Journal of Computer Science 8 (7): 1191-1197, 2012
ISSN 1549-3636

Comparative Study of Performance in Cryptography Algorithms (Blowfish and Skipjack)

[1]Ali Ahmad Milad, [2]Hjh Zaiton Muda,
[1]Zul Azri Bin Muhamad Noh, [1]Mustafa Almahdi Algaet
[1]Department of Computer System and Communication,
Universiti Teknikal Malaysia Melaka, 76100, Melaka, Malaysia
[2]Department of Computer Science,
Universiti Putra Malaysia, 34300, Serdang, Selangor, Malaysia

Abstract: Problem statement: The main goal guiding the design of any encryption algorithm needs to be secured against unauthorized attacks. For all applied applications, performance and the cost of implementations are also important concerns. A data encryption algorithm would not be of much use if it is secure enough but slow in performance because it is a common repetition to embed encryption algorithms in other applications such as e-commerce, banking and online transaction processing applications. Inserting of encryption algorithms in other applications also prevents a hardware implementation and is thus a major cause of tainted overall performance of the system. **Approach:** In this study, the performance of the two of the popular secret key encryption algorithms (Blowfish and Skipjack) was compared. **Results:** Blowfish and Skipjack, had been implemented and their performance was compared by encrypting input files of varying contents and sizes. The algorithms had been implemented in a uniform language C#, using their standard specifications to allow a fair comparison of execution speeds. **Conclusion:** The performance results have been summarized and a conclusion has been presented. Based on the experiments, we can conclude that the Blowfish is the best performing algorithm for implementation.

Key words: Cryptography algorithms, blowfish algorithm, skipjack algorithm, encryption, decryption, feistel network, S-boxes, private key algorithm, Data Encryption Standard (DES), public key

INTRODUCTION

Privacy is a sensitive subject that touches everyone. Common techniques to safe guard privacy are: when writing a private letter an envelope is used to send it, when using a credit card a secret code number is used and when speaking to someone in private. In case of computers, sensitive data is public (e.g., assessment grades, financial accounts). With the Internet, the computer can be used like a telephone or like a post office, with the disadvantage that everybody connected to the network could have access to the data. This is why, especially with computers, privacy is important.

Different levels of security (computer security, network security) have to be measured. Cryptography can be compared to an electronic safe where private data have been hidden. The cryptographer need always think about the intruder. Cryptography can be compared to the chess game, in that we must think not only of our own schemes, but also of our adversaries. Cryptography usually uses a lot of mathematical formulae and logical meanings. The science is quite novel for the public, that is why it is a very difficult subject, but now more people are interested in it and a lot of books dealing with the subject have been written and it is now easy to find good cryptography information. It seems that the strongest cryptography algorithms now exist to the public even if it is very difficult to understand them. Certainly, the best way to know if a cryptography algorithm is strong is to make its source code and documentation available to the public. If no one can break it, then it is safe to use it.

Algorithms types:

Private key algorithm: A private key algorithm uses one password (or one private key) to encrypt a message, to decrypt it the same password is used. The same algorithm or a different one can be used to crypt and decrypt.

Corresponding Author: Ali Ahmad Milad, Department of Computer System and Communication,
Universiti Teknikal Malaysia Melaka, 76100, Melaka, Malaysia

Public key algorithm: A public key algorithm consists of a public key (B) used to encrypt a message and a private key (A) used to decrypt the message, for one public key there is one private key (A I, B I) and only the private key that belongs to the public key can decrypt a message encrypted by the public key. Due to this, everyone can use the public key. If someone wants to send message, they encrypt the message with the receiver's public key and only the receiver, who knows the private key, can decrypt this message.

This algorithm can also be used to sign a message to prove that it is really the sender who is sending a message, to do so the sender encrypts the message with his private key that can be decrypted only with the public key.

There are a lot of cryptography algorithms have been created; it is not the aim of this research. To go into great detail about cryptography, so only two of the cryptography algorithms are going to be quickly explained to give a general idea of how to encrypt and decrypt a messages or files.

Blowfish: Symmetric block cipher, designed by Bruce Schneier and included in a large number of cipher suites and encryption products. Blowfish provides a good encryption rate in software and no effective cryptanalysis of it has been found to date. Blowfish has a 64-bit block size and a variable key length from 32 up to 448 bits. It is a 16-round Feistel cipher and uses large key-dependent S-boxes.

Skipjack: Is a block cipher-an algorithm for encryption-developed by the U.S. National Security Agency (NSA). Initially classified. It was originally intended for use in the controversial Clipper chip. Subsequently, the algorithm was declassified and now provides a unique insight into the cipher designs of a government intelligence agency. Skipjack uses an 80-bit key to encrypt or decrypt 64-bit data blocks. It is an unbalanced Feistel network with 32 rounds. It was specially designed to replace the Data Encryption Standard (DES).

In this study we have been discuss in details Blowfish algorithm and Skipjack algorithm because these two algorithms have not been cracked yet.

There are different ways to attack a cryptography algorithm. If the algorithm security is only based on its secret, once someone finds the algorithm source code it will be very easy to break it. Sometimes cryptography algorithms can have weak points. For example, with algorithms that just consist of adding the same number to all the password letters they are easier to break. This is because it is simple; all that is to be done it to find the number used. This attack needs strong cryptography knowledge and understanding. It is only used for bad cryptography algorithms, but as with everything relating to computer science; it is very difficult to totally avoid making errors, so this attack is always the first one attempted. If no weak points are found the only attack that can be done is the "brute force" attack. This attack is based on the cipher text generated by a cryptography algorithm. If the attacker can get into the password database and even if all the passwords in it are encrypted, software exists that simply try every possible passwords, encrypts them and compares the cipher text generated with the one held in the password database. Of course, it would take too long to try all the possible passwords. Present computers are not fast enough and there are too many possibilities. Keeping these in mind, cryptographers need to provide the world with a new encryption standard. Many of the unbroken are protected by patents. If the world is to have a secure, unpatented and freely- available encryption algorithm, we need to develop several candidate encryption algorithms now. These algorithms can then be subjected to years of public scrutiny and cryptanalysis. Then, the hope is that one or more candidate algorithms will survive this process and can eventually become a new standard. The purpose of the thesis is to discuss the requirements for a standard encryption algorithm.

Literature review: Some cryptography algorithms have been designed in the past. Discussing all of them is outside the scope of this study. However the main ones have been listed according to their category.

Message digest algorithms: As an Internet standard (RFC 1321), message digest algorithms have been active in a wide variety of security applications and is also usually used to check the integrity of files. A message digest hash is typically a 32- characters hexadecimal numbers. Recently, a number of projects have been created message digest "rainbow tables" which are simply accessible online and can be used to opposite many hashes into strings that collide with the unique input.

MD5: Designer by Ron Rivest and Digest length: 16 bytes, Block size: 64 bytes, Max.final block size: 55 bytes, State size:16 bytes. (Rivest, 1992; Schneier, 1996; Berson, 1992; RSA Laboratories Security Dobbertin *et al*., 1996).

Ripemed-320: Designers by Dobbertin *et al*., 1996, Alias: "RIPEMD", Digest length: 40 bytes, Block size: 64 bytes, Maximum Final block size: 55 bytes, State size: 20 bytes. (Dobbertin *et al*., 1996; Menezes *et* *al*., 1997). This message digest is not claimed to provide a security level higher than RIPEMD-160. SHA-384, SHA-512 or Eddy is used instead.

Symmetric key algorithms: Symmetric-key algorithms are a class of algorithms for cryptography that use pettily related cryptographic keys for both decryption and encryption. The encryption key is pettily related to the decryption key, in that they may be equal or there is a simple convert to go between the two keys. The keys, in training, represent a shared secret between two or more gatherings that can be used to maintain a private information link. Other terms for symmetric-key encryption are single-key, one-key and private key encryption. Using the latter term can sometimes conflict with the term private key in public key cryptography.

AES256: (Advanced Encryption Standard): designers: Joan Daemen, Vincent Rijmen and Alias. AES256 is defined as Rijndael with a 128-bits block size and 14 rounds, Key length: 256 bits, Block size: 16 bytes (Savard, 2000; Biham, 1999; Baudron *et al*., 1999; Carter *et al*., 1999; Messerges, 2000; Courtois and Pieprzyk, 2002). There are some claims that the Extended Sparse Linearization (XSL) (Surhone *et al*., 2011) attack can break the AES algorithm. Since AES is already usually used in commerce and government for the transmission of secret information, finding a technique that can shorten the total of time it takes to recover the secret message without having the key would have wide effects. Opinions vary on whether the attack works because the process is experiential and very technical and so it has been proved difficultly to evaluate its complexity. In addition, the process is expected to have a high work-factor, which unless lessened, means the technique would not reduce the energy to break AES very much in comparison to an in-depth search. Then, even if the attack has been analyzed correctly, it is unlikely to affects the real-world security of block ciphers in the close future. However, the attack has been caused some experts to express greater unease at the algebraic simplicity of the current AES algorithm.

DES (Data Encryption Standard): Designers: Don Coppersmith, Horst Feistel and Walt Tuchmann, U.S. National Security Agency and Published: 1976, Block size: 8 bytes, (NBS, 1981; Schneier, 1996; Menezes *et al*., 1997; Biham and Shamir, 1991a, 1991b; Biham and Shamir, 1993; Matsui, 1994a, 1994b; 1995; Biham and Biryukov, 1997; Knudsen, 1995; Knudsen and Mathiassen, 2001; Surhone *et al*., 2011). The fixed 56-bits operative key length is too short to stop brute-force attacks (Wiener, 1994), DES is also weak to differential and linear cryptanalysis.

MATERIALS AND METHODS

There are a lot of algorithms for block ciphers cryptography such as DES, AES, Blowfish, Skipjack, Safer and Tiger. In this study we have been discuss in details Blowfish algorithm and Skipjack algorithm because these two algorithms have not been cracked yet. Blowfish: In 1993, Schneier (1993) published the Blowfish block cipher. At that time, the current Data Encryption Standard (DES) was known to be weak to cryptanalysis and brute-force attacks. Other cryptographic algorithms were available to substitute DES, but many of the cryptographic algorithms were either safe by patents or considered proprietary. Schneier developing Blowfish to be a publicly accessible cryptographic algorithm with the possible to replace DES. Schneier also fortified others to evaluate the performance and security of Blowfish. To date, the security of Blowfish has not been cooperated. Blowfish is a 64-bits symmetric block cipher that uses a variable-length key from 32-448-bits (14 bytes). The algorithm was developed to encrypt 64-bits of plaintext into 64-bits of cipher text powerfully and securely. The operations carefully chosen for the algorithm were table lookup, modulus, addition and bitwise exclusive-or to minimize the time necessary to encrypt and decrypt data on 32-bit mainframes. An aware attempt was made in planning the algorithm to keep the operations simple and relaxed to code while not cooperating security. As with DES, Blowfish includes a 16 round Feistel network for encryption and decryption. But during each round of Blowfish, the left and right 32-bits of data are improved unlike DES which only changes the right 32-bits to become the next round's left 32-bits. Blowfish combined a bitwise exclusive-or process to be achieved on the left 32-bits before being changed by the F function or propagated to the right 32-bits for the next round. Blowfish also combined two exclusive-or operations to be performed after the 16 rounds and an exchange operation. This operation is different from the variation function performed in DES. Schneier (1993) did not include a permutation because from his analysis the variation did not provide any further encryption. For the F function, Blowfish combined table lookup, modular addition and exclusive-or to reduce the time required to perform the operations. Other more complicated and simpler designs were considered, but the operations selected were determined to be an effective method to provide the appropriate level of encryption. The major differences between DES and Blowfish are that Blowfish uses a variable-length key and that the S-boxes are derived from the key. Unlike the 56-bit key for DES, Blowfish can have a key that choices from 32 to 448-bits. This variable length key is then used in the sub-key generation and S-boxes. A variable-length key would make cryptanalysis more difficult for possible attackers. One of the main disadvantages of Blowfish is the time required to reset the algorithm with the key. Once the algorithm has been initialized although, it can encrypt and decrypt data efficiently; therefore, Blowfish would be more applicable for applications that change the key value rarely and encrypt or decrypt large streams of data. Storing the P-array and S-boxes could be an issue for small devices with limited memory, but with the increases in memory technology, this should be less of an issue. Blowfish was initially designed to encrypt and decrypt 64-bits to be compatible to DES, but Blowfish can be climbed to use 128-bits for input. Many of the Blowfish applications have combined this change. The Blowfish algorithm uses a key, which can be up to 448 bits in length, to compute sub-keys, which are used in the actual encryption and decryption. The sub keys used in Blowfish are the P-array and S-boxes. The P-array involves of 18 32-bits values while the 4 S-boxes consist of 256 32-bits values. The original values of the P-array and the 4 S-boxes are initialized with the hexadecimal segment of pi. The P-array is primed first and then followed by the S-boxes. The hexadecimal fraction of pi was selected because it produced a random number for the initialization. Another type of initialization might be used but the initialization values must be random. According to Schneier (1993), designs in the initialization values can result in a weaker cipher. After initialization, the values of the P-array and S-boxes are changed using the key. The key, which can contain up to 448-bits, is segmented into 32-bit values. If the key is less than 448-bits, the key is recurring. For example, if the key was “1234” and each number represented 8 bytes, the first 32-bit value would be “1234” and the next 32- bit value would be “1234.” This operation would be repeated to obtain the required 448-bit key. For a 448-bit key, there would be 14 32-bits values, which mean the key-array (K) has 4 32-bits values less than the P-array. The key is then used to reset the P-array. An exclusive-or operation is performed between each of the 18 32-bits P-array values and a 32-bits value of the key. P1 is XOR-ed with the first 32-bits of the Key (K1) and then P2 is XOR-ed with the next 32- bits of the Key (K2). This process is continued until P15 because the key-array has only 14 32-bits. Giving to Stallings [2003], an exclusive-or operation would be lead with P15 and the first 32-bit key value (K1), which is improper. The key value is actually the next value of the key repetition, which would basically be K15. For example, if the original key was “123”, where each number signifies 8-bits and K14 equaled “1231”, the next 32-bit key value for the exclusive-or operation with P15 would be “2312” (K15) and not “1231” (K1) as requested by Stallings. After the P-array has been initialized with the key, the P-array and S-boxes values are then modified using the encryption routine of Blowfish. First, two 32-bits values consisting of zeros are encrypted. Other original values could be selected but these values must be fixed to stop an attacker from making the same cipher text with two different keys (Schneier, 1993). The cipher text from the encryption of the zero values is used to substitute the P-array values P1 and P2. The cipher text is also used as the input for the following encryption round. The cipher text from this round then substitutes P3 and P4 and the encryption routine is implemented again with the cipher text as input. This process is continued until all 18 values in the P-array and all 256 values for each of the 4 S-boxes is substituted with cipher text. A total of 521 encryptions are preformed to find all the P-array and S-box values. Now that the P-array and S-boxes values have been established, plaintext can now be encrypted. For encryption, the 64-bits plaintext is parted into a left and right half each consisting of 32-bits. The encryption routine consists of a 16 rounds Feistel network. In the first round, an exclusive-or operation is achieved between the left 32-bits (LE-0) and the 32-bit P1 of the P-array. This value changes the next 32-bits right value (RE-1) and this value is also introduced into the F function. The F function takes the 32-bits input and separates it into 4 bytes (8-bits each). These four values are then used for table lookup in their own S-Boxes. Blowfish only uses four 8-bits values for plotting to the S-box values. The 32-bit values of the S-boxes are then manipulated giving to the following formula: 32-bit Output = ((((S[1][a] +S[2][b]) mod 2^32) ⊕ S[3][c])+S[4][d])mod 2^32. The 32-bit value of S-box 1 is additional to the 32-bits value of S-box 2. The modulus of this result by 2^32 is taken as the input for the exclusive-or operation to achieved with the 32-bits value of S-box 3. The result of the exclusive-or operation is then added to the 32- bits value from S-box 4 and the modulus 2^32 are then performed. A bitwise exclusive-or operation is achieved on the final 32-bits output from the F function and the right half of the data (RE-0). The result of this operation becomes the left half 32-bits input for the next round (LE-1). The results of round 1 can be explained by the following equations:

$$\mathrm{LE}-1 = \mathrm{F}(\mathrm{LE}-0 \oplus \mathrm{P1}) \oplus \mathrm{RE}-0$$

$$\mathrm{RE}-1 = \mathrm{LE}-0 \oplus \mathrm{P1}$$

Round 2 is then achieved with inputs LE-1 and RE-1. This process is repetitive for a total of 16 rounds. The universal equations to describe the rounds are as follows:

$$\mathrm{LEi} = \mathrm{F}(\mathrm{LEi}-1 \oplus \mathrm{Pi})\mathrm{REi}-1 \quad \text{where } 1 \le i \le 16$$

$$\mathrm{REi} = \mathrm{LEi}-1 \oplus \mathrm{Pi} \quad \text{where } 1 \le i \le 16$$

After finishing the 16 rounds, LE-16 and RE-16 values are changed. An exclusive-or operation is then achieved between the swapped LE-16 and P18 and also with the swapped RE-16 and P17 to obtain LE-17 and RE-17, singly. The 32-bits values of LE-17 and RE-16 are combined to obtain the 64-bits cipher text.

The decryption process for Blowfish is nearly identical to the encryption process except the P-array values are upturned. For decryption process, the bitwise exclusive or operation is achieved between the first left 32-bit value (LE’-0) of the cipher text and P18. In the encryption, this process would have been achieved with P1. The decryption process has been repeated for the 16 rounds. LE’-16 and RE’-16 are then exchanged and a bitwise exclusive-or operation is achieved with P1 and LE’-16 and also with P2 and RE’-16 to obtain LE-17 and RE-17, individually. LE-17 and RE-17 are then joint to obtain the original plaintext.

Skipjack: Skipjack is the secret (symmetric) key algorithm encrypts and decrypts data in 64-bit blocks, using an 80-bit key called crypto variable. It takes a 64-bits block of plaintext as input and outputs a 64-bits block of cipher text. The change involves performing 32 steps or iterations of a complex, nonlinear function. As the number of rounds growths, the security of the algorithm growths exponentially. The algorithm can be used in any one of the four streaming styles (Electronic Code Book, Output Feedback, Cipher feedback and Cipher Chaining). Skipjack is an algorithm that was advanced in 1987 and put into service in 1993. It is a previously secret NSA encryption algorithm. Skipjack is a typical of a family of encryption algorithms developed in 1980 as part of the NSA suite of "Type I" algorithms, which are suitable for keeping all levels of classified data. Type I algorithms are typically very secure and are usually classified as secret. Skipjack was used to encrypt complex, but not classified, government data. It was implemented in two government encryption devices: the Clipper chip and Fortezza PC card. These devices have many uses and are usually employed by agencies such as the FBI and NSA. They deliver a high level of security for sensitive communications while allowing the capture of telecommunications by law enforcement officials for such things as criminal surveys. Skipjack is an iterated block cipher with 32 rounds of two kinds, called Rule A and Rule B. Each round is defined in the form of a linear feedback shift register with additional non-linear keyed G permutation. Rule B is fundamentally the opposite of Rule A with minor positioning differences. Skipjack applies eight rounds of Rule A, followed by eight rounds of Rule B, followed by another eight rounds of Rule A and followed by another eight rounds of Rule B.

The original meanings of Rule A and Rule B are given in Fig. 1, where counter is the round number (in the range 1-32) and where G is a four rounds Feistel permutation whose F function is defined as an 8×8bits S box (called the F table) and each round of G is keyed by eight bits of the key.

Step Rule A does the following:

- G permutes w1
- The new w1 is the XOR or of the G output, the counter and w4
- Words w2 and w3 shift one register to the right, that is they become w3 and w4 respectively
- The new w2 is the G output
- The counter is incremented by one

Rule B workings similarly: To encrypt, the algorithm starts the counter at 1. The algorithm phases Rule A for 8 steps, then changes to Rule B for 8 more times. The algorithm again changes to Rule A for next 8 steps and finishes the encryption with 8 steps in Rule B. The counter is incremented by one after each step. To decrypt, the algorithm starts the counter at 32. The algorithm steps into Rule B^{-1} for 8 steps, Rule A^{-1} for 8 steps, Rule b^{-1} for 8 more steps and lastly steps Rule A^{-1} for another 8 steps. The counter is decremented by one after each step.

G-Permutation uses the 10-bytes cryptovariable. G-function is a four-rounds Feistel structure. The round function is a fixed byte-substitution table called the F-table. For the Cryptovariable schedule. It is 10 bytes long and used in natural order. The key schedule is repeated in the sense that the same set of four bytes of the sub keys (entering a single G permutation) are repeated every five rounds and there are only five such sets. In addition, the key bytes are separated into two sets: the even bytes and the odd bytes. The even bytes continuously enter the even rounds of the G permutation, although the odd bytes always enter the odd round of the G permutation. To analysis this algorithm. Brute Force Attack: this attacker efforts every possible key on a piece of cipher text until an comprehensible translation into plaintext is obtained. On an average, half of all possible keys must be tried to complete success. It is also called comprehensive search. The resources required to achieve an exhaustive search depend on the distance of the keys. A key of length N bits has 2^N possibilities. Skipjack uses 80-bits keys, which means there are 2^{80} (around 10^{24}) or more than 1 trillion possible keys.

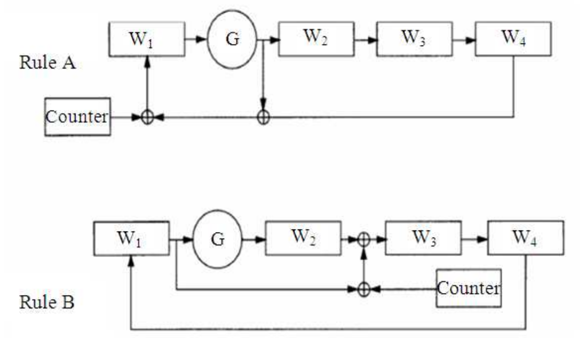

Fig 1: Skipjack Stepping Rules

RESULTS AND DISCUSSION

We feat the advantages object-oriented languages to implement Blowfish and Skipjack algorithms because both of them easy to keep and modify existing code, they afford a suitable execution time and since large files may be involved in encryption process, the .NET Framework's compost collector manage the allocation and release of memory for the application to avoid “out of memory” state to occur during the encryption process.

SkipJack and Blowfish are implemented using Microsoft Visual C# 2008 Express Edition and Microsoft .NET Framework 3.5. The main function of this software is to encrypt and decrypt files using Skipjack and Blowfish algorithm with 64-bits block size. SkipJack and Blowfish cipher deals with any type of files such as .txt, .doc and .jpg. Firstly, SkipJack cipher reads the plain file stream and pads it to complement of 64. Secondly, chop the file stream into 16-bits blocks, each 4 blocks is inserted into the algorithm as plaintext to be encrypted. Finally, the output decrypted text is saved into an output file. Blowfish for encryption, the 64-bits plaintext is divided into a left and right half each containing of 32-bits. The encryption routine consists of a 16 rounds Feistel network. In the first round, an exclusive-or operation is achieved between the left 32-bits (LE-0) and the 32-bits P1 of the P-array. This value becomes the next 32-bits right value (RE-1) and this value is also inserted into the F function. The F function takes the 32-bits input and divides it into 4 bytes (8-bits each). These four values are then used for table lookup in their respective S-Boxes. Blowfish only uses four 8-bits values for plotting to the S-box values. After running the program many times on different type of files we observe that the time needed to encrypt a file is the same time of decrypt it.

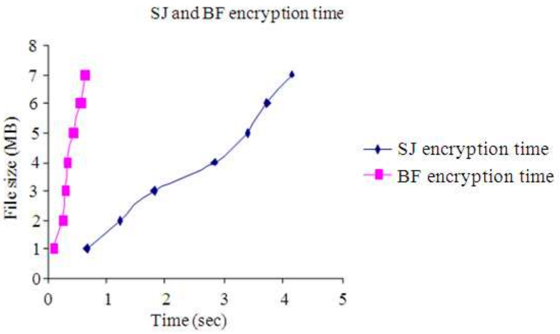

Fig. 2: Skipjack and Blowfish encryption

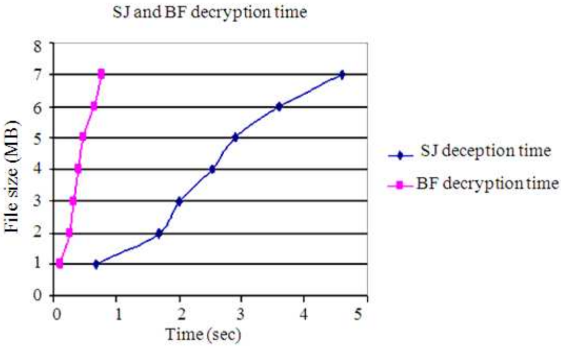

Fig. 3: Skipjack and Blowfish decryption

As we seen here at Fig. 2 and 3 to compare between the encryption and decryption time for Skipjack and Blowfish algorithms. Based on the experiments that we have been implemented, we can conclude that the Blowfish is the best performing algorithm than the Skipjack.

CONCLUSION

In this study, the algorithms Blowfish and Skipjack were executed and their performance was compared by encrypting input files of changing contents and sizes. The algorithms were implemented in a uniform language C# 2008, using their standard qualifications. In the end, the results were presented which conclude that the Blowfish is faster than Skipjack. However, security was not catered for, in practice and one would consider the security first. A proposed direction for the future work could be to analyze the performance/security trade-off in better complexity. For case, an algorithm with more complex rounds and a larger number of rounds is usually careful more secure. The impact of these and other such factors on the overall performance of an algorithm needs to be measured.

REFRENCES

Baudron, O., H. Gilbert, L. Granboulan, H. Handschuh and A. Joux *et al*., 1999. Report on the AES Candidates. Proceeding of the Presented at the 2nd AES Conference, Mar. 22-23, Rome, Italy, pp:1-15.

Berson, T., 1992. Differential cryptanalysis mod 2^{32} with applications to MD5. Proceedings of the 11th Annual International Conference on Theory and Application of Cryptographic Techniques, (TACT' 92), ACM, USA., pp: 71-80.

Biham, E. and A. Biryukov, 1997. An improvement of Davies' attack on DES. J. Cryptol., 10: 195-206. DOI: 10.1007/s001459900027

Biham, E. and A. Shamir, 1991a. Differential cryptanalysis of DES-like cryptosystems. Proceedings of the 10th Annual International Cryptology Conference on Advances in Cryptology, (AC' 91), Springer-Verlag, London, pp: 2-21.

Biham, E. and A. Shamir, 1991b. Differential cryptanalysis of the full 16-round DES. Proceedings of the 12th Annual International Cryptology Conference on Advances in Cryptology, (AC' 91), Springer-Verlag, London, pp: 487-496.

Biham, E. and A. Shamir, 1993. Differential Cryptanalysis of the Data Encryption Standard. 1st Edn., Springer-Verlag, New York, ISBN-10: 0387979301, pp: 188.

Biham, E., 1999. A note on comparing the AES Candidates, Proceedings of the Presented at the 2nd AES Conference (AESC' 99), pp: 1-8.

Carter, G., E. Dawson and L. Nielsen, 1999. Key schedule classification of the AES candidates. Queensland University of Technology.

Courtois, N. and J. Pieprzyk, 2002. Cryptanalysis of block ciphers with overdefined systems of equations. Proceedings of the 8th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology, (ASIACRYPT' 02), ACM Press, UK, pp: 267-287.

Dobbertin, H., A. Bosselaers and B. Preneel, 1996. RIPEMD-160: A strengthened version of RIPE MD. Katholieke Universiteit Leuven.

Knudsen, L. and J.E. Mathiassen, 2001. A chosen-plaintext linear attack on DES. Proceedings of the 7th International Workshop on Fast Software Encryption, (FSE' 00), ACM Press, UK, pp: 262-272.

Knudsen, L., 1995. New Potentially Weak Keys for DES and LOKI. Adv. Cryptology-EUROCRYPT Proc., 950: 419-424.

Matsui, M., 1994a. Linear cryptanalysis method for DES cipher. Proceedings of the Workshop on the Theory and Application of Cryptographic Techniques on Advances in Cryptology, (EUROCRYPT' 93), ACM Press, USA, pp: 386-397.

Matsui, M., 1994b. The first experimental cryptanalysis of the data encryption standard. Proceedings of the 14th Annual International Cryptology Conference on Advances in Cryptology, (CRYPTO' 94), ACM, UK, pp: 1-11.

Matsui, M., 1995. On correlation between the order of s-boxes and the strength of DES. Adv. Cryptol. DOl: 10.1007/Bfb0053451

Menezes, A.J., P.C.V. Oorschot and S.A. Vanstone, 1996. Handbook of Applied Cryptography. 1st Edn., CRC Press, Boca Raton, ISBN: 0849385237, pp: 780.

Messerges, T.S., 2000. Securing the AES finalists against power analysis attacks. Proceedings of the 7th International Workshop on Fast Software Encryption, (FSE' 00), ACM Press, UK, pp: 150-164.

NBS, 1981. Guidelines for Implementing and Using the NBS Data Encryption Standard. 1st Edn., National Bureau of Standards, United States, pp: 39.

Rivest, R., 1992. The MD5 Message-digest algorithm.

Savard, J., 2000. Towards the 128-bit Era: AES Candidates. A Cryptographic Compendium. John J. G. Savard.

Schneier, B., 1993. Description of a New Variable-Length Key, 64-bit Block Cipher (Blowfish). Proceedings of the 1st Software Encryption, Cambridge Security Workshop, (SECSW '93), Springer-Verlag London, UK, pp: 191-204.

Schneier, B., 1996. Applied Cryptography: Protocols, Algorithms and Source Code in C. 2dn Edn., Wiley and Sons, ISBN-10: 0471128457, pp: 758.

Surhone, L.M., M.T. Tennoe and S.F. Henssonow, 2011. National Institute of Science and Technology. 1st Edn., Betascript Publishing, ISBN-10: 6136174987, pp: 136.

Wiener, M.J., 1994. Efficient DES Key Search. 1st Edn., School of Computer Science, Ottawa, pp: 49.