Technium Vol. 11, pp.72-86 (2023)
ISSN: 2668-778X
www.techniumscience.com

Performance Analysis of a Hybrid Security Algorithm for Secure Cloud Environment

John Mulopwe Lukusa[*,1] and Fatma Tansu Hocanın[2]
[1]Information Technologies Department, Cyprus International University, North Cyprus, johnlukusam@gmail.com
[2]Computer Engineering Department, Cyprus International University, North Cyprus

Abstract. Encryption algorithms are critical in ensuring the security of information in today's rapidly evolving internet and network applications. These algorithms are especially important in ensuring information security for internet-based dynamic applications and storage methods like cloud computing, which require high reliability, scalability, and autonomy to enable ubiquitous access. Given the prevalence of cloud computing in our everyday routines, data security has emerged as a major concern, particularly for non-technical users. To address this concern, the effectiveness of two widely used symmetric key encryption algorithms, AES and Blowfish, in terms of throughput, power consumption, and encryption speed, has been analyzed and compared. Based on the simulation results, it is clear that AES outperforms Blowfish when security flaws are put into consideration. It should be noted, however, that AES requires more processing power. As a result, a proposed efficient hybrid security algorithm seeks to combine the benefits of both approaches in an encryption system, thereby improving data security in a cloud environment while optimizing resource utilization. According to the performance analysis, the proposed hybrid approach is an excellent solution for encrypting and decrypting files with large block sizes and longer keys, as well as defending against Man in the Middle (MitM) attacks.

Keywords. AES, Algorithm, Analysis, Blowfish, Encryption, Performance.

1. Introduction

Cloud storage has become a fundamental platform for storing vast amounts of data in the digital age, where people's lives are heavily influenced by technology. However, data security are the major challenge in cloud computing, particularly for corporations concerned about the security of sensitive data. While a strong encryption algorithm such as AES is commonly regarded as adequate, it still has theoretical flaws. Various chain ciphers have been proposed as solutions to mitigate the risk of data compromise to address this issue. As cloud computing becomes more prevalent in our daily lives, ensuring secure data storage has become an urgent concern, particularly for non-technical users. Utilizing the full potential of cloud computing components such as virtualization, multi-tenant cloud storage, and cloud networks can provide an efficient and secure method of data protection [1].

Cloud service providers provide their services in a variety of models, including SaaS, PaaS, and IaaS, with a variety of deployment options. Regardless of the model chosen, both providers and clients are concerned about security. Cloud service security encompasses several areas, including computing securities, data storage security, virtualization securities, and network security. To address these concerns, the National Institute of Standards and Technology (NIST) has established comprehensive guidelines for ensuring cloud service security, providing providers with a road map for protecting their client's data [1,2]. In distributing legislation for cloud security standards, the Cloud Security Alliance (CSA) has also reiterated the importance

of authentications, secrecy, integrity, as well, availability. Cloud computing not only provides a variety of services, but it also can defend against threats and attacks that may harm a system. Throughout the architecture and service tiers, cloud security addresses data integrity, source availability, and confidentiality. Different encryption levels can be used to further classify and secure data based on sensitivity, as not all data require the same level of protection.

The goal of this research is to improve cloud security by incorporating AES(s) and Blowfish(s) encryption techniques. This multi-layered Défense system capitalizes on the strengths of both encryption methods, ensuring a higher level of protection for cloud data. This proactive approach aims to maintain the security of sensitive information even in the face of potential breaches or cyber-attacks by implementing a combination of encryption techniques.

2. Literature Review

The ability of the information technology (IT) sector to contribute to cyberinfrastructure in a truthful, valuable, and cost-effective manner is critical to its success. Cloud computing is a network-based computing paradigm that allows programs and applications to run on linked servers, providing scalable on-demand computing resources and services [3]. It provides numerous benefits, including cost savings, security, privacy, dependability, increased processing power, storage, flexibility, scalability, and lower IT infrastructure overhead costs [4,5]. Various implementation models, such as public, private, communal, and hybrid cloud, provide organizations with a variety of options for leveraging cloud services [6]. Cloud computing's main characteristics include on-demand self-services, broad network access, resource pooling, rapid flexibility, and pay-per-use systems [3,6].

However, there are some drawbacks to cloud computing, such as the need for fast connectivity and reliance on cloud service providers for direct access to resources. These barriers are expected to diminish as cloud computing evolves, potentially resembling a traditional client-server network paradigm in the absence of service providers [7]. Collaboration between IT employees and executives is essential for a unified cloud strategy that aligns with the company's goals and produces positive business results. Cloud computing works best when it is seamlessly integrated into existing infrastructure and operations, allowing businesses to make the most of their existing procedures and equipment.

Mobile cloud computing, which combines mobile and cloud technologies to deliver computational power to mobile devices; cloud quantum computing, which uses quantum computing over the internet; hybrid cloud solutions, which are known for their adaptability and cost-effectiveness; and automation, which streamlines processes and boosts productivity by eliminating time-consuming tasks [8]. Organizations can leverage the benefits of cloud technology to deliver efficient and dependable IT solutions by considering the appropriate cloud computing models [2, 7 – 9, 10].

2.1. Barriers and Solutions in Cloud Computing Adoption

Several issues are impeding cloud computing adoption. Adoption is hampered by compliance with standards and regulations, and a lack of knowledge in cloud computing-based IT management contributes to hesitation in adopting the technology [11]. Users have little control over the resources provided by cloud service providers, so security is a major concern. Data encryption can address data security tampering and privacy concerns, but it may increase processing costs and make data retrieval more difficult [11]. The lightweight-secure-conjunctive-keyword-search (LCKS) methodology enables authorized keyword searches in hybrid cloud environments [12]. Blockchain technology can protect against internal cloud service provider attacks, but careful cloud service provider selection is required [12]. Data

privacy is a major issue in cloud computing, particularly in the context of the Internet of Things (IoT) [13]. The data-information-knowledge-wisdom (DIKW) architecture addresses privacy concerns. Protecting the security and privacy of patient's medical data in the cloud is critical in the healthcare sector, and alternative solutions such as a medical cyber-physical system with elliptic curve encryption can ensure data privacy [14]. To maintain high levels of security for sensitive data, cryptographic algorithms such as DES, Blowfish, RSA, and AES must be considered [13].

2.2. Cryptographic Algorithms: Safeguarding Data with Encryption

Table 1. Comparison of Cryptographic Algorithms in Terms of Memory Usage and Encryption Time.

Researches	Algorithm	Memory Used	Description Time	Encryption Time
Mahmud [24]	DES	18.2 kb	1000ms	1300ms
Abirami [11]	3DES	20.7 kb	800ms	1550ms
Patil [14]	AES	14.7 kb	600ms	600ms
Purwinarko [10]	Blowfish	9.38 kb	450ms	500ms
Ajmal [25]	Blowfish	4 kb	120ms	150ms
Ajmal [25]	AES	14 kb	39ms	24ms
Fatima [26]	AES	200 kb	------	6ms

Cryptography is a technique for ensuring message secrecy and protecting individuals' and organizations' privacy by encrypting data and restricting access to only authorized recipients [15]. Cryptographic algorithms are procedures that use complex mathematics and encryption keys to convert plaintext into encrypted ciphertext, making messages difficult to decipher without the proper keys [16].

The Data Encryption(s) Standard(s) (DES), a symmetric block encryption technique published by NIST, is a well-known cryptographic algorithm. DES employs a 64-bit key, with 56 bits dedicated to independent key bits and 8 bits dedicated to error detection. However, due to vulnerabilities, DES has been compromised and is no longer considered secure [14]. Triple DES (3DES) is a DES variant that applies the DES cipher to each data block three times, increasing security. It has a 64-bit block size and a key length of 112 or 168 bits. 3DES was created to increase the size of DES keys and to withstand brute force attacks [14]. Blowfish is a free symmetric key block cipher that was created as an alternative to traditional encryption methods. It is suitable for both home and business use, with key lengths ranging from 32 to 448 bits. Blowfish is regarded as a reliable encryption method that can, in some cases, replace DES [14].

Rivest, Shamir, and Adelman created RSA, an asymmetric public-key cryptosystem. It entails making a private key for decryption and a public key for encryption. RSA is based on the challenge of factoring large prime numbers and has key sizes ranging from 1024 to 4096 bits [14]. The Advanced Encryption Standard (AES) is a symmetric key block encryption algorithm that was introduced in 1998. It operates on a 44x44 matrix of 128-bit blocks and supports data and key lengths of 128, 192, or 256 bits. AES employs multiple transformation rounds and can be implemented in both hardware and software [14]. Block ciphers use cryptography algorithms such as Cipher Block Chaining (CBC) and Electronic Code Book (ECB). Before encryption, CBC incorporates feedback by XORing each plaintext block with the previous ciphertext block, whereas ECB operates on each block independently. CBC can

recover from bit errors but not from synchronization errors, whereas ECB is vulnerable to certain attacks [14].

The choice of a cryptographic algorithm is determined by specific requirements. Blowfish is appropriate for applications with limited memory space, whereas AES provides strong confidentiality and integrity. Understanding the advantages and disadvantages of each algorithm is critical for selecting the best one for a given application [14]. Authentication and protection against Man-in-the-Middle attacks can be effectively achieved by employing cryptographic algorithms such as AES and Blowfish in hybrid security techniques such as those used in mobile applications [10].

3. Proposed Model for Secure Cloud Computing Environment (PCCE)

Cloud computing systems keep data on remote servers that are accessible via the internet, providing advantages such as less reliance on local storage and lower operational costs. However, Concerns about privacy and securities arise as a result of the possibility of unauthorized entry into service providers. Existing systems frequently encrypt all data with the same key size, regardless of data confidentiality, increasing costs and processing time. To tackle these concerns, this study proposes a secure cloud computing paradigm that is specifically designed for cloud-based dynamic groups. To ensure appropriate levels of confidentiality for different types of data, the proposed model categorizes data based on its security requirements and employs multiple security techniques with variable key sizes. The model prioritizes data security, privacy, and data confidentiality, as well as fine-grained access control, user revocation, and scalability and efficiency [8, 17 – 18].

3.1. The System Functionality

The proposed model for cloud data security employs a classification system with three levels of categorization. To protect data, it employs different encryption techniques at each level. Level 1 data is non-sensitive and available to the public, whereas level 2 data, such as personal and educational documents, is encrypted with AES-256. Level 3 data, which is critical, is encrypted using AES-256 and Blowfish in a two-cipher cascade process. With client-side encryption and decryption, the model ensures data confidentiality and security. The method improves data security and is suitable for cloud storage [8, 10, 19 – 22]. The proposed model is depicted graphically in Fig. 1, which shows the categorization and encryption techniques used at each level. The diagram highlights the client-side encryption and decryption procedures at all three levels (Level 1, Level 2, and Level 3), as well as the associated encryption/decryption methods. The following is a detailed description of these steps:

Technium Vol. 11, pp.72-86 (2023)
ISSN: 2668-778X
www.techniumscience.com

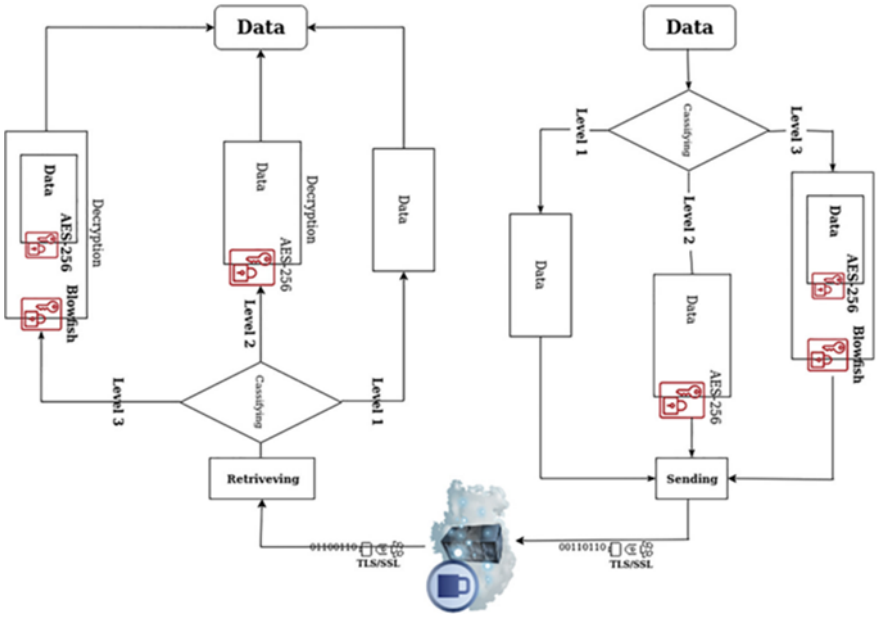

Fig. 1. Schematic representation of the suggested model

Step 1: Client-side encryption

Client-side encryption is divided into three groups in the proposed model based on data importance and encryption method used (Fig. 1). This method enables end users to categorize and encrypt their data before transmitting it, ensuring that their data remains secure and inaccessible to the cloud providers or third parties.

Step 2: Cloud storage

During this phase, data is stored and transferred to the cloud. The proposed model encrypts data before transmission using AES and Blowfish encryption techniques, eliminating the need for additional encryption by the cloud provider. This ensures that data is protected at all times during transmission and storage in the cloud [23].

Step 3: Client-side

Decryption Client-side decryption, like encryption, applies data categorization to the encrypted data and uses the encryption keys in reverse order. It is important to note that the keys used for data encryption are accessible to end users.

To distribute plaintext material and increase the complexity of the relationship between plaintext, encrypted text, and encryption key, the AES encryption method employs substitution-permutation (SP) networks [10]. Blowfish implementation, on the other hand, is key-dependent and employs the Cipher Block Chaining (CBC) mode for added security. CBC processes data blocks individually, preventing the entire message from being decrypted even if some blocks are obtained by an attacker.

The proposed model improves the security of data stored in the cloud by combining these encryption techniques and the CBC mode. It provides a practical solution for protecting critical information during cloud storage from unauthorized access or theft.

The logical operator exclusive or (XOR) returns true only when one operand is true and the other is false. It is used to accurately represent the required conditions in logical condition formation.

Technium Vol. 11, pp.72-86 (2023)
ISSN: 2668-778X
www.techniumscience.com

3.2. Application of Encryption and Decryption Algorithms on the Proposed Model

3.2.1. AES-256 encryption. AES 256 exemplifies symmetric key encryption, a cryptographic technique that uses a single secret key for both data encryptions and decryptions. This encryption processes consist of several steps designed to ensure the secure transformation of information. To begin, a secure key is generated by analyzing a random byte stream using algorithms such as SHA-256, resulting in a 256-bit key. Following that, the length of the input file is increased to match the AES block size, which aids in maintaining compatibility during the encryption process. An Initialization Vector (IV) is generated to enable the Cipher Block Chaining (CBC) mode. The IV is stored in the output file, along with the encrypted blocks, with the IV at the beginning. The file is then read in blocks, with each block encrypted using the AES algorithm. The encrypted block is appended to the output file to ensure the data's integrity and confidentiality. This procedure is repeated until all of the blocks have been encrypted. In the context of the study, the file belongs to a specific class in which the input is analyzed using SHA-256 in generating a usable 256-bit key. Following the byte stream analysis, the input file is padded, IV generated, and CBC mode is used. The IV is included in the output file and can be used for subsequent decryption. Finally, a loop is started to gradually encrypt data blocks extracted from the padded input file, ensuring the confidentiality of the transmitted data.

3.2.2. AES-256 decryption. AES decryption uses the same symmetric key encryption to retrieve the original, unencrypted data systematically. During the decryption phase, the pre-stored encryption key is loaded and the Initialization Vector (IV) from the encrypted file is accessed. The IV is obtained by examining the first block, ensuring the data's security. The decryption loop decrypts each encrypted file block and writes the results to an output file. This procedure is repeated until all blocks have been decrypted. The correct keys are required for the extraction of the original file, ensuring authorized access to the data. The proposed model addresses data security concerns by providing a secure and reliable method for decrypting and accessing the original data. The decryption loop reverses the encryption loop, systematically decrypting encrypted data back into its original form. When the decryption processes are finished, the resulting data are unpadded, bringing the decryption phase to a close.

```
private class X_Param {}

X_Param X_Param_obj = new X_Param(IV.getBytes(StandardCharsets.UTF_8));
Cipher encryption_cipher = Cipher.getInstance("Blowfish/CBC/PKCS5Padding");
SecretKeySpec secret_key = new SecretKeySpec(KeyData, "Blowfish");
encryption_cipher.init(Cipher.ENCRYPT_MODE, secret_key, X_Param_obj);
```

Fig. 2. Blowfish encryption argument formula

3.2.3. Blowfish encryption. The Blowfish encryption algorithm, like AES-256, is a symmetric method that uses secret keys for both encryptions and decryptions. Creating a key, padding the input data, creating an initialization vector (IV), generating S-boxes and a subkeys array, using the Cipher Block Chaining (CBC) mode, and saving the encrypted output are all steps in using Blowfish encryption. The process is reversed for decryption, using the same key, IV, and S-boxes. To enhance cloud data securities, the proposed method combines AES-256 and Blowfish encryption in cascade mode. An IV is created after the input data is padded. In the encryption process, S-boxes and a subkey array are used. The security is increased further by encrypting the data block by block.

Technium Vol. 11, pp.72-86 (2023)
ISSN: 2668-778X
www.techniumscience.com

```
private class X_Param {}

X_Param initializing_vector = new X_Param(IV.getBytes(StandardCharsets.UTF_8));
Cipher encryption_cipher = Cipher.getInstance("Blowfish/CBC/PKCS5Padding");
SecretKey encryption_secret_key = new SecretKeySpec(KeyData, "Blowfish");
encryption_cipher.init(Cipher.ENCRYPT_MODE, encryption_secret_key, initializing_vector);
```

Fig. 3. Blowfish decryption argument formula

3.2.4. Blowfish Decryption. Blowfish decryption employs the same symmetric key encryption as encryption. Loading the IV and key, reading the input files, performing block-by-block decryptions in CBC mode, removing padding from the output, and creating an output file for the unpadded data are all steps in blowfish decryption. The original IV is required during the decryption process to ensure proper alignment with the Blowfish cipher's block size. When used in cascade mode, the Blowfish keys are generated independently of the AES-256 key. The user is responsible for securely maintaining the IV. Loading the IV and key is critical for decrypting the data in the same way it was encrypted, allowing the original file to be recovered. As a result, the permissible arguments for the CBC mode decryption command are determined.

4. Performance Analysis of The Proposed Model

Table 1. Key Length, Block Size, Number of Rounds.

Heading level	Key Length (Nκ words)	Block Size (Nb words)	Number of Rounds (Nγ)
AES 128	4	4	10
AES 192	6	4	12
AES 256	8	4	14

The AES algorithms operate on input and output values of 0 and 1, respectively, with a 128-bit block size, and a key length ranging from 128 to 256 bits. As shown in Fig4, the AES key expansion process involves creating unique columns in the enlarged key based on the binary key size divided by 32.

$$B^7x^7+b^6x^6+b^5x^5+b^4x^4+b^3x^3+b^2x^2+b^{11}+b0=\sum b^ix \qquad (1)$$

Using an S-box, XORing with a Rcon number, and computing the next set of bytes for key expansion is all part of this process. The BLOWFISH algorithm, on the other hand, provides customizable key lengths ranging from 32 to 448 bits and is well-known for its data security effectiveness. It employs a Feistel network with 16 rounds and key-data dependent replacement and permutation operations. On 32-bit microprocessors, the BLOWFISH algorithm is especially fast. Figure 5 shows the conversion of hexadecimal to binary, where subkeys are represented in hexadecimal format.

Technium Vol. 11, pp.72-86 (2023)
ISSN: 2668-778X
www.techniumscience.com

```
public int[] schedule_core(int[] in, int rconpointer) {
        in = leftrotate(in, 1);
        int hex;
        for (int i = 0; i < in.length; i++) {
            hex = in[i];
            in[i] = sbox[hex / 16][hex % 16];
        }
        in[0] ^= rcon[rconpointer];
        return in;
    }
```

Fig. 4. Return column from the key scheduling

```
private String hexToBin(String plainText)
    {
        String binary = "";
        Long num;
        String binary4B;
        int n = plainText.length();
        for (int i = 0; i < n; i++) {

            num = Long.parseUnsignedLong(
                plainText.charAt(i) + "", 16);
            binary4B = Long.toBinaryString(num);

            // each value in hexadecimal is 4 bits in binary.
            binary4B = "0000" + binary4B;

            binary4B = binary4B.substring(binary4B.length() - 4);
            binary += binary4B;
        }
        return binary;
    }
```

Fig. 5. Hexadecimal to binary conversion

4.1. Simulation Procedure

In terms of speed, the throughput of an encryption technique is calculated using encryption time. The encryption scheme's throughput is determined by dividing the entire plaintext.

Table 2. Comparative Encryption and Decryption Algorithm Execution Times (in Milliseconds).

	Encryption		Decryption	
Input Size in (Kbytes)	Blowfish	AES	Blowfish	AES
49	36	56	38	63
59	36	38	26	58
100	37	90	52	60
321	45	164	92	149
899	64	258	102	171
5345.28	122	1237	149	655
7310.33	107	1366	140	882

22335	155	1370	142	885
42000	165	1530	190	998
99000	190	1720	210	1208
Average Time (Sec)	91.90	542.30	98.15	360.38
Throughput (Mb/Sec)	152.25	25.90	142.58	38.83

The execution times of the proposed model's encryption and decryption algorithms were evaluated for various packet sizes, and the results are shown in Table 4.2. The table compares the total encryption time to the encrypted plaintext size in Kbytes. The power consumption of the encryption technology was found to decrease as the throughput value increased. The encryption and decryption methods were carefully implemented to ensure that the results were fair and accurate. The algorithms were evaluated using various data block sizes ranging from 0.5MB to 20MB.

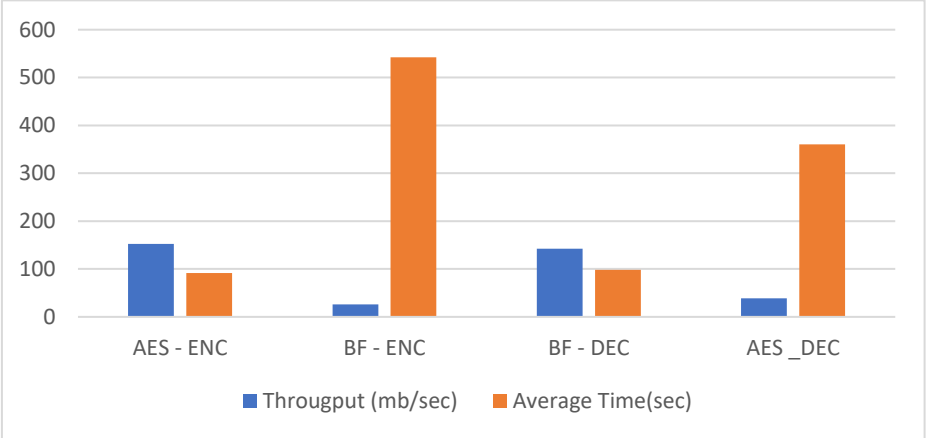

Fig. 6. Comparative Encryption and Decryption Algorithm Execution Times

4.2. Simulation Results

The simulation results for this compassion point at the encryption and decryption stages are displayed in Fig. 7 and Table 3. The result shows that the Blowfish method outperforms other algorithms considering the processing time. It is also worth noting that two modes of operations, electronic code books, and cipher block chaining, were used with a block cipher, and the performance results are indicated in Fig 7.

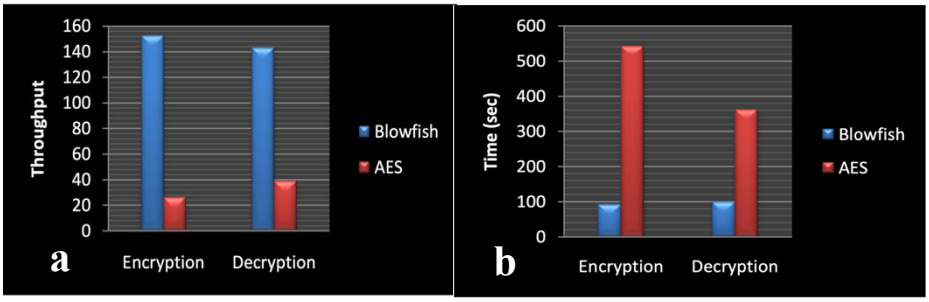

Fig. 7. AES and Blowfish of (a) Algorithm Throughput and (b) time comparison (MB/sec)

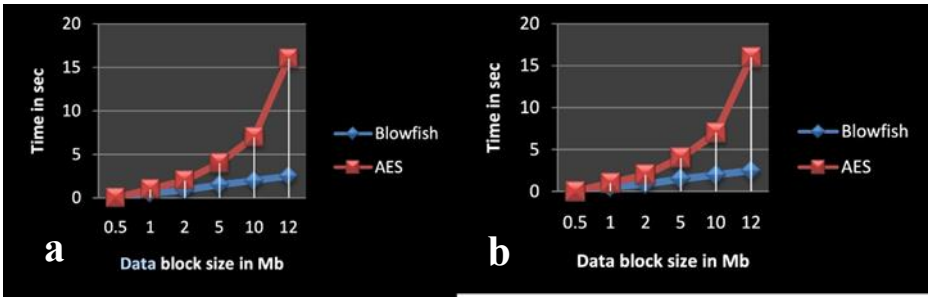

Fig. 8. (a) The ECB and (b) The CBC mode results

Furthermore, the result shows that the Blowfish method outperforms the AES algorithms as regards processing time. It also demonstrates that AES consumes greater resources when data blocks are relatively large. CBC takes longer to process than ECB due to its key-chaining structure. The results in Fig.8 show that the extra time is not significant for many applications, given that CBC is far superior to ECB in terms of protection. Although the difference between the two modes is difficult to detect with the naked eye, the findings revealed that the average difference between ECB and CBC is only 0.059896 seconds, as shown in Fig 8.

Table 3. Results Compare to Other Studies

	Mahmud [24]	Abirami [11]	Patil [14]	Purwinarko [10]	Ajmal [25]	Ajmal [25]	Fatima [26]	Study Results	
Algorithm	DES	3DES	AES	Blowfish.	Blowfish	AES	AES	AES	Blowfish
Memory used	18.2 kb	20.7 kb	14.7 kb	9.38 kb	4 kb	14 kb	200 kb	12MB	12MB
Description Throughput time.	1000ms	800ms	600ms	450ms	2200ms	39ms	------	550ms	98ms
Encryption Throughput time.	1300ms	1550ms	600ms	500ms	150ms	24ms	6ms	350ms	100ms

4.2.1. ***AES and Blowfish KEYS***.

An Intel Core i5 2.5GHz processor, 8GB of RAM, and the Windows 10 64-bit operating system are used on the client side of the experiment. Applications written in Java 2010 run on an Intel Xeon Processor E3-1200 v5 Series, Core i3, 64GB DDR4 Memory, and the CentOS Linux 7 operating system in the database server. During testing, a query will be run to process the database record data, and the results will be presented in the form of a PDF report. Before the medical record data is encrypted, 256 SHA hashing procedures will ensure the PDF report files' integrity.

Technium Vol. 11, pp.72-86 (2023)
ISSN: 2668-778X
www.techniumscience.com

To compare the encryption's performance in terms of speed and throughput, the RSA and ECC key generation procedures will be used. Figure 9 depicts the results of data encryption techniques. According to testing results, blowfish (BF) is faster than AES; this conclusion is consistent with the literature, although AES-BF hybrid techniques were thought to be more efficient than AES methods.

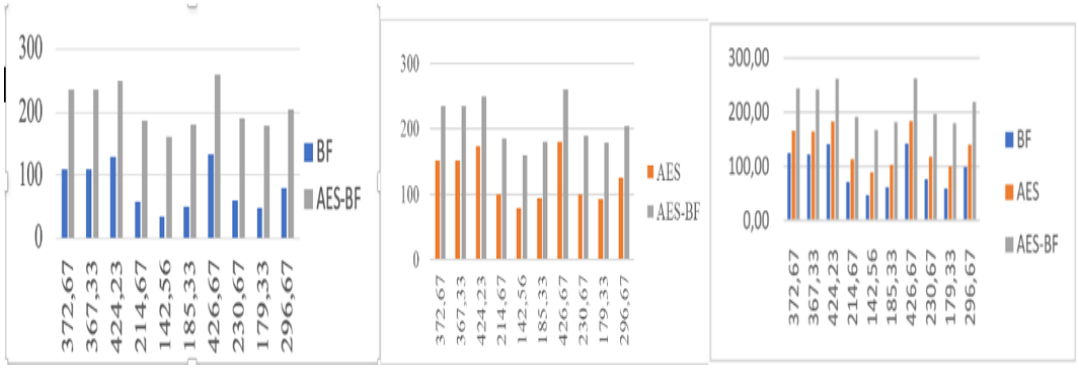

Fig. 9. (a) RSA and (b) ECC encryption key graph (time vs file size)

While the hybrid AES-BF methodology takes longer to decrypt data than it does to encrypt it, the private key BF and public methods take longer to generate the decryption key than AES approach.

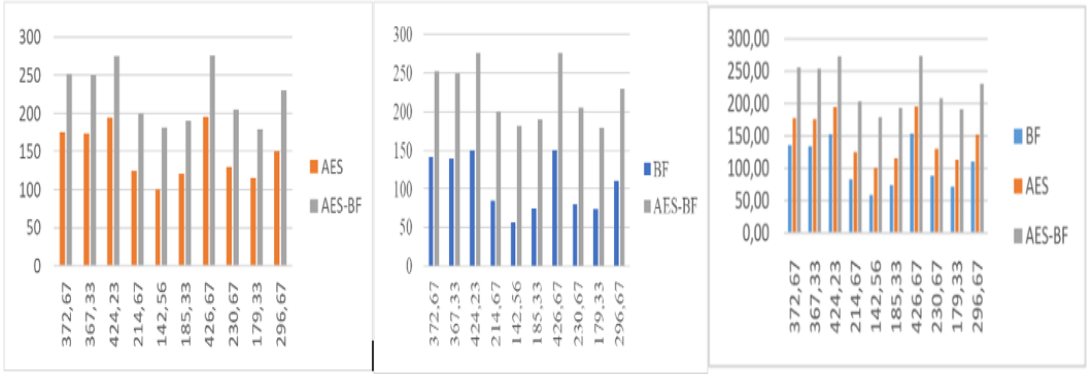

Fig. 10. (a) BF with AES-BF, (b) AES with AES-BF and (C) BF, AES with AES-BF decryption key graph (time vs file size)

The performance test results are displayed in terms of throughput expressed in bits per second. The encryption and decryption processes were tested, and BF was found to be faster than AES at encrypting and decrypting data files. AES-BF was discovered to be more powerful than AES alone. AES was found to be more efficient than both BF and AES-BF, as evidenced by the graphs in Fig 14 that compare the results.

4.2.2. Results Comparison between BF with AES-BF and with AES-BF keys.

In comparison to other studies, the proposed approach of combining AES and Blowfish encryption and decryption techniques for data security in cloud computing produced better results in terms of both time and size. The use of a two-cipher cascade, with AES-256 as the primary and Blowfish as the secondary encryption methods, provides a higher level of security for sensitive data. Client-side encryption, cloud storage, and client-side decryption are required

for data security. During the first stage, the client side encrypts the data with both AES-256 and Blowfish ciphers and two separate keys before sending it to the cloud for storage. The encrypted data is finally decrypted on the client side using the same two keys that were used in the encryption process. Because the critical information is already encrypted using the AES and Blowfish methods, this approach does not entirely rely on the cloud provider for additional encryption. Overall, the study's findings show that the proposed method is a viable solution for improving data security in cloud computing.

Table 4. Comparison Between the Results and Other Studies is shown in Table 4

Studies	Algorithm	Memory used	Description Throughput time	Encryption Throughput time
Study Results	Blowfish Key	426MB	15 x 10^4ms	13 x 10^4ms
	Blowfish with AES Key	426MB	14 x 10^4ms	15 x 10^4ms
	AES Key	426MB	198 x 10^3ms	18 x 10^4ms
	AES with BF Key	426MB	17 x 10^4ms	19 x 10^4ms
	AES + BF with BF Key	426MB	27 x 10^4ms	24 x 10^4ms
	AES + BF with AES Key	426MB	25 x 10^4ms	26 x 10^4ms

The study strategy outperformed others because it provides numerous advantages in key management implementation.

1. Encrypt the data with Blowfish using a randomly generated key.
2. Encrypt the Blowfish key with AES using a user-provided key.
3. Store the encrypted data and encrypted key together.
4. When decrypting the data, first decrypt the Blowfish key with AES.
5. Decrypt the data using the decrypted Blowfish key.

This method accepts a key and some data and returns an encrypted string. If an error occurs during the encryption process, it throws an exception. The first step is to generate a random 16-byte Blowfish key and store it in a SecretKeySpec object.

```
byte[] blowfishKeyBytes = new byte[16];
SecureRandom secureRandom = new SecureRandom();
secureRandom.nextBytes(blowfishKeyBytes);
SecretKeySpec blowfishKeySpec = new SecretKeySpec(blowfishKeyBytes,
BLOWFISH_ALGORITHM);
```

Then creates a Cipher object for Blowfish encryption, initializes it with the generated Blowfish key, and uses it to encrypt the input data.

```
Cipher blowfishCipher = Cipher.getInstance(BLOWFISH_ALGORITHM);
blowfishCipher.init(Cipher.ENCRYPT_MODE, blowfishKeySpec);
byte[] encryptedData = blowfishCipher.doFinal(data.getBytes());
```

And creates a SecretKeySpec object for the user-provided key, creates a Cipher object for AES encryption, and uses it to encrypt the Blowfish key.

```
SecretKeySpec aesKeySpec = new SecretKeySpec(key.getBytes(), AES_ALGORITHM);
Cipher aesCipher = Cipher.getInstance(AES_ALGORITHM);
aesCipher.init(Cipher.ENCRYPT_MODE, aesKeySpec);
byte[] encryptedBlowfishKey = aesCipher.doFinal(blowfishKeyBytes);
```

Then concatenate the encrypted Blowfish key and encrypted data into a single-byte array.

```
byte[] encryptedBytes = new byte[encryptedBlowfishKey.length +
encryptedData.length];
System.arraycopy(encryptedBlowfishKey, 0, encryptedBytes, 0,
encryptedBlowfishKey.length);
System.arraycopy(encryptedData, 0, encryptedBytes,
encryptedBlowfishKey.length, encryptedData.length);
```

And then concatenated byte array in Base64 returns it as a string.

```
return Base64.getEncoder().encodeToString(encryptedBytes);
```

After that, we create a SecretKeySpec object for the user-supplied key, a Cipher object for AES decryption, use it to decrypt the encrypted Blowfish key, and create a new SecretKeySpec object for the decrypted key. We also create a Cipher object for Blowfish decryption, initialize it with the decrypted Blowfish key, and use it to decrypt the encrypted data. The decrypted data is returned as a string.

```
SecretKeySpec aesKeySpec = new SecretKeySpec(key.getBytes(), AES_ALGORITHM);
Cipher aesCipher = Cipher.getInstance(AES_ALGORITHM);
aesCipher.init(Cipher.DECRYPT_MODE, aesKeySpec);
byte[] decryptedBlowfishKeyBytes = aesCipher.doFinal(encryptedBytes,
0, 16);
SecretKeySpec blowfishKeySpec = new
SecretKeySpec(decryptedBlowfishKeyBytes, BLOWFISH_ALGORITHM);

Cipher blowfishCipher = Cipher.getInstance(BLOWFISH_ALGORITHM);
blowfishCipher.init(Cipher.DECRYPT_MODE, blowfishKeySpec);
byte[] decryptedDataBytes = blowfishCipher.doFinal(encryptedBytes, 16,
encryptedBytes.length - 16);
String decryptedData = new String(decryptedDataBytes);
return decryptedData;
```

5. Conclusion

This research explores a hybrid security method for establishing a secure cloud environment. The method achieved a maximum throughput of 98ms for Blowfish and 550ms for AES while using 20 MB of memory in the ECB and CBC models. The AES and BF models produced 426mb with $27x10^4$ms for AES and 426mb with $25x10^4$ms for Blowfish, respectively. In the absence of pipelining, this hybrid method proved to be particularly effective.

The study focused on the performance of two symmetric encryption key techniques, AES and Blowfish, in terms of encryption speed, throughput, and power consumption. Table 4

clearly shows that Blowfish outperforms AES due to its lack of known security flaws, making it an excellent standard encryption method. However, due to its higher processing resource demands, AES outperforms Blow-fish in sprint speed, resulting in a weaker overall performance for Blowfish. As a result, Blowfish may be better suited for wireless setups that deal with small packets.

It is worth noting that the Blowfish method has a security issue with weak keys that AES does not have. As a result, combining AES and Blowfish in a hybrid approach is an excellent solution for encrypting and decrypting files with large block sizes and longer keys, such as 128-bit blocks and 128, 192, and 256-bit keys. AES provides greater security in the long run. AES with Blowfish is a hybrid method that protects against Man in the Middle (MitM) attacks. To summarize, combining AES and Blowfish encryption and decryption techniques has proven to be an effective method for securing data in cloud storage. In comparison to other studies, the results show that this method not only provides a greater standard of security but also reduces computation time and file size impact. This demonstrates the proposed method's suitability for addressing security concerns about sensitive data in cloud computing environments. It provides a practical and dependable means of protecting sensitive data from unauthorized access and breaches by employing an efficient and secure two-cipher cascade.

References

[1] James, D., & Girish Tere.: Cloud-Computing. The University of Mumbai, (2018).
[2] Keijo, H., Adnan, A., Johan, L., & Tommi, M.: Introduction to Cloud Computing Technologies. TUCS General Publication (2014). https://doi.org/10.13140/2.1.1747.8082
[3] Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., & Zaharia, M.: A view of cloud computing. Communications of the ACM, 53(4), 50-58 (2010).
[4] Hashizume, K., Rosado, D. G., Fernández-Medina, E., & Fernandez, E. B.: An analysis of security issues for cloud computing. Journal of internet services and Applications, 4, 1-13 (2013).
[5] Zhang, Q., Cheng, L., & Boutaba, R.: Cloud computing: state-of-the-art and research challenges. Journal of internet services and Applications, 1, 7-18 (2010).
[6] Mell, P., & Grance, T.: The NIST definition of cloud computing (2011).
[7] Malik, A., Hiekkanen, K., Dhir, A., & Nieminen, M.: Impact of privacy, trust and user activity on intentions to share Facebook photos. Journal of Information, Communication and Ethics in Society (2016).
[8] Dowling, J.: Introduction to Cloud Computing (2019).
[9] Gandhi, P.: Data visualization techniques: Traditional data to big data. In Data Visualization: Trends and Challenges Toward Multidisciplinary Perception (pp. 53–74). Springer Singapore (2020). https://doi.org/10.1007/978-981-15-2282-6_4
[10] Purwinarko, A., & Hardyanto, W.: A Hybrid Security Algorithm AES and Blowfish for Authentication in Mobile Applications. Scientific Journal of Informatics, 5(1), 2407–7658 (2018a). http://journal.unnes.ac.id/nju/index.php/sji
[11] Abirami, M., & Chellaganeshavalli, S.: Performance Analysis of AES and Blowfish Encryption Algorithm. In International Journal of Innovative Research in Science, Engineering and Technology, An ISO, Vol. 3297, Issue 11 (2007). www.ijirset.com
[12] Sampath, D. M., Ch, U. K., & T, P.: Generating Cipher Text using BLOWFISH Algorithm for Secured Data Communications. International Journal of Innovative Technology and Exploring Engineering, 9(2), 117–121 (2019). https://doi.org/10.35940/ijitee.a5063.129219

[13] Sicari, S., Rizzardi, A., & Coen-Porisini, A.: 5G In the Internet of things era: An overview on security and privacy challenges. Computer Networks, 179, 107345 (2020).
[14] Patil, P., Narayankar, P., Narayan, D. G., & Meena, S. M.: A Comprehensive Evaluation of Cryptographic Algorithms: DES, 3DES, AES, RSA, and Blowfish. Procedia Computer Science, 78, 617–624 (2016). https://doi.org/10.1016/j.procs.2016.02.108
[15] Qadir, A. M., & Varol, N.: A review paper on cryptography. 7th International Symposium on Digital Forensics and Security, ISDFS 2019, October, 1–6 (2019). https://doi.org/10.1109/ISDFS.2019.8757514
[16] Gupta, A., & Walia, N. K.: Cryptography Algorithms : A Review. International Journal of Engineering Development and Research, 2(2), 1667–1672 (2014). http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=FEF3E8340DC536679E3C83BF43F1616C?doi=10.1.1.674.7141&rep=rep1&type=pdf last accessed 2023/06/06
[17] Ivan, P., Tommi, M., & Adnan, A.: Introduction to Cloud Computing Technologies. ACM Papers (2013). https://doi.org/10.13140/2.1.1747.8082
[18] Tawalbeh, L. A., Haddad, Y., Khamis, O., Benkhelifa, E., Jararweh, Y., & AlDosari, F.: Efficient and secure software-defined mobile cloud computing infrastructure. International Journal of High-Performance Computing and Networking, 9(4), 328-341 (2016).
[19] Purwinarko, A., & Hardyanto, W.: A Hybrid Security Algorithm AES and Blowfish for Authentication in Mobile Applications. Scientific Journal of Informatics, 5(1), 2407–7658 (2018b). http://journal.unnes.ac.id/nju/index.php/sji
[20] Masud, M., & Huang, X.: An e-learning system architecture based on cloud computing. World Academy of Science, Engineering, and Technology, 62, 74-78 (2012).
[21] Masud, M. A. H., & Huang, X.: A novel approach for adopting cloud-based e-learning system. In 2012 IEEE/ACIS 11th International Conference on Computer and Information Science, pp. 37-42, IEEE (2012).
[22] Ghosh, A. M., & Grolinger, K.: Edge-cloud computing for Internet of Things data analytics: Embedding intelligence in the edge with deep learning. IEEE Transactions on Industrial Informatics, 17(3), 2191-2200 (2020).
[23] Ul Haq, M. N., & Kumar, N.: A novel data classification-based scheme for cloud data security using various cryptographic algorithms. International Review of Applied Sciences and Engineering, September (2021). https://doi.org/10.1556/1848.2021.00317
[24] Mahmud, A. H., Angga, B. W., Tommy, Marwan, A. E., & Siregar, R.: Performance analysis of AES-Blowfish hybrid algorithm for the security of patient medical record data. Journal of Physics: Conference Series, 1007(1) (2018). https://doi.org/10.1088/1742-6596/1007/1/012018
[25] Ajmal, A., Ibrar, S., & Amin, R.: Cloud computing platform: Performance analysis of prominent cryptographic algorithms. Concurrency and Computation: Practice and Experience, 34(15), 1–18 (2022). https://doi.org/10.1002/cpe.6938
[26] Fatima, S., Rehman, T., Fatima, M., Khan, S., & Ali, M. A.: Comparative Analysis of Aes and Rsa Algorithms for Data Security in Cloud Computing. 14 (2022). https://doi.org/10.3390/engproc2022020014