Symmetric Encryption Algorithms: Review and Evaluation study

Mohammed N. Alenezi [1], Haneen Alabdulrazzaq[1], and Nada Q. Mohammad[1]
[1]Computer Science & Information Systems Department, Public Authority for Applied Education & Training, Kuwait

***Abstract*:** The increased exchange of data over the Internet in the past two decades has brought data security and confidentiality to the fore front. Information security can be achieved by implementing encryption and decryption algorithms to ensure data remains secure and confidential, especially when transmitted over an insecure communication channel. Encryption is the method of coding information to prevent unauthorized access and ensure data integrity and confidentiality, whereas the reverse process is known as decryption. All encryption algorithms aim to secure data; however, their performance varies according to several factors such as file size, type, complexity, and platform used. Furthermore, while some encryption algorithms outperform others, they have been proven to be vulnerable to specific attacks. In this paper, we present a general overview of common encryption algorithms and explain their inner workings. Additionally, we select ten different symmetric encryption algorithms and conduct a simulation in Java to test their performance. The algorithms we compare are AES, BlowFish, RC2, RC4, RC6, DES, DESede, SEED, XTEA, and IDEA. We present the results of our simulation in terms of encryption speed, throughput, and CPU utilization rate for various file sizes ranging from 1MB to 1GB. We further analyze our results for all measures that have been tested, taking into account the level of security they provide.

Keywords: Information security, Encryption, Decryption, Cryptography, Symmetric, Block-Cipher, Hashing.

1. Introduction

With the increased usage of data exchange and communication through the Internet, it becomes crucial to secure data from cyber-attacks. Nowadays, providing data confidentiality and privacy has presented a significant challenge for researchers and professionals in the realm of cybersecurity. Data confidentiality means protecting data against unauthorized access or theft. It can be achieved with the help of cryptography through data encryption and decryption. The aim of cryptography is to secure critical data or documents on a hard disk, or when it is transferred through an insecure communication channel.

Data encryption is the art of securing messages by converting them to hidden texts, whereas the inverse process of retrieving original texts from hidden texts is called decryption. Encryption/decryption is made possible with the help of some keys. Every encryption algorithm aims to make the decryption process as difficult as possible without the help of the key used in encryption. Figure 1 shows the general idea of encryption and decryption. There are three types of cryptographic techniques: **symmetric key**, **asymmetric key**, and **hashing** shown in figure 2.

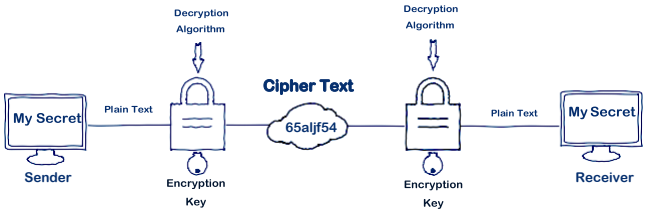


Figure1. General Idea of Encryption and Decryption

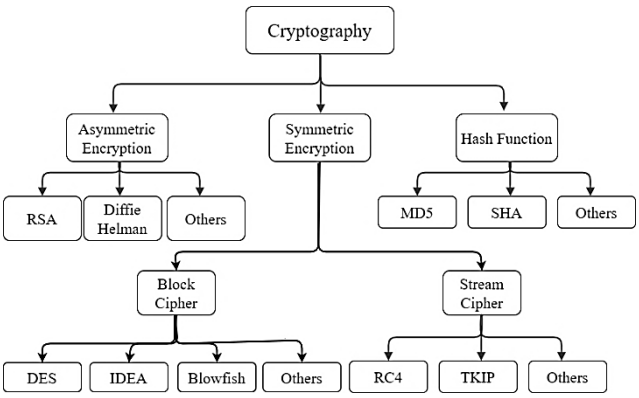


Figure2. Basic Classification of Cryptography

In the symmetric key technique, both Encryption and decryption are done based on a single key called a private key. It is also referred to as a secret key. A secure channel is required for sharing this private key between the sender and receiver. Symmetric key cryptographic algorithms are divided into two types based on the input data: block ciphers and stream ciphers. In block cipher-based systems, data is being processed or encrypted on a fixed-length group of bits called a block, whereas in stream cipher-based systems, data is being processed on a stream of bits. Figure 3 illustrates the process of Symmetric Encryption.

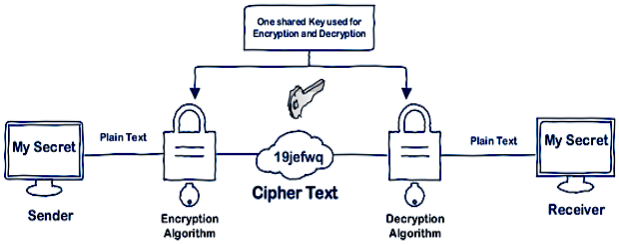


Figure3. Symmetric Encryption

Asymmetric key cryptographic systems require two keys, one is kept secret, and the other is a public key. Encryption is accomplished with the use of a public key, whereas the secret key is used to decrypt the encrypted text. Both of these keys are mathematically related. Although asymmetric systems provide a higher level of security, they might not be well suited for large sized documents. This is because the speed is slow compared with symmetric key-based systems and they also record a higher rate of CPU utilization. Figure 4 illustrates the process of Asymmetric Encryption.

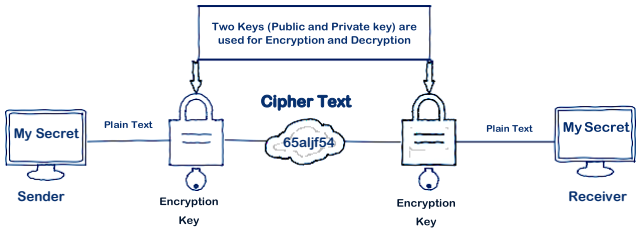


Figure 4. Asymmetric Encryption

The third type of cryptographic algorithms is hashing. In hashing, an input message is mapped into a compact fixed-size bit string called a hash. Hash functions are one-way functions which are mathematical algorithms that map the input message of arbitrary size into a fixed-size hash or message *digest.* Figure 5 presents the general concept of the hash function. Hash functions are mainly used for password storage and data integrity check. The most widely used hash functions are:

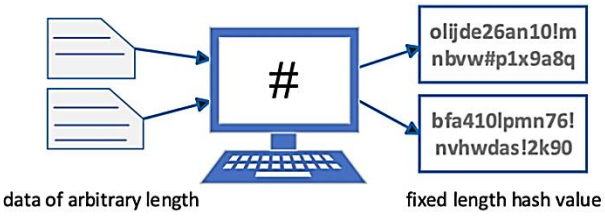


Figure 5. Hash function general concept

- Secure Hashing Algorithm(SHA)
- RACE Integrity Primitives Evaluation Message Digest (RIPEMD)
- Message Digest Algorithm (MD)
- Whirlpool

Digital signatures are mathematical techniques or algorithms that are used to validate the authenticity and integrity of information or messages such as an email, a credit card transaction, or a digital document. It acts like an electronic fingerprint to uniquely identify users and to protect user data. Using a digital signature ensures that a message or document was not modified from the time it was signed. It is done by applying hashing to the document or message and then encrypting the document with the sender's secret key. Digital signatures use Public Key Infrastructure (PKI) to strengthen security. PKI represents the policies and standards which support the distribution of public keys and the identity validation of individuals or entities with digital certificates.

The remainder of this paper is organized as follows: Section 2 shows the related work conducted by various researchers in comparing different encryption algorithms. Section 3 presents an overview of the inner workings of common encryption algorithms. Our performance and analysis are described in Section 4. Finally, the conclusion is presented in Section 5, where we summarize our findings.

2. Related Work

There is a variety of encryption algorithms available to provide privacy of data and confidentiality. Any encryption algorithm will secure data; however, choosing an appropriate algorithm depends on several factors such as performance measures, system specifications, complexity, and the level of security provided. Several researchers have evaluated the performance of various encryption algorithms using different parameters. In this section, we discuss some of the work found in the literature.

Abood and Guirguis [1] made a comparative study of currently available encryption algorithms like AES, DES, TDES, DSA, RSA, ECC, EEE, and CR4 based on their performance in security, key size, complexity and time. Based on this study, AES, BlowFish, RC4, E-DES, and TDES are the fastest algorithms in terms of encryption, time, speed and flexibility. They concluded AES is the most reliable algorithm in terms of speed of Encryption, decoding complexity, key length, security, as well as flexibility.

Riman and Abi-Char [2] have analyzed the performance of four block cipher algorithms such as AES, DES, 3DES, and E- DES based on speed, block size, and key size. They concluded that E-DES outperforms all the other three models based on the input files and experimental results. It encrypts/decrypts the data faster than the other algorithms that were tested. In comparison to DES, E-DES showed an improvement in two areas; more straightforward implementation and more significant key and input blocks to provide security.

Dixit et al. [3] explained the various available encryption and decryption methods and compared them in terms of development, number of rounds, key length, block size, attacks found, level of security, possible keys, time required to check all possible keys, etc. They have made a comparison between traditional as well as hybrid encryption techniques such as DSA-RSA, AES-RC4, RC4-AES-SERPENT, SERPENT-RC4, and AES-ECC. They concluded that AES-ECC reduced time and space complexity and DSA-RSA hybrid algorithm had better performance and throughput.

Bhanot and Hans [4] compared and analyzed different data encryption algorithms in both symmetric and asymmetric categories, to find the best performing algorithm. They compared the algorithms based on development, key length, number of rounds needed for encryption and decryption, block size, various types of attacks found, level of security, and encryption speed. In their study, they observed that the strength of each algorithm could be determined by key management, type of cryptography, number of keys, number of bits used in a key, etc. They have concluded that BlowFish and ECC had better performance results. They also stated that there was no successful attack reported on BlowFish at the time, whereas ECC has been successfully attacked.

Wahid et al. [5] performed an analysis of various encryption algorithms such as DES, 3DES, AES, RSA, and BlowFish, based on their performance, weaknesses, and strengths. In

their study, they concluded that BlowFish is better in terms of memory usage, time, and mitigation of attacks. However, if confidentiality and integrity are the main concerns, then AES becomes the better choice.

Kaur and Mahajan [6] have conducted a comparison of symmetric key algorithms on both a local system and a cloud system. For the two systems, they implemented four symmetric key algorithms: AES, DES, BlowFish, and DESede. The local implementation was done using Java on eclipse and the cloud implementation used eclipse-SDK and Google App Engine. Their evaluation used input lengths of 10KB, 13KB, 39 KB, and 56 KB. They found that the speed-up ratio of DES and BlowFish reflected a small change with an increase in input size, whereas, for AES, it decreases. They noted that DESede was more time consuming and BlowFish was the lowest in terms of time consumption. Their conclusion states that performance-wise, BlowFish, AES, and DES are the better algorithms and AES demonstrated high security with the least time consumption.

Hendi et al. [7] devised a light weight cryptosystem referred to as Simple and Highly Secure Encryption Decryption algorithm (SHSED) for data storage on cloud computing. Their system is based on the IDEA encryption algorithm and its performance was compared against AES, DES, and LED. The proposed algorithm performed better than AES and LED, however, its performance was slightly slower than DES.

Tyagi and Ganpati [8] conducted a theoretical study of four popular symmetric algorithms, such as DES, 3DES, AES, and BlowFish. They compared these algorithms based on various factors like speed, block size, security against attacks, confidentiality, throughput, power consumption, key size, etc. Based on their study, BlowFish had better performance when considering encryption time, decryption time, and throughput. They also concluded that 3DES was the lowest in terms of performance.

Princy [9] analyzed various symmetric key algorithms such as AES, DES, 3DES, BlowFish, RC4, and RC6 with regards to security, performance, processing time, and number of rounds. The results showed that BlowFish delivered more privacy and security in data transmission over an unsafe channel when increasing its key size from 128 to 448.

Mathur and Kesarwani [10] made a comparison of performance between DES, 3DES, AES, RC2, RC6, and BlowFish. They evaluated the performance of these algorithms based on key length, encoding method, data type, and packet size. They found that the encoding methods do not influence the encryption or decryption processes of these algorithms. BlowFish outperformed all the other algorithms when the packet size was changed. Moreover, they showed that RC2 had low performance and throughput in comparison to the other algorithms. RC2, RC6, and BlowFish faced a significant disadvantage over the other algorithms when the data type of input changed from text to image. They also concluded that higher key length would influence both power and time consumption.

Nema and Rizvi [11] analyzed DES, 3DES, AES, BlowFish, Twofish, Threefish, RC2, RC4, RC5, and RC6 based on throughput, scalability, security, memory usage, power consumption, speed, and flexibility. Their results show that BlowFish was the most efficient algorithm in terms of security, flexibility, memory usage, as well as performance.

Marwaha et al. [12] analyzed DES, Triple DES, and RSA based on level of security, time taken for encryption/decryption, and throughput. The performance of the algorithms varies with different input sizes. They summarized that the speed and throughput of DES are better than that of 3DES. Moreover, DES showed less power than 3DES and RSA. 3DES provided more confidentiality and scalability overall, but in comparison to DES and RSA; it consumed more power with less throughput.

Nadeem and Javed [13] implemented and compared DES, 3DES, AES, and BlowFish using Java and evaluated their performance based on varying input types and sizes, execution speeds, and different hardware platforms. Based on their comparison, BlowFish had better performance than the rest. They ranked these algorithms based on execution time: BlowFish (fastest), DES, AES, Triple DES (slowest). The execution speed of the block cipher based algorithms increased when increasing the size of blocks and decreasing the size of the key. However, in stream cipher algorithms, speed decreases when increasing the block size. They also concluded that the security provided by an algorithm increases with the number of encryption rounds, although it slowed down the speed of an algorithm.

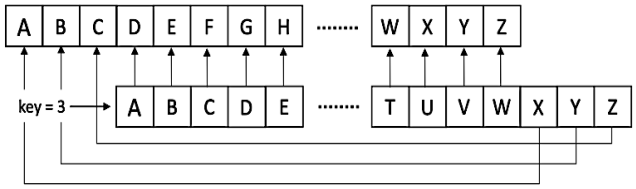
Sun [14] presented a recent survey on most privacy protection techniques proposed in the literature for cloud systems. The work organizes different techniques available in the literature for cloud systems. The survey found several techniques that fall under Attribute-based Encryption (ABE), Key Policy Attribute-based Encryption (KP-ABE), (KP-ABE), Ciphertext Policy Attribute-based Encryption (CP-ABE) and many other techniques. The survey highlights current challenges related to several proposed protection technologies for the cloud. The main challenges listed are: Trust, Access Control, and Encryption. Therefore, encryption for cloud-based systems remains as a current challenge for researchers.

3. Common Encryption Algorithms

There are many encryption methods being used in cryptography. In this section, we detail some common encryption algorithms based on both stream and block ciphers as well as explain the different modes of block cipher-based encryption.

3.1 Caesar

Caesar cipher [15], [16] is one of the most straightforward symmetric block cipher encryption schemes; therefore, it is easy to break. The Roman ruler Julius Caesar created and used this encryption scheme to send military orders to his legions. It is a substitution cipher where encryption and decryption keys are the same. The keys used in this scheme are integers and the most commonly used integer, is 3. In this encryption technique, each alphabet is shifted right or left by a key-value, as shown in figure 6 (with key=3).



Key	3
Plain Text	MY INFORMATION
Cipher Text	PB LQIRUPDWLRQ

Figure 6. Shifting method in Caesar Encryption

3.2 Data Encryption Standard(DES)

DES [16], [17] is one of the basic symmetric key block cipher algorithms which takes plain texts as blocks each one carrying 64 bits and converts ciphertexts using keys of 64 bits. Out of these 64 bits, 8 bits of the key are used for odd parity which will not count in key length. Therefore there exist 2^{56} possible ways to find the correct key. The DES algorithm performs two permutations (initial permutation and final permutation) and 16 processing steps, each of which is called a round, and for each round, a different key is used.

DES is based on two cryptographic operations: substitution and transposition. In each round of DES, some substitutions and transpositions are performed. Before starting the first round, an initial permutation is applied to the plain text. For example, an initial permutation replaces the first bit of the plain text with the 58th bit, and the second with the 50th bit, and so on. The resultant permuted block is divided into two halves, both having 32 bits and each one is going through 16 rounds of encryption processes. The final permutation is applied to the combined block to get the ciphertext. DES has been reported vulnerable and as such was replaced with 3DES [18]. The overall working of DES is explained in figure 7 [16], [17]. There are three modes of operation for DES. They are ECB, CBC, and CFB. We explain these modes in detail in Section 3.14.

3.3 Triple Data Encryption Standard(3DES)

Triple-DES [19] is a block cipher encryption algorithm. As its name indicates, 3DES applies DES three times to each data block to enhance the security of the encrypted data. Since the security of 3DES is three times better than that of DES, it is now considered more preferable than DES. However, it does consume a considerable amount of time in comparison with its predecessor.

3DES works in the same way as DES, in a loop with length 3. Initially, the original plain text is encrypted with one key, the resulting ciphertext is again encrypted using another key, and finally, it is performed again with a third key. The four modes of operation for Triple DES are shown in table 1.

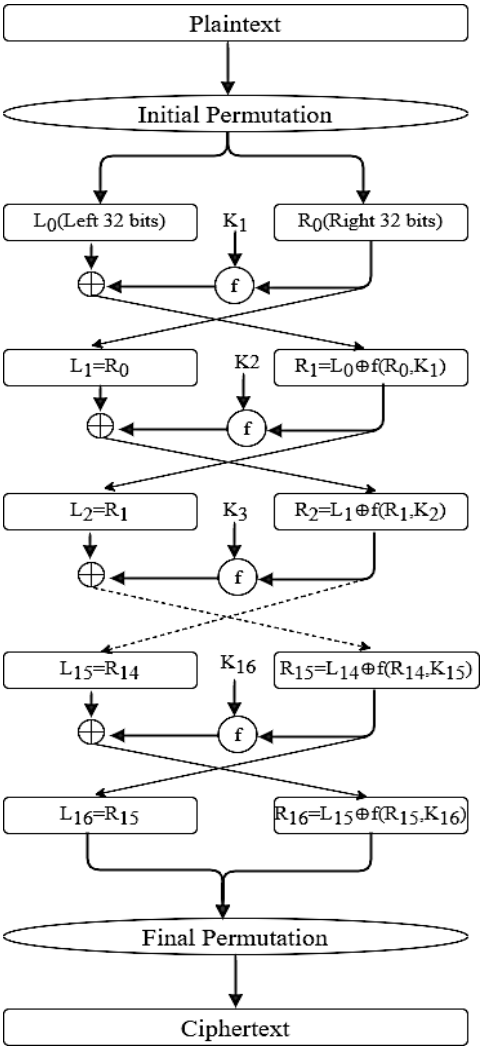


Figure7. The process of DES Algorithm

3.4 Advanced Encryption Standard (AES)

Advanced Encryption Standard (AES) [16], [20] is a block cipher algorithm that came as a replacement for DES and Triple DES. It encrypts and decrypts a 128-bit block of data. Based on the choice of key size, 128 bits, 196 bits, or 256 bits, AES can take 10, 12, or 14 rounds for encryption. Each round consists of four operations: substitute bytes, shift keys, mix column and add round key. However, mix column operation is not performed in the last round. Separate round keys generated from the given cipher key are used in each round of encryption. Data to be encrypted is divided into blocks. Each block is represented as an array of data which is known as a state array. AES is not vulnerable like DES and is also known to provide a good level of security [18]. The encryption process of AES is shown in figure 8 [16], [20].

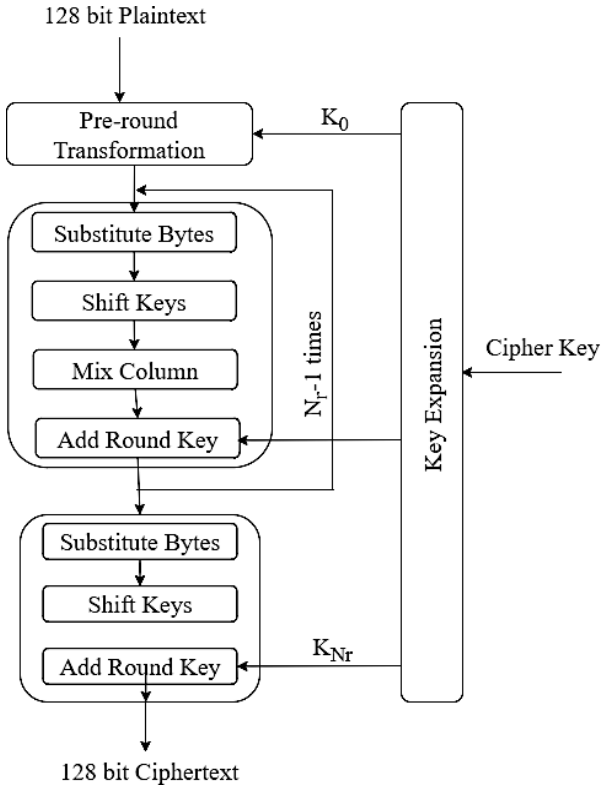


Figure 8. Encryption Process of AES

3.5 BlowFish

BlowFish is a block cipher-based encryption algorithm whose key length varies from 32-bits to 448-bits. Each block handles 64-bits of data [16], [21]. BlowFish encrypts the data through 16 rounds of operations. At each round, data undergoes a key-dependent permutation in P-block and substitution in S-block. Each S-block carries 32-bits of data. Figure 9 shows the BlowFish function F, which splits 32-bit data into four quarters; each carrying 8-bits [16], [21]. These quarters would be the inputs for the S-block. In S-blocks, XOR and Modulo 2^{32} operations are performed to get the final encrypted data. The reverse process is done to decrypt the data.

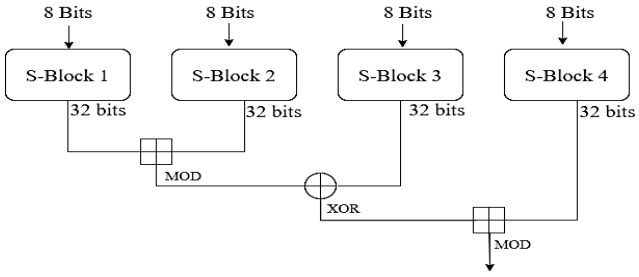


Figure 9. BlowFish function F

3.6 Twofish

Twofish [16], [20], [21] is also a block cipher based symmetric encryption system that works in a similar manner to BlowFish. Unlike BlowFish however, Twofish is considered to be flexible. Twofish allows users to customize encryption speed, key setup time, code size and works fast in an 8-bit CPU as well as in smart cards, embedded chips, etc. It is freely available to use as it is un-patented, license-free software. Twofish encrypts the documents of 128-bit block size with key sizes of 128, 198, or 256 bits in 16 rounds of encryption. The building blocks of Twofish are shown in figure 10 [21].

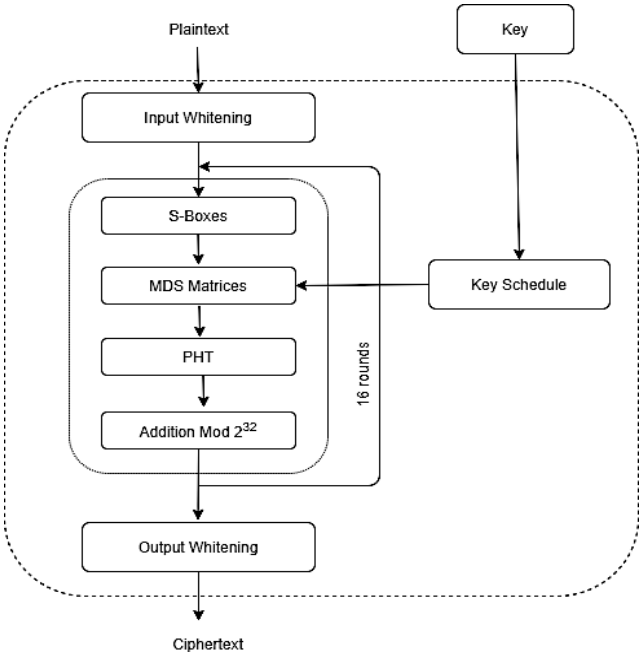


Figure 10. The building blocks of Twofish

The actual processing of each round of Twofish starts and ends with pre-whitening and post-whitening (meaning text blocks are XORed with additional subkeys), respectively. Two 32-bit words are given as input to function F, which is split into four bytes and sent to four different key-dependent S- blocks. The outputs of these four S-blocks are combined with the help of a Maximum Distance Separable (MDS) matrix to form a 32-bit word. Then these two 32-bit words are combined by using a Pseudo Hadamard Transform (PHT), two round subkeys are added, and then the right half of the text is XORed with it. Before and after the XOR operation, a 1-bit rotation is performed. After repeating these rounds 16 times, the last swap is reversed, and an XOR operation is performed between four keywords with another four keywords to get the final encrypted text.

3.7 Threefish

Threefish [16], [17], [21] is a tweak-able block cipher based encryption standard that takes three inputs: a key, a tweak, and plain text, to be encrypted. Threefish uses the same length key as the data block size for encrypting a block of data. This encryption method is used for data blocks of size 256, 512, and 1024 bits. Threefish scheme produces encrypted data by repeating the same sequence of operations 72 times (or rounds) except for 1024-bit block of data, which takes 80 rounds. A 128-bit tweak value is used for all of these data block sizes. Operations of Threefish encryption standards are of three types: addition, XOR, and rotations. Threefish is also free to users since it is an unpatented and license-free encryption standard. Figure 11 shows in detail how each round of Threefish-256 works [22].

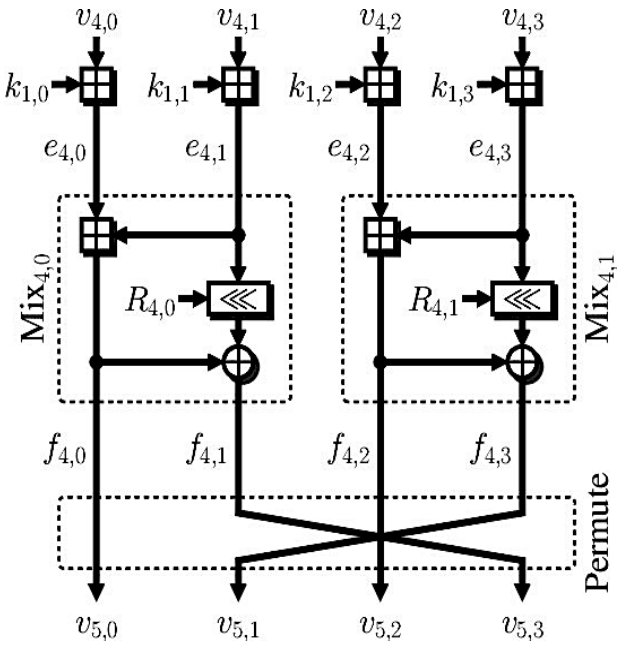


Figure 11. One round of Threefish-256

3.8 International Data Encryption Algorithm (IDEA)

IDEA [23] is a block cipher encryption algorithm that processes 64-bit data blocks with the help of a 128-bit key. This 64-bit data block is divided into four equal sub-blocks, each of size 16 bits. Each of these sub-blocks undergoes eight rounds of repeated sequences of operations and one output transformation phase. For each round of operation, this system needs six unique keys, which are all generated from the 128- bit original key. The output of each round is given as the input to the next round except in the eighth round. The output of the eighth round is given to the output transformation phase, which performs only arithmetic operations, and it needs four keys. The output transformation phase produces the final cipher key. The entire process of encryption needs 52 keys. The process of IDEA is depicted in figure 12 [23].

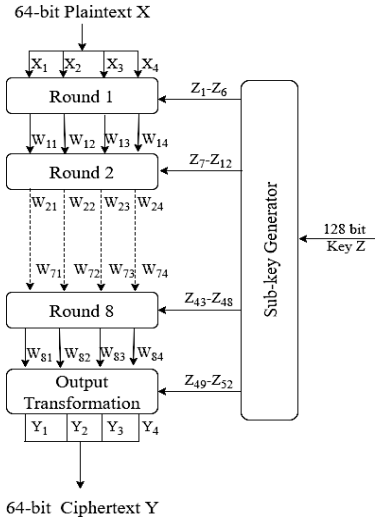


Figure 12. The Process of IDEA Encryption Algorithm

3.9 Rivest Cipher 2(RC2)

RC2 [16], [24] is a symmetric block cipher, also known as ARC2. It handles data blocks of 8 bytes (64-bits), and each data block is divided into four words each of size 2 bytes (16- bits), represented as R[0], R[1], R[2], R[3]. The entire process of encryption and decryption is done on this array as input and output are also stored in the same array. RC2 uses a key of variable length - from a byte to 128-bytes. After accepting a key value to RC2, it expands this key value to get new 128 key bytes to use in both encryption and decryption. RC2 also accepts another value as input, called key bit limit, to identify maximum adequate key size, represented in bits. RC2 has a heterogeneous round structure with two mashing rounds and 16 mixing rounds. RC2 is mainly based on four operations: AND, NOT, XOR, and modular addition.

Every 64-bit data block is encrypted using 64 words of the expanded key. Each mixing or mashing operation consists of 4 mixing or four mashing operations, respectively.

3.10 Rivest Cipher 4(RC4)

RC4 [16], [24] is a symmetric stream cipher algorithm in which each character is encrypted one at a time, commonly used in wireless routers. The key length of RC4 varies from 40 to 2048-bits. To get a more robust encrypted text, 16-byte keys are preferred. Data blocks are XORed with keystream bytes one by one to encrypt the data. The working of RC4 is mainly relayed on the creation of keystream bytes, which is entirely independent of plain text.

The overall working of encryption using RC4 is depicted in figure 13 [16]. An S-block of size 8*8 (whose entries are permutations of numbers from 0 to 255) and a state table of 256 bytes long (initialized with variable length key from 1 to 256 bytes) are generated as an initial step of RC4. This state table is used for the creation of pseudo-random bytes and pseudo-random stream. The plaintext is XORed with this generated pseudo-random stream to get ciphertext.

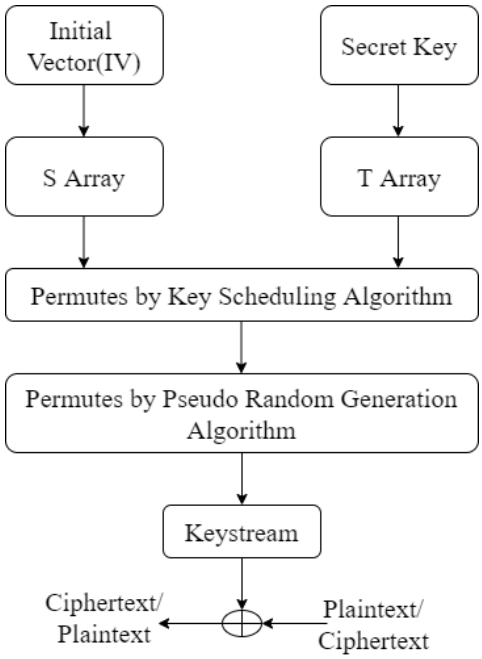


Figure 13. RC4 Encryption Algorithm

The entire working of RC4 is done in 2 phases: keystream generation and actual encryption. The encryption key is generated by using state array and key. It performs several mixing operations, and each consists of swapping and modulo operations.

3.11 Rivest Cipher 5 (RC5)

RC5 [24] is also a symmetric key block cipher. It encrypts the data as block sizes of 32, 64, or 128 bits, but the more suitable size is 64 bits. The key length of RC5 ranges from 0 to 2040 bits, 128 bits is the most suggested one. RC5 can be implemented in both software and hardware since it performs only simple operations that can be performed by a microprocessor. The entire process of RC5 is depicted in figure 14 [24].

It uses two 32 bit registers A and B to store plain as well as ciphertexts; initialized with plaintext and after encryption, it is replaced with the ciphertext. It can take any rounds 1-255 to perform encryption (usually, it takes 12 rounds).

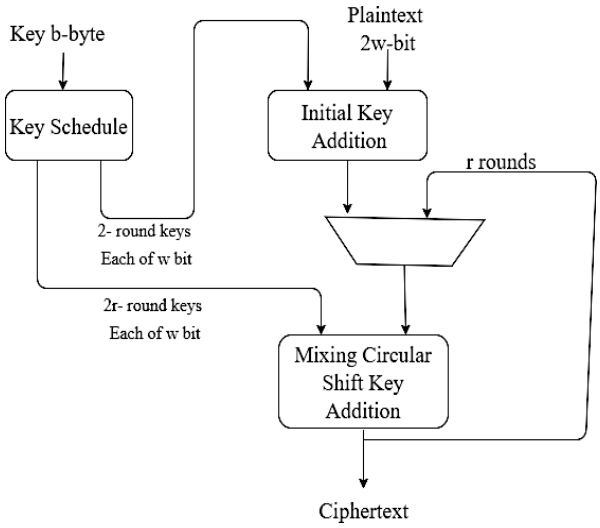


Figure 14. RC5 Encryption Algorithm

3.12 Rivest Cipher 6(RC6)

RC6 [24] is a block cipher similar to RC5, which uses all operations that RC5 uses in addition to multiplication. It performs encryption in 20 rounds of subsequent operations. RC5 and RC6 are parameterized algorithms. RC6 is represented as RC6w/r/b; where w is word size in bits, r is the number of rounds to complete the encryption process, and b is the size of the encryption key in bytes. The basic operations of RC6 are addition, subtraction, XOR, multiplication, left rotation, and right rotation.

In comparison with other algorithms, the variants in the family of RC encryption algorithms were proven to be vulnerable against certain types of attacks.

3.13 Rivest-Shamir-Adleman Algorithm(RSA)

RSA is a widely used asymmetric or public key- based cryptosystem. RSA is considered to be one of the secure encryption algorithms used [18]. It encrypts the data in one particular round. It is a block cipher that uses two different keys for encryption and decryption. The security of RSA depends on the factoring problem, which is the practical difficulty in factoring the product of two prime numbers. Anyone with good knowledge of prime numbers is able to decrypt the data. RSA algorithm creates both private and public keys as follows. Let the two prime numbers be p and q; it calculates n as the product of p and q and $\psi\ (n) = (p - 1)\ (q - 1)$. Then the algorithm chooses e as $1 < e < \psi(n)$, where e and n are co-prime. Once e is selected, the algorithm calculates a value for d as $(d * e)\ \%\psi\ (n) = 1$. The resultant private key is (d, n), and the public key is (e, n). Encryption and decryption are done using equations 1 and 2, respectively.

$$C = M^e (mod\ n) \qquad (1)$$

$$M = C^d (mod\ n) \qquad (2)$$

3.14 Modes of Block Cipher based Encryption

When the same key is used for encrypting multiple blocks of data, intruders can easily break the message. To overcome this issue, we need to avoid creating an identical ciphertext block from the identical plain text by giving an additional input to each block of encryption, which is the mixture of plain text and ciphertext from the previous block. This idea is called block cipher modes of operation [25]–[27]. Multiple encryption modes are used when we are encrypting a large stream of data using block cipher based methods without affecting its security. Each mode has its pros and cons. The encryption modes widely used are: Electronic Code Book (ECB), Cipher Block Chaining (CBC), Propagating or Plaintext Cipher Block Chaining (PCBC), Cipher Feedback (CFB), Output Feedback (OFB) and Counter (CTR).

1) ECB: In ECB [28], each block is encrypted and decrypted separately, as shown in figure 15.

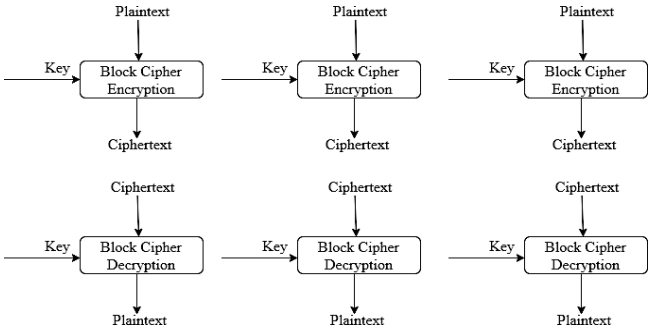


Figure 15. Encryption and Decryption in ECB mode

2) CBC: In CBC mode [25], each plaintext block is XORed with a previously created ciphertext block. Due to this chaining, each ciphertext block depends on its previous block. The first block is XORed with a random initialization vector that has the same length as the plaintext block. Figure 16 explains encryption and decryption using CBC mode. In this mode, we cannot recover the plain text from ciphertext if a single-bit transmission error in plaintext occurs. However, if a single bit error occurred in ciphertext, it will not affect the entire text; it will damage only two plaintext blocks.

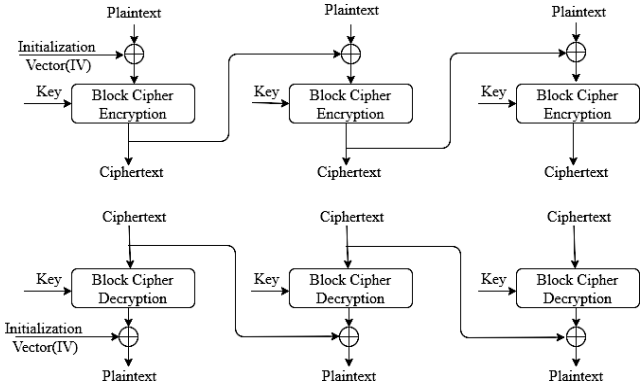


Figure 16. Encryption and Decryption in CBC mode

3) PCBC: PCBC [29] is similar to CBC mode. It mixes plaintext and ciphertext blocks of the previous block with the plaintext of the current block. Here, a single-bit transmission error will affect the entire decryption, and the plaintext cannot be recovered. The way PCBC mode works is shown in figure 17.

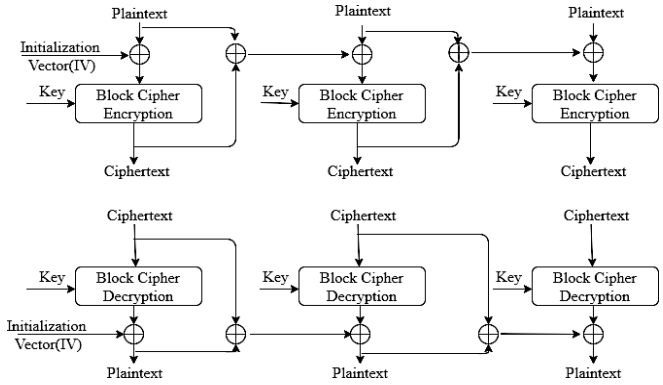


Figure 17. Encryption and Decryption in PCBC mode

4) CFB: In CFB mode [30], the ciphertext data from the previous block is encrypted first and then added to the plaintext of the current block. It uses the same encryption procedure for both encryption and decryption depicted in figure 18. A single-bit transmission error in plaintext block will damage all the subsequent ciphertexts, but single-bit errors in ciphertexts affect only two subsequent blocks.

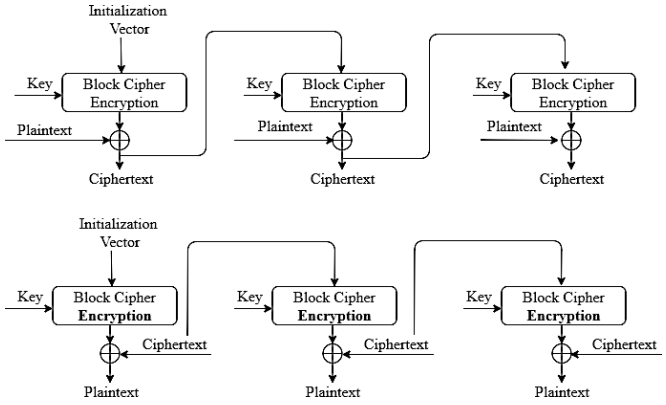


Figure 18. Encryption and Decryption in CFB mode

5) OFB: OFB mode [31] is similar to the way stream cipher works. The ciphers in OFB mode create keystream bytes to encrypt subsequent blocks. A single-bit transmission error in this mode will damage the corresponding plain or ciphertext bit only. Figure 19 shows the encryption and decryption in OFB mode.

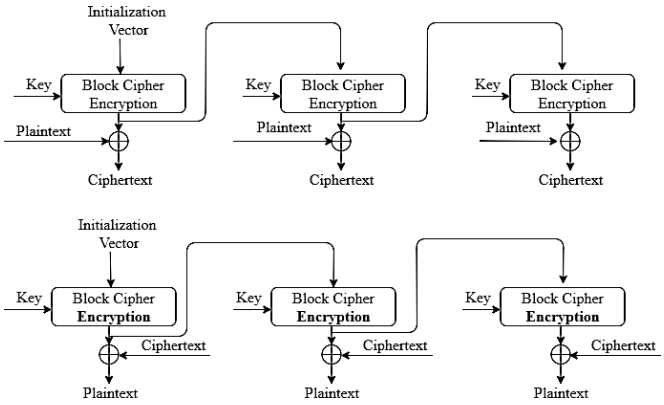


Figure 19. Encryption and Decryption in OFB mode

6) CTR: CTR [32] works in a similar manner to a stream cipher. It uses additional input for encrypting the plaintext; this additional input is created by adding an increasing counter with a nonce value (means number used once). Figure 20 shows encryption and decryption in CTR mode.

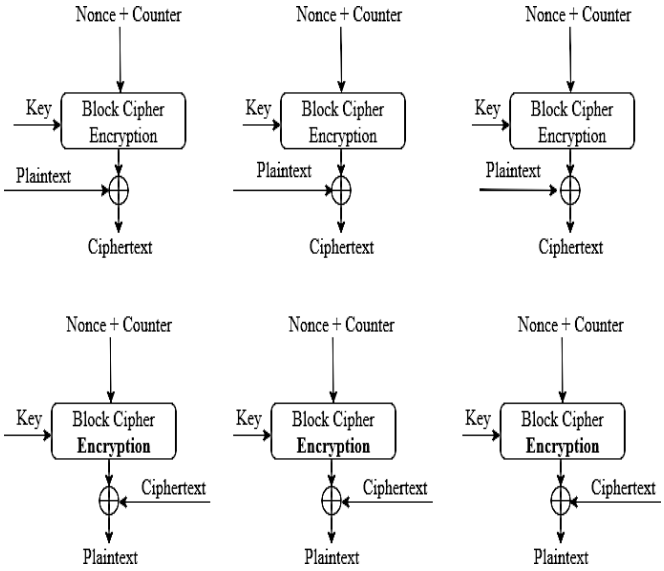


Figure 20. Encryption and Decryption in CTR mode

4. Performance and Analysis

4.1 Simulation and System Setup

In order to evaluate and compare the performance of the presented encryption algorithms, a simulation in Java programming language is created. Java by default offers Java Cryptography Extension (JCE) [33] to support encryption. However, not all the encryption algorithms are available within JCE. To facilitate testing other encryption algorithms not included in JCE, we incorporated the use of Bouncy Castle [34] which offers a wide range of encryption algorithms. The algorithms tested in our simulation are AES, BlowFish, RC2, RC4, RC6, DES, DESede, SEED, XTEA, and IDEA. The block cipher mode selected in our simulation is Cipher Block Chaining (CBC). In this simulation, the speed of encryption (execution time) and the throughput are evaluated with respect to different file sizes. The sizes tested are 1MB, 10MB, 100MB, 500MB and 1GB.

Furthermore, the percentage of CPU usage is calculated for each encryption algorithm. Encryption time is considered to be an essential metric in evaluating any encryption algorithm. Based on [35], the throughput can be calculated using equation 3.

$$Throughput = \frac{c}{t} \qquad (3)$$

where c represents the total encrypted plaintext in bytes and t represents the encryption time. The input parameters are standardized for all encryption algorithms to ensure fairness in the conducted comparison. To reduce the variance in the results, we have adopted using an average of 10 runs for each algorithm's results. Table 2 shows the parameters of the system that were used in the simulation.

4.2 Results and Discussion

The performance measures that were tested in our simulation are: execution time, throughput, and CPU utilization, for the encryption algorithms listed in section 4.1. Table 3 shows the results of execution time and throughput. The results of CPU utilization rate are presented in table 4. Figures 21 to 25 illustrate the encryption time, throughput, and CPU usage for the encryption algorithms under consideration at different input sizes 1GB, 500MB, 100MB, 10MB, and 1MB, respectively.

The simulation results for encryption time in table 3 reflect the speed of the encryption algorithms that were tested. The algorithm that takes the least amount of time to encrypt a plain text file is considered the fastest. The results for execution time illustrated in figures 21(a) to 25(a) show that the RC4, RC6, and AES are the fastest algorithms to produce encrypted data. RC4 and RC6 have previously been known to be vulnerable to attacks, while AES has withheld its level of security, as it has been approved by the US National Institute of Standards and Technology (US-NIST). Therefore, to ensure security of encrypted data, AES seems to be the better choice. DESede was the slowest in terms of execution time; however, it does perform DES encryption thrice. Therefore, if time is not a concern, it can be a suitable candidate to consider for highly confidential data. All encryption algorithms that were tested demonstrated a proportional increase in execution time when linked with the increase in the file sizes being tested.

An increase in throughput indicates less power consumption by an encryption algorithm. The results for throughput are illustrated in figures 21(b) to 25(b). The encryption algorithms, arranged in decreasing order of throughput value are: RC4, RC6, AES, BlowFish, SEED, DES, IDEA, XTEA, RC2, and DESede. It should be emphasized that choosing an appropriate encryption algorithm takes many factors into account. Weighing the factors against each other can help in choosing an adequate algorithm. Since RC4 and RC6 algorithms have been compromised before; AES would be a better fit when taking into account all the factors being considered.

The percentage of CPU utilization of the encryption algorithms for the different file sizes tested are shown in figures 21(c) to 25(c). For the most part, all the values are comparable, with minimal variances detected. However, distinct differences are found in CPU utilization for file size 1GB. The CPU utilization rates of all the tested algorithms, at different file sizes, are listed in table 4. The CPU utilization rate is closely coupled with the system specifications and setup at hand. The same algorithms being tested might produce different results on other systems than the results observed here.

To summarize our findings, RC4, RC6, and AES algorithms, have produced better results compared with their counterparts. However, these results should not be taken at face value. There are many factors that come into play when choosing an appropriate encryption algorithm to implement. One such factor is the level of security needed for the data. Highly classified or confidential data require a higher level of security to be implemented and therefore a more complex encryption algorithm might be favorable despite the execution time it takes. Another factor to consider is whether the encryption will take place at the file level or the application level. The performance of an encryption algorithm at the application level will produce larger overhead in terms of performance. Therefore, choosing an encryption algorithm based on the system setup and specifications at hand become vital. Applications that are executed in real-time might favor encryption speed over complexity. Furthermore, some encryption algorithms have been considered vulnerable against different types of attacks, therefore, striking a balance between the factors being considered when selecting an encryption algorithm for implementation, becomes essential.

5. Conclusion

Encryption algorithms play a pivotal role in providing security in today's digital exchange of data. There are various ways to compare encryption algorithms and demonstrate both their strengths and weaknesses. In order to choose an appropriate encryption algorithm, users can consider different factors such as speed, throughput, complexity, CPU utilization, security level, etc. In this paper, we provided an overview of several encryption algorithms detailing the inner workings of each one. Furthermore, we compared and analyzed the results produced by ten encryption algorithms: AES, BlowFish, DES, DESede, SEED, IDEA, RC2, RC4, RC6, SEED, and XTEA in terms of encryption time, throughput, and CPU utilization. Simulation of these algorithms was performed at different plaintext file sizes such as 1GB, 500MB, 100MB, 10MB, and 1MB. From our results, we observed that RC4, RC6 and AES have produced the best results in terms of encryption time and throughput. We have determined that AES is the better candidate for its performance as well as the level of security it provides. Our results are reflected only for the chosen parameters in our experimental setup. It should be noted, when selecting an appropriate encryption algorithm, factors other than performance measures must be considered. A good encryption algorithm provides a balance between the reported performance measures, the required level of security, and the nature of the data or application being encrypted.

References

[1] O. G. Abood and S. K. Guirguis, "A survey on cryptography algorithms," *International Journal of Scientific and Research Publications*, vol. 8, no. 7, pp.

410–415, 2018.

[2] C. Riman and P. E. Abi-Char, "Comparative analysis of block cipher-based encryption algorithms: A survey," *Information Security and Computer Fraud*, vol. 3, no. 1, pp. 1–7, 2015.

[3] P. Dixit, A. K. Gupta, M. C. Trivedi, and V. K. Yadav, "Traditional and hybrid encryption techniques: a survey," in *Networking Communication and Data Knowledge Engineering*. Springer, 2018, pp. 239–248.

[4] R. Bhanot and R. Hans, "A review and comparative analysis of various encryption algorithms," *International Journal of Security and Its Applications*, vol. 9, no. 4, pp. 289–306, 2015.

[5] M. N. A. Wahid, A. Ali, B. Esparham, and M. Marwan, "A comparison of cryptographic algorithms: Des, 3des, aes, rsa and BlowFish for guessing attacks prevention," *J Comp Sci Appl Inform Technol*, vol. 3, no. 2, pp. 1–7, 2018.

[6] G. Kaur and M. Mahajan, "Evaluation and comparison of symmetric key algorithms," *International Journal of Science, Engineering and Technology Research (IJSETR)*, vol. 2, no. 10, pp. 1960–1962, 2013.

[7] A. Y. Hendi, M. O. Dwairi, Z. A. Al-Qadi, and M. S. Soliman, "A novel simple and highly secure method for data encryption-decryption," *International Journal of Communication Networks and Information Security*, vol. 11, no. 1, pp. 232–238, 2019.

[8] N. Tyagi and A. Ganpati, "Comparative analysis of symmetric key encryption algorithms," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 4, no. 8, pp. 63–70, 2014.

[9] P. Princy, "A comparison of symmetric key algorithms des, aes, BlowFish, rc4, rc6: A survey," *International Journal of Computer Science & Engineering Technology (IJCSET)*, vol. 6, no. 5, pp. 328–331, 2015.

[10] M. Mathur and A. Kesarwani, "Comparison between des, 3des, rc2, rc6, BlowFish and aes," in *Proceedings of National Conference on New Horizons in IT-NCNHIT*, vol. 3, 2013, pp. 143–148.

[11] P. Nema and M.A.Rizvi, "Critical analysis of various symmetric key cryptographic algorithms," *International Journal on Recent and Innovation Trends in Computing and Communication*, vol. 3, no. 6, pp. 4301–4306, June 2015.

[12] M. Marwaha, R. K. Bedi, A. Singh, and T. Singh, "Comparative analysis of cryptographic algorithms," *International Journal of Advanced Engineering Technology*, pp. 16–18, 09 2013.

[13] A. Nadeem and M. Y. Javed, "A performance comparison of data encryption algorithms," *International Conference on Information and Communication Technologies*, pp. 84–89, 2005.

[14] P. J. Sun, "Privacy protection and data security in cloud computing: a survey, challenges, and solutions," *IEEE Access*, vol. 7, pp. 147 420–147 452, 2019.

[15] F. Zhang, Z.-y. Liang, B.-l. Yang, X.-j. Zhao, S.-z. Guo, and K. Ren, "Survey of design and security evaluation of authenticated encryption algorithms in the caesar competition," *Frontiers of Information Technology & Electronic Engineering*, vol. 19, no. 12, pp. 1475–1499, 2018.

[16] W. Stallings, "The principles and practice of cryptography and network security 7th edition, isbn-10: 0134444280," *Pearson Education*, vol. 20, no. 1, p. 7, 2017.

[17] W. Tuchman, *A Brief History of the Data Encryption Standard*. USA: ACM Press/Addison-Wesley Publishing Co., 1997, Ch. 16, pp. 275–280.

[18] N. Advani, C. Rathod, and A. M. Gonsai, "Comparative study of various cryptographic algorithms used for text, image, and video," in *Emerging Trends in Expert Applications and Security*. Springer, 2019, pp. 393–399.

[19] R. P. Adhie, Y. Hutama, A. S. Ahmar, M. Setiawan *et al.*, "Implementation cryptography data encryption standard (des) and triple data encryption standard (3des) method in communication system based near field communication (nfc)," in *Journal of Physics: Conference Series*, vol. 954, no. 1. IOP Publishing, 2018, p. 012009.

[20] J. Katz and Y. Lindell, *Introduction to Modern Cryptography*, ser. Chapman & Hall/CRC Cryptography and Network Security Series. CRC Press, 2014. [Online]. Available: Link.

[21] B. Schneier, "The BlowFish encryption algorithm," Link, 2008, [Online; Accessed 1 July 2020]. [Online]. Available: Link.

[22] N. At, J.-L. Beuchat, and I. San, "Compact implementation of threefish and skein on fpga," in *2012 5th International Conference on New Technologies, Mobility and Security (NTMS)*. IEEE, 2012, pp. 1–5.

[23] S. Basu, "International data encryption algorithm (idea)– a typical illustration," *Journal of global research in Computer Science*, vol. 2, no. 7, pp. 116–118, 2011.

[24] S. Charbathia and S. Sharma, "A comparative study of rivest cipher algorithms," *International Journal of Information & Computation Technology. ISSN*, vol. 4, pp. 0974–2239, 2014.

[25] M. Dworkin, "Recommendation for block cipher modes of operation. methods and techniques," National Inst of Standards and Technology Gaithersburg MD Computer security Div, Tech. Rep., 2010, [Online; accessed 27 June 2020]. [Online]. Available: Link.

[26] P. Rogaway, "Evaluation of some blockcipher modes of operation," *Cryptography Research and Evaluation Committees (CRYPTREC) for the Government of Japan*, 2011, [Online; accessed 27 June 2020]. [Online]. Available: Link.

[27] A. J. Malozemoff, J. Katz, and M. D. Green, "Automated analysis and synthesis of block-cipher modes of operation," in *2014 IEEE 27th Computer Security Foundations Symposium*. IEEE, 2014, pp. 140–152.

[28] W. Stallings, "Nist block cipher modes of operation for confidentiality," Cryptologia, vol. 34, no. 2, pp. 163–175, 2010.

[29] C. J. Mitchell, "Cryptanalysis of two variants of pcbc mode when used for message integrity," in Australasian Conference on Information Security and

Privacy. Springer, 2005, pp. 560–571.
[30] T. Syben, “Introduction to block cipher,” Link, April 2011, [Online; accessed 30 June 2020].[Online]. Available: Link.
[31] S. Almuhammadi and I. Al-Hejri, “A comparative analysis of aes common modes of operation,” in 2017 IEEE 30th Canadian conference on electrical and computer engineering (CCECE). IEEE, 2017, pp. 1–4.
[32] D. Bujari and E. Aribas, “Comparative analysis of block cipher modes of operation,” in International Advanced Researches & Engineering Congress, 2017, pp. 1–4.
[33] Java, “Cryptography extension (jce),” Link, [Online; accessed 30 June 2020]. [Online]. Available: Link
[34] BouncyCastle, “The legion of the bouncy castle,” Link, 2014, [Online; accessed 30 June 2020]. [Online]. Available: Link.
[35] A. A. Tamimi, “Performance analysis of data encryption algorithms,” Retrieved October, vol. 1, 2008. Available: Link.

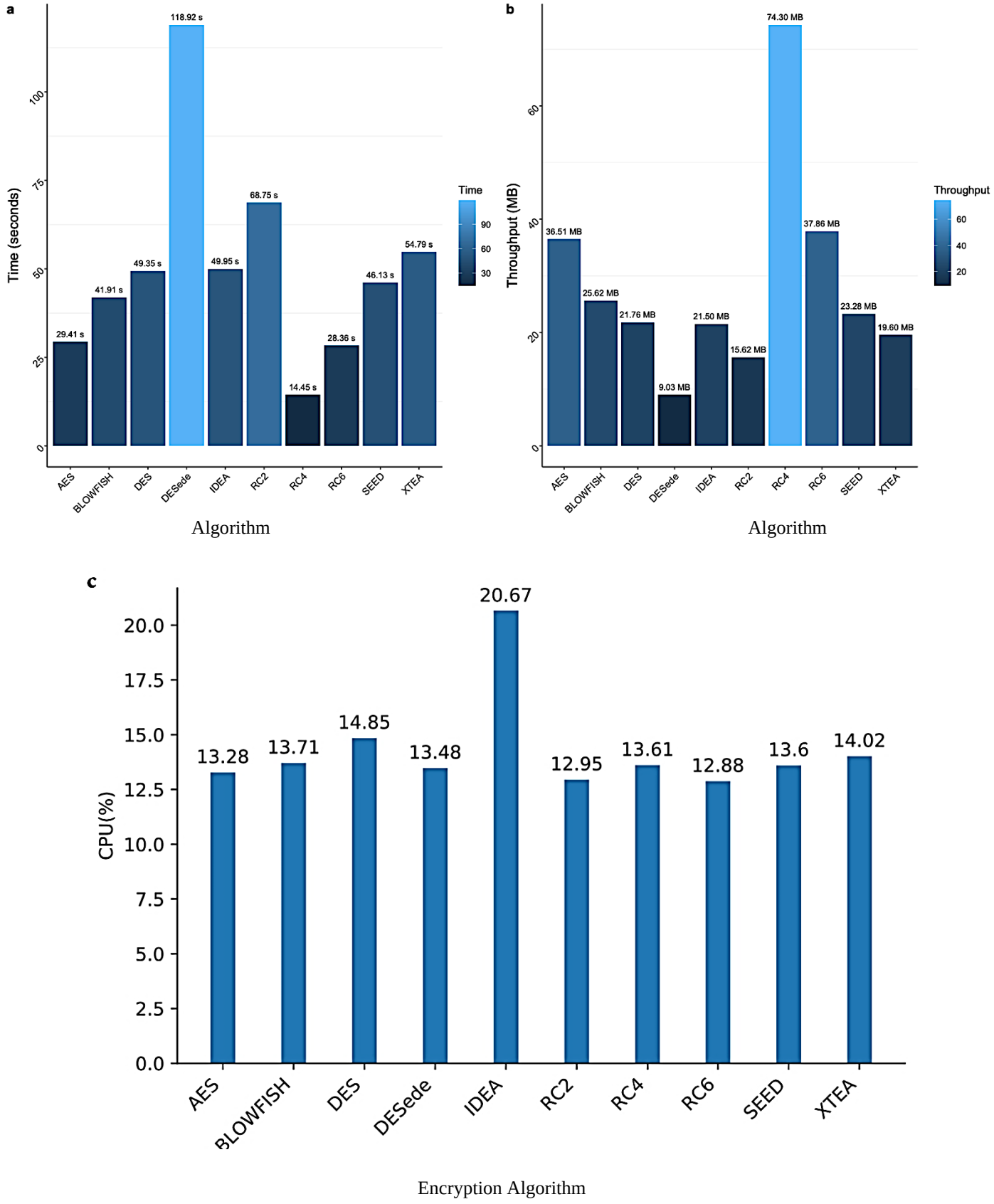


Figure 21. (a) Time, (b) Throughput and (c) CPU usage for different encryption algorithm with a file size: 1GB

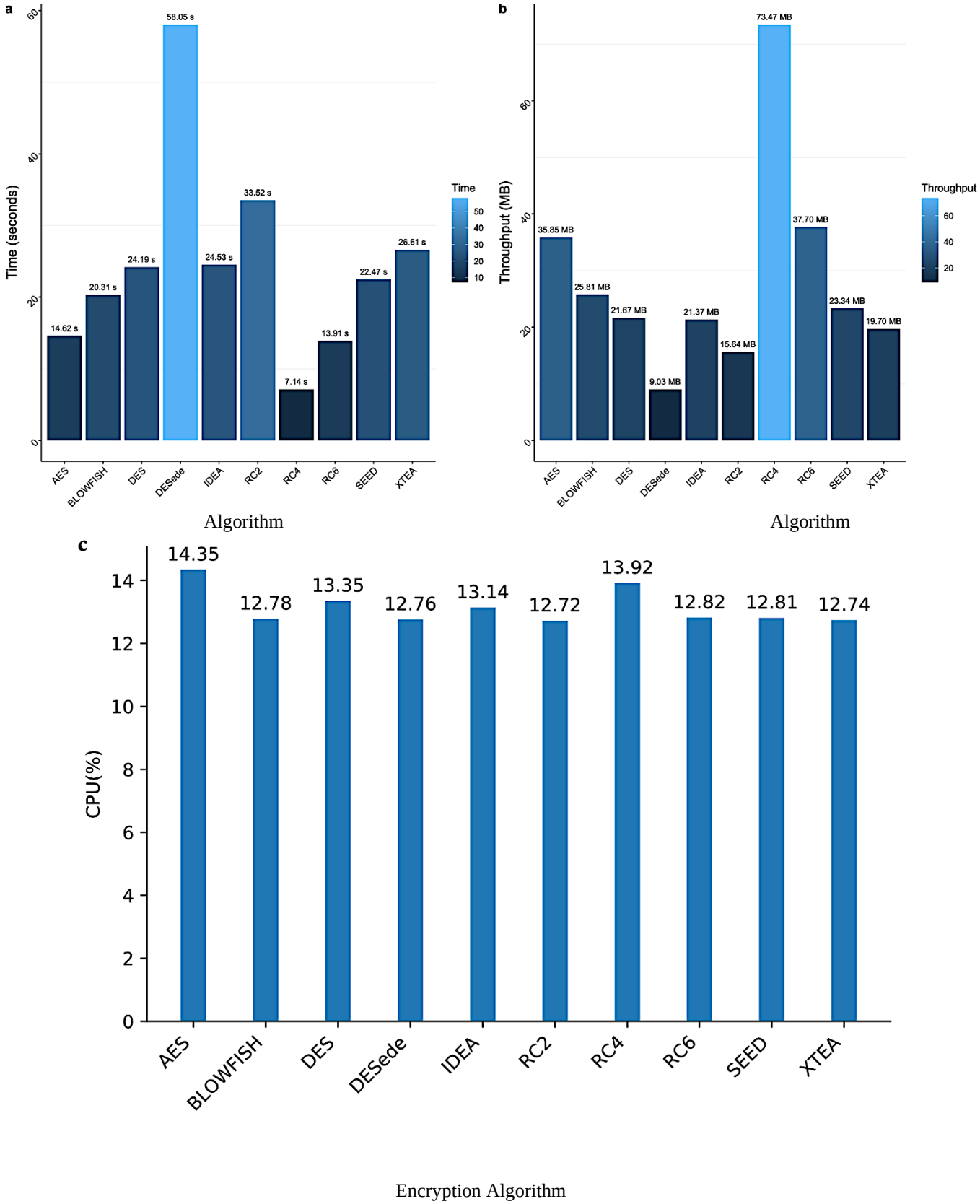


Figure 22. (a) Time, (b) Throughput and (c) CPU usage for different encryption algorithm with a file size: 500MB

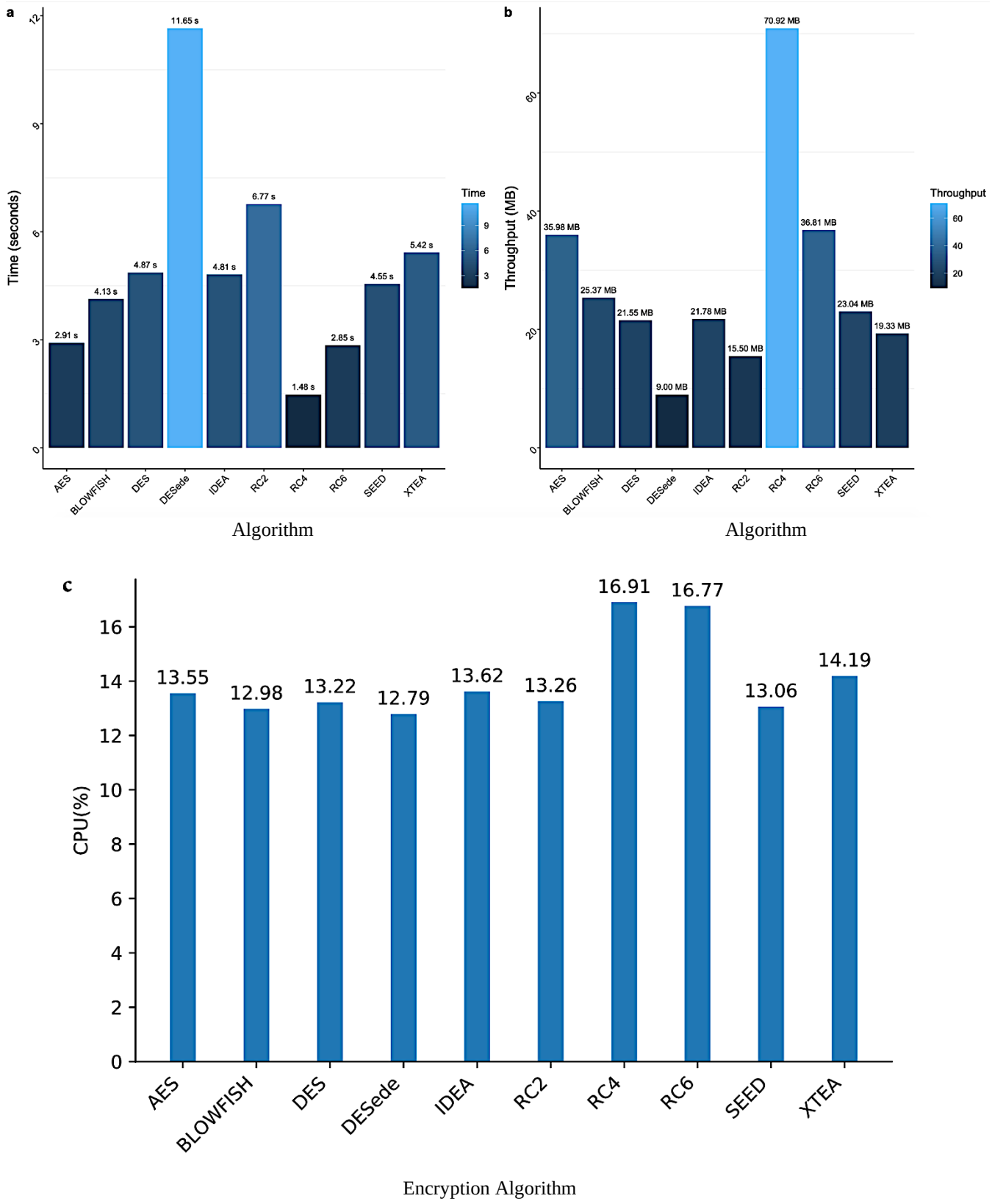


Figure 23. (a) Time, (b) Throughput and (c) CPU usage for different encryption algorithm with a file size:100MB

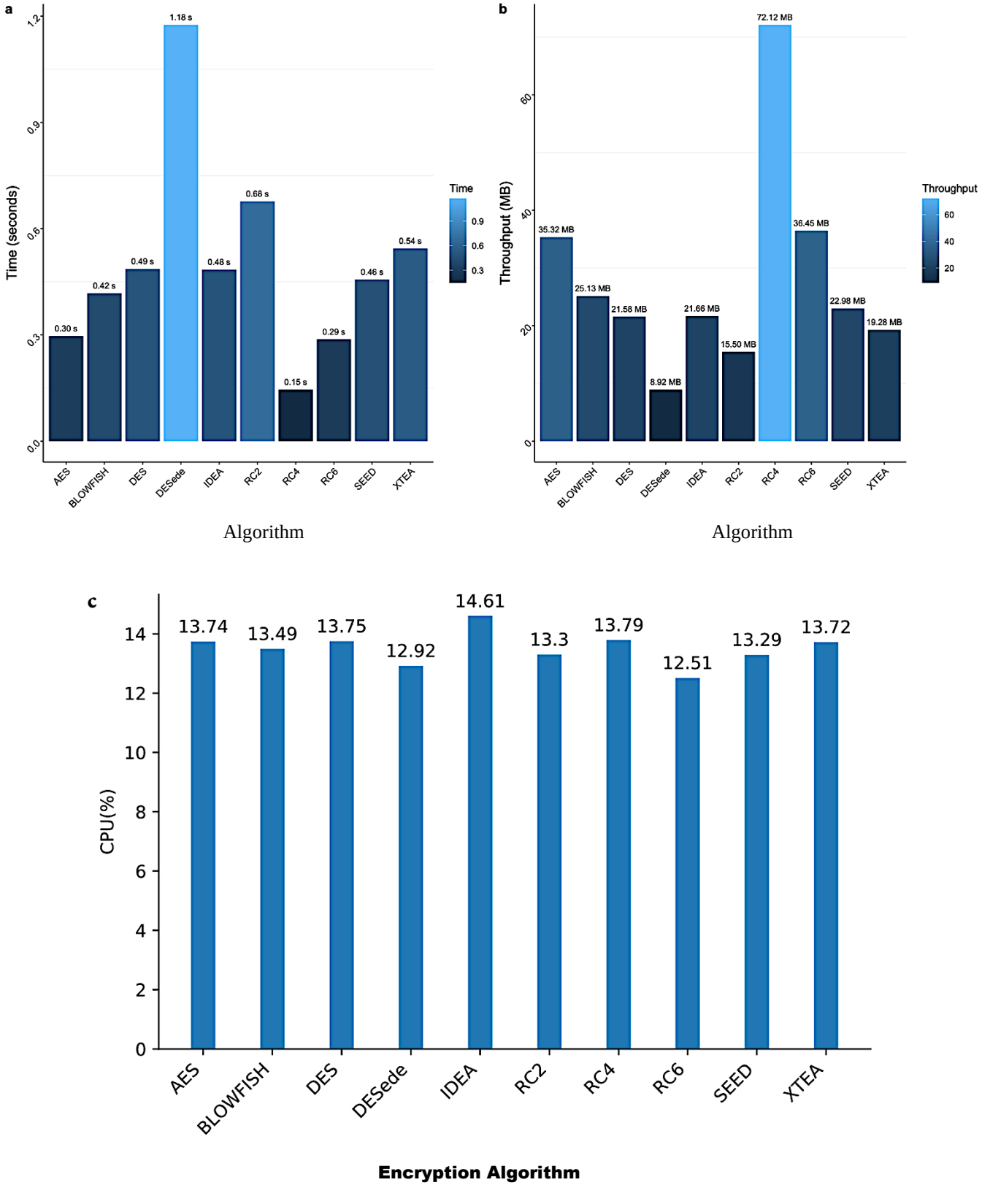


Figure 24. (a) Time, (b) Throughput and (c) CPU usage for different encryption algorithm with a file size: 10MB

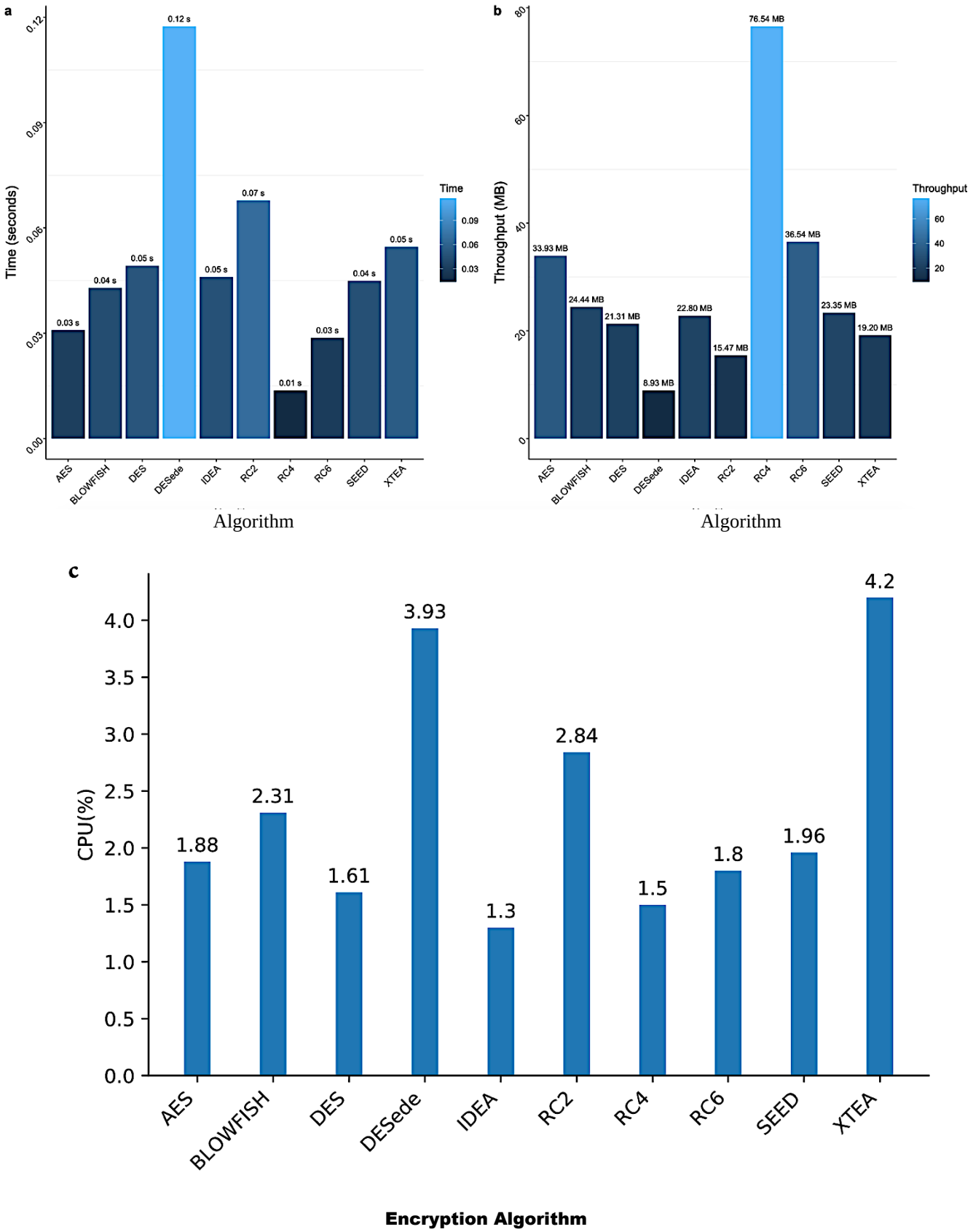


Figure 25. (a) Time, (b) Throughput and (c) CPU usage for different encryption algorithm with a file size:1M

Table 1: Triple DES modes of operation

Mode	Definition
EEE3	a block of data is encrypted three times using three different keys.
EDE3	The block of data is encrypted using one key, then decrypt with another key, and finally encrypted with a third key.
EDE2	It is like EDE3, with only two keys being used.
EEE2	It is like EEE3, with only two keys, first and last keys are the same.

Table 2: System and Experiment Setup

Component	Parameter
Programming language	Java
Application platform	Java SE JDK 14.01 & Bouncy Castle 1.65
Operating system	Windows 10, 64-bit
Computer specs	CPU Intel® Xeon® CPU X5570 @ 2.93 GHz and 32 GB for RAM
Encryption algorithms	AES, BlowFish, RC2, RC4, RC6, DES, DESede, SEED, XTEA, and IDEA
File type	plaintext
File sizes	1MB, 10MB, 100MB, 500MB, and 1GB

Table 3: Encryption Time and Throughput for Various Algorithms

Algorithms	Encryption Time (in seconds)					Throughput (in MB)				
	1GB	500MB	100MB	10MB	1MB	1GB	500MB	100MB	10MB	1MB
AES	29.41	14.62	2.91	0.3	0.03	36.51	35.85	35.98	35.32	33.93
BlowFish	41.91	20.31	4.13	0.42	0.04	25.62	25.81	25.37	25.13	24.44
DES	49.35	24.19	4.87	0.49	0.05	21.76	21.67	21.55	21.58	23.31
DESede	118.92	58.05	11.65	1.18	0.12	9.03	9.03	9	8.92	8.93
IDEA	49.95	24.53	4.81	0.48	0.05	21.5	21.37	21.78	21.66	22.8
RC2	68.75	33.52	6.77	0.68	0.07	15.62	15.64	15.5	15.5	15.47
RC4	14.45	7.14	1.48	0.15	0.01	74.3	73.47	70.92	72.12	76.54
RC6	28.36	13.91	2.85	0.29	0.03	37.86	37.7	36.81	36.45	36.54
SEED	46.13	22.47	4.55	0.46	0.04	23.28	23.34	23.04	29.98	23.35
XTEA	54.79	26.61	5.42	0.54	0.05	19.6	19.7	19.33	19.28	19.2

Table 4: CPU Utilization Percentage for Various Algorithms

Algorithms	CPU Utilization Percentage				
	1GB	500MB	100MB	10MB	1MB
AES	13.28	14.35	13.55	13.74	1.88
BlowFish	13.71	12.78	12.98	13.49	2.31
DES	14.85	13.35	13.22	13.75	1.61
DESede	13.48	12.76	12.79	12.92	3.93
IDEA	20.67	13.14	13.62	14.61	1.3
RC2	12.95	12.72	13.26	13.3	2.84
RC4	13.61	13.92	16.91	13.79	1.5
RC6	12.88	12.82	16.77	12.51	1.8
SEED	13.6	12.81	13.06	13.29	1.96
XTEA	14.02	12.74	14.19	13.72	4.2