For generating the complete set of key-expansion, a total of 521 iterations is required [6].

[1]The Blowfish Encryption Algorithm : `https://www.schneier.com/academic/blowfish/`

The study on the DES and Blowfish Encryption algorithm by Tingyuan and Teng, both the encryption algorithm has resistance against the differential and linear cryptanalysis [8]. the encryption speed of the blowfish algorithm is much higher than DES [8]. Thus the time for the encryption should be taken into consideration when evaluating the performance of the proposed model. Mousa mentioned that the speed of the encryption and decryption is affected by file size, but not by changing the key-length [7].

2.2 Modified Blowfish Algorithm

Vaudenay, reported the first analysis on the blowfish cryptanalysis of the key-length and the Feistel rounds in 1996. Any blowfish encrypted data can be cracked with an adequate piece of information describing the F function or with a weak key-expansion [4]. Later in 2007, Kara and Manap reported on a new class of weak keys in the blowfish algorithm, enhancing the previous work of Vaudenay. This report is about the recovery of the unknown key in $2^{(k+32-16r)/64}$ where the pre-computation phase roughly costs $r \cdot 2^{k-11}$ steps [9].

A modified approach to the S-Box permutation in the blowfish algorithm based on the Fisher-Yates Shuffle(FYS) or Knuth Shuffle(KS) was reported by Corpuz *et al.,* with modified function F = (S1+S4)Mod 2^{32} XOR (S2+S3)Mod 2^{32}. The values of S-Box are randomized within the 256 elements [10]. This method has proved to provide better performance than the original algorithm with 72% of the encryption time and 48% of decryption time for the five different files. Later the same team deployed the encryption algorithm in the cloud computing platform resulting in the performance of the modified algorithm with 440% efficiency in encryption and 308% efficiency in decryption [11].

Analyzing the four different cases of the Modified function F of the blowfish algorithm, Vaibhav and Singh have concluded that all the instances give better performance than the original and case with function F = (S1+S3)Mod 2^{32} XOR (S2+S4)Mod 2^{32} provides the best security [12].

Dulla *et al.,* have designed an enhanced blowfish algorithm by reducing the number of Feistel rounds to 14 and increasing the block size to 128-bits and variable key length [13]. Though the team increased the block size to 128-bits, they used two separate blowfish algorithm for the encryption. The enhanced algorithm improved the performance with an average of 11.36% in encryption and 9.8% in decryption [13]. Reducing Feistel rounds may lead to leakage of data due to the weak key as proposed by Vaudenay.

Ariel Roy Reyes *et al.,* have researched on modifying the blowfish algorithm supporting 128 bit Block size called Blowfish-128 [14]. As shown in Figure 1, the block selector sends each 128-bit for encryption and divided into two equal lengths, 64-bit data. Based on the sum of the generated numbers, the block is sent to either Crypto-algorithm Processor or Inverted Crypto-algorithm Processor. There isn't any descriptive explanation on the Crypto-algorithm Processors and its operation.

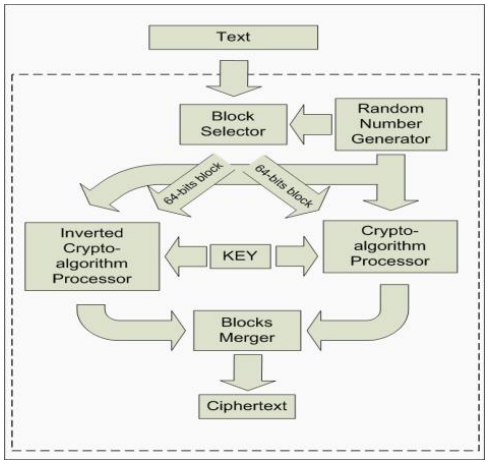

Figure 1: Blowfish-128 [14]

Blowfish algorithm gives better performance for the encryption and decryption when the execution is based on the parallel computation in the Graphical Processing Unit GPU instead of the Central Processing Unit CPU [15]. Analysis proves that the speed of the encryption and decryption after the data is moved from the host memory to GPU memory is 3 to 7 times higher, but the key expansion happens in the host memory as it consumes time for the encryption [15].

2.3 Other Enhanced Algorithms

Reviewing the previous researchers work based on the blowfish algorithm and its enhancements provides some information to design the algorithm without losing its property and standard. Further review will be on the papers that enhanced other similar algorithms. Kalaiselvi and Anand Kumar have designed two modified AES algorithms, one with the Genetic Algorithm, another with Neural network in S-Box. In the Genetic Algorithm, enhanced cipher uses crossover and mutation operators for encryption and decryption [16]. Dynamic block bit size and multi-state tables can control multi-level keys in the Twofish algorithm. From the statistical results produced by Muhajer and Monem, the complexity of the modified Twofish algorithm is comparatively higher than the original Twofish algorithm [17].

There isn't any analysis of the modified algorithm to prove the strongness and withstand against brute-force attacks. Similar to the previous paper [17], Wang and Jiang reported a design with 3-DES Encryption using the Genetic Algorithm [18]. Genetic Algorithm can be used for generating an optimized key sequence, A block of 192 bits having three 64-bits block each encrypted with three different key Sequences K1, K2, and K3.

Another paper proposed by Catalini *et al.,* on modifying the Twofish algorithm, to increase the security and efficiency by introducing an additional block in each round of encryption. The Additional block required to be simple and well designed based on LFSR, one such algorithm called "Snow cipher" has been adopted [19]. To verify the performance on the security, standardized procedures by NIST were followed. This analysis is achieved by computing the Normalized Cross-Correlation function with the following equation [19].

where A(x,y) and B(x,y) are luminance functions of the original and ciphered image, respectively. x and y are spatial coordinates.

$$NCC(u,v) = \frac{\sum_{x,y}\left[A(x,y)-\langle A\rangle_{u,v}\right]\cdot\left[B(x-u,y-v)-\langle B\rangle\right]}{\sqrt{\sum_{x,y}\left[A(x,y)-\langle A\rangle_{u,v}\right]^2\cdot\sum_{x,y}\left[B(x-u,y-v)-\langle B\rangle\right]^2}} \quad (1)$$

Twofish-256 is a modified block cipher of 256-bit block encryption. Twofish algorithm is an advanced version of the Blowfish algorithm with 128-bit of encryption. Lung Su *et al.,* proposed Twofish-256 with 16 256bit keys instead of 8 keys sequence [20].

2.4 S-Box Enhancement

de Souza and Luiz have researched on compacting the S-Box module for the Twofish cryptographic algorithm. The compact Feistel function can support the key-size of 128, 192, 256 bits. With the increase in the key-length, the S-Box function expands [21]. The compact S-Box module is shown in Figure 2, where the switch after the XOR of key decides for the next operation based on key-length.

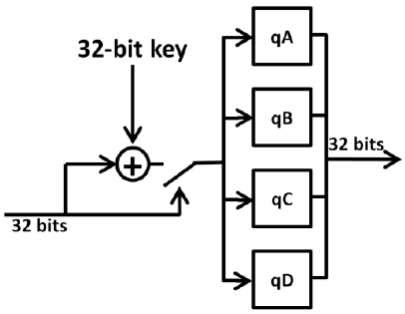

Figure 2: S-Box compact module [21]

S-Box plays a vital role in the cryptographic algorithm. The architecture of the S-Box depends on the algorithm. In the two non-memory based S-Box, the Galois field-based S-Box was designed by Rijmen to reduce the overall computation cost in AES S-Box [22] and combinational logic based S-Box using the 16x1 multiplexer or 4x1 multiplexer.

2.5 Performance Analysis

Further, we will discuss the evaluation of performance on different parameters. To evaluate the performance of different block cipher, Kubadia *et al.,* have used metrics like encryption time, decryption time, throughput, and Cipher to Text Ratio [23]. Cipher to Text Ratio is to determine the extra padded bytes generated by the algorithm.

Similarly, [24], [25] and [26] were also analyzed based on the evaluation of the encryption, decryption speed, power consumption, and memory used. Priyadarshini *et al.,* researched the comprehensive evaluation of cryptographic algorithms includes symmetric and asymmetric ciphers [27]. The evaluation parameters used were Encryption and Decryption time - to determine the time taken by both encryption and decryption. Memory utilized by each algorithm. Avalanche effect- a small change in input makes a significant

change in output. To determine the Avalanche effect, we use the Hamming distance(sum of bit by bit XOR of ascii value) [27]. Entropy is the measure of randomness or uncertainty in the output. Entropy can be determined by Shannon's formula [27].

3 Methodology

Further, author will discuss on the methodology to achieve the proposed modified blowfish algorithm. The previous section helped in learning the design and structure of the blowfish algorithm. Later, we studied the previously modified blowfish algorithms and other modified algorithms like AES, DES, Twofish. Some of the papers like [4], [10], [20] helped in understanding the algorithm and its limitations. Other papers like [14] and [13] didn't find much helpful, and the modification in the design changes the basic functionality of the algorithm. Some of the evaluation methods author reviewed will be used in our performance analysis. Author used the source code of the blowfish algorithm written in java by Markus Hahn in Septemeber 2002. The source code is available in the GitHub[2] by author Matt Iversen.

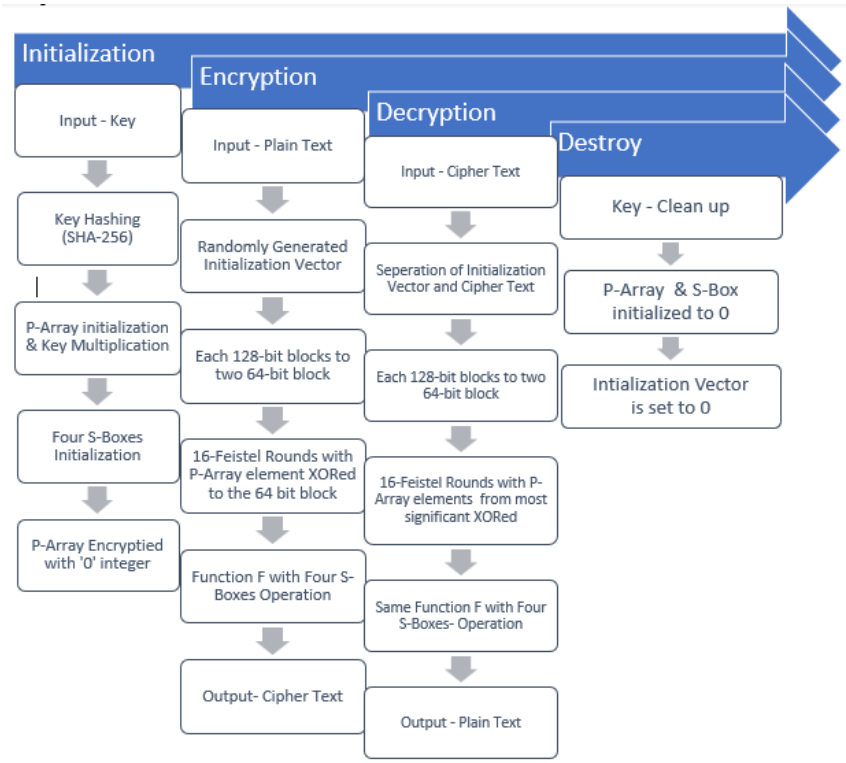

Figure 3: High Level Design of Enhanced Blowfish Algorithm

3.1 Initialization & Key Expansion

Although the process of modifying the algorithm is on the encryption and decryption, there is an initialization phase called the key-expansion. The modified blowfish symmetric block cipher algorithm encrypts with each block of 128-bits. Any size of the key

[2] https://gist.github.com/Ivoz/1342537

as input will be converted into 256-bit by hashing using the SHA-256 algorithm. The key is encrypted by the SHA-256 encryption algorithm to increase the strength of the initialization. Initialization of the P-array and S-Box is from the hexadecimal value of pi(π). P-Array has 18 elements of a 64-bit key set from P1 to P18. Four 64-bit S-Boxes has 256 entries each. S-Box used here converts the 8-bit data to 64-bit data.

Around 1000 digit of pi value is taken from a website[3] and converted to hexadecimal value using java program. The result is used to initialize the P-Array and S-Boxes. Consequent 64 bits of the key are XORed with each element of P-array. After 256-bit of the key, this key cycle is repeatedly used for all the elements in P-array.

3.2 Encryption

After the key-expansion of the algorithm, we will design the encryption phase. The given data that needs to be encrypted is divided into multiple 128-bit blocks, and a new Initialization Vector of 128-bit is randomly generated for each data that needs to be encrypted. Each block of 128-bit data is XORed with the generated Initialization Vector, and the result is sent to the encryption block.

In the encryption block, the resulted 128-bit block is divided into two 64-bit blocks. These two 64-bit blocks are sent into the 16 Feistel rounds, and in each round, each element of P-Array is XORed. Each round has the operation of function F for the S-Box calculation. All the four S-Boxes are used here to convert the 8-bit file to 64-bit data. Finally, after all the operation in the encryption block, the two 64-bit encrypted data is combined to make it into 128-bit data. The result is stored in the initialization vector variable and XORed with the next 128-bit of data and, the encryption loop continues till the last. Finally, the last remaining bits are padded with 0 to make it 128-bit value. The output byte array and Initialization Vector are converted to a hexadecimal string and returned as ciphertext.

3.3 Decryption

In the decryption phase, from the given cipher data, the first 128-bit data is stored in the Initialization Vector. The second part of the cipher data is to be converted to the plain text using the decryption scheme of the blowfish algorithm. Each 128 bit of data from the second part of the ciphertext is sent to the decryption block.

The decryption block is as same as the encryption block with 16 Feistel rounds. Each 128-bit ciphertext is divided into two 64 bits for the decryption. The reverse operation of the encryption happens in the 16 rounds. The most significant element in the P-array i.e., the 18th element of the p-array is XORed to cipher followed by the previous element from the array. The same function F used in the encryption scheme is used for the decryption block, with the S-Box generating the decrypted cipher value for the specified block. The resulted value is XORed with the Initialization Vector and returned as the decrypted text. In the decryption scheme, there isn't a need for padding at the last set of bit, as the ciphertext will always be a multiple of 128. After the cycle completes the resulted byte array is converted to the plain text using UNC string conversion.

[3] `http://www.geom.uiuc.edu/~huberty/math5337/groupe/digits.html`

3.4 Performance Metrics

To determine the performance of the modified blowfish algorithm is better than the original version, we will use some of the parameters used in the previous papers as discussed in section 2.5. Some of the parametric metrics are

Time: One of the important metrics to be concentrated when analyzing the performance is the time taken for each action by the algorithm. Time taken for the key Expansion, Encryption, and Decryption are calculated for both the original and modified algorithm to find the difference with the help of different key sizes and plain text length.

Memory: Another metric, we will be evaluating the performance is based on the memory consumption of the algorithm in each phase. An average of different input is taken and determined for both the original and modified algorithms.

4 Design Specification

In this section, we will discuss the architecture and specifications of the algorithm with the variable declaration.

Key - In the initialization phase, a new object is created based on the specific key provided by the user with datatype "string" of any length. The provided key is hashed to a 256-bit byte array using SHA-256.

P-Array - P-Array consists of 18 elements, each of 64-bit value. We will use the hexadecimal pi value718BCD5882154AEES to initialize the P-Array. For example, P1 is stored with the first 64-bit of pi(π in hexadecimal) value 243F6A8885A308D3, P2 is stored with the next 64 bit i.e., 13198A2E03707344. The process continues until the P18 value.

S-Box - Next 64-bit of pi value, after initializing P18, is used for the S-Box1 1st element. Similarly, All the 256 elements of the S-Box1 is initialized, followed by other S-Boxes are filled each with 256 elements. Similar to the P-Array all the elements used are stored in "Long" data type.

4.1 Modified Function F(n)

The detailed construction of the modified function F() is shown in the Figure 4. Consider 'n' be the 64-bit block from the encryption or decryption block sent to function. As the S-Box is designed to convert the 8-bit value to 64-bit value, we divide n into two halves, namely nL and nR, 32-bit each. Both the nL and nR are divided into four 8-bit values. Each 8-Bit value is sent to the four S-Boxes, shown in the Figure 4. First 8-bit of nL and first 8-bit of nR is XORed. Similarly, corresponding nL and nR values are XORed. Function(F) is given by

$$S_1\text{out} = (S_1(\text{n}L_1) \text{ XOR } S_1(\text{n}R_1))$$

$$S_2\text{out} = (S_2(\text{n}L_2) \text{ XOR } S_2(\text{n}R_2))$$

$$S_3\text{out} = (S_3(\text{n}L_3) \text{ XOR } S_3(\text{n}R_3))$$

$$S_4\text{out} = (S_4(\text{n}L_4) \text{ XOR } S_4(\text{n}R_4))$$

$$\mathbf{F(n) = (((}S_1\mathbf{out} + S_2\mathbf{out)mod64 \ XOR} \ S_3\mathbf{out)} + S_4\mathbf{out)mod64}$$

where nL_1, nL_2, nL_3, nL_4 are First, Second, Third, Forth 8-bit of nL, respectively.

nL_1, nL_2, nL_3, nL_4 are First, Second, Third, Forth 8-bit of nL, respectively.

$S_1()$, $(S_2()$, $(S_3()$, $(S_4()$ are S-Box1, S-Box2, SBox3, and SBox4, respectively.

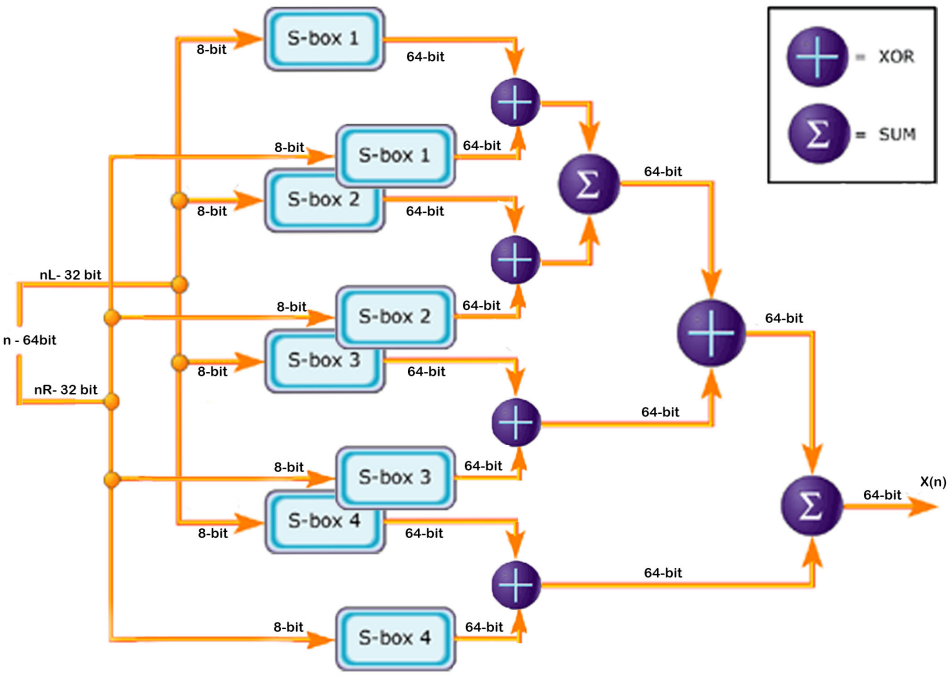

Figure 4: Modified Function F

4.2 Encryption

In this section, we will discuss the architecture of the blowfish encryption algorithm as shown in Figure 5. Given plain text is divided into multiples of 128 bits and sent to the encryption block.

1. Consider X be the 128-bit data sent to the encryption block. It is divided into two 64-bit block X_L and X_R.
2. For : n = 1 to 16
   $P_n' = X_L$ XOR P_n
   $F_n' = \mathrm{F}(P_n')$ where F(x) is the modified function
   $X_L = F_n'$ XOR X_R
   $X_R = P_n'$
3. After 16th iteration
   $X_L = X_L$ XOR P_{18}
   $X_R = X_R$ XOR P_{17}
4. Both the 64bit output from the encryption block is combined to make a 128-bit encrypted block of cipher data.

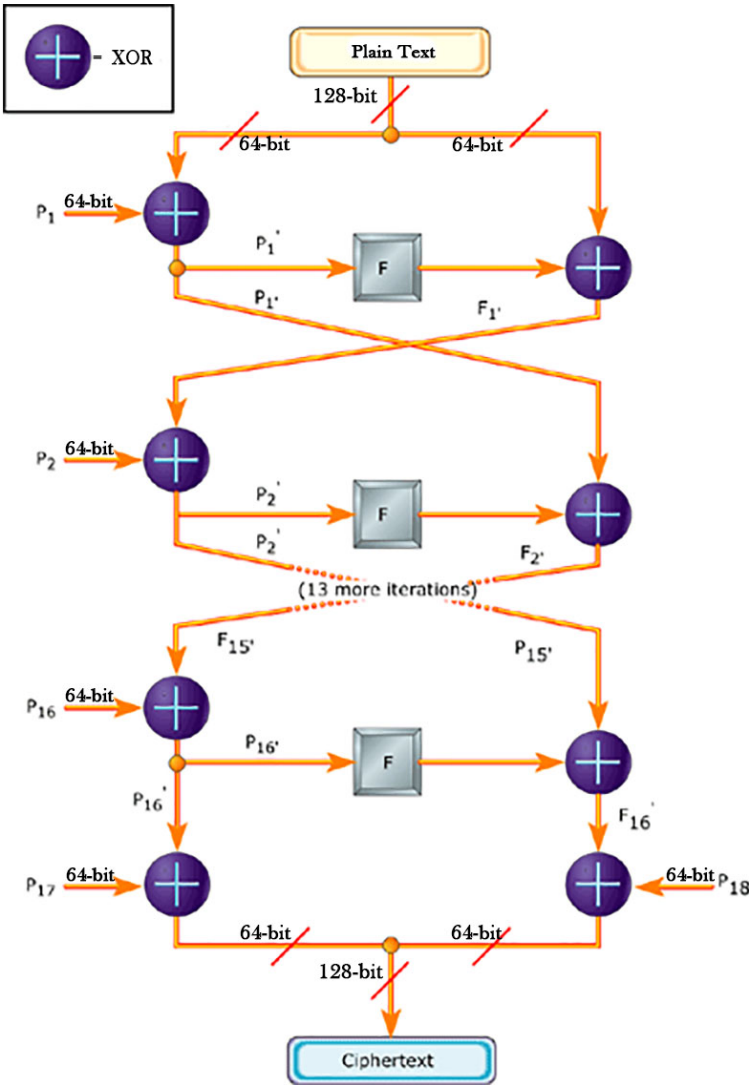

Figure 5: Blowfish Encryption

4.3 Decryption

Same as the encryption block, the decryption block has 16 Feistel rounds of encryption. After the separation of the Initialization Vector, the remaining ciphertext is sent to the decryption block of 128-bits.

1. Consider X be the 128-bit cipher data sent to the decryption block. It is divided into two 64-bit block X_L and X_R.
2. For : n = 1 to 16

   $P'_{(19-n)} = X_L$ XOR $P_{(19-n)}$ —— i.e., starts from P_{18} till P_3

   $F'_n = \mathrm{F}(P'_{(19-n)})$ where F(x) is the modified function

   $X_L = F'_n$ XOR X_R

   $X_R = P'_{(19-n)}$
3. After 16th iteration

   $X_L = X_L$ XOR P_1

   $X_R = X_R$ XOR P_2

4. Both the 64bit output from the decryption block is combined to make a 128-bit of plain text.

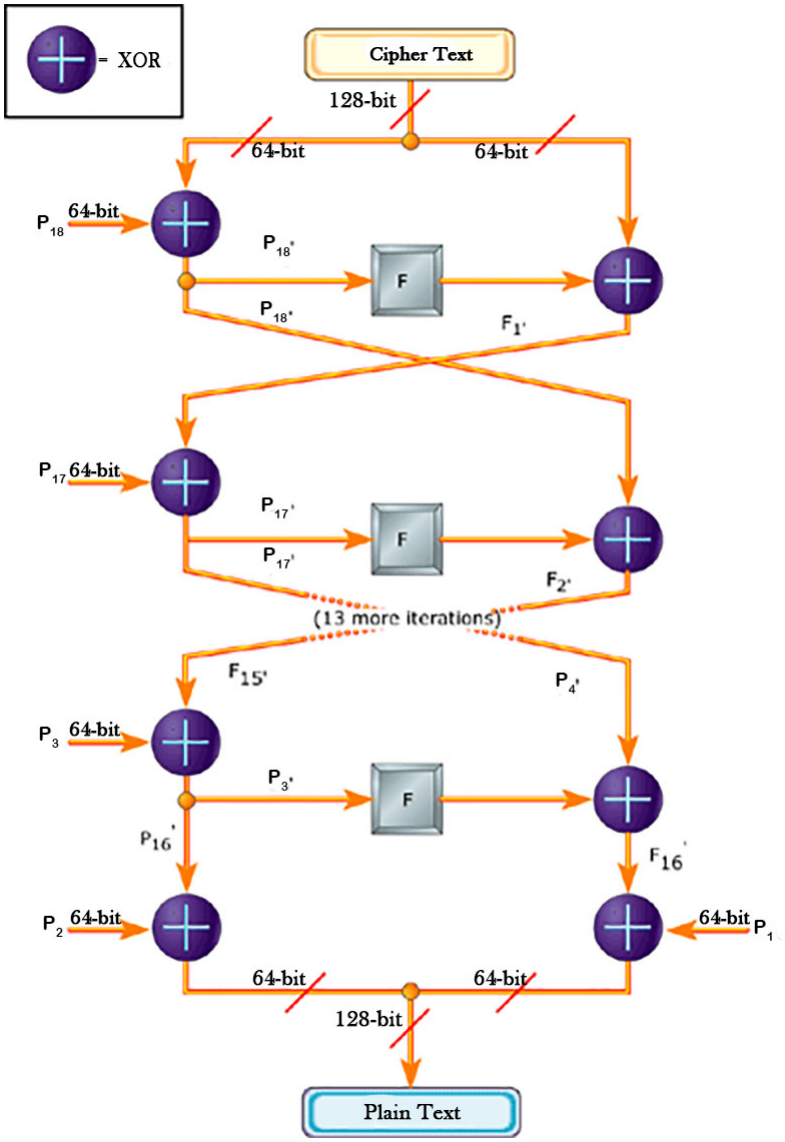

Figure 6: Blowfish Decryption

5 Implementation

5.1 System Specification

The blowfish algorithm is designed using Java programming language. The system configuration is as follows.

Model : HP Pavilion Sleekbook
Processor : Intel(R) Core(TM) i5-3317U CPU @1.70GHz
Operating System: Windows 10 Home
RAM : 12 GB
Hard Disk : 750GB

IntelliJ IDEA version 202.1.4 is used to program the java code for the modified algorithm. This source code is written in a single java class. Further, we will discuss the important methods and variables used in the implementation of the algorithm.

5.2 Methods and Variables

For storing the 128-bit data, we use BigInteger. BigInteger class provides analogs to primitive integer operations. BigInteger allows storing any number of bits. We use methods like `byteArrayToBigInteger()` and `bigIntegerToByteArray()` to convert any of the input to 128-bit integer. For the 64-bit and 32-bit storage, we use Long and Integer data types, respectively. Also to separate long values from BigInteger we use the methods `longLo64()` and `longHi64()`. Similarly, `intLo32()` and `intHi32()` is used to convert the long value to two integers of 32-bits.

5.3 Input and Output

The public Java class is created with the name ModifiedBlowfish, and the parameterized constructor uses the key as an argument. To initialize the key we use the following code.

```
ModifiedBlowfish blowfish = new ModifiedBlowfish("12345");
```

12345 represent the key sent to the ModifiedBlowfish class as the argument. This initializes the ModifiedBlowfish class followed by the hashing of key using SHA-256. MessageDigest[4] class is a one-way hashing function provided by oracle as an inbuilt class. `digest.update()` method is used to provide the input for hashing using SHA. `digest.digest()` method is used to retrieve the 256-bit hashed data.

```
MessageDigest digest = MessageDigest.getInstance("SHA-256");
```

Public method `encryptString()` is implemented for calling the encryption process with the string that needs to be encrypted as a parameter. For example,

`blowfish.encryptString("abcde");` where 'abcde' is the plain text that needs to be encrypted. Output for the same string varies everytime we run the code. This is due to the Initialization Vector randomly generated by the encryption method for each processing. One of the output returned after encryption for the plain text is `"d5d1046fda09796c4c88d5b4eb636efad0b89be737b6c0a3e8ff16ad07627db"`.

Ciphertext returned has 256-bit of data, where the first 128-bit i.e, the first 32 characters `"d5d1046fda09796c4c88d5b4eb636efa"` represents the Initialization Vector. Following 128-bit `"d0b89be737b6c0a3e8ff16ad07627dbb"` is the ciphertext encrypted by using the blowfish algorithm for the plain text "abcde".

Similarly, public method `decryptString()` is implemented for the decryption process with the input as ciphertext. Ciphertext length is in multiples of 128-bits. The first 128-bit refers to the Initialization Vector, and the following text is the encrypted ciphertext.

Public void method `destroy()` is called to destroy or delete all the initialized factors like P-Array, S-Boxes, and Initialization Vector to zero. After the cleanup, both encryption and decryption dependencies are ignored, and a new object has to be created for using the blocks.

[4]https://docs.oracle.com/javase/7/docs/api/java/security/MessageDigest.html

6 Evaluation

In this section, to measure the performance of both the original and modified blowfish algorithm, we use the parameters like Time, Memory. In java code to find the time taken for the operation of each method, we use the inbuilt method `System.nanoTime()`. The difference in time before and after the execution is the actual time taken to process.

To determine the used memory of each operation, we use the inbuilt Class called `Runtime`. Some of the necessary methods in Runtime class are `gc()`, used to clear the garbage collector data, `totalMemory()` returns the total memory of the system. `freeMemory()` gives the available memory in the system. Hence the difference is used to determine the memory used by the code to run.

6.1 Case Study 1: Time Taken for Key Expansion

To evaluate the performance based on the time taken for the key expansion, we determine the time taken for the initialization of the array. As already mentioned, code `ModifiedBlowfish blowfish = new ModifiedBlowfish("key");` is used to initialize the key. We evaluate both the original algorithm and the modified algorithm. The determined time difference is measured in nanoseconds(10^{-9}).

As shown in Table 1, the time taken for the modified algorithm is almost 8 to 10 times higher than the original algorithm, and the increase in the key length doesn't affect the time taken by the algorithm. This is because we hash any key given by the user to 256-bits. Time taken for the key expansion by the modified Algorithm is higher but unnoticeable. The average time taken for the initialization of the blowfish algorithm is 0.8ms(milliseconds).

	Time Taken	
Key Length	Modified Algorithm(ns)	Original Algoirthm(ns)
0	1418209	152669
50	1084229	135512
100	1080227	117010
150	777279	91120
200	723287	105119
250	776783	93467
300	756137	108748
350	663487	87503
400	644478	89595
450	641505	97339
500	677470	140607

Table 1: Time Taken for Key Expansion

6.2 Case Study 2: Time Taken for Encryption and Decryption

Similarly, on evaluating the time taken for the encryption and decryption, based on different lengths of plaintext, the original blowfish algorithm is 8 to 10 times faster than the time taken by the modified blowfish algorithm. All the time description is calculated in nanoseconds(10^{-9}). The average time taken for the encryption and decryption phase is 0.2ms(milliseconds).

	Encryption		Decryption	
Plain text	Original Algorithm(ns)	Modified Algoirthm(ns)	Original Algorithm(ns)	Modified Algoirthm(ns)
0	35529	304808	14752	161311
50	20107	229966	24457	182158
100	17349	240570	22051	223074
150	28817	229415	28394	136424
200	13980	144714	44872	170814
250	39887	208551	34822	198007
300	28207	183551	31729	225145
350	30643	216414	70739	256715
400	26708	249431	214371	299852
450	37625	264102	131048	370908
500	45696	315707	69922	381983

Table 2: Time Taken for Encryption and Decryption

6.3 Case Study 3: Memory Allocation

The memory used in the blowfish algorithm is calculated using Runtime class in java. As shown in Table 3, the memory allocation is only for the key Expansion and encryption, but not for decryption. On comparing the memory usage by original and modified algorithm, the modified blowfish algorithm consumes two times the memory required for the original version.

Memory Allocated	Original Algorithm(kb)	Modified Algoirthm(kb)
Key Expansion	495.92	974.12
Encryption	84.04	185.64
Decryption	0	0

Table 3: Memory Allocation

6.4 Discussion

Using metrics like the time consumption and memory utilized by the algorithm in each phase, we can conclude that the original blowfish algorithm is faster and less memory consuming. The modified algorithm uses double the memory size of the original algorithm. In terms of encryption quality and security perspective, the modified algorithm is comparatively powerful as the encryption uses a 128-bit Initialization vector and 64-bit P-array and S-Box elements.

Although the time taken for the encryption and decryption is 8 to 10 times the original algorithm, it maintains the same speed throughout all the sizes of data. When the encryption or decryption time is high, brute force attacks can be controlled. The delay in the encryption or decryption speed is unnoticeable by a human. Thus the modified encryption is considered to be performing better than the older version.

One of the disadvantages is the memory usage of the modified blowfish algorithm. The memory capacity is two to three times higher than the original blowfish algorithm. Modified Algorithm may require excess memory than the prior.

7 Conclusion and Future Work

Thus the enhancement of performance and security of the blowfish algorithm by modifying the block size of the cipher from 64-bit to 128-bit is successfully implemented and evaluated. In terms of evaluation, though the memory consumption of the modified algorithm is large, the security perspective is handled perfectly to overcome the disadvantages of the weak key expansion.

The design of the S-Box generating 8-bit to 64-bit block is simple and other different methods are not evaluated. An evaluation of different S-Box design can be done to generate 16-bit to 64-bit, which is quite difficult. This is because the memory usage will be exponential on using such S-Box. An optimized S-Box can be designed for future work. I have used BigInteger to convert store the 128-bit data, which also increases the memory when initialized each time. This can be optimized by replacing the BigInteger with a user-defined 128-bit data type.

References

[1] J. Katz and Y. Lindell, *Introduction to Modern Cryptography*, 2nd ed. A Chapman & Hall Book, 2014.

[2] E. Conrad, S. Misenar, and J. Feldman, *CISSP Study Guide, Chapter 4 - Domain 3: Security Engineering (Engineering and Management of Security)*, 3rd ed., 2015.

[3] P. Patil, P. Narayankar, D. G. Narayan, and S. M. Meena, "A comprehensive evaluation of cryptographic algorithms: Des, 3des, aes, rsa and blowfish," *International Conference on Information Security & Privacy (ICISP2015)*, vol. 78, pp. 617–624, 2016.

[4] S. Vaudenay, "On the weak keys of blowfish," *Ecole Normale Superieure*, 1995.

[5] C. Tana, X. Dengb, and L. Zhang, "Identification of block ciphers under cbc mode," *8th International Congress of Information and Communication Technology (ICICT)*, 2018.

[6] S. K. Chinta, "Blowfish," 2015.

[7] A. Mousa, "Data encryption performance based on blowfish," pp. 131–134, 2005.

[8] T. Nie and T. Zhang, "A study of des and blowfish encryption algorithm," *TENCON 2009 - IEEE Region 10 Conference, Singapore*, pp. 1–4, 2009.

[9] O. Kara and C. Manap, "A new class of weak keys for blowfish," *Fast Software Encryption. FSE 2007. Lecture Notes in Computer Science*, vol. 4593, pp. 1–4, 2007.

[10] R. Corpuz, B. Gerardo, and R. Medina, "A modified approach of blowfish algorithm based on s-box permutation using shuffle algorithm," 2018, pp. 140–145.

[11] R. Corpuz, R. Medina, and B. Gerardo, "Using a modified approach of blowfish algorithm for data security in cloud computing," pp. 157–162, 2018.

[12] V. Poonia and D. N. S. Yadav, "Analysis of modified blowfish algorithm in different cases with various parameters," pp. 1–5, 2015.

[13] G. L. Dulla, B. D. Gerardo, and R. P. Medina, "An enhanced blowfish (ebf) algorithm for securing x64filemessage content," pp. 1–6, 2018.

[14] A. R. Reyes, E. Festijo, and R. Medina, "Blowfish-128: A modified blowfish algorithm that supports 128-bit block size," 06 2018.

[15] T. Mahajan and S. Masih, "Enhancing blowfish file encryption algorithm through parallel computing on gpu," in *2015 International Conference on Computer, Communication and Control (IC4)*, 2015, pp. 1–4.

[16] K. Kalaiselvi and A. Kumar, "Enhanced aes cryptosystem by using genetic algorithm and neural network in s-box," in *2016 IEEE International Conference on Current Trends in Advanced Computing (ICCTAC)*, 2016, pp. 1–6.

[17] S. M. Kareem and A. M. S. Rahma, "A novel approach for the development of the twofish algorithm based on multi-level key space," *Journal of Information Security and Applications*, vol. 50, p. 102410, 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2214212619304119

[18] L. Wang and G. Jiang, "The design of 3-des encryption system using optimizing keys," in *2019 China-Qatar International Workshop on Artificial Intelligence and Applications to Intelligent Manufacturing (AIAIM)*, 2019, pp. 56–58.

[19] G. Catalini, F. Chiaraluce, L. Ciccarelli, E. Gambi, P. Pierleoni, and M. Reginelli, "Modified twofish algorithm for increasing security and efficiency in the encryption of video signals," in *Proceedings 2003 International Conference on Image Processing (Cat. No.03CH37429)*, vol. 1, 2003, pp. I–525.

[20] Shun-Lung Su, Lih-Chyau Wuu, and Jhih-Wei Jhang, "A new 256-bits block cipher - twofish256," in *2007 International Conference on Computer Engineering Systems*, 2007, pp. 166–171.

[21] O. de Souza Martins Gomes and R. L. Moreno, "A compact s-box module for 128/192/256-bit symmetric cryptography hardware," in *2016 9th International Conference on Developments in eSystems Engineering (DeSE)*, 2016, pp. 94–97.

[22] K. B. Anuroop, A. James, and M. Neema, "Hardware software codesign for a hybrid substitution box," in *2017 30th International Conference on VLSI Design and 2017 16th International Conference on Embedded Systems (VLSID)*, 2017, pp. 423–428.

[23] A. Kubadia, D. Idnani, and Y. Jain, "Performance evaluation of aes, arc2, blowfish, cast and des3 for standalone systems : Symmetric keying algorithms," in *2019 3rd International Conference on Computing Methodologies and Communication (ICCMC)*, 2019, pp. 118–123.

[24] T. Nie, C. Song, and X. Zhi, "Performance evaluation of des and blowfish algorithms," in *2010 International Conference on Biomedical Engineering and Computer Science*, 2010, pp. 1–4.

[25] M. Panda, "Performance analysis of encryption algorithms for security," in *2016 International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES)*, 2016, pp. 278–284.

[26] M. Iavich, S. Gnatyuk, E. Jintcharadze, Y. Polishchuk, A. Fesenko, and A. Abisheva, "Comparison and hybrid implementation of blowfish, twofish and rsa cryptosystems," in *2019 IEEE 2nd Ukraine Conference on Electrical and Computer Engineering (UKRCON)*, 2019, pp. 970–974.

[27] P. Patil, P. Narayankar, D. Narayan, and M. S M, "A comprehensive evaluation of cryptographic algorithms: Des, 3des, aes, rsa and blowfish," *Procedia Computer Science*, vol. 78, pp. 617–624, 12 2016.