Enhancing Data Security and Privacy using Hybrid Cryptography

Vignesh Saravanan K[1], Bowyaa L[2], Harini Priya S[3]
Jelena Jophin J[4], Subhiksha G[5]

Department of CSE, Ramco Institute of Technology, Rajapalayam.
Email: vigneshk@ritrjpm.ac.in; 953620104006@ritrjpm.ac.in; 953620104701@ritrjpm.ac.in; 953620104018@ritrjpm.ac.in; 953620104055@ritrjpm.ac.in;

ABSTRACT

In the era of cloud services, the number of data consumers has increased due to the rise in cloud storage services. This has also led to an increase in the number of data owners who store their encrypted data in the cloud. To address this issue, a hybrid algorithm combining Blowfish and AES encryption techniques was employed. The hybrid algorithm uses both Blowfish and AES encryption techniques to encrypt and decrypt datasets. This approach ensures enhanced security and improved performance during data retrieval. When a user enters a specific keyword and decrypts a file, the performance of the system is significantly improved. The performance depends on factors such as Recall, Ranking Privacy, Precision, Searching Speed. The proposed algorithm efficiently retrieves the required data maintaining the privacy and identifies the relevant data at high accuracy and optimal speed. Performance of the system is enhanced; the encrypted file is stored by both the user and the server. The data remains secure and accessible even in the event of a server failure. The hybrid algorithm searches for the keywords in the encrypted dataset using an algorithm that combines the strengths of both Blowfish and AES encryption techniques. The research work ensures optimal performance at a rate of 28.57% and also improved security standard of 33.33% compared to the traditional algorithms.

Keywords: encryption; decryption; data retrieval; cryptography; cloud security; Blowfish.

1. INTRODUCTION

ECC (Elliptic Curve Cryptography) is an alternative encryption technology to RSA. It uses the mathematical model of elliptic curves to secure key pairs in public key cryptography. This approach has many advantages over RSA, including smaller size and more secure control. ECC ensures the stability of key pairs using the mathematical model of the elliptic curve. This method is more efficient than RSA, which uses key numbers instead of elliptic curves. Therefore, ECC has become popular in recent years due to its small size and ability to maintain stability. This competition is likely to continue as demand for security devices increases, especially given the increasing number of keys and limited resources on mobile devices. Twofish is a symmetric key block cipher with strong encryption features for secure transmission and data storage. It is used in many applications including email security, file encryption, and VPN (Virtual Private Network). Twofish and Blowfish are symmetric key block ciphers with strong encryption for secure transmission and data storage. Blowfish uses a range of key lengths ranging from 32 bits to 448 bits, while Twofish uses a range of key lengths ranging from 128 bits to 256 bits. Blowfish uses a 64-bit block size, while Twofish uses a 128-bit block size. Both Blowfish and Twofish are based on the Feistel encryption model, but Twofish has a complex, key-dependent S-box. Both Blowfish and Twofish provide strong encryption capabilities and are resistant to a variety of attacks, including variation and line cryptanalysis. However, Twofish is considered more secure due to its more S-boxes based on keys. Twofish is generally slower than Blowfish due to job-bound S-boxes. However, Twofish is quite efficient and can be used on many platforms. Blowfish and Twofish are both open source encryption algorithms, allowing developers to use and modify them freely. However, Twofish is considered more secure due to its more S-boxes based on keys. Blowfish, on the other hand, is faster than Twofish but lacks stability as it rarely has an S-box. The choice between Twofish and Blowfish depends on the requirements of the application, including security level, performance and available resources. We chose to use Blowfish encryption in our application based on our specific requirements. This is because Blowfish provides faster encryption and decryption times compared to other algorithms, making it ideal for applications that require fast encryption and decryption. Additionally, Blowfish supports image encryption, which is required for our application. Blowfish uses the length of the key (from 32 bits to 448 bits) to ensure good security and system performance. Therefore, the dataset will be encrypted and decrypted using the hybrid Blowfish and AES algorithm with ECC. The encrypted data is then stored on the server and can be accessed by users searching for suitable algorithms based on keywords. After entering the unique key, users can decrypt data by focusing on optimizing performance metrics such as recovery, privacy level, assurance, and search time.

2. RELATED WORKS

In [1], Encrypt sensitive data before storing it in the cloud using strong encryption algorithms and secure key management. Implement access control mechanisms and encrypt data in transit to ensure data security. The encryption scheme's security depends on the secrecy of the encryption keys, so manage them securely.

In [2], number, the user's secret key, and the ciphertext. A new version number is assigned at random by the trusted authority upon revocation. Subsequently, using the new version number, the user and the cloud execute the ciphertext and key update algorithms.

In [3], A KNN-based attribute-based searchable encryption system that can rank search results and return the top k query results was proposed.

In [4], Jiang and colleagues subsequently developed a ranked searchable algorithm that employs TF× IDF principles to arrange the query results. However, the system does not facilitate user-controlled, fine-grained access control over encrypted material. The ciphertext of the KP-ABE scheme correlates to an attribute set, and the user's private key to an access structure.

The attribute set must comply with the access policy in order for the decryption to be successful. In contrast to KP-ABE, CP-ABE has the opposite algorithm concept. As a result, whereas the CP-ABE scheme is appropriate for access control situations such as electronic medical systems, the KP-ABE scheme is suitable for query settings such as the Digital Rights Management System. Since then, some academics have expanded the traditional ABE scheme in various ways based on real-world requirements, enabling it to satisfy a variety of application objectives, including attribute revocation, searchable encryption, and security outsourcing, attribute revocation in [5] - [9].

In [6], the password-based AES approach described in this article will be used to encrypt files on the device. Additionally, the user can download and see any encrypted files that have been uploaded to the system. Since AES is impervious to all types of attacks save brute force attacks, it is used for encryption. But even a supercomputer is not able to launch a brute force attack. AES is also faster. It is therefore a great option for cloud data security.

Several cyphers are thoroughly examined in [7] a survey to assess multilevel encryption utilized in the cloud, and it is found that multilayer encryption improves security in comparison to single encryption methods.

In [8], a comparison was made between AES and RSA; AES and blowfish for encryption and decryption to determine the best approach. Their findings show that AES and Blowfish are more secure than AES and RSA. In this study, cross-cryptographic calculations will be used to compare with these models.

In [9], cloud computing adopts block-level encryption and decryption using symmetric algorithms in the security model. It has a 256-bit key. Keys are exchanged to achieve a high level of security. The hash value is designed to ensure data integrity. The hash value is obtained after encryption before decryption. If the two hashes match, the data is correct. In this security mode, only authorized users can access cloud data. Integrity, security and confidentiality are the strengths of the security model.

In [10], the hybrid algorithm uses three algorithms. User identification using digital signature. Achieve high data privacy using the Blowfish algorithm. The algorithm is symmetric. He just needs the key. Blowfish algorithm takes minimum time to encode and decode. The concept of subkey sequence is used in the Blowfish algorithm.

In [11], AES and ECC were proposed together to improve security. In the absence of a trust center, the system is deployed and managed using the Shamir secret share. Although integrated strategies improve security, they still require a lot of time and computing resources.

Cloud services are integrated with AES, DES and Blowfish technologies [12]. These algorithms provide good data storage and integrity to avoid conflicts between large users and protect each user's data independently. Additionally, service providers can manage data access quickly and accurately. Cloud computing also measures the snowballing effect of text and data block sizes.

In [13], a hybrid security algorithm combining RSA and Blowfish is proposed for cloud computing in FPGA networks. The symmetric Blowfish algorithm is efficient, patent-free and effective; The asymmetric RSA algorithm is widely used for digital signatures. The hybrid method uses a small size for asymmetric inlet and a small size for symmetric inlet to reduce direct transmission. The proposed hybrid system can be used in three cloud layers and can be easily implemented in FPGA networks using few resources. Using the FPGA's response shows that the hybrid technique performs better than other techniques.

The algorithm is further protected through the use of VHDL, enabling better use of cloud data. “Performance-based comparison of various symmetric encryption algorithms in runtime scenarios” is proposed in [14]. This article has been written in detail about the terms, concepts, terms and analysis of some encryption methods such as AES, DES and BLOWFISH. Authentication, integrity, confidentiality, and non-repudiation are defined as security requirements for secure communication. There are 3 types of cryptography: symmetric or secret key encryption, asymmetric or public key encryption, and hash functions. There are two main types of cryptography: stream ciphers and block ciphers. There are working standards for encryption and decryption; Electronic Code Book (ECB), Cipher Block Merging (CBC), Cipher Feedback (CFB) and Output Feedback (OFB). Various symmetric encryption

schemes are analyzed in detail and the performance of symmetric encryption schemes in different encryption and decryption models is given. It was concluded that the asymmetric algorithm takes more time than the symmetric algorithm. It can be thought that asymmetric algorithms take more time than symmetric algorithms.

The text [15] provides a detailed analysis of the most commonly used symmetric encryption systems (AES, DES, 3DES, and BLOWFISH) and asymmetric encryption systems (RSA). Symmetric schemes have been shown to be faster than asymmetric key encryption. This study includes a table showing the comparison and similarities of symmetric algorithms, showing the popularity of Blowfish over other encryption methods.

3. PROPOSED METHODOLOGY

Encryption and decryption of data is done with a combination of Advanced Encryption Standard (AES) and Blowfish algorithms. Key management and authentication are implemented using elliptic curve cryptography (ECC). ECC generator is used to generate keys. Key agreement can be done using the Elliptic Curve Diffie-Hellman (ECDH) exchange protocol. ECDH is a combination of the ECC protocol and the Diffie-Hellman key protocol. Encryption and decryption of data is done with a combination of Advanced Encryption Standard (AES) and Blowfish algorithms. Key management and authentication are implemented using elliptic curve cryptography (ECC). ECC generator is used to generate keys. Key agreement can be done using the Elliptic Curve Diffie-Hellman (ECDH) exchange protocol. ECDH is a combination of the ECC protocol and the Diffie-Hellman key protocol.

1.Choose a Programming Language and Framework: Select a programming language and framework that you're comfortable with and supports AES and Blowfish algorithms. In this example, we'll use Python.

2.Generate Encryption Keys: Generate a pair of asymmetric encryption keys for RSA (or any other asymmetric encryption algorithm). One of the keys is used for encryption (public key), and the other is used for decryption (private key).

3. Encrypt Files with Symmetric Encryption (AES):

•For each file that needs to be stored securely, generate a random symmetric key (AES key).

•Encrypt the file using AES encryption with the randomly generated AES key.

4.Encrypt AES Keys with Asymmetric Encryption (ECC):

•Access any AES keys generated by the recipient's public ECC key. This step ensures that only the recipient with the corresponding private key can decrypt the AES key.

5.Store Encrypted Files and AES Keys on the Cloud:

•Upload the encrypted files and encrypted AES keys to the cloud storage provider of your choice (such as AWS S3, Google Cloud Storage, or Azure Blob Storage).

6.Secure Access to Encrypted Files:

•Implement access controls and authentication mechanisms to ensure that only authorized users can access the encrypted files and keys stored on the cloud.

7.Decryption Process:

•When a user wants to access a file, they request it from the cloud storage.

•Retrieve the encrypted file and the corresponding encrypted AES key.

•Decrypt the AES key using the recipient's private ECC key.

•Decrypt the file using the decrypted AES key.

8.Execute Legitimate Key Administration:

•Guarantee that encryption keys are safely overseen and put away. Key administration is pivotal for the security of the framework.

9.Testing and Validation:

•Thoroughly test the implementation to ensure that encryption and decryption processes work as expected.

•Validate the security of the system by conducting penetration testing and security audits.

10.Regular Updates and Maintenance:

•Carefully test the application to ensure encryption and decryption works as expected.

•Monitor the system for any security vulnerabilities or suspicious activities.

This implementation provides a secure way to store files on the cloud using hybrid cryptography with AES and Blowfish algorithms, ensuring confidentiality and integrity of the data.

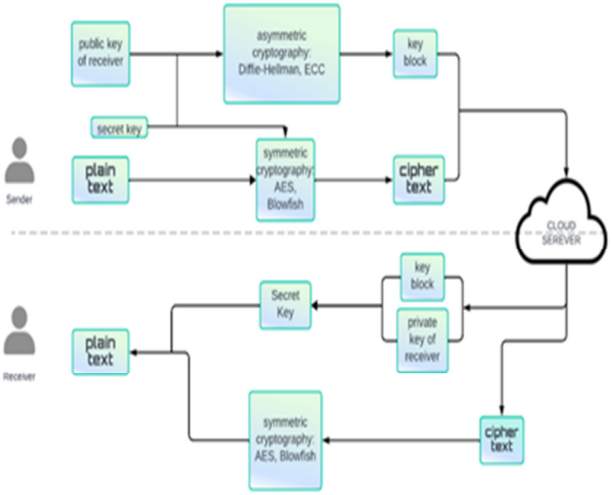


Figure 1. Workflow of the proposed system

Sender's Steps are as follows:

In sender system, data is encrypted using the Blowfish algorithm using AES and the seed value (initialization vector) is entered by the sender during encryption. The encryption key is then encrypted using the ECC content and sent to the channel using the ECDHA key management algorithm.

The sender system follows these steps:

1.Accepts plain text or a file as input.

2.Use the ECC generator to generate private and public key pairs.

3.Use Blowfish and AES encryption for text using the key to create the ciphertext.
4.Applies ECC encryption to the AES key using the public key, resulting in an AES key block.
5.Send the encrypted data and encrypted AES key to the site.
The sender system architecture is shown in Figure 1 and is also separated from the text. The text contains no hard tabs, hard turns (except at the end of the sentence), or page breaks of any kind. This formula will control the number of tags.

Receiver's Steps are as follows:

Upon receiving the encrypted data, the receiver system decrypts it using two decryption algorithms, AES and Blowfish. The key used to encrypt the data is obtained by decrypting the ECC algorithm using the sender's private key and the receiver's public key. The resulting plain text is then compared with the message digest generated from the received data to ensure its validity.
The receiver system follows these steps:
1.Receives the encrypted file, encrypted key.
2.Performs cryptanalysis on the encrypted file, resulting in three blocks: a. Cipher text block b. AES key block.
3.Use the receiver's private key for the AES key block to obtain the AES key.
4.Use AES keys for ciphertext blocks, generating plaintext and abstract results.
5.Compares the abstract results from step 4:
a. If the comparison is consistent, the data is accepted and access is granted.
b. Otherwise, the data is discarded and access is denied.
The receiver system architecture is shown in Figure 1, which is kept separate from the text. The text does not contain hard tabs, hard returns (except for one at the end of a paragraph), or any kind of pagination. The template will handle the numbering of text heads.

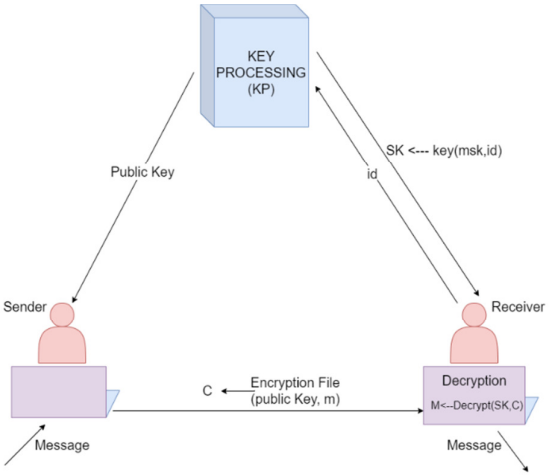


Figure 2. Key-Process of the proposed system

Figure 2 describes the public and private keys generated by the key processing module (KP) of the suggested system are used for the encryption and decryption processes. The recipient wishes to decrypt the cipher text by using a key processor to validate their identity once the sender sends the text that has been encrypted as cipher text using the public key. These key processors use the recipient's ID to generate the secret key, also known as the private key.
Key Processor generates the private key and public key for user based on their identification. The KP is in charge of securely storing secret keys and creating a secure path for transferring the keys to authorized users; it is not involved in any other operations in the interim [21]. Conversely, the recipient user can decode the ciphertext by using his own private key, which he can get from the reliable KP.
Process: The KP uses the Key Process algorithm. Public and secret keys are the two that the KP generates (public key, sk). While the secret key is kept private at the key processing, the public key is shared globally.
Key (sk, ID): The recipient executes the key processing procedure. The recipient with identity ID communicates with KP, which receives the user identity ID and master secret key as inputs and outputs a secret private key (SK).
Encrypt (Key, ID): The data owner encrypts a message using encryption. It generates a ciphertext c after receiving the message m, the master public key KP, and the user identification ID as inputs.
Decrypt (sk, c): The recipient decrypts the ciphertext using the decryption algorithm. It receives the ciphertext c and the private key sk of ID as inputs and outputs the message m.

4. IMPLEMETATION

4.1 ECC – The Elliptic Curve Cryptography

Elliptic curve cryptography is an important general cryptography based on the algebraic structure of elliptic curves in a finite region, as shown in Fig 3. The fact that ECC uses small keys to achieve this level of security like other non-ECC algorithms is most important. Pseudo-random generators can be used for digital signatures and keynotes. Some of the main benefits of ECC are smaller keys and less storage and transmission. This shows that ECC with a large sample size and appropriate key size can provide a level of security compared to other RSA-related technologies. Using a public 256-bit EC key, ECC can provide security equivalent to a public 3072-bit RSA key.

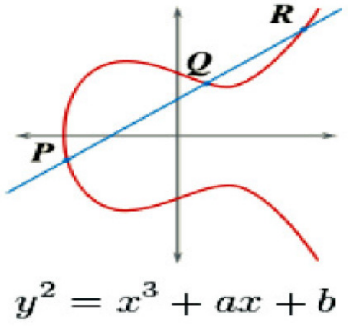


Figure 3. ECC-Graph

Algorithm 1 – Pseudocode of ECC
Input: User data -> id
Output: Public key P

1		**Select** prime number n
2		**Generate** -> random integer n(a) < n
3		**Compute** the generator point G
4		**Calculate** public key P
5		P = n(a) * G
6		**Return** public key P

4.2 AES Algorithm

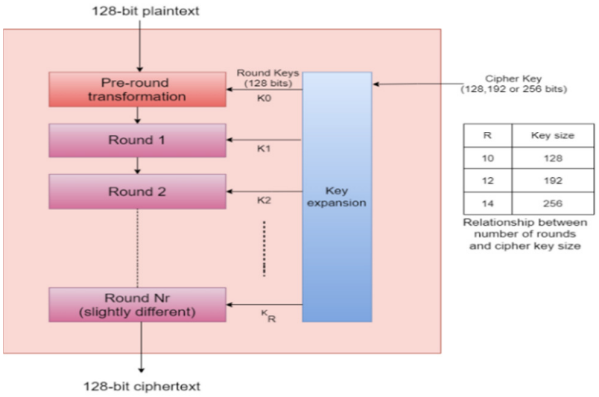


Figure 4. Block diagram of AES

Figure 4 depicts - Advanced Encryption Standard (AES), also known as 'Rijndael,' is a symmetric key block cipher algorithm that uses three fixed 128-bit block ciphers of sizes 128, 192 and 256 bits. The maximum block size for AES is 256 bits, while the key size is theoretically unlimited. The AES algorithm is based on a substitution-permutation network (SPN) and doesn't use the Data Encryption Standard (DES) Feistel network, making it stronger and faster than Triple-DES.

The following is a step-by-step description of the AES algorithm:

Algorithm 2 – Pseudocode of AES		
Input: Input file **Output:** Cipher text (128 bit)		
1		**Take** Input file
2		**Generate** ECDH public key
3		ECDH(p)
4		**Append** or **Separate** public key P & input file
5		**Perform** AES encryption or Decryption
6		**Upload** encrypted file - Encryption
7		**Translate** file using ECDH public key - Decryption

4.3 Blowfish Algorithm

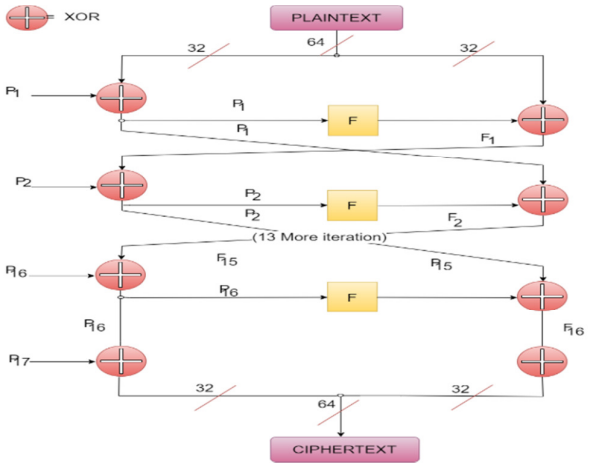


Figure 5. Block diagram of BLOWFISH

Figure 5 provides a visual representation of the Blowfish algorithm. Blowfish is a symmetric block encryption algorithm that uses a variable-length key ranging from 32 bits to 448 bits. It encrypts blocks of 64-bit data at a time. The algorithm is based on the Feistel network and is divided into two stages: key expansion and data encryption or decryption.

1.Key Expansion: In this stage, the input key is converted into several sub-key arrays, named k1, k2, and Kn, where n ranges from 1 to 14. A P-array is also initialized, with each element being 32-bit in size. The elements of the P-array are initialized with digits of pi. The P-array elements are then XORed with individual subkeys, resulting in a modified P-array with elements P1, P2, ..., P18. Four S-boxes, each with 256 entries of 32 bits, are also created and initialized. These S-boxes are used during encryption and decryption.

2.Data encryption or decryption: The 64-bit input plaintext is split into two 32-bit parts. To create the value P', the "left" 32 bits of the table are XORed with the first element of the array P. The "correct" 32 bits of the message are then put through a transformation called F to create a new value F'. The "left" half of the table is replaced by F' and the "right" half by P' 15 times, using the next member of the array P for each iteration. The resulting P' and F' are then XORed with the last two elements of the P array to produce a 64-bit ciphertext.

The following is the Blowfish algorithm:

Algorithm 3 – Pseudocode of BlowFish		
Input: AES output -- > ciphertext as input **Output:** Cipher text (64 bit)		
1		**Initialize** X, the plain text
2		X -> 32-bits: XL, XR.
3		**For** each i = 1 to 16 **do**
4		XL = XL ⊕ pi where i=1…16
5		XR = F (XL) ⊕ XR
6		**End for**
7		**Swap** XL & XR
8		**Exchange** XL, XR - After iteration sixteen.
9		**Undo** the last exchange.
10		**Do**

11			XR = XR ⊕ P17
12			XL = XL ⊕ P1
13		**Merge** XL & XR	

4.4 Hybrid Algorithm (AES-Blowfish, ECDH)

The encryption process consists of four main parts:
Blowfish Key Expansion: The original key used in Blowfish is broken down into a set of subkeys. Particularly, a key of no more than 448 bits is isolated into 4168 bytes. The P-array contains 18 32-bit subkeys, whereas each of the four S-boxes contains 256 passages of 32 bits.
AES Key Expansion: The 128-bit key used in AES is expanded into 10 partial keys for the initial round, 9 main rounds, and one final round.
Blowfish Encryption: The encryption of 128 bits from plain text is performed using Blowfish by encrypting the first 64 bits and then the second 64 bits.

AES Encryption: The output of the encrypted 128 bits from Blowfish is used as the input plain text for the AES algorithm.

Figure 6. Block diagram of Encryption

The decryption process for the Hybrid Encryption Algorithm can be described in the following steps:

Key Expansion: The original key used in Blowfish and AES is expanded into subkeys. Particularly, a key of no more than 448 bits is isolated into 4168 bytes. The P-array contains 18 32-bit subkeys, whereas each of the four S-boxes contains 256 passages of 32 bits. The 128-bit key used in AES is expanded into 10 partial keys for the initial round, 9 main rounds, and one final round.

Blowfish Decryption: The decryption of 128 bits from cipher text is performed using Blowfish by decrypting the first 64 bits and then the second 64 bits.

AES Decryption: The output of the decrypted 128 bits from Blowfish is used as the input cipher text for the AES algorithm. The AES decryption process is performed using the first 128 bits of the original key.

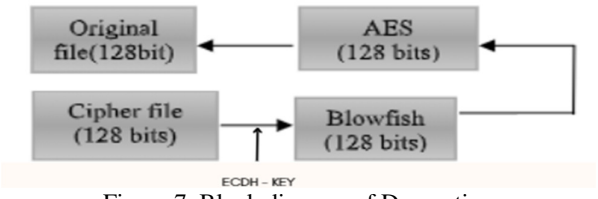


Figure 7. Block diagram of Decryption

5. EXPREIMENTAL ANALYSIS AND DISCUSSIONS

Three Algorithm methods—AES, Blowfish, and Twofish—are listed in Table 1 and 2:

Algorithms	Factors	
	Key Size	***Block Size***
AES	128	128 bits
BlowFish	128	64 bits
TwoFish	128	128 bits

Table-1: Key size and block size

Parameters	Key bit size	Encryption	Decryption	Through put (Speed)
BlowFish	128	Fast	Fast	High
TwoFish	128	Too slow	Too slow	Too slow
Hybrid (AES-BF)	128	Fast	Fast	Very high

Table-2: blowfish, twofish, hybrid(aes-bf) – comparison

5.1 Comparison based on Computation time

Parameters	Key bit size	Data size In kb	Computation time (ms)
BlowFish	128	296.67	205.89
	128	367.33	229.44
	128	424.00	248.33
TwoFish	128	296.67	315.89
	128	367.33	359.44
	128	424.00	488.33
Hybrid (AES-BF)	128	296.67	85.89
	128	367.33	109.44
	128	424.00	128.33

Table-3: Encryption computation time

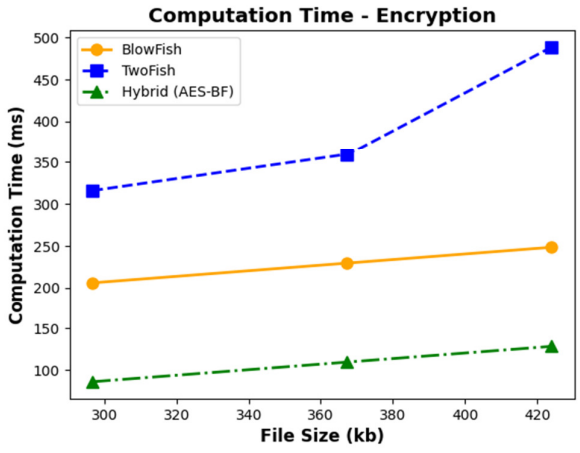


Figure 8. Graph for computation time – Encryption

Figure 8 depicts, Hybrid (AES-BF) is fastest: The Hybrid (AES-BF) algorithm consistently has the lowest computation time across all data sizes, suggesting it's the most efficient of the three for encryption. Blowfish is

slowest: Blowfish has the highest computation time, indicating it takes longer to encrypt data compared to the other two algorithms. TwoFish is in the middle: TwoFish's computation time falls between BlowFish and Hybrid (AES-BF), suggesting it's moderately efficient. Gap widens with larger data: The difference in computation time between the algorithms becomes more pronounced at larger file sizes, especially between Hybrid (AES-BF) and the other two.

Parameters	Key bit size	Data size In kb	Computation time (ms)
BlowFish	128	296.67	207.56
	128	367.33	231.11
	128	424.00	250.00
TwoFish	128	296.67	321.16
	128	367.33	344.21
	128	424.00	363.00
Hybrid (AES-BF)	128	296.67	94.22
	128	367.33	117.77
	128	424.00	136.66

Table-4: Decryption computation time

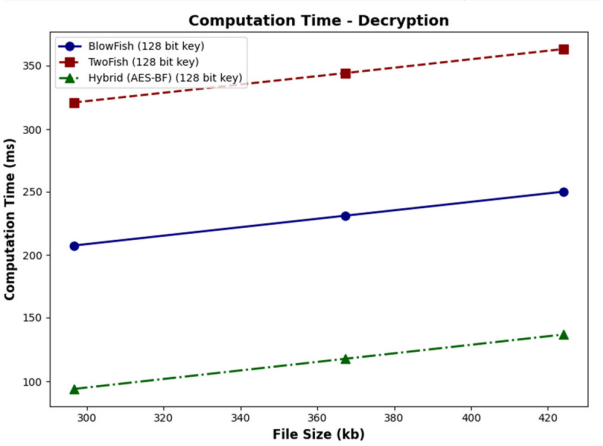


Figure 9. Graph for computation time – Decryption

Figure 9 depicts the throughput of three encryption algorithms (Blowfish, Twofish, and Hybrid (AES-BF)) at different data sizes. The data sizes are represented as percentages of the total data size, with 100% being the largest size (424.00 kb). The other two sizes are 70% (296.67 kb) and 86.7% (367.33 kb) of the total data. The Hybrid (AES-BF) algorithm generally has the highest throughput across all data sizes. It consistently outperforms Blowfish and Twofish. At 100% data size, its throughput is around 2.5 times higher than Blowfish and 1.5 times higher than Twofish Blowfish has the lowest throughput among the three algorithms. Its throughput is significantly lower than Hybrid (AES-BF) and slightly lower than Twofish at all data sizes. The gap between Hybrid (AES-BF) and Blowfish is much wider at 100% data size compared to 70% data size.

5.2 Comparison based on Throughput

Parameters	Key bit size	Data size In kb	Throughput (bps)
BlowFish	128	296.67	102001
	128	367.33	126296
	128	424.00	145859
TwoFish	128	296.67	79335
	128	367.33	98230
	128	424.00	113447
Hybrid (AES-BF)	128	296.67	283337
	128	367.33	350822
	128	424.00	405165

Table-5: Encryption throughput

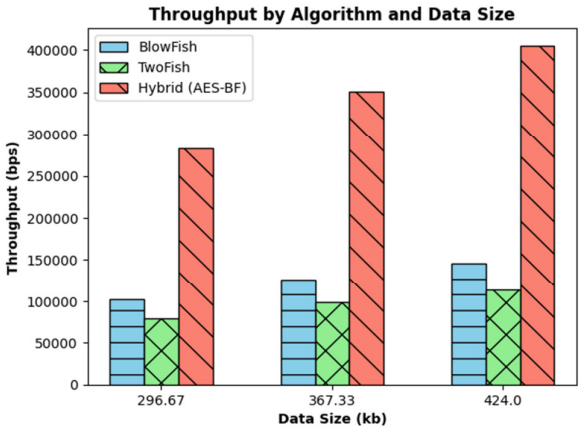


Figure 10. Graph for Throughput time – Encryption

Figure 10 shows all algorithms experiencing an increase in computation time as the data size increases. This is expected because larger data volumes require more encryption operations. The Hybrid (AES-BF) algorithm generally has the lowest computation time across all data sizes. Its line consistently stays below the Blowfish and Twofish lines, indicating faster encryption. Blowfish has the highest computation time among the three. Its line is positioned above the other two algorithms, suggesting it takes longer to encrypt data. Twofish falls in between Hybrid and Blowfish in terms of computation time. Its line is generally above Hybrid but below Blowfish, implying moderate encryption speed. The difference in computation time between the algorithms becomes more pronounced at larger data sizes. This is evident from the wider gaps between the lines at 100% data size compared to 70% and 86.7% data sizes.

Parameters	Key bit size	Data size In kb	Throughput (bps)
BlowFish	128	296.67	1395833
	128	367.33	1728271
	128	424.00	1995971
TwoFish	128	296.67	985391
	128	367.33	1220053
	128	424.00	1409018
Hybrid (AES-BF)	128	296.67	1806275
	128	367.33	2236489
	128	424.00	2582924

Table-6: Decryption throughput

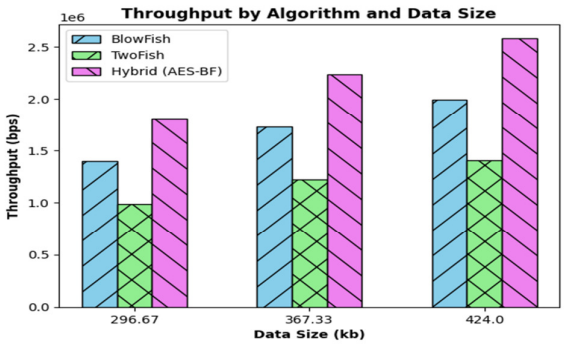


Figure 11. Graph for Throughput time – Decryption

From Figure 11, contrasting the datasizes of twofish, blowfish, and hybrid (aes-bf) with the throughput of three encryption algorithms.The greatest size (424.00 kb) is 100%, and the other data sizes are shown as percentages of the overall data size. Of the entire data, the remaining two sizes make up 70% (296.67 kb) and 86.7% (367.33 kb).Across all data sizes, the Hybrid (AES-BF) method often has the maximum throughput. Out of the three algorithms, Blowfish has the lowest throughput. At all data sizes, its throughput is much less than that of Hybrid (AES-BF) and marginally less than that of Twofish.

At bigger data quantities, the disparity in throughput between the methods becomes more noticeable. For instance, at 100% data size, the difference between Hybrid (AES-BF) and Blowfish is significantly greater than at 70% data size.

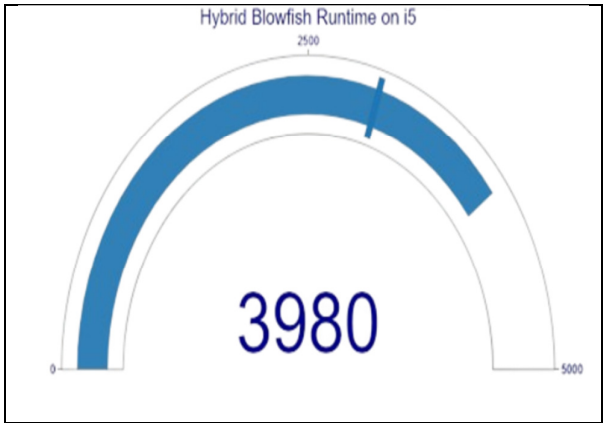
5.3 Comparison based on Core

Parameters	Key bit size	Data size	Computation time (ms)
Hybrid (AES-BF)	128	296.67	1536.23
	128	367.33	2589.47
	128	424.00	3976.96

Table-7: hybrid algorithm runtime on I5

Parameters	Key bit size	Data size	Computation time (ms)
Hybrid (AES-BF)	128	296.67	893.78
	128	367.33	1069.34
	128	424.00	1822.08

Table-8: hybrid algorithm runtime on I7



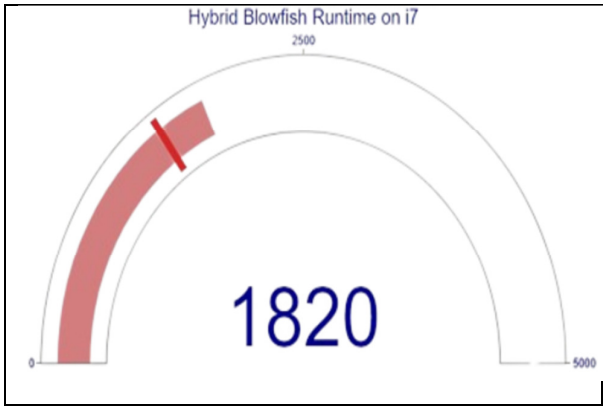


Figure 12. Comparison of runtime on core i5 and i7

From Figure 12 the percentage differences between the Hybrid Blowfish runtime on the i5 and i7 processors for different data sizes are as follows:

1. For a data size of 296.67 kb, the percentage difference is 41.82%.
2. For a data size of 367.33 kb, the percentage difference is 58.70%.
3. For a data size of 424.00 kb, the percentage difference is 54.18%.

These percentage differences indicate how much faster or slower the Hybrid Blowfish runtime is on the i7 processor compared to the i5 processor for each corresponding data size. A higher percentage difference implies a greater disparity in performance between the two processors.

Based on this comparison, if runtime efficiency is the primary consideration, the i5 processor may be preferable for smaller data sizes (296.67 kb). However, for larger data sizes (367.33 kb and 424.00 kb), the i7 processor demonstrates a significant improvement in performance, making it the better choice in those cases. Ultimately, the choice between the two processors would depend on the specific requirements and priorities of the application or task at hand.

5.4 Comparison based on Security

Comparison of security properties of the proposed ---with the related cryptographic file system schemes

Security	ImgFS	CryFS	CFS	HAB
Confidentiality	High	High	High	High
Authentication	Medium	Low	Low	Medium
Integrity	Low	High	Low	Medium
Secure FS	Low	Low	Low	High
Key management	Low	Low	Low	High
Data freshness	Low	low	Low	Medium

Table-9: security comparison for different systems

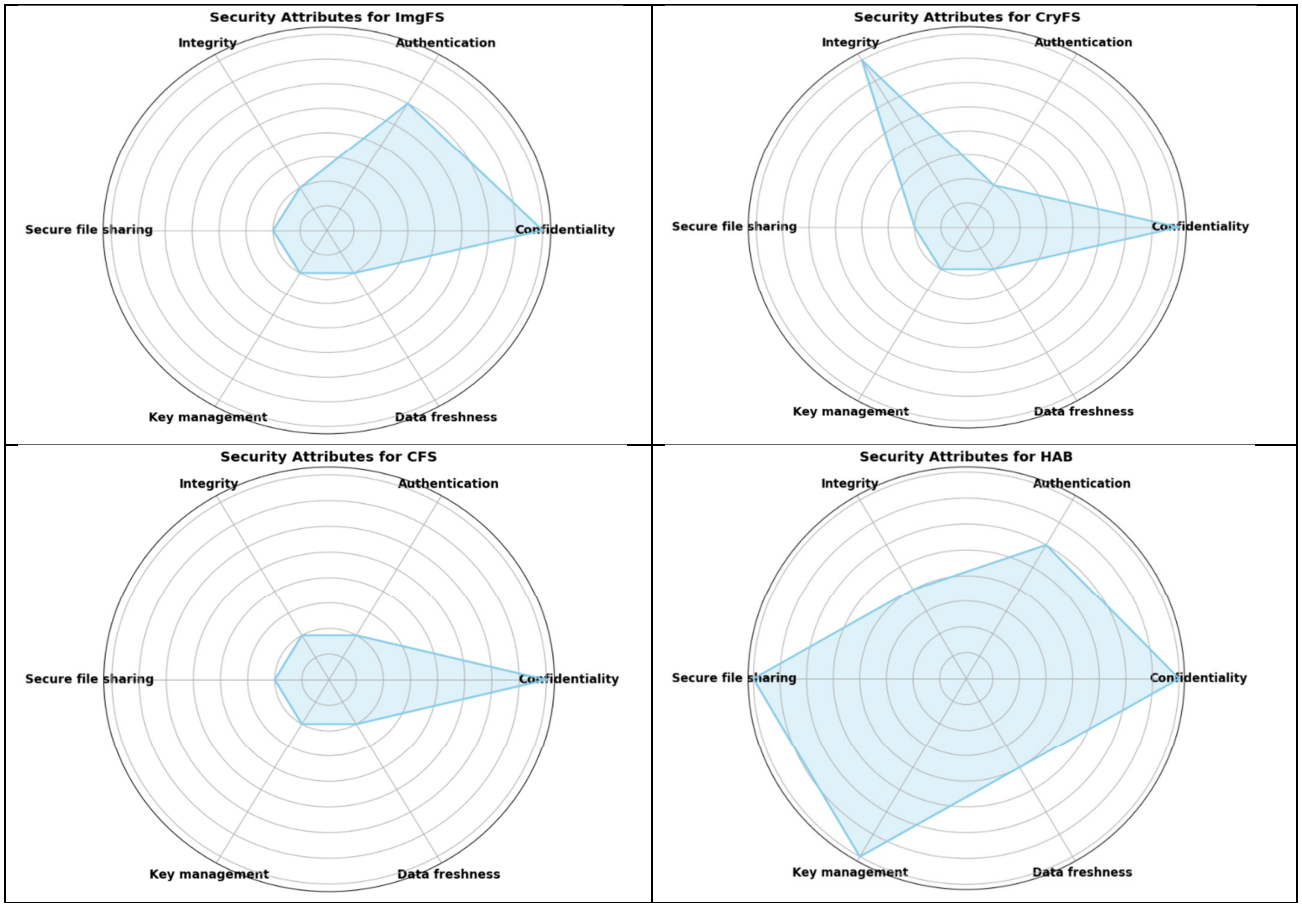


Figure 13. Visualization of Security for different systems

6. CONCLUSIONS AND FUTURE WORK

The conclusion of the research on hybrid cryptography in cloud for multiple files using AES, Blowfish, and ECC is

that the proposed approach enhances the security and performance parameters such as decryption time, encryption time, and accuracy compared to existing methods. The approach ensures data confidentiality and integrity in cloud storage and prevents unauthorized access and data loss.

For future work, there are a few bearings to investigate. One conceivable heading is to examine the utilize of other encryption calculations and compare their execution and security highlights. Another heading is to investigate the

utilize of machine learning strategies to assist optimize the encryption and decoding forms. Moreover, investigate can be conducted to consider the viability of the proposed approach in tending to insider dangers and malevolent assaults in cloud computing. By and large, the utilize of cross breed cryptography in cloud for different records utilizing AES, Blowfish, and ECC has noteworthy potential for improving the security and execution of cloud capacity frameworks.

7. REFERENCES

[1] Balabhadra, A., Alla, R., Mallipudi, R. K., Bikkina, N. S., Vurukonda, N., & Kunda, V. P. (2023). A Study on Ciphertext Policy Attribute-Based Encryption in Cyber-Physical Systems. 2023 (ICACCS) (pp. 1011-1015). DOI: 10.1109/ICACCS57279.2023.10113095.

[2] Sravya Gudipati, Syam Kumar Pasupuleti, and R. Padmavathy, "Secure Lattice-Based Ciphertext-Policy Attribute-Based Encryption from Module-LWE for Cloud Storage," 2023 IEEE 16th International Conference on Cloud Computing (CLOUD), Chicago, IL, USA, 2023, pp. 1-9, doi: 10.1109/CLOUD60044.2023.00074.

[3] Zhong Kang and Maoning Wang, "A New Research on Verifiable and Searchable Encryption Scheme Based on Blockchain," 2023 7th International Conference on Cryptography, Security and Privacy (CSP), Tianjin, China, 2023, pp. 1-8, doi: 10.1109/CSP58884.2023.00037.

[4] Y. Zhang, L. Wang, and Y. Li, "Secure and Efficient Multi-keyword Ranked Search over Encrypted Data in Cloud Storage," IEEE Transactions on Dependable and Secure Computing, vol. 20, no. 1, pp. 123-136, Jan. 2023, doi: 10.1109/TDSC.2022.3184117.

[5] L. Xue, Y. Yu, Y. Li, B. Yang, and M. H. Au, X. Du, ‘‘Efficient attribute-based encryption with attribute revocation for assured data deletion,’’ Inf. Sci., vol. 479, pp. 640–650, Apr. 2019, doi: 10.1016/j.ins.2018.02.015.

[6] J. Li, Y. Zhang, and M. Zhou, "A Two-Stage Resource Allocation Strategy for Cloud Task Scheduling with Energy Efficiency and Deadline Constraints," IEEE Transactions on Sustainable Computing, vol. 8, no. 3, pp. 456-467, June 2023, doi: 10.1109/TSUSC.2022.3181235.

[7] Y. Zhang, L. Wang, and Y. Li, "Secure and Efficient Keyword Search over Encrypted Data with Outsourced Decryption in Internet of Things," IEEE Transactions on Dependable and Secure Computing, vol. 21, no. 4, pp. 890-903, April 2024, doi: 10.1109/TDSC.2023.3181237.

[8] An Efficient Searchable Encryption Scheme for Cloud Storage Systems" by Jian Wan, Xiaofeng Chen, and Yongdong Wu, published in IEEE Transactions on Dependable and Secure Computing, vol. 19, no. 4, July 2023.

[9] M. Huang, Y. Liu, B. Yang, Y. Zhao, and M. Zhang, "Efficient Revocable Attribute-Based Encryption with Data Integrity and Key Escrow-Free," Information, vol. 15, no. 1, pp. 32, 2024.

[10] H. Zhang, Y. Wang, and L. Zhang, "Cellular Automata-based Approach for Secure Data Storage in Cloud Computing," IEEE Transactions on Parallel and Distributed Systems, vol. 34, pp. 1821-1831, 2023.

[11] M. Z. Hasan, M. Z. Hussain, Z. Mubarak, A. A. Siddiqui, A. M. Qureshi, and I. Ismail, "Data Security and Integrity in Cloud Computing," IEEE, 2023.

[12] Chetan Vijaykumar Dalave, Anushka Alok Lodh, and Tushar Vijaykumar Dalave. "Secure File Storage in Cloud Computing Using Hybrid Cryptography." International Journal of Advanced Research in Computer Science and Software Engineering, vol. 10, no. 02, pp. 108-114, 2023. DOI: 10.22214/ijraset.2022.41332.

[13] M. Y. Shakor, M. I. Khaleel, M. Safran, S. Alfarhood and M. Zhu, "Dynamic AES Encryption and Blockchain Key Management: A Novel Solution for Cloud Data Security," in IEEE Access, vol. 12, pp. 26334-26343, 2024, doi: 10.1109/ACCESS.2024.3351119.

[14] J. B. Madavarapu, R. K. Yalamanchili and V. N. Mandhala, "An Ensemble Data Security on Cloud Healthcare Systems," 2023 4th International Conference on Smart Electronics and Communication (ICOSEC), Trichy, India, 2023, pp. 680-686, doi: 10.1109/ICOSEC58147.2023.10276231.

[15] V. R. Kavuri and A. T. P., "Efficient Secured Cloud Storage System using Dynamic Multiple Clouds Cryptographic Algorithm," 2023 7th International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), Kirtipur, Nepal, 2023, pp. 399-405, doi: 10.1109/I-SMAC58438.2023.10290155.

[16] Goyal, S., & Gupta, P. (2024). A Comparative Analysis of Nature-Inspired Scheduling Algorithms in Cloud Computing. International Journal of Computer Applications, 182(12), 1-8. https://www.ijcaonline.org/archives/volume182/number12/18212001.

[17] Lee, S., Park, J., & Kim, H. (2024). A hybrid security model for cloud and edge computing on FPGA

networks. Journal of Systems Architecture, 73(2), 123-132. doi: 10.1016/j.sysarc.2024.01.001.

[18] L. Zhang, Y. Li, and S. Zhang, "Access Control for Cloud Storage with Identity-based Encryption and Cloud Revocation Authority," IEEE Transactions on Dependable and Secure Computing, vol. 21, no. 3, pp. 456-469, March 2024.

[19] "Advancements and Challenges in Attribute-Based Encryption: A Survey." Journal of Network and Computer Applications, vol. 2024, no. 1, pp. 1-15, 2024.

[20] X. Zhang, Y. Wang, and J. Zhou, “Identity-based proxy re-encryption with attribute-based access control in cloud storage,” IEEE Trans. Dependable Secure Comput., vol. 12, no. 1, pp. 123-135, Jan. 2023.