Publisher: Engineering College, Al Muthanna University

Open Access

DOI:10.52113/3/eng/mjet/2026-14-02-/25-43, Vol. (14), Issue (2), (2026)

Muthanna Journal of Engineering and Technology MJET

Submitted 15 January 2026, Accepted in revised form 02 April 2026, Published online 15 April 2026

The Effect of Block Cipher Modes on Confidentiality and Authentication in Network Communication Security

Bushra Jaber M.Jawad

College of Administration and Economics, University of Kerbala, Karbala, Iraq
**Corresponding author E-mail: bushra.j@uokerbala.edu.iq*

Abstract

High-performing authenticated encryption schemes are essential for modern communication systems to provide both confidentiality and authentication. However, although many schemes have been evaluated for their performance and are in wide-spread use today, there are relatively few examples of comparative studies showing how authenticated encryption schemes compare against each other with respect to their security bounds or resistance to forgery. Furthermore, most performance comparisons have been made between abstract modes of operation that do not provide guaranteed security from either authenticated encryption or integrity protection; and thus there is no definitive security assessment for these modes of operation. In this paper, we provide a comprehensive analytical and experimental comparison between three algorithms and three modes of operation (GCM, CCM, and EAX), using multiple variables related to performance (encryption time, decryption time and throughput), as well as formal performance metrics (authentication strength, probability bounds of forgery, and integrity through tag manipulation). The results of the study illustrates that there are significant differences in performance between GCM, CCM, and EAX regarding their throughput. The EAX value recorded using the AES algorithm was the best performer, while Blowfish had the second best performance, and the CAST algorithm had the lowest. For all comparisons, the authentication failure probability was always below the established cryptographic threshold for authentication success. This research provides a comparative performance-security outline for each algorithm-mode, and the findings can serve as a reference for choosing the appropriate algorithm-mode for authenticated encryption systems.

Keywords: *Symmetric Cryptography; Block cipher algorithms; Modes of Operation; GCM Modes; CCM Modes; EAX Modes.*

1. Introduction

Cryptography refers to the principles, systems, and algorithms required to protect communication and the sharing of information when exchanging information between one or more parties[1],[2]. It is typically described as protecting sensitive information so that it is not revealed to unauthorized third parties and, thus, protecting to some degree the confidentiality, integrity, and authenticity of that information via the communication of different formats. The main objectives are to: confidentiality, integrity, authentication and non-repudiation;[3], [4], confidentiality is to denote that the information is only viewable by the intended entity,[5] [6], integrity is to mean that the message sent is only what the sender intended [7], [8]. Authentication denotes that a message can be proven to be either true or false based on credentials [9], [10]. Additionally, non-repudiation denotes that the sender cannot deny sending the message or contents of that message [11], [12]. Furthermore, there are various different types of cryptography, such as: symmetric-key and asymmetric-key algorithms, hashes and digital signatures [13]. A symmetric-key algorithm uses the same key for both encryption of the information and decryption of the information[14]. With symmetric-key algorithms you will need to keep the key secret from everyone, including any adversaries. Two examples of symmetric-key algorithms are AES and DES [15],[16]. Asymmetric-key algorithms, on the other hand, use a key pair. One of the keys is public and can be used by

anyone to encrypt the information while the other key, the private key, is kept to decrypt the information. One example of an asymmetric-key algorithms is RSA. Block ciphers can block encrypt/decrypt, and when dealing with a large amount of data, you can also just block encryption instead of encrypting all of the data: you can accomplish that with some modes of operation and block ciphers. The modes of operation are ECB, CBC, CTR, OFB, and CFB [17], [18]. The modes of running a block cipher give methods for processing and running the block cipher on larger than a block of data, that also more securely and more quickly.

1.1 Types of Cryptography

Cryptography encompasses the processes and methods of converting information into a form that conceals its substance, thereby ensuring that its content remains undisclosed to unintended recipients. It facilitates the secure emission, transmission, and storage of information, thereby enabling entities to communicate without apprehensions about the fear of security compromise or the need for surveillance.

Cryptography is typically classified as symmetric, asymmetric, hash functions, and digital signatures [19]. Symmetric cryptography uses the same key for encrypting and decrypting. Asymmetric cryptography uses a key pair for a process, where one key is public and the other is private, also known as public key cryptography. Each key is used for one direction of the process.[20], [21] [22].

Block ciphers constitute a category of symmetric-key encryption techniques that transform fixed-length blocks of plaintext into ciphertext by concealing the original data with a key-based cryptographic transformation. Since messages often exceed the size of a single block, various block cipher modes of operation have been developed to enhance security and efficiency. These modes enable block ciphers to securely process messages of arbitrary length, transforming encryption and decryption into repetitive operations on multiple blocks [2],[23].

1.1.1 Symmetric Cryptography

Symmetric cryptography utilizes a common key for encryption and decryption sensitive information. Terminology in symmetric cryptography often includes: single key, private key, secret key or shared key cryptography. Symmetric encryption can employ either stream ciphers or block ciphers. There are numerous algorithms, including, AES, DES, 3DES, and Twofish that primarily differ in block size, key size, and the number of rounds [24], [25], [19], [26], [27], [28], [29].

Block ciphers are one of the fundamental components of symmetric cryptography. A block cipher encrypts a fixed-size block of data and generally operates on blocks sizes of 64 bits or 128 bits, and applies an invertible transformation using the secret key. The security of a block cipher is predicated on the choice of a pseudorandom permutation from the set of all permutations on the block space [30]. For the most part, the goal is to encrypt data that is greater than a single block, therefore modes of operations define how to securely apply the block cipher to large message [31], [23], [32].

1.1.2 Asymmetric Cryptography

Asymmetric cryptography uses a pair of keys - a public and a private key [33]- to provide confidentiality, authentication and non-repudiation [26], [19]. The public key encrypts the message; the private key signs the message. The operations can be inverted for decryption and signature verification. Hence, a sender with the recipient's public key can securely send a confidential message. The sender possessing his or her own private key allows the sender to create a verifiable and non-repudiable signature. However, asymmetric cryptography is often too slow to use on its own, so in practice, symmetric cryptography is used for bulk data and asymmetric cryptography is used solely to transfer a symmetric key [34].

1.2 Modes of Operation for Block Ciphers

Block ciphers operate on fixed-length strings of bits, such as 64 bits for DES or 128 bits for AES. Plaintext is divided into blocks, a block cipher transformation is executed via a mode of operation for each block, and the outputs from each encryption operation are combined to form the full ciphertext. To decrypt, the reverse operations are performed on the ciphertext to recover the plaintext [35]. A mode of operation is an algorithm which utilizes a block cipher to provide information services such as confidentiality or authenticity. The most well-known are ECB, CBC, CTR, OFB, and CFB [23] [36].

1.2.1 Electronic Code Book (ECB) mode

In this mode each block of plaintext is independently encrypted with the block cipher using a secret key. ECB is simple, allows concurrent encryption, but is really only appropriate for random/plaintext messages or short messages. ECB does not hide data patterns well.

1.2.2 Cipher Block Chaining (CBC) mod

In CBC confidentiality is enhanced in that each block of plaintext is XOR-ed with the preceding ciphertext block before it is encrypted. This ensures that identical blocks of plaintext will encrypt to different blocks of ciphertext. The first block of plaintext is XOR-ed with a random initialization vector (IV). CBC cannot be used for parallel encryption, but it supports parallel decryption [36], [37]

1.2.3 Counter (CTR) mode

CTR creates key streams by encrypting the successive values of a counter via a block cipher; the key stream produced from the countered value is XOR-ed with plaintext. The simplicity of counter mode allows the possibility of parallelism and random access, which is beneficially supportive of these techniques. CTR is secure as long as the same values of the counter are not reused.

1.2.4 Output Feedback (OFB) mode

OFB converts a block cipher to a synchronous stream cipher. The IV is encrypted and the output subsequently becomes the next input into the block cipher, or the output stream is XOR-ed to plaintext. As a result of its properties, OFB may also provide for error-tolerant encryption.[18]

1.2.5 Cipher Feedback (CFB) mode

CFB is analogous to OFB, except the previous outputs of the ciphertext block are used as input into the same block cipher to produce the output keystream. For this reason cfb is also classified as self-synchronizing stream cipher.

1.2.6 EAX Mode

EAX is an AEAD scheme combining the ECBC and EMAC modes, both derived from a fixed-length block-cipher PRF P. Its design emerged from efforts to safeguard the broad applicability of the ECBC encryption mode when a specific authentication function was neglected. EAX lends itself to reliable encryption of data streams within IPsec and a range of multicast modes compatible with the Kerberos V5 protocol and generic IP on IPv4 or IPv6 networks. Key-generation protocols enable default security settings when cryptographic parameters remain unspecified or cannot be agreed upon [38].

1.2.7 CCM Mode (Counter with CBC-MAC) mode

CCM provides a way of authenticated encryption based on the standard counter mode of encryption and the cipher block chaining message authentication code. CCM provides confidentiality and integrity of the authenticated ciphertext, and is commonly used in any application where IEEE 802.15.4 and the Constrained Application Protocol (CoAP) are often used. [39]. CCM utilizes counter mode encryption and the CBC-MAC mode of authentication. Note that counter mode can be parallelizable, so only the MAC, its associated data, and the plaintext will be processed sequentially. The MAC is calculated over the nonce, associated data, and the plaintext prior to decryption to protect against decryption of a maliciously authenticated ciphertext. The ciphertext will be accompanied by the tag and remaining fields, including a cleared version of the nonce. The length of the tag can be between four byes and 16 bytes, and shorter versions will tend to have weaker security. Common applications would include the various IEEE 802.15.4 standards or any application using the Constrained Application Protocol (CoAP).[40], [41], [42].

1.2.8 GCM Mode

GCM is a widely adopted mode for commercial and governmental protocols such as TLS, SSH, IPsec, and 5G. Its popularity indicates both importance in contemporary information security and efficiency of design. The GCM authentication scheme is at first blush an uncomplicated method of authenticated encryption. The mode engages counter mode for encryption, ensuring each cipher text block is tagged uniquely, then checks for integrity through the GHASH function . The GCM and core encryption stages after the initial counter mode preparation illustrate how previous counter encryption is subject to tag calculation. The provided guidance on nonce handling addresses the very serious challenges around nonce reuse and pairs it with an exceedingly simple - although easy to do incorrectly - encryption scheme. Because GCM can compute operations concurrently to a modest degree, data administration operations can occur concurrently with counter-based operations. This parallel encoding and decryption mechanism will be beneficial for high-throughput applications in both hardware and software. [43] [44] [42]

Although symmetric encryption algorithms have been extensively studied, there are still several open questions. Much of the literature has focused on traditional modes of operation (such as CBC, CFB, and CTR). Research studies on new modes of operation or hybrids, such as EAX mode, GCM and CCM, have been few and far between. In addition, published studies have not compared multiple encryption types under the same experimental conditions and produced inconsistent results on performance and efficiency.

In addition, encryption time scalability with increasing file size has not been systematically investigated, making it difficult to predict performance in large-scale or cloud environments. The balance between security strength and computational efficiency also remains an open challenge. Finally, there is a lack of comprehensive comparative studies on lightweight encryption techniques suitable for IoT and real-time applications that require minimal latency and high efficiency.

2. Related Works

Recent developments in networked systems and the Internet of Things (IoT) have prioritized lightweight cryptography as traditional encryption algorithms can impose high computational overheads on resource-constrained environments. Many research papers studied the potential to simultaneously consider security strength and time in these settings by optimizing encryption mechanisms or implementing new operation modes encryption devices.

AES block cipher using ECM, CBC, OFB, and CFB modes is analyzed in order to differentiate properties of the cipher text and assess the complexity of block construction for cipher text schemes through the periodic regularities method. The characteristics of four block modes of operation is considered within two analyses: the first analysis defined periodicity concerning cipher text, while the second involved the principle of repeated cipher iterations responding to the properties of cipher text using specific control input data. Results of the analysis of cipher text patterns in relationship to blocks and encryption iterations, as well as in derived formulas, to establish periodicity with respect to blocks and encryption iterations. The derived formulas present additional complexity to understanding patterns and trends within the sequence of cipher text derived from the first iteration of encryption for each mode presented with various key sizes [37].

A particularly interesting paper introduced a new operational mode which alternates between multiple ciphers using symmetric keys in a structured, pre-defined order to take advantage of increasing efficiency in encryption and decryption while still maintaining a high level of security. The paper described patterned cipher block (PCB) operations which also included integrity checks to assess for decrypted and subsequently encrypted ciphertexts, and a two-round handshaking protocol for the efficient exchange of key and pattern information. The design combines optimized pattern techniques to improve cryptographic performance and safety in other communication methods from delay-sensitive messaging contexts to high-security networks. [45]

The selection of which one to use depends on several variables, with the most important being speed, desired level of security, and the type of application used in the system. The findings also make us think that the CBC mode has the most desirable speed regardless of key size since it is superior to any other mode of operation. The study hypothesizes that performance testing of AES is a critical step to improve encryption design efficiency. The performance test would help end-users and system designers determine which application settings to use to maximize AES performance, efficiency, and security. Another mechanism for encrypting and decrypting different sizes of files using multiple keys generated is introduced.[46]

The study also investigates the impact of using multiple keys on symmetric algorithm encryption's security and performance levels. The multiple keys were already created by using cryptographic hash functions, MD5 and SHA-2 [47]. The algorithms to be used in the testing were Blowfish and CAST, and both were tested in three modes of operation: CBC, CFB, and CTR. The results of the tests show Blowfish, with the produced multiple keys, worked best using CBC mode, although Blowfish encrypted and decrypted slowly. Using Blowfish with both CBC and CTR modes yielded the greatest throughput, and then Blowfish. [48]

In [49], the authors proposed a tree-CBC (TCBC) block encryption method, in which firstly, preprocesses the smart grid plaintext based on the data's security level. Second, it restructures the CBC mode’s chain structure into a tree structure, where data blocks with a higher security level are placed deeper in the encryption tree. Third, since the overall security and encryption time of data encrypted with TCBC depend on the number of forks in the encryption tree. The authors also compared their scheme to standard modes of encryption: CFB, OFB, CTR, and PCBC.

The paper presents an atomic level of cryptanalysis for AES in the encryption of blocks and multimedia data. The analysis is conducted on modes of AES, i.e. CBC, ECB, CFB and OFB modes. Several analyses is reported and only for passwords and block encryption, as multimedia data such as audio, videos and images. Foremother, we study its runtimes by comparing it with the tested algorithm in the same hardware [18].

The paper discusses the modeling of four AES modes: ECB, CFB, OFB, and CBC, focusing instead on the performance of the mentioned modes in Cryptool2. The paper employs a modeling approach to investigate the modes of the Advanced Encryption Standard (AES) algorithm. The methodology focuses on tracking the content of cryptographic transformations at each stage to enhance understanding of the algorithms. The research introduces two analytical principles: periodicity of ciphertext and repeated cipher iterations, It compares the characteristic properties of ciphertext across these modes, highlighting their differences [50].

Recent studies have investigated the use of chaotic systems and frequency-domain transformations to enhance processing capabilities for block cipher modes in image encryption. In fact, a recently developed and improved block cipher image cryptosystem is based on three block cipher authentication modes (CBC, CFB, OFB) combined with bit-reversals and the two-dimensional (2D) chaotic Baker map in the Fourier domain (DFT). This hybrid approach aims to improve the diffusion and confusion properties of block ciphers to increase resilience against statistical and differential attacks. In this way, this work adds to emerging literature that begins to highlight the combination of chaos theory and existing cryptographic schemes as a promising option for creating strong yet efficient image encryption methods [51].

Despite numerous studies on AES and other symmetric algorithms, most existing work has focused on traditional operation modes such as CBC and CTR. Very few have analyzed mode EAX or compared multiple encryption methods under identical conditions. Therefore, this research addresses these gaps by proposing a comparative framework to evaluate encryption performance using the EAX mode with three distinct algorithms, which are AES, Blowfish and CAST. In addition, modes of operation GCM, CCM and EAX with AES.

3. Proposed Work

The proposed work aims to evaluate and compare the performance of block cipher encryption algorithms under different modes of operation, through two main phases that represent the core of the proposed analytical study. In the first phase of the proposed work, the focus is on the AES algorithm to study the effect of changing modes of operating on overall performance, considering that it is the fastest and most secure. Three different modes of operation CCM, GCM, and EAX are selected to be applied to the same set of data size using the same encryption keys.

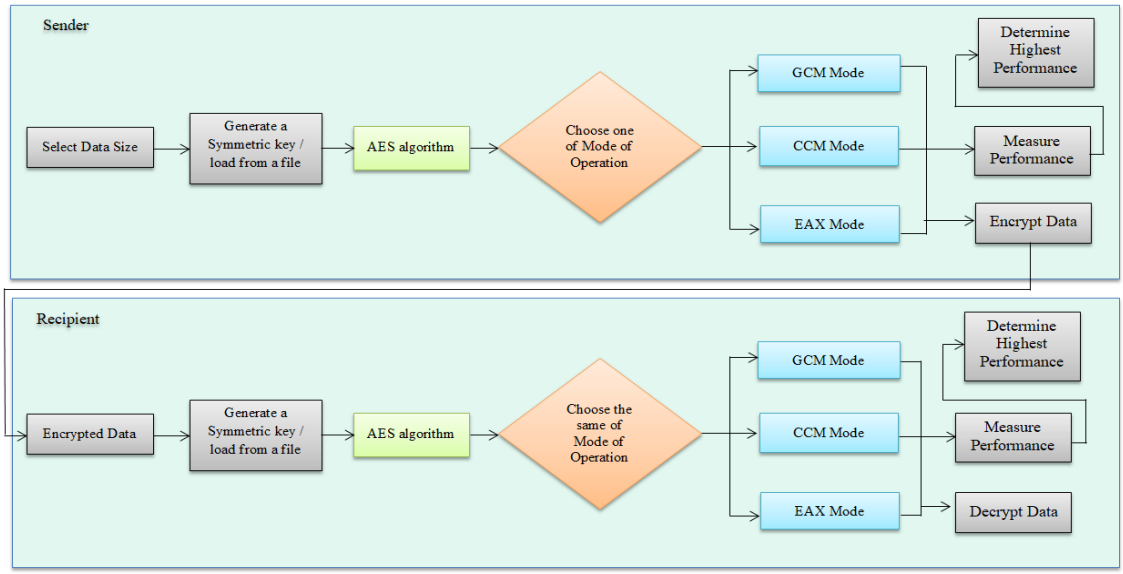

Fig. 1: Flow Diagram of the First Phase of proposed Work (AES with GCM,CCM and EAX mode)

The Second phase: Comparison of three algorithms under EAX operating mode. Three different algorithms (AES, CAST, and Blowfish) is tested using EAX operating mode. Different data sizes in a set of files are encrypted and decrypted to measure the actual performance of each algorithm. Figures 1 and 2 explain the steps of two phases of the proposed work.

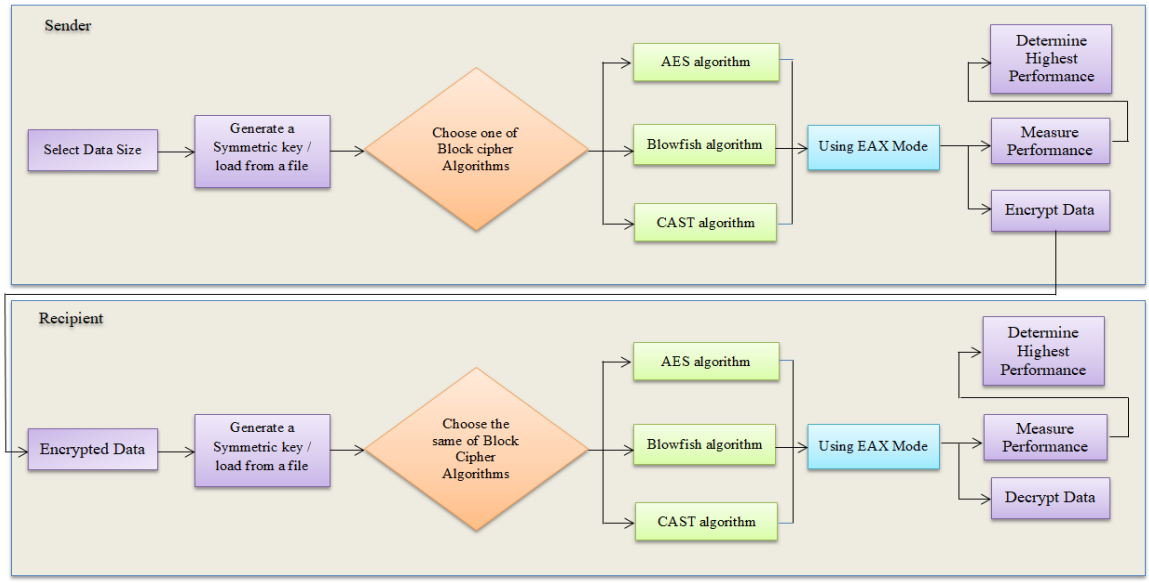

Fig. 2: Flow Diagram of the Second Phase of proposed Work (EAX mode for AES ,Blowfish, CAST)

To evaluating performance of the modes, four metrics is selected in our proposed work, these are encryption time, decryption time, encryption throughput and decryption throughput. The selected Modes are integrated the authentication and encryption process. Thus they are ensure the confidentiality and authentication. As shown in the general codes for using these modes:

cipher_enc = Algorithm.new(key, AESAlgorithm.MODE_(EAX,CCM,GCM)

The above code only to choose the mode and without addition word to mention the authentication. While in the next code is added tag that indicates to verify the integrity.

Ciphertext.with.tag = encrypt_AES.Mode_CCM (plaintext, nonce, AAD)

AES algorithm.mode_CCM (an encryption scheme that integrated encryption and authentication)[52].

nonce: Produced random value.
plaintext: The initial text that will be encrypted.
AAD: refers to Associated Authenticated Data which utilizes for authentication process only and not encrypted.
Output: ciphertext_and_tag = cipher text + add the authentication tag that confirms data integrity.
GCM Function: It mainly archives encryption process using Counter Mode (CTR). It adds an integrity verification layer using Galois Field Authentication. The result is secure and authenticated encryption [41].
ciphertext, tag = cipher_enc.encrypt_and_digest(data)
Explanation: encrypt_and_digest():
Encrypts the data, then it calculates an authentication tag that ensures the integrity of the ciphertext.
Output: ciphertext: The resulting ciphertext.
tag: The authentication tag used later during decryption to verify that the data has not been modified.
decrypted_data = cipher_dec.decrypt_and_verify(ciphertext, tag)
Explanation: decrypt_and_verify():
Decrypts the ciphertext, then verifies the authentication tag.
If the tag matches, it means that: The data has not been modified. The key and nonce are correct.
If they do not match, an error will be thrown, indicating that the data has been corrupted or tampered with.

Integrity / Forgery Bound:

If the tag length is t bits, then the probability of successfully forging a single tag is:[53],[40],[54],[55]

$$P_{forgery} \leq 2^{-t} \quad (1)$$

When performing independent forgery attempts:
Practical example :
t = 128 bit
q = 100

$$P_{total} \leq \frac{q}{2^t} \quad (2)$$

$$P_{forgery} \leq \frac{100}{2^{128}} \quad (3)$$

$$P_{total} \approx 100 \times 2^{-128} \quad (4)$$

This is a statistically negligible probability.
The following algorithms represent the steps of the proposed work and evaluating performance of studies algorithms with the GCM, CCM, EAX modes.
Pseudocode 1 and 2 illustrate pseudocode encryption and decryption time for AES algorithm in three modes of operation GCM,CCM and EAX.

Pseudocode 1: Encryption Time of three modes GCM,CCM and EAX

```
Input: FileSet = {F1, F2, F3, F4, F5}    // Data of different sizes  : 100kb, 500kb, 1MB, 3MB, 5MB
      Modes = {GCM, CCM, EAX}     // Three modes selected
Output: EncryptionTime[Mode][FileSize]   // Array for storing Encryption times

Begin

    Initialize all timers and performance metrics
    For each Mode in Modes do
       For each File in FileSet do
          Load Original_File(File)
          Start_Timer()
          Encrypted_Data ← Encrypt(Method1, Original_File, Mode)
          Stop_Timer()
          Time ← Calculate_Elapsed_Time()
          Store EncryptionTime[Mode][FileSize(File)] = Time
          Save(Encrypted_Data, Mode, FileSize(File))
       End For
    End For

   //Evaluating Performance
   For each Mode in Modes do
      Plot(FileSize, EncryptionTime[Mode])

   End For
      Display ("encryption performance For Each Modes and Best Mode")
End
```

Pseudocode 2: Decryption Time of three modes GCM,CCM and EAX of AES Algorithm

```
Input: FileSet = {F1, F2, F3, F4, F5}   // Data of different sizes  : 100kb, 500kb, 1MB, 3MB, 5MB
       Modes = {GCM, CCM, EAX}   // Three modes
Output: DecryptionTime[Mode][FileSize] // Array for storing decryption times
Begin
  Initialize all timers and performance metrics
  For each Mode in Modes do
    For each File in FileSet do
      Load Encrypted_File(File, Mode)
      Start_Timer()
      Decrypted_Data ← Decrypt(AES algorithm, Encrypted_File, Mode)
      Stop_Timer()
      Time ← Calculate_Elapsed_Time()
      Store DecryptionTime[Mode][FileSize(File)] = Time
       Verify Integrity(Decrypted_Data)  // Data verification after decryption
    End For
  End For
  Compare DecryptionTime among Modes
  Display("Best decryption performance achieved by: ", Best_Mode)
End
```

Algorithm 1 and 2 display the steps of the calculating encryption and decryption throughput of the three modes

Algorithm 1: Encryption Throughput of AES Algorithm with GCM, CCM and EAX Modes

```
Input:  File_Set = {F1, F2, ..., Fn},
        Key_Set = {K1, K2, K3},
        Method =  AES ,          Mode = { GCM, CCM, EAX }
Output: Encryption Throughput for each Mode and File Size
1. Initialize encryption parameters and timers.
2. For each file Fi in File_Set do:
   a. Measure FileSize = size(Fi).
   b. For each encryption Mode Mi ϵ { GCM, CCM, EAX }do:
        i.   Select key Ki from Key_Set according to Mi.
        ii.  Start timer T_start.
        iii. Encrypt Fi using AES algorithm with Ki under Mode Mi.
        iv.  Stop timer T_end.
        v.   Compute EncryptionTime = T_end - T_start.
        vi.  Compute Encryption Throughput = File Size / Encryption Time.
        vii. Store throughput result for Mi and Fi.
3. Plot Encryption Throughput (MB/s) versus File Size (MB) for all algorithms.
4. Compare the performance of each algorithm.
5. Output the highest-performing AES with selected mode as the optimal encryption configuration.
```

Algorithm 2: Decryption Throughput of AES Algorithm with GCM, CCM and EAX Modes

```
Input: Encrypted_Files = {E1, E2, ..., En},
Key_Set = {K1, K2, K3},
Method = AES , Mode = { GCM, CCM, EAX }
Output: Decryption Throughput for each mode
1. Initialize decryption parameters and timers.
2. For each encrypted file Ei in Encrypted_Files do:
   a. Measure FileSize = size(Ei).
   b. For each decryption Mode Mi { GCM, CCM, EAX }do:
        i.   Select the corresponding decryption key Ki from Key_Set.
        ii.  Start timer T_start.
        iii. Decrypt Ei using AES algorithm with Ki under Mode Mi.
        iv.  Stop timer T_end.
        v.   Compute DecryptionTime = T_end - T_start.
        vi.  Compute Decryption Throughput = File Size / Decryption Time.
        vii. Store throughput result for Mi and Ei.
3. Plot Decryption Throughput (MB/s) versus File Size (MB) for all algorithms.
4. Compare the performance of each algorithm.
5. Output the highest-performing AES with selected mode as the optimal decryption configuration.
```

In the second phase, the third mode EAX, which performed the least among the three, will be tested with more than one block cipher algorithm. Here, three algorithms are chosen: AES, Blowfish and CAST. Pseudocode 3 and 4 depict the steps of the calculating encryption and decryption time.

Pseudocode 3: Encryption Time of EAX Mode for AES, Blowfish and CAST Algorithms

```
Input: FileSizes = [0.1, 0.5, 1, 3, 5] MB
      Algorithms = [AES, Blowfish, CAST]
      Mode = "EAX"
Output: Encrypted data, Plot of Encryption Time vs data Size
Begin
   For each method in Algorithms do
       For each size in FileSizes do
            File ← Generate Or LoadFile(size)
             StartTime ← GetCurrentTime()
             EncryptedFile ← Enecrypt(File, algorithm, Mode_EAX)
             EndTime ← GetCurrentTime()
             EncryptionTime ← EndTime - StartTime
             Store(Algorithm, size, DecryptionTime)
        End For
   End For
```

Pseudocode 4: Decryption Time of EAX Mode for AES, Blowfish and CAST Algorithms

```
Input: FileSizes = [0.1, 0.5, 1, 3, 5] MB
      Algorithms = [AES, Blowfish, CAST]
      Mode = "EAX"
Output: decrypted data, Plot of DecryptionTime vs dataSize
Begin
   For each method in Algorithms do
       For each size in FileSizes do
            File ←LoadFile(size)
            # from Previous pseudocode,  EncryptedFile ← Encrypt(File, method, Mode)
            StartTime ← GetCurrentTime()
            DecryptedFile ← Decrypt(EncryptedFile, method, Mode)
            EndTime ← GetCurrentTime()
            DecryptionTime ← EndTime - StartTime
            Store(Method, size, DecryptionTime)
        End For
   End For
   Plot(DecryptionTime vs FileSize for each Method)
End
```

Algorithms 3 and 4 explain steps of computing encryption and decryption throughput of AES, Blowfish and CAST algorithms in EAX mode

Algorithm 3: Encryption Throughput of EAX Mode for AES, Blowfish and CAST Algorithms

```
Input:  File_Set = {F1, F2, ..., Fn},
        Key_Set = {K1, K2, K3},
        Method = { AES, Blowfish, CAST }
        Mode = EAX
Output: Encryption Throughput for each Algorithm and File Size
1. Initialize encryption parameters and timers.
2. For each file Fi in File_Set do:
   a. Measure FileSize = size(Fi).
   b. For each encryption method Mi ∈ { AES, Blowfish, CAST } do:
        i.   Select key Ki from Key_Set according to Mi.
        ii.  Start timer T_start.
        iii. Encrypt Fi using Mi with Ki under Mode EAX.
        iv.  Stop timer T_end.
        v.   Compute EncryptionTime = T_end - T_start.
        vi.  Compute Encryption Throughput = File Size / Encryption Time.
        vii. Store throughput result for Mi and Fi.
3. Plot Encryption Throughput (MB/s) versus File Size (MB) for all algorithms.
4. Compare the performance of each algorithm.
5. Output the highest-performing algorithm as the optimal encryption configuration.
```

Algorithm 4 : Decryption Throughput of EAX Mode for AES, Blowfish and CAST Algorithms

Input: Encrypted_Files = {E1, E2, …., En}
Key_Set = {K1, K2, K3},
Method = { AES, Blowfish, CAST }
Mode = EAX
Output: Decryption Throughput for each Algorithm and File Size
1. Initialize decryption parameters and timers.
2. For each encrypted file Ei in Encrypted_Files do:
   a. Measure FileSize = size(Ei).
   b. For each decryption Method Mi ∈ { AES, Blowfish, CAST}do:
      i. Select the corresponding decryption key Ki from Key_Set.
      ii. Start timer T_start.
      iii. Decrypt Ei using AES algorithm with Ki under Mode EAX.
      iv. Stop timer T_end.
      v. Compute DecryptionTime = T_end - T_start.
      vi. Compute Decryption Throughput = File Size / Decryption Time.
      vii. Store throughput result for Mi and Ei.
3. Plot Decryption Throughput (MB/s) versus File Size (MB) for all algorithms.
4. Compare the performance of each algorithm.
5. Output the most efficient decryption algorithm for Mode EAX.

As pertains to the proposed work specified in this section's detailed description concerning encryption algorithms (including different modes of operation) and their ability to increase the effectiveness of data protection using operations in sequence, there is a need for practical evidence demonstrating how well the proposed project actually works via experimental tests. Therefore, the next section includes a description of the working conditions where the proposed algorithm/test setup is implemented, including both hardware and software configurations, along with the metrics used to assess their performance. As such, this section will act as a preliminary foundation from which all analysis will be conducted on the experimental outputs that will be detailed in subsequent sections; specifically: encryption time, decryption time, throughput measurements and overall purpose relative to evaluation of the overall effectiveness of the proposed work, as compared against all other relevant approaches.

4. Experimental Setup

4.1 Programming Environment

The results of implementing the proposed work depend on the software and the hardware used. A laptop with core Intel i7 processor and 16GB of RAM will be used. Operating system is Windows 11 Professional. In order to facilitate Python development, PyCharm will serve as the IDE for writing code and implementation of the algorithm. The testing framework consists of both proprietary libraries and free open source libraries to allow for accurate measurement of the results of the proposed solution. The implementation was conducted using the PyCryptodome cryptographic library in Python.
With the programming environment and tools used now fully defined, we move on to presenting and analyzing the results, to evaluate the performance of the proposed methods and discuss their impact according to the specified experimental criteria, reflecting the effectiveness of the methodology used.

4.2 Experimental Configuration

- Algorithm: AES-128
- Modes: EAX, GCM, CCM
- Message: Fixed plaintext sample,
- AAD: Static associated data
- Trials per mode: 100
- Warm-up rounds: 10

4.3 Attack Model: Tag Forgery

In this attack scenario:
During process of decryption message is used ‘ decrypt_and_verify()’. If verification yields as false (ValueError), an attack would be detected.
Mathematically:
If $Tag' \neq Tag$ then: Reject = Verify (Ciphertext, Tag')

4.2 Evaluation Metrics

4.2.1 Performance Metrics

- Encryption Time :The length of time to change the original document into a secure version using the correct key.
- Decryption Time: The amount of time it takes to restore the original document when it is encrypted.
- Throughput: The rate at which data can be processed by the encryption process (usually in B/s, MB/s).

4.2.2 Security Evaluation

Two primary metrics were used:

- **Failure Rate (%)**

$$\text{Failure Rate} = \frac{\text{Detected Forgeries}}{\text{Total Trails}} \times 100$$

Represents how often forged tags are successfully detected.

- **Detection Time (µs)**

Measured using high-resolution performance counters to evaluate:

- Average verification rejection time
- Standard deviation

5. Results and Discussions

This section will present the experimental results of the proposed work, along with a discussion.

5.1 Performance evaluation

5.1.1 Experimental Results of the First Phase of the Proposed Work

In this phase, encryption time and decryption performance are measured, as detailed below for AES algorithm with three modes GCM, CCM and EAX.

- **Encryption Time Analysis of AES algorithm for AES in GCM, CCM, EAX mode**

Figures 3 illustrates the encryption time of AES algorithm with the three modes of operation GCM, CCM and EAX, across varying data sizes (0.1- 5) MB.
The encryption time follows the same fluidity and patterns. The time taken seems to increase almost linearly with an increase in file size.
• GCM Mode is always the fastest, then followed by the CCM Mode, then the final Mode is EAX.
• GCM Mode is the best for the encryption process and the slowest mode is EAX.

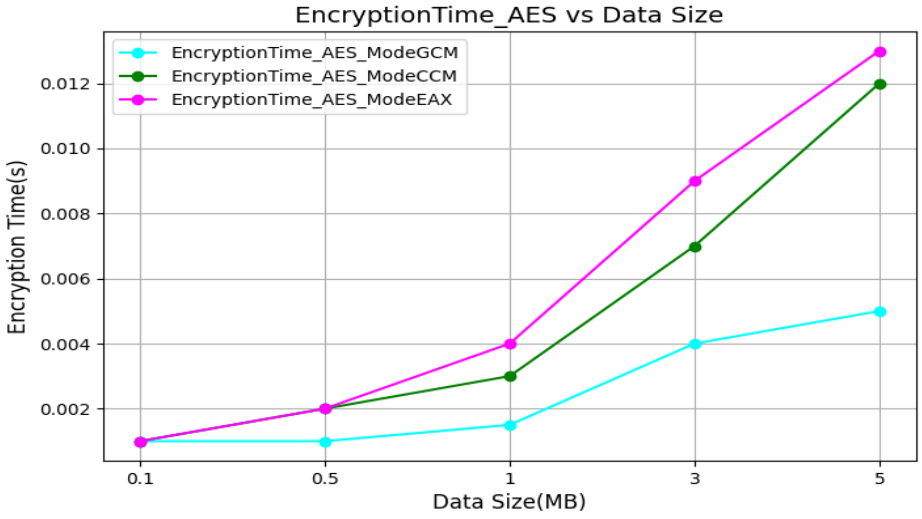

Fig. 3 Encryption Time of AES in GCM,CCM and EAX Mode

- **Decryption Time Analysis of AES algorithm for AES in GCM, CCM, EAX mode**

Figure 4 illustrates the decryption time of AES algorithm with the three modes of operation GCM, CCM and EAX, across varying data sizes (0.1- 5) MB.

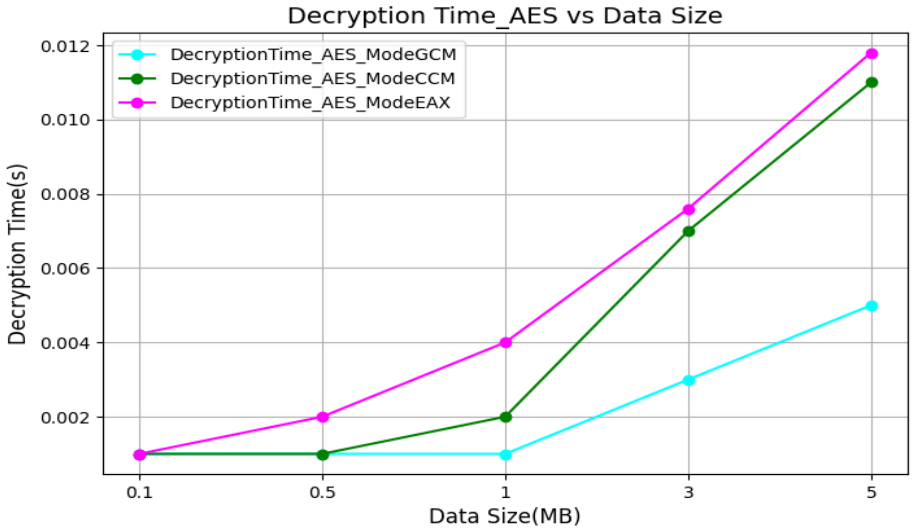

Fig. 4 Decryption Time of AES in GCM,CCM and EAX Mode

It's apparent that the time required to decrypt increases with file size across each of the three modes. Mode GCM is consistently the fastest for all sizes, followed by CCM, and Mode EAX also is the slowest. Hence, Mode GCM is the most efficient for speed during board's decryption.

Table 1: Comparing between three modes for (encryption vs. decryption) time

Aspect:	Encryption	Decryption
Overall Time:	Convergent values for both operations	very convergent as well
Fastest Mode:	GCM mode in both cases	GCM mode in both cases
Slowest Mode:	Always EAX mode	Always EAX mode
Gradual Behavior:	Semi-linear as the size of file increases	semi-linear as well

It's obvious that the time taken for both encryption and decryption is nearly directly related, suggesting that the AES is well balanced in each direction.
All three modes are equally affected by file size, but Mode GCM has the highest performance by time for both encryption and decryption.

Final Summery : GCM mode is the best. GCM mode has the lowest time for encryption and decryption across all file sizes. Mode GCM shows more consistent performance than the other modes tested with increasing file size.
Thus, one could argue that mode GCM performs the best if the application requires speed and performance efficiency, but the other modes could potentially be more secure (depending on if the time differences are related to an increase in process complexity). Thus, in terms of time performance only, the clear GCM mode is the best.

- **Encryption Throughput Analysis of AES in GCM, CCM and EAX mode**

The performance analysis in Figure 5 shows that encryption throughput for all three modes is related to the data size and steadily increases in throughput with data size. This means that with increasing data size the system works more efficiently for large data sizes, possibly cause the initial and static operations have less impact overall with larger data.

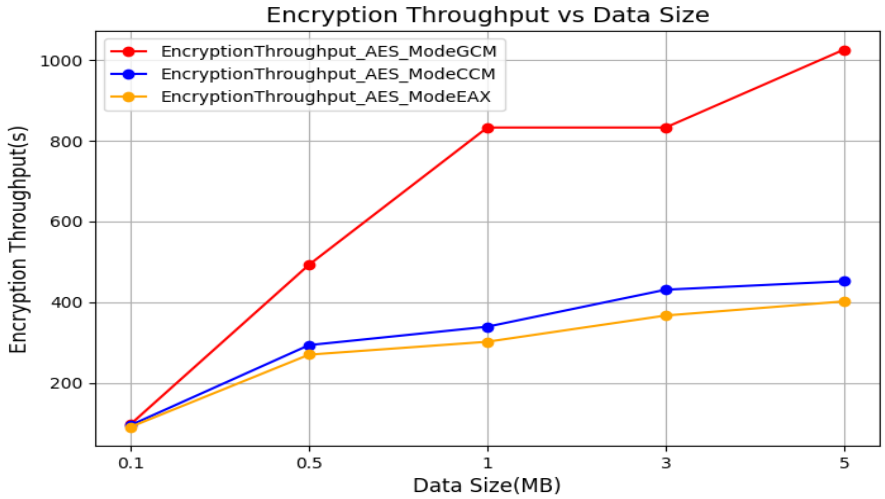

Fig. 5 Encryption Throughput of AES in GCM,CCM and EAX Mode

- Mode GCM is the best by a significant margin, reaching speeds of over 1000 MB/s at a size of 5 MB, a substantial improvement compared to modes in Figures 5 and 6.
- Modes CCM and EAX had nearly the same performance. However, the results for Mode CCM are slightly better than Mode EAX in most measures and Mode CCM had a higher average speed.
- The differences in speed between restated become even more apparent as file size increased, with mode GCM also appearing to be the most consistent and efficient with larger files.

- **Decryption Throughput Analysis of AES in GCM, CCM and EAX**

Figure 6 shows that encryption throughput for all three modes:

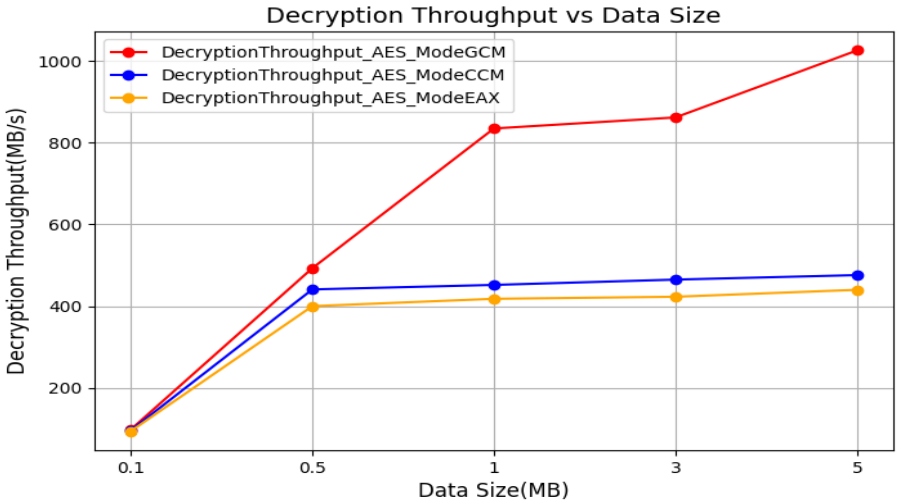

Fig. 6 Decryption Throughput of AES in GCM,CCM and EAX Mode

The decryption curves look and behave much like the encryption curves; we see performance increase with file size, and the mode rankings maintain their classification order in terms of efficiency. In addition, Mode GCM performed better in terms of transfer speed, indicating that only Mode GCM encrypts and decrypts efficiently. CCM and EAX Modes are similar, but mode CCM performed slightly better across all file sizes. The proposed system, in fact, provides a good balance between encryption and decryption, with no significant lag in either direction. Furthermore, GCM mode encryption and decryption curves are very similar, indicating a good performance for both CCM Mode and EAX Mode have differences in encryption and decryption. These results are expected. The way the work in the calculation is nearly identical and takes approximately the same amount of time.

AES algorithm in GCM Mode. It is fast in both directions, with the time impact on larger file processing greatly reduced. If a fast system is necessary, this could be a good choice for large files. Mode CCM is a better option where stability is more important to the user than maximum speed.

Table 2: Comparing between three modes for (encryption vs. decryption) throughput

Standard	GCM Mode	CCM Mode	EAX Mode
Throughput Encryption	Highest & Most Stable	Medium	Lowest
Throughput Decryption	Highest	Medium	Lowest
Trend with Increasing Volume	Best in efficiency and speed	Good Performance	Acceptable or not bad Performance

5.1.2 Experimental Results of the Second Phase Analysis Study

In this phase, encryption time and decryption performance are measured, as detailed below

- **Encryption Time Analysis of EAX Mode for AES, Blowfish and CAST algorithms**

Figures 7 and 8 shows encryption time and decryption time for three algorithms AES, CAST, Blowfish with Mode EAX. In Figure 7, the relationship between encryption time (sec) and data size (MB) is represented. As shown in Figures, 7 and 8 the encryption time increases gradually with increasing across file size in all three algorithms, a normal behavior driving by larger the amount of data to be encrypted.

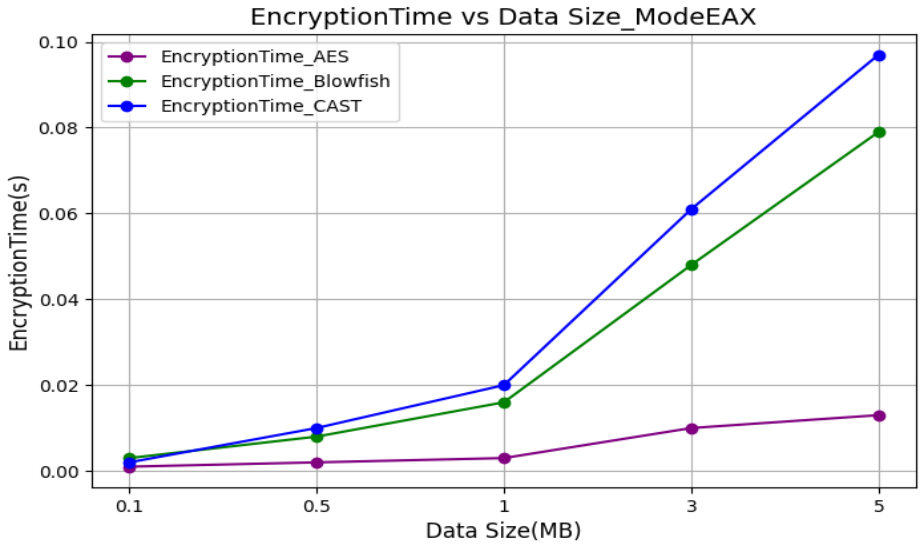

Fig. 7 Encryption Time of EAX Mode for AES, Blowfish and CAST algorithms

When comparing the three algorithms:

- AES algorithm: Achieved the lowest encryption time in all file sizes, at 5MB taking only about 0.013 seconds.
- Blowfish algorithm is faster than CAST and slower than AES (approximately 0.096 seconds at 5 MB).
- CAST algorithm: consumes highest time from the other.

Thus, AES is fastest encryption algorithm and extremely effective at encrypting data quickly.

- **Decryption Time Analysis for EAX Mode of AES, Blowfish and CAST algorithm**

Figure 8 shows the impact of file size on encryption time. The larger file sizes consume longer decryption times. As the following comparison between algorithms :

- AES algorithm: Achieved the shortest decryption time across all sizes (about 0.011 seconds at 5MB).
- Blowfish algorithm: ranked second.
- CAST algorithm: recorded the longest decryption time, exceeding 0.1 seconds at 5MB.

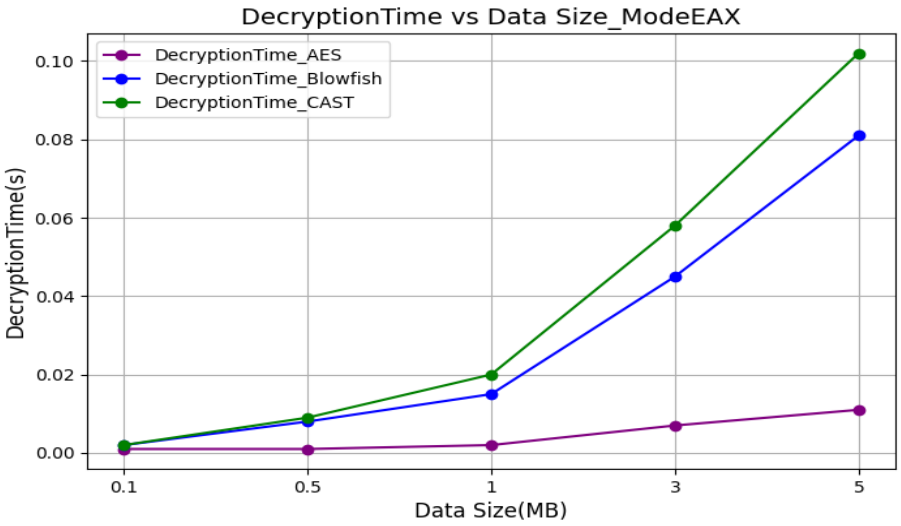

Fig. 8 Decryption Time of AES, Blowfish and CAST algorithms with EAX Mode

AES remains the most efficient decryption algorithm in terms of time.
Overall comparison between encryption and decryption formats:
In both cases, the AES algorithm consistently outperformed others in speed for both encryption and decryption. The difference in encryption and decryption times across algorithms is minimal, indicating similar efficiency in both directions. CAST demonstrates slower performance in both.

Table 3: Comparing and concluding (encryption vs. decryption) Performance of three algorithms

Standard	GCM Mode	CCM Mode	EAX Mode
Throughput Encryption	Highest & Most Stable	Medium	Lowest
Throughput Decryption	Highest	Medium	Lowest
Trend with Increasing Volume	Best in efficiency and speed	Good Performance	Acceptable or not bad Performance

Final Result:
AES algorithm is the best performance within EAX mode of operation because it has achieved the lowest time in both encryption and decryption, and is therefore best suited for applications that require high speed and efficiency in data processing.

- **Encryption Throughput Analysis for AES, Blowfish and CAST algorithms with EAX Mode**

Figure 9 demonstrates the data transfer speed associated with encryption throughput using three algorithms:

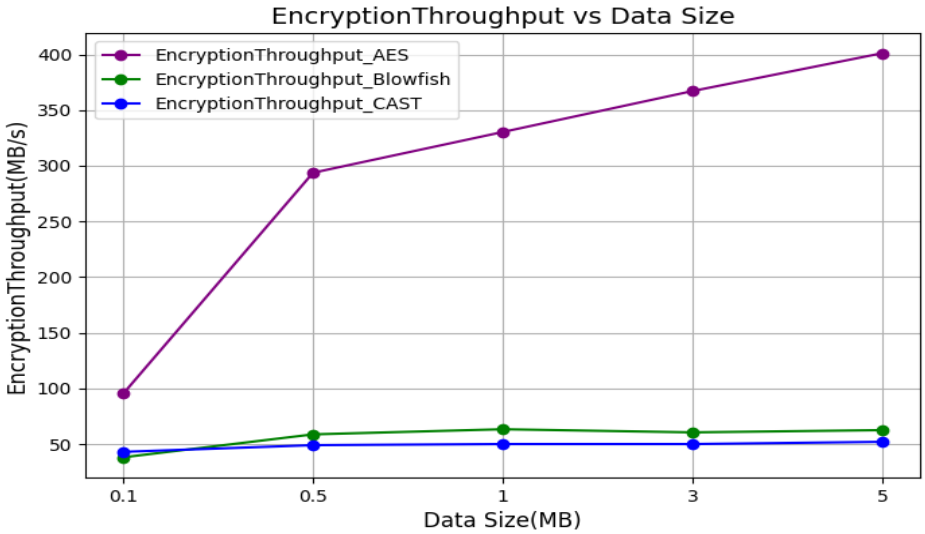

Fig. 9 Encryption Throughput of EAX Mode of AES, and CAST algorithms

- AES: This algorithm performs extremely well in comparison to the other two. The speed starts at about 95 MB/s when the file size is approximately 0.1 MB but quickly climbs to 400 MB/s at 5 MB. This indicates that the algorithm is quite efficient in encrypting files as the file size increases.

- Blowfish: This algorithm has the lowest starting performance (Blowfish approximately 45 MB/Sec) and rises steadily to about 60 MB/s at a 1 MB file size then remains relatively stable. This demonstrates that performance appears relatively constant and does not increase significantly as file size increases.
- CAST: This algorithm has the weakest performance that remains stable, ranging from ~45 to 50 MB/Sec across all file sizes. No improvement occurs as the size increases.

- **Decryption Throughput Analysis for AES, Blowfish and CAST algorithms with EAX Mode**

Figure 10 describes the decryption throughput, or data transfer speed, for the same three algorithms.

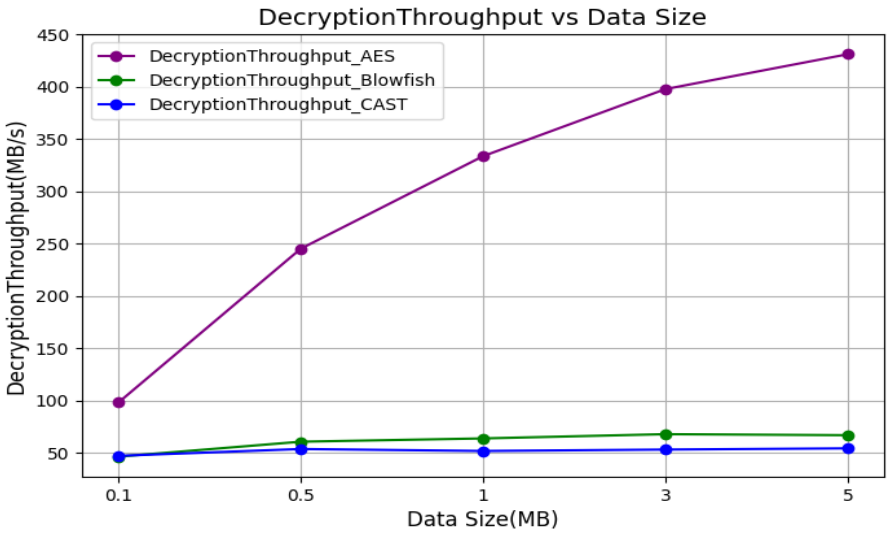

Fig. 10 Decryption Throughput of AES, Blowfish and CAST algorithms in EAX Mode

- AES: Again, in all testing occasions, yielded the highest average performance starting above 100 MB/s and ending at around 440 MB/s for the 5 MB file. Therefore, it came with the best, and most improved, decryption performance with the larger file sizes.
- Blowfish: Averaged approximately 40 MB/s going to around 60-65 MB/s, and then leveled off Blowfish had very similar performance behavior to that of the encryption behavior.
- CAST: Again, CAST is consistently and repeatedly the weakest at each performance test, and also leveled off at around 45-50 MB/Sec.

Table 4: Comparing between three algorithms for (encryption vs. decryption) Performance and final conclusion

Algorithm	Encryption performance	Decryption Performance	general evaluation
AES	Fastest and best	Fastest and best	Best performance
Blowfish	Medium	Medium	Balanced
CAST	Slowest	Slowest	The least efficient

Final Summary: AES is the best algorithm, reason for superiority: the highest throughput in both encryption and decryption is achieved. Its efficiency improves with increasing data size (i.e., it is suitable for large files).

5.2 Security evaluation

Evaluation process starts with the reading of text file and consider it as a plaintext message. After that, it encrypts the file using three AEAD modes in AES: EAX, GCM, and CCM. Then it performs a Tag Forgery Attack: an attempt to decrypt the file using a forged tag. It measures:
Failure Rate (%): The percentage of times the attack is detected (generating a ValueError).
Average Detection Time (μs): The average time taken to detect the forged tag in microseconds.
Std of Detection Time (μs): The standard deviation of detection times.
Expermantail results display in Figures 11, 12 and 13, and Tables 5, 6 and 7.

Figures 11,12 and 13 depict the comparisons of average tag forgery detection times for three modes GCM,CCM and EAX .

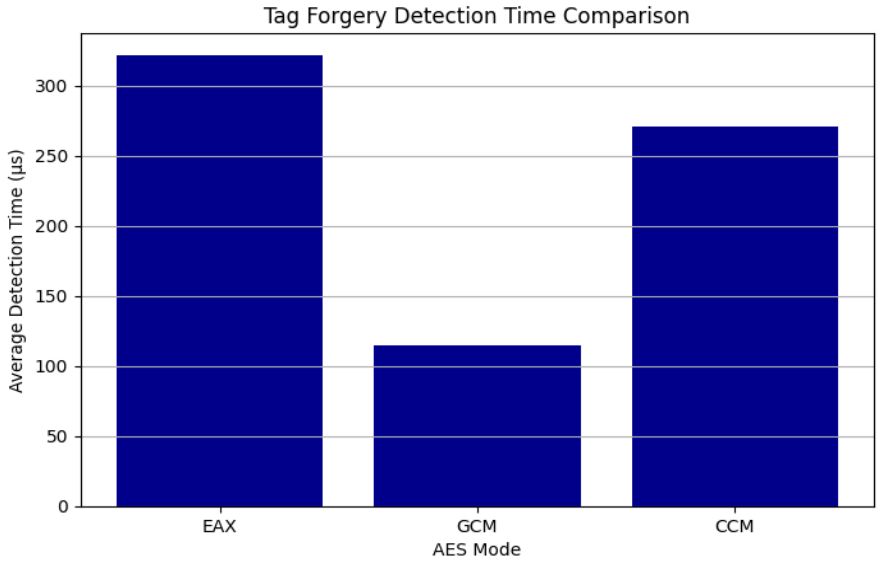

Fig. 11 Average Tag Forgery Detection Time for three modes GCM, CCM and EAX at data size 100kb

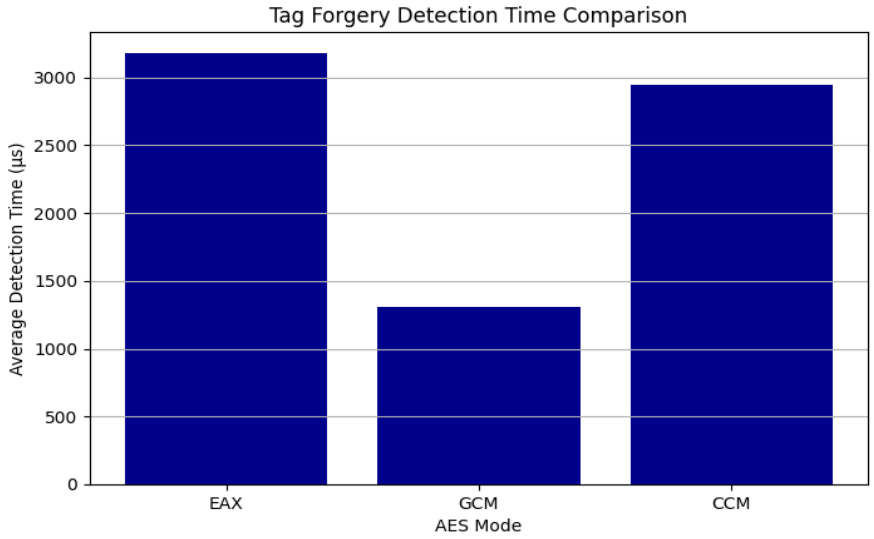

Fig. 12 Average Tag Forgery Detection Time for three modes GCM, CCM and EAX at data size 1MB

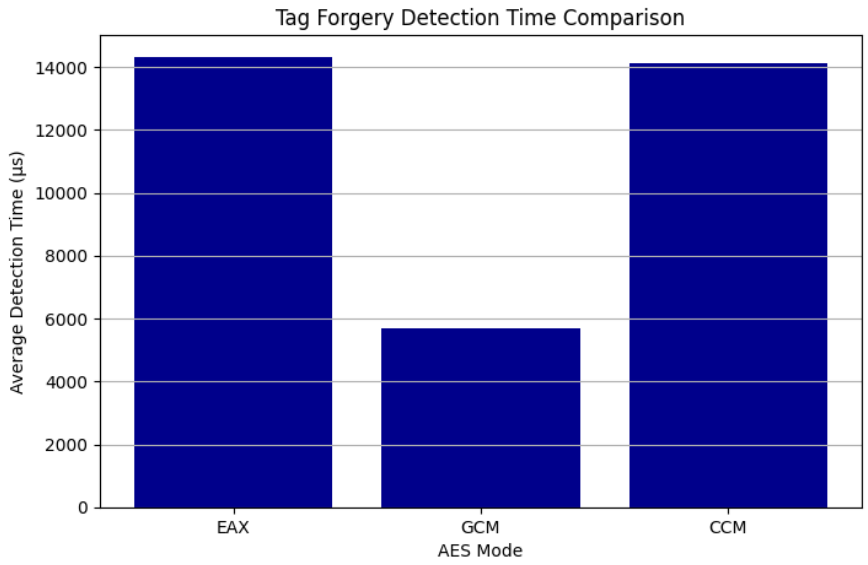

Fig. 13 Average Tag Forgery Detection Time for three modes GCM, CCM and EAX at data size 5MB

Tables (5 to7) show the security evaluation results against Tag Forgery attacks for EAX, GCM, and CCM modes at three data sizes: 100KB, 1MB, and 5MB. The analysis focuses on three indicators:

- Fail Rate % (Forgery failure rate)
- Fail Avg. (μs) (Average forgery detection time)
- Fail Std. (Standard deviation of detection time)

Table 5. GCM, CCM and EAX modes Security Evaluation - Tag Forgery (μs) at data size 100kb

Mode	Fail Rate%	Fail Avg.	Fail Std.
EAX	100.0	321.20	69.70
GCM	100.00	114.87	21.78
CCM	100.00	270.77	89.77

Table 6. GCM, CCM and EAX modes Security Evaluation - Tag Forgery (μs) at data size 1MB

Mode	Fail Rate%	Fail Avg.	Fail Std.
EAX	100.0	3177.14	821.27
GCM	100.00	1305.31	416.34
CCM	100.00	2943.28	649.19

Table 7. GCM, CCM and EAX modes Security Evaluation - Tag Forgery (μs) at data size 5MB

Mode	Fail Rate%	Fail Avg.	Fail Std.
EAX	100.0	14303.26	1440.52
GCM	100.00	5677.98	1560.38
CCM	100.00	14102.53	2278.53

First: Forgery Failure Rate (Fail Rate%)
All three modes achieved: Fail Rate = 100%
This means that all forgery attempts were successfully detected at all data sizes.
Important security implication: This confirms that all three modes achieve integrity and authenticity.
This aligns with the theoretical security characteristics of AEAD modes.
The probability of successful forgery in practice remains theoretically around $\approx 2^{128}$ (at a Tag length of 128 bits), a computationally negligible probability. Therefore, in terms of authentication strength, there is no practical difference between the three modes.

Second: Analysis of Failure Detection Time (Fail Avg.)

- At 100KB

GCM is clearly the fastest.
EAX is the slowest.
The difference is evident even at small sizes.

- At 1MB

All values increased almost linearly with increasing size.
GCM remains the fastest.
CCM slightly outperforms EAX.

- At 5MB : The gap widens.

GCM is approximately: ≈ 2.5 times faster than CCM and EAX
CCM and EAX are very close at large sizes.

Third: Standard Deviation Analysis (Fail Std.)
At small sizes: The deviation is relatively low. However, at 5MB: it increases significantly.
Notes: Fluctuation is related to CPU and memory load.
There is no security instability. CCM exhibits the highest fluctuation at large sizes.

6. Conclusion and Future Study

This study offered a structured evaluation of the proposed cryptographic method by integrating performance benchmarking with explicit security validation. The results approve linear scalability across increasing data sizes, while revealing clear efficiency differences among the evaluated AEAD modes. AES-GCM consistently achieved superior throughput and lower latency due to its parallelizable authentication structure, whereas CCM and EAX demonstrated higher computational overhead stemming from their sequential MAC constructions. From a security perspective, the conducted Tag Forgery experiments demonstrated complete resistance to adversarial manipulation, with zero successful forged-tag acceptances across all tested data sizes. The empirical results are consistent with the theoretical authentication bound of 2^{-128}, confirming that the probability of successful forgery remains computationally negligible under standard assumptions. Moreover, AES-GCM showed faster rejection latency, reinforcing its suitability for high-performance secure environments requiring rapid adversarial detection. Future research will extend the security evaluation by implementing and experimentally analyzing a Bit-Flipping attack scenario to further examine integrity preservation and error propagation

behavior under active message modification. This will provide deeper insight into robustness against manipulation at the ciphertext level and enhance the adversarial validation framework.

References

[1] N. A. A. S. Bolaji and A. B. Abubakar, "Comparative Analysis of Encryption Algorithms," *Covenant Journal of Informatics & Communication Technology*, vol. 6, no. 1, pp. 16–30, 2018, doi: 10.47672/ejt.1312.

[2] C. Tan, X. Deng, and L. Zhang, "Identification of block ciphers under CBC mode," *Procedia Computer Science*, vol. 131, pp. 65–71, 2018, doi: 10.1016/j.procs.2018.04.186.

[3] R. Yadav, "Analysis of Cryptography in Information Technology," *Interantional Journal of Scientific Research in Engineering and Management*, vol. 07, no. 03, pp. 1–6, 2023, doi: 10.55041/ijsrem18379.

[4] D. Kodzo, M. Hodowu, D. R. Korda, and E. Danso Ansong, "An Enhancement of Data Security in Cloud Computing with an Implementation of a Two-Level Cryptographic Technique, using AES and ECC Algorithm," *International Journal of Engineering Research & Technology (IJERT)*, vol. 9, no. March 2021, pp. 2278–0181, 2020.

[5] E. Jincharadze, "Critical Analysis of Some Cryptography Algorithms," *Scientific and Practical Cyber Security Journal (SPCSJ)*, vol. 1, no. 2, p. 2017, 2017.

[6] S. Srilaya and S. Velampalli, "Performance evaluation for des and AES algorithms-an comprehensive overview," *2018 3rd IEEE International Conference on Recent Trends in Electronics, Information and Communication Technology, RTEICT 2018 - Proceedings*, pp. 1264–1270, 2018, doi: 10.1109/RTEICT42901.2018.9012536.

[7] Bhavya Daya, "Network security: History, importance, and future," *University of Florida Department of Electrical and …*, p. 13, 2013.

[8] P. Williams, I. K. Dutta, H. Daoud, and M. Bayoumi, "A survey on security in internet of things with a focus on the impact of emerging technologies," *Internet of Things (Netherlands)*, vol. 19, p. 100564, 2022, doi: 10.1016/j.iot.2022.100564.

[9] S. A. Hamad and Q. Z. Sheng, "Realizing an Internet of Secure Things : A Survey on Issues and Enabling Technologies," no. X, pp. 1–21, 2020, doi: 10.1109/COMST.2020.2976075.

[10] D. Puthal, S. Nepal, R. Ranjan, and J. Chen, "DLSeF: A dynamic key-length-based efficient real-time security verification model for big data stream," *ACM Transactions on Embedded Computing Systems*, vol. 16, no. 2, 2016, doi: 10.1145/2937755.

[11] D. P. R. P. Ekta Agrawal, "A More Effective Approach Securing Text Data Based On Private Key Cryptography," *International Journal on Recent and Innovation Trends in Computing and Communication*, vol. 5, no. 3, pp. 163–168, 2017.

[12] L. Zhang and L. Wang, "A hybrid encryption approach for efficient and secure data transmission in IoT devices," *Journal of Engineering and Applied Science*, vol. 71, no. 1, pp. 1–18, 2024, doi: 10.1186/s44147-024-00459-x.

[13] V. Esther Jyothi, B. D. C. N. Prasad, and R. K. Mojjada, "Analysis of Cryptography Encryption for Network Security," *IOP Conference Series: Materials Science and Engineering*, vol. 981, no. 2, 2020, doi: 10.1088/1757-899X/981/2/022028.

[14] U. T. Pius, C. Onyebuchi, O. P. Chinasa, and E. F. Adoba, "A Cloud-Based Data Security System using Advanced Encryption (AES) and Blowfish algorithms," *Journal of Scientific and Engineering Research*, vol. 5, no. 6, pp. 59–66, 2018.

[15] S. Kumar, "Review on Network Security and Cryptography," *International Transaction of Electrical and Computer Engineers System*, vol. 3, no. 1, pp. 1–11, 2015, doi: 10.12691/iteces-3-1-1.

[16] R. Saha, G. Geetha, G. Kumar, and T. H. Kim, "RK-AES: An Improved Version of AES Using a New Key Generation Process with Random Keys," *Security and Communication Networks*, vol. 2018, 2018, doi: 10.1155/2018/9802475.

[17] P. Prajapati and K. Chaudhari, "KBC: Multiple Key Generation using Key Block Chaining," *Procedia Computer Science*, vol. 167, no. 2019, pp. 1960–1969, 2020, doi: 10.1016/j.procs.2020.03.224.

[18] M. Boussif, "On The Security of Advanced Encryption Standard (AES)," *8th International Conference on Engineering, Applied Sciences, and Technology, ICEAST 2022 - Proceedings*, pp. 83–88, 2022, doi: 10.1109/ICEAST55249.2022.9826324.

[19] M. A. Al-Shabi, "A Survey on Symmetric and Asymmetric Cryptography Algorithms in information Security," *International Journal of Scientific and Research Publications (IJSRP)*, vol. 9, no. 3, p. p8779, 2019, doi: 10.29322/ijsrp.9.03.2019.p8779.

[20] P. Patil, P. Narayankar, D. G. Narayan, and S. M. Meena, "A Comprehensive Evaluation of Cryptographic Algorithms: DES, 3DES, AES, RSA and Blowfish," *Procedia Computer Science*, vol. 78, no. December 2015, pp. 617–624, 2016, doi: 10.1016/j.procs.2016.02.108.

[21] W. Al-Nbhany and A. Zahary, "A Comparative Study among Cryptographic Algorithms: Blowfish, AES and RSA," *International Arab Conference on Information Technology*, no. May, 2016.

[22] U. Thirupalu and E. K. Reddy, "Performance Analysis of Cryptographic Algorithms in the Information Security," no. March, 2022, doi: 10.13140/RG.2.2.16273.51047.

[23] D. Bujari and E. Aribas, "Comparative Analysis Of Block Cipher Modes Of Operation," *International Advanced Researches & Engineering Congress*, no. November 2017, pp. 2–5, 2017.

[24] H. K. A. Alsuwaiedi and A. M. S. Rahma, "A new modified DES algorithm based on the development of binary encryption functions," *Journal of King Saud University - Computer and Information Sciences*, vol. 35, no. 8, p. 101716, 2023, doi: 10.1016/j.jksuci.2023.101716.

[25] M. N. Alenezi, H. Alabdulrazzaq, and N. Q. Mohammad, "Symmetric encryption algorithms: Review and evaluation study," *International Journal of Communication Networks and Information Security*, vol. 12, no. 2, pp. 256–272, 2020.

[26] M. B. Yassein, S. Aljawarneh, E. Qawasmeh, W. Mardini, and Y. Khamayseh, "Comprehensive study of symmetric key and asymmetric key encryption algorithms," *Proceedings of 2017 International Conference on Engineering and Technology, ICET 2017*, vol. 2018-Janua, pp. 1–7, 2017, doi: 10.1109/ICEngTechnol.2017.8308215.

[27] J. Zhang, X. Wang, B. Liu, J. Wang, X. Li, and L. He, "Research on symmetric encryption and decryption algorithm of shared data based on chaotic mapping and permutation," *2023 IEEE 6th International Conference on Information Systems and Computer Aided Education, ICISCAE 2023*, pp. 302–305, 2023, doi: 10.1109/ICISCAE59047.2023.10391859.

[28] H. Alabdulrazzaq and M. N. Alenezi, "Performance Evaluation of Cryptographic Algorithms: DES, 3DES, Blowfish, Twofish, and Threefish," *International Journal of Communication Networks and Information Security (IJCNIS)*, vol. 14, no. 1, 2022, doi: 10.17762/ijcnis.v14i1.5262.

[29] B. A. Buhari *et al.*, "Performance and Security Analysis of Symmetric Data Encryption Algorithms: AES, 3DES and Blowfish," *International Journal of Advanced Networking and Applications*, vol. 16, no. 04, pp. 6473–6486, 2025, doi:

10.35444/ijana.2025.16404.
[30] A. M. Qadir and N. Varol, "A review paper on cryptography," *7th International Symposium on Digital Forensics and Security, ISDFS 2019*, pp. 1–6, 2019, doi: 10.1109/ISDFS.2019.8757514.
[31] M. Imdad, S. N. Ramli, and H. Mahdin, "An Enhanced Key Schedule Algorithm of PRESENT-128 Block Cipher for Random and Non-Random Secret Keys," *Symmetry*, vol. 14, no. 3, pp. 1–22, 2022, doi: 10.3390/sym14030604.
[32] M. Ebrahim, S. Khan, and U. Bin Khalid, "Symmetric Algorithm Survey: A Comparative Analysis," vol. 61, no. 20, pp. 12–19, 2014.
[33] D. Ali, A. M. Tripathi, and K. Saini, "A Study On Network Security and Cryptography," *Proceedings - IEEE 2024 1st International Conference on Advances in Computing, Communication and Networking, ICAC2N 2024*, no. January, pp. 511–514, 2024, doi: 10.1109/ICAC2N63387.2024.10894984.
[34] U. H. Shaikh, M. M. Abbas, S. A. Lahad, M. Razi, and M. Shaikh, "A Comparative Survey of Symmetric and Asymmetric Key Cryptography Algorithms," *2nd International Multidisciplinary Conference on Emerging Trends in Engineering Technology-2024 (2nd IMCEET-2024)*, no. October, pp. 257–262, 2024.
[35] V. Kumar, A. Sharma, I. Ntroduction, and I. J.- August, "A Survey on Various Cryptography Techniques," vol. 3, no. 4, pp. 307–312, 2014.
[36] A. Fenyi, J. G. Davis, and K. Riverson, "Comparative Analysis of Advanced Encryption Standard , Blowfish and Rivest Cipher 4 Algorithms," *International Journal of Innovative Research and Development*, vol. 3, no. 11, pp. 384–392, 2014.
[37] Z. Alimzhanova, D. Nazarbayev, A. Ayashova, and A. Kaliyeva, "Analysis of Ciphertext Behaviour Using the Example of the AES Block Cipher in ECB, CBC, OFB and CFB Modes of Operation, Using Multiple Encryption," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 13758 LNAI, pp. 621–629, 2022, doi: 10.1007/978-3-031-21967-2_50.
[38] M. M. Parelkar and K. Gaj, "Implementation of EAX mode of operation for FPGA bitstream encryption and authentication," *Proceedings - 2005 IEEE International Conference on Field Programmable Technology*, vol. 2005, pp. 335–336, 2005, doi: 10.1109/FPT.2005.1568588.
[39] E. López-Trejo, F. Rodríguez-Henríquez, and A. Díaz-Pérez, "An FPGA implementation of CCM mode using AES," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 3935 LNCS, no. December 2005, pp. 322–334, 2006, doi: 10.1007/11734727_26.
[40] J. Katz and Y. Lindell, *INTRODUCTION TO MODERN CRYPTOGRAPHY: Second Edition*. 2014. doi: 10.1201/b17668.
[41] N. Mouha and M. Dworkin, *Report on the Block Cipher Modes of Operation in the NIST SP 800-38 Series*. 2024.
[42] M. Dworkin, "Recommendation for block cipher modes of operation: the CCM mode for authentication and confidentiality," *NIST Special Publication 800-38C*, 2007.
[43] F. Piper and S. Murphy, *Understanding cryptography*. 2013. doi: 10.1093/actrade/9780192803153.003.0002.
[44] A. Hamza and B. Kumar, "A Review Paper on DES, AES, RSA Encryption Standards," *Proceedings of the 2020 9th International Conference on System Modeling and Advancement in Research Trends, SMART 2020*, pp. 333–338, 2020, doi: 10.1109/SMART50582.2020.9336800.
[45] S. Oh, S. Park, and H. Kim, "Patterned Cipher Block for Low-Latency Secure Communication," *IEEE Access*, vol. 8, pp. 44632–44642, 2020, doi: 10.1109/ACCESS.2020.2977953.
[46] M. N. Alenezi, H. Alabdulrazzaq, H. M. Alhatlani, and F. A. Alobaid, "On the performance of AES algorithm variants," *International Journal of Information and Computer Security*, vol. 23, no. 3, pp. 322–337, 2024, doi: 10.1504/IJICS.2024.138494.
[47] B. J. M. Jawad and S. Al-Alak, "DSK: The Proposed Method for Deriving Secret Keys for encrypting of the block in the networks," *2023 1st International Conference on Advanced Engineering and Technologies, ICONNIC 2023 - Proceeding*, pp. 259–262, 2023, doi: 10.1109/ICONNIC59854.2023.10467460.
[48] Bushra Jaber M.Jawad and S. Al-alak, "Design and Implementation of Multi-key Blowfish and CAST Algorithm: Comparative Study with CBC, CFB and CTR Modes," *Wasit Journal of Computer and Mathematics Science*, vol. 2, no. 4, pp. 87–98, 2023, doi: 10.31185/wjcms.203.
[49] Y. Li, C. Song, J. Dong, and H. Zheng, "An efficient encryption method for smart grid data based on improved CBC mode," *Journal of King Saud University - Computer and Information Sciences*, vol. 35, no. 9, p. 101744, 2023, doi: 10.1016/j.jksuci.2023.101744.
[50] O. Manankova and M. Yakubova, "Modeling of the Modes of Operation of the AES Algorithm in the Cryptool 2 Environment," *In: Arai, K. (eds) Intelligent Computing. SAI 2023. Lecture Notes in Networks and Systems*, vol. 739, pp. 462–469, 2023, doi: https://doi.org/10.1007/978-3-031-37963-5_32.
[51] M. Y. M. Yassin, A. A. Nasir, M. Taha, and N. Iqbal, "Enhanced DFT-Based Chaotic Image Block Cipher with Several Modes of Operation," *2023 12th International Conference on Image Processing Theory, Tools and Applications, IPTA 2023*, 2023, doi: 10.1109/IPTA59101.2023.10320057.
[52] W. Stallings, *NETWORK SECURITY ESSENTIALS: APPLICATIONS AND STANDARDS*, Fourth Edi. 2010.
[53] D. A. Mcgrew, W. T. Drive, S. Jose, and J. Viega, "The Galois / Counter Mode of Operation (GCM)," 2005.
[54] J. Katz and Y. Lindell, *Introduction to modern cryptography*. 2007. doi: 10.1201/9781420010756.
[55] D. A. Mcgrew and S. R. Fluhrer, "Multiple forgery attacks against Message Authentication Codes," 2005.