A Project Report on

# “COMPARATIVE ANALYSIS OF VARIOUS AES MODES OF OPERATION FOR IMAGE ENCRYPTION”

# ABSTRACT

In the digital era of today, data is considered as the new oil, which tells us the importance of data in the current scenario. With the involvement of lots and lots of data, there comes the need for safeguarding the data. Security in the digital data communication is the need of the hour. These data can be of any type, say, text, image, audio, video and so on.

There is also widespread use of digital images in the current era. Images are highly useful, better to understand than only text and also has many advantages. With these benefits, there are also various threats. Images can be stolen, manipulated and then misused. Therefore, there is need to make sure that these images are kept and transferred in a safe manner.

With the rapid increase of the Internet users, network security becomes very important and necessity. Cryptography plays a major role in network security. There are various cryptographic systems, developed by various researchers. Some of these systems consume considerable amounts of resources, like, memory, CPU time, encryption and decryption time.

In this project, the various block cipher modes of operation on AES, which is considered as the standard encryption system by the National Institute of Standards and Technology (NIST). The comparison is done in terms of encryption time, decryption time, and throughput with variable data packets sizes.

The results of the comparison are summarized and our observations are highlighted to help making informative decision when choosing the mode of operations for different applications with symmetric-key ciphers.

# TABLE OF CONTENTS

# LIST OF TABLES



# LIST OF FIGURES

# CHAPTER 1

# INTRODUCTION

# 1. INTRODUCTION

## 1.1. Overview

With the introduction of Internet, there was a boom in the digital world. The rapid advancement in the technology led to cheaper electronic gadgets and cheaper internet. This resulted in involvement of vast amount of data and increment in digital communication. These data must be safeguarded and secure communication sessions must be provided to the users of digital world. The security of data transmitted across a global network has turned into a key factor on the network performance measures. So, the confidentiality and the integrity of data are needed to prevent eavesdroppers from accessing and using transmitted data.

Cryptographic algorithms are very important in information security where data is encrypted at the sender side and decrypted at the receiver side. The Advances Encryption Standard (AES) is a symmetric block cipher adopted by the National Institute of Standards and Technology (NIST) in 2001. A block cipher processes the plaintext of a fixed length, known as the block size. If the length of the plaintext is larger than the block size, it must be divided into several blocks. Typically, the last block of the plaintext must be padded to match the block size. Modes of operation are formal descriptions of the way that the block encryption on a plaintext with size larger than a block size is handled. They also provide desired services such as authenticity, integrity, and confidentiality.

In this project, different modes of operation on AES is compared for best performance. The performance evaluation of these modes is based on the following metrics: encryption time, decryption time, and throughput with variable data packet size. This study fills the gaps in previous studies regarding the modes of operation on AES.

### 1.1.1. Advanced Encryption Standard (AES) Algorithm

The AES algorithm (also known as the Rijndael algorithm) is a symmetrical block cipher algorithm that takes plain text in blocks of 128 bits and converts them to ciphertext using keys of 128, 192, and 256 bits. Since the AES algorithm is considered secure, it is in the worldwide standard.

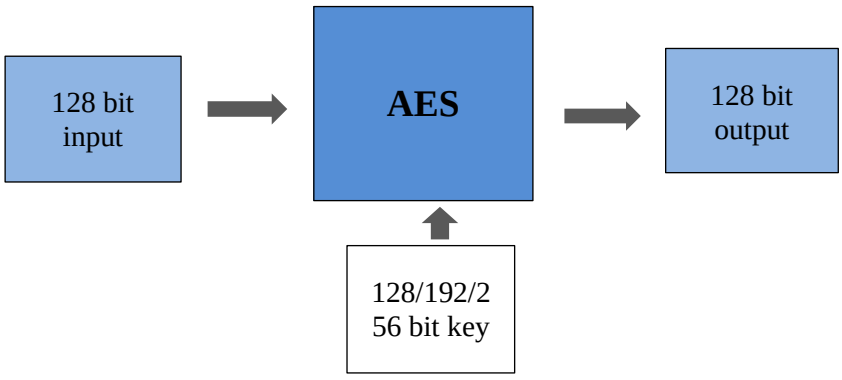
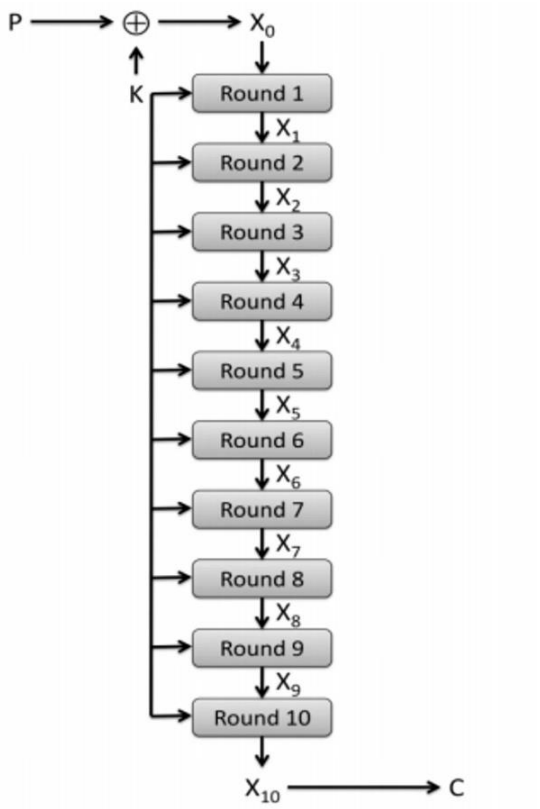



**Figure 1.1.1.1: Block sizes in AES Algorithm**

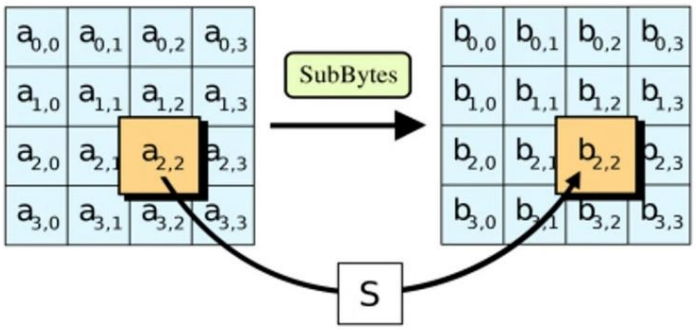
The AES algorithm uses a substitution-permutation, or SP network, with multiple rounds to produce ciphertext. The number of rounds depends on the key size being used. A 128-bit key size dictates ten rounds, a 192-bit key size dictates 12 rounds, and a 256-bit key size has 14 rounds. Each of these rounds requires a round key, but since only one key is inputted into the algorithm, this key needs to be expanded to get keys for each round, including round 0.




Each round in the algorithm consists of four steps.

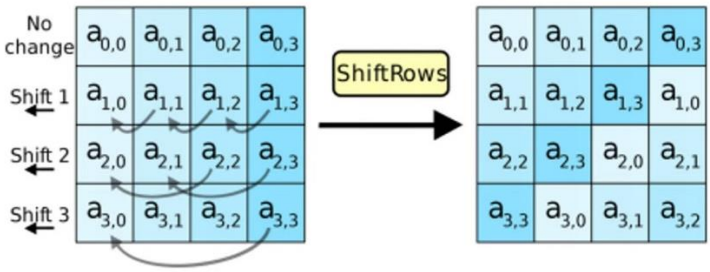
*i) Substitution of the bytes*

In the first step, the bytes of the block text are substituted based on rules dictated by predefined S-boxes (short for substitution boxes).



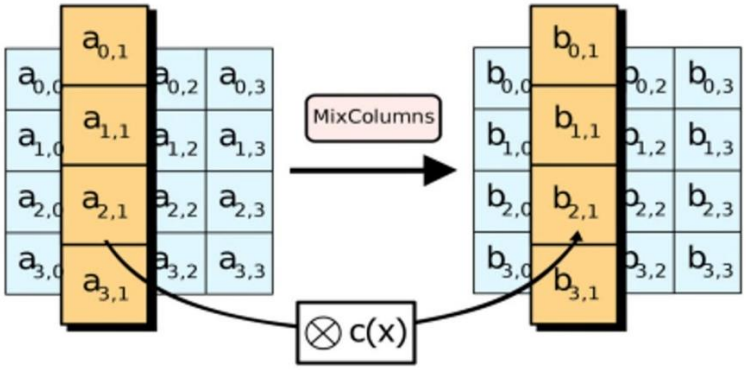

*ii) Shifting the rows*

Next comes the permutation step. In this step, all rows except the first are shifted by one, as shown below.



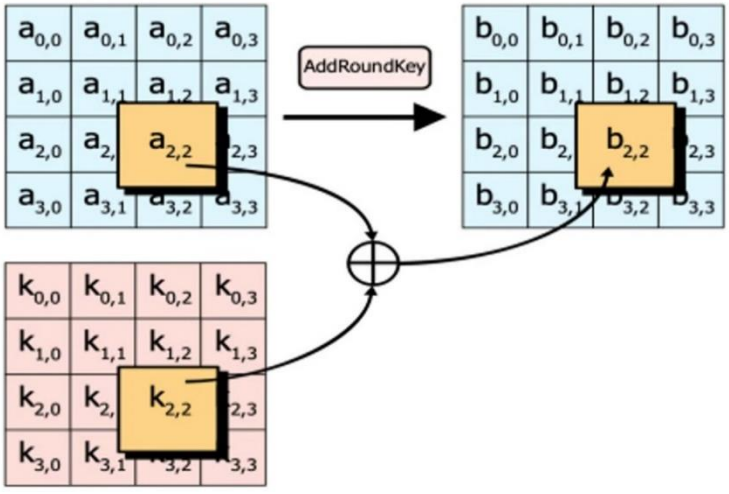

*iii) Mixing the columns*

In the third step, the Hill cipher is used to jumble up the message more by mixing the block's columns.




*iv) Adding the round key*

In the final step, the message is XORed with the respective round key.




Based on the recommendation of the National Institute of Standards and Technology (NIST), there are five main block cipher modes of operation: the Electronic Code Book (ECB), Cipher Block Chaining (CBC), Cipher FeedBack (CFB), the Output FeedBack (OFB), and the Counter (CTR) modes. Few more modes of operation include Offset CodeBook (OCB) mode, Galois Counter Mode (GCM) and Counter with CBC-MAC (CCM).

### 1.1.2. Electronic Codebook (ECB) Mode

ECB is the simplest AES block cipher mode. In the ECB mode, the Bi block is encrypted according to the following formula:

$$C_i = E_K(B_i)$$

where $E_K$ denotes the block encryption algorithm using key *K* and $C_i$ is the cipher corresponding to $B_i$.

Decryption using the ECB mode is equally simple with the following formula:

$$Bi = DK(Ci)$$

where DK denotes the block decryption algorithm using key K.

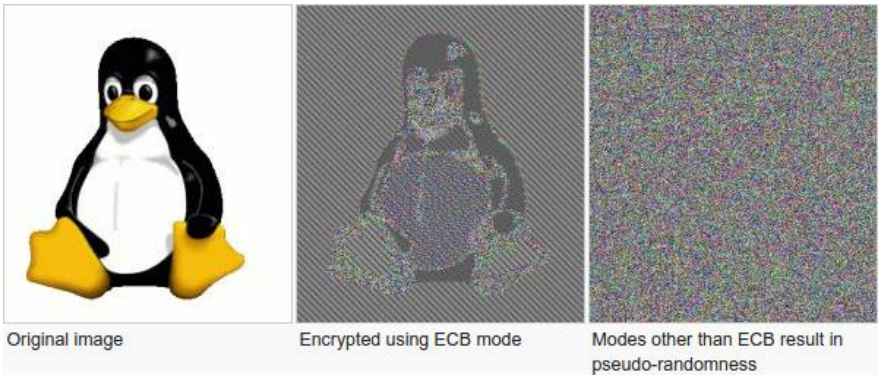



**Figure 1.1.2.1: Encryption in ECB Mode**

The most obvious advantage of using the ECB mode is how simplistic it is. The other main advantage is that ECB can tolerate the loss of blocks without affecting other available blocks. This advantage is relevant in the case of blocks being sent over a network as packets. This resilience is made possible by the fact that any Bi block does not depend on any of its adjacent blocks.

Despite its advantages, ECB is looked down upon due to the fact that the encryption algorithm is entirely deterministic. In simpler terms, identical blocks will have the same ciphers under ECB mode, which may reveal patterns the blocks have; so, ECB doesn't wholly hide its details. This is a security threat to its users.

### 1.1.3. Cipher Block Chaining (CBC) Mode

CBC is an AES block cipher mode that trumps the ECB mode in hiding away patterns in the plaintext. CBC mode achieves this by XOR-ing the first plaintext block ($B_1$) with an initialization vector before encrypting it. CBC also involves block chaining as every subsequent plaintext block is XOR-ed with the ciphertext of the previous block.

If we summarize this process into a formula, it would look like:

$$C_i = E_K (B_i \oplus C_{i-1})$$

where $E_K$ denotes the block encryption algorithm using key *K*, and $C_{i-1}$ is the cipher corresponding to $B_{i-1}$. Here, we are assuming $C_0$ to be the initialization vector.

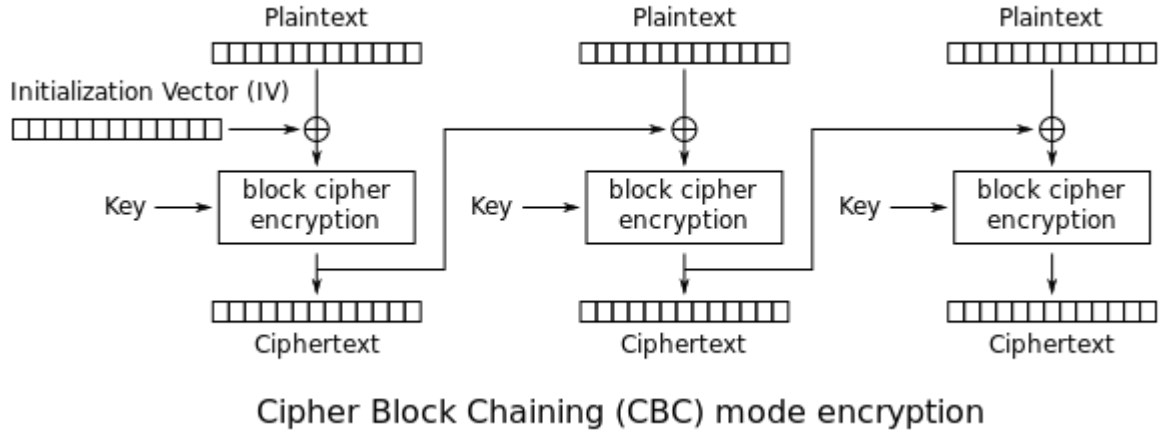



**Figure 1.1.3.1: Encryption in CBC Mode**

Similarly, decryption using the CBC can be done using:

$$B_i = D_K (C_i) \oplus (C_{i-1})$$

where $D_K$ denotes the block decryption algorithm using key K. The same initialization vector ($C_0$) will be used for decryption.

The greatest advantage CBC has over ECB is that, with CBC mode, identical blocks do not have the same cipher. This is because the initialization vector adds a random factor to each block; hence, why the same blocks in different positions will have different ciphers.

Although CBC mode is more secure, its encryption is not tolerant of block losses. This is because blocks depend on their previous blocks for encryption. So, if block Bi is lost, the encryption of all subsequent blocks will not be possible. This chained behavior also means that the encryption of blocks needs to be done sequentially, not in parallel. However, these disadvantages do not extend to decryption, which can be done in parallel if all ciphertext blocks are available and can tolerate block losses.

### 1.1.4. Cipher Feedback (CFB) Mode

CFB is an AES block cipher mode similar to the CBC mode in the sense that for the encryption of a block, $B_i$, the cipher of the previous block, $C_{i-1}$ is required. CFB also makes use of an initialization vector like CBC. The main difference is that in CFB, the ciphertext block of the previous block is encrypted first and then XOR-ed with the block in focus.

The CFB in the form of a formula:

$$C_i = E_K(C_{i-1}) \oplus B_i$$

where $E_K$ denotes the block encryption algorithm using key *K* and $C_{i-1}$ is the cipher corresponding to $B_{i-1}$. Here, we are assuming C0 to be the initialization vector.

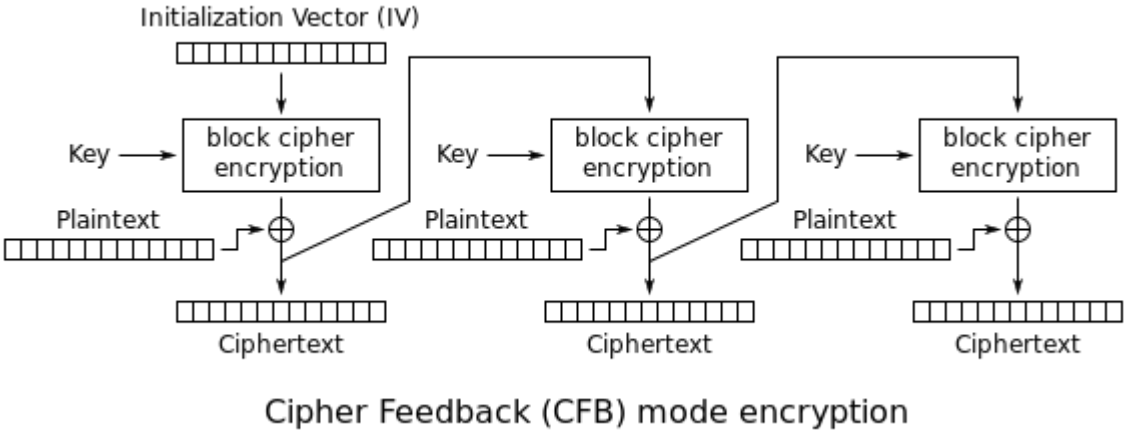



**Figure 1.1.4.1: Encryption in CFB Mode**

Similarly, decryption using the CFB can be depicted as:

$$B_i = E_K(C_{i-1}) \oplus (C_i)$$

It is essential to note that the decryption algorithm is not used here.

The main advantage of the CFB mode is that, since it doesn't use a decryption algorithm, it is generally faster than the CBC mode. CFB encryption is also non-deterministic, which means it does not reveal any patterns the plaintext may have.

The disadvantages of CFB are identical to those of the CBC mode. The encryption cannot tolerate block losses, nor can multiple blocks be encrypted in parallel. However, decryption is both loss-tolerant and can be parallelized.

### 1.1.5. Output Feedback (OFB) Mode

OFB is an AES block cipher mode similar to the CFB mode. What mainly differs from CFB is that the OFB mode relies on XOR-ing plaintext and ciphertext blocks with expanded versions of the initialization vector. This process can be seen as a one-time pad and the expanded vectors as pad vectors.

The following formula depicts how a sequence of pad vectors is created:

$$V_i = E_K(V_{i-1})$$

where $E_K$ denotes the block encryption algorithm using key *K* and $V_i$ and $V_{i-1}$ are adjacent vectors. In the above formula, we are assuming $V_0$ to be the initialization vector.

Once the sequence of pad vectors is generated, encryption with the OFB mode can be carried out using the following formula:

$$C_i = V_i \oplus B_i$$

Decryption is carried out in a similar way:

$$B_i = V_i \oplus C_i$$

Like the CFB mode, OFB also makes use of a single encryption algorithm for both encryption and decryption.

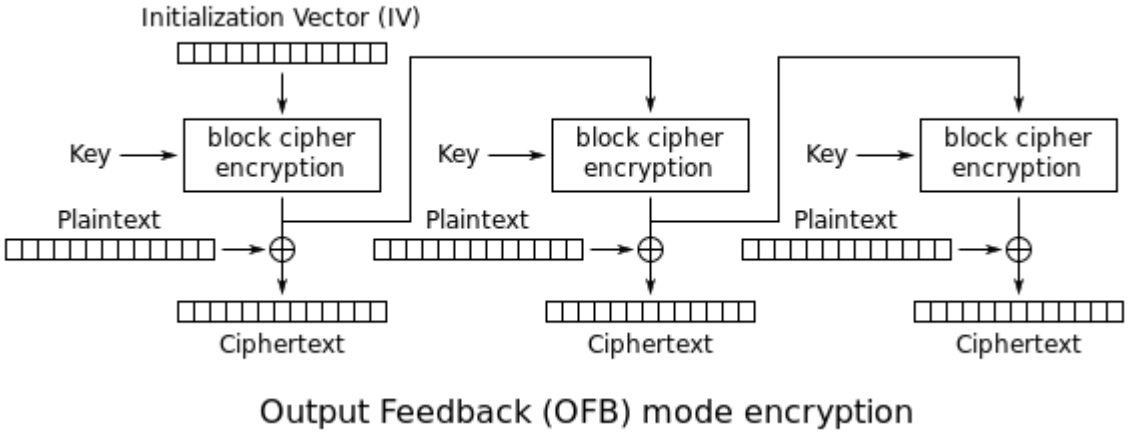



**Figure 1.1.5.1: Encryption in OFB Mode**

Since blocks are independent of one another using the OFB mode, both encryption and decryption of blocks can be done in parallel once the pad vectors have been generated. The lack of interdependency also means that the OFB mode is tolerant to the loss in blocks.

A significant drawback of the OFB is that repeatedly encrypting the initialization vector may produce the same state that has occurred before. This is an unlikely situation, but in such a case, the plaintext will start to be encrypted by the same data as it was previously.

### 1.1.6. Counter (CTR) Mode

CTR is a popular AES block cipher mode in which every step can be done in parallel. CTR is similar to OFB as it also involves XOR-ing a sequence of pad vectors with the plaintext and ciphertext blocks. The main difference lies in how these pad vectors are generated.

In the CTR mode, we start off with a random seed, s, and compute pad vectors according to the formula:

$$V_i = E_K(s+i-1)$$

where $E_K$ denotes the block encryption algorithm using key K, $V_i$ is a pad vector, and i is the vector's offset starting from 1.

Now that the vectors have been generated, encryption similar to the OFB mode can proceed using the following formula:

$$C_i = V_i \oplus B_i$$

Decryption is carried out in a similar way:

$$B_i = V_i \oplus C_i$$

Note that, CTR, like the CFB and OFB modes, also makes use of a single encryption algorithm for both encryption and decryption.

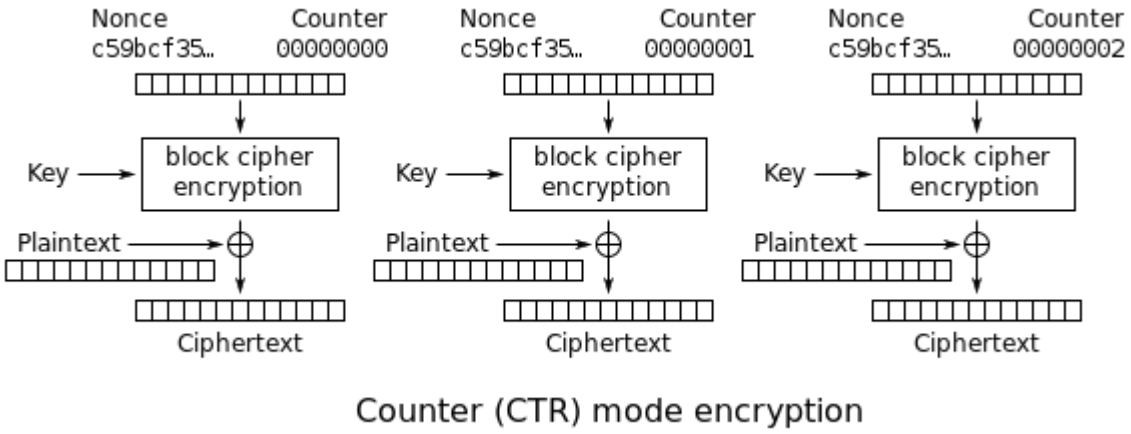



**Figure 1.1.6.1: Encryption in CTR Mode**

Since blocks are independent of one another with the CTR mode, once the pad vectors have been generated, both encryption and decryption of blocks can be parallelly done. The lack of interdependency also means that the CTR mode is tolerant to a loss in blocks. The CTR mode is considered to be very secure and efficient for most purposes.

A serious disadvantage of the CTR is that a synchronous counter needs to be maintained at both receiving and sending ends. Losing track of this counter could lead to the incorrect recovery of plaintext.

### 1.1.7. Offset Codebook (OCB) Mode

OCB is a blockcipher-based mode of operation that simultaneously provides both privacy and authenticity for a user-supplied plaintext. Such a method is called an authenticated-encryption scheme. What makes OCB remarkable is that it achieves authenticated encryption in almost the same amount of time as the fastest conventional mode, CTR mode, achieves privacy alone. Using OCB one buys privacy and authenticity about as cheaply as one used to pay for achieving privacy alone. Despite this, OCB is simple and clean, and easy to implement in either hardware or software. OCB accomplishes its work without bringing in the machinery of universal hashing, a technique that does not seem to lend itself to implementations that are simple and fast in both hardware and software.

Somewhat more precisely, OCB solves the problem of nonce-based authenticated-encryption with associated-data (AEAD). The associated-data part of the name means that when OCB encrypts a plaintext it can bind it to some other string, called the associated data, that is authenticated but not encrypted. The nonce-based part of the name means that OCB requires a nonce to encrypt each message. OCB does not require the nonce to be random; a counter, say, will work fine. Unlike some modes, the plaintext provided to OCB can be of any length, as can the associated data, and OCB will encrypt the plaintext without padding it to some convenient-length string, an approach that would yield a longer ciphertext.

Assume using a blockcipher of E=AES with 128-bit keys. The block length is n=128 bits. Let's assume a nonce of 96 bits. Other values can be accommodated, but 96 bits (12 bytes) is the recommended nonce length. Let M be the message you want to encrypt. Let A be the associated data. Let K be the OCB encryption key, which is just a key for the underlying blockcipher, AES. Let N be the 96-bit nonce—for example, a counter that gets incremented with each message to be encrypted.

Let's first look at the setting when |M| is a positive multiple of 128 bits. Then break M into 128-bit blocks M = M1 ... Mm. Encryption then works as shown in the following picture. Read each column top-to-bottom, and then go across left-to-right. We'll defer describing the Init and Inc functions for just a moment. The value Checksum is the 128-bit string Checksum = M1 ⊕…⊕ Mm. We'll describe how to compute the value Auth in a moment. It's determined by A and K. The number τ, the tag length of the scheme, is, like the blockcipher E, a parameter of the mode. It's a number $0 \le \tau \le 128$. Now the picture:

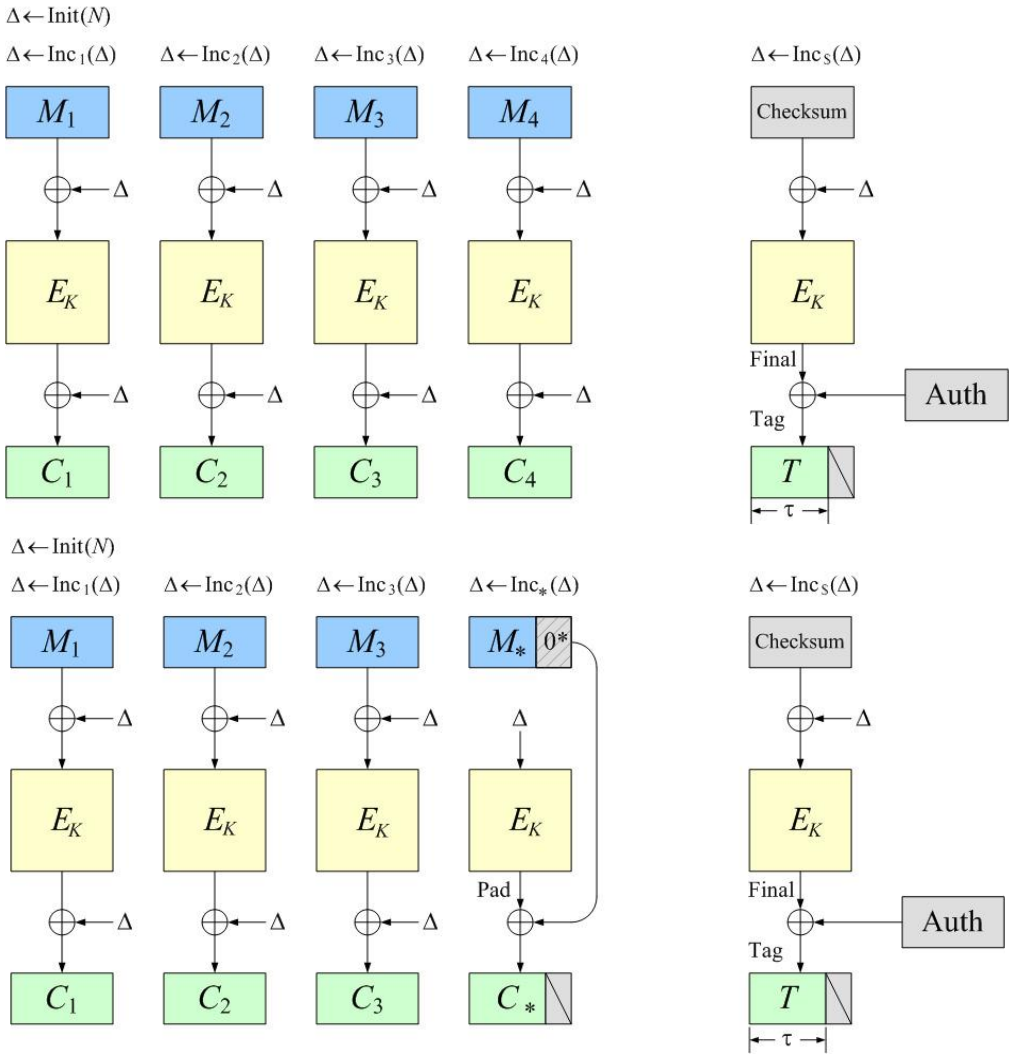



**Figure 1.1.7.1: Encryption in OCB Mode**

The ciphertext is the 128m + τ bit string CT = C1 C2 … Cm T.

We are given a 96-bit nonce N. We save the last six bits as the number Bottom (a value between 0 and 63) and then we create a 128-bit value Top by prepending to N the the 32-bit constant 0x00000001 and zeroing out the last six bits. Let $K_{top} = E_K$ (Top) and let Stretch be the 256-bit string Stretch = $K_{top} \| (K_{top} \oplus (K_{top} <<8))$. By S << i we mean the left shift of the 128-bit string S by i positions (leftmost bits vanish, zero bits come in on the right). The value denoted above as Init(N), the initial value for Δ, is the first 128 bits of Stretch << Bottom.

Here, if N is counter, $K_{top}$ is going to change only once every 64 calls. As a consequence, if we keep $K_{top}$ around as long as needed, then, 63 out of 64 times, all that will be needed to compute Init(N) is a logical operation to extract the lowest six bits of N and then a logical shift of the string Stretch by these many bits. Just a few cycles. The other one time out of 64 we will need a blockcipher call. The amortized cost to compute Init, meaning the average cost over a sequence of successive calls, is extremely low, and we don't need to implement anything like a GF (2128) multiply.

Now let's explain how the Inci function works. First, for S a 128-bit string, let double(S) = S << 1 if the first bit of S is 0, and let double(S) = (S << 1) ⊕135 otherwise (where 135 denotes the 128-bit string that encodes that decimal value). For those of you for whom this means something, double(S) is just the multiplication of S by the field point "x" when using a particular representation for the finite field with 2128 points. Given all these K-derived L-values, we set $L_* = E_K$ (0128), L$=double(L∗), L[0]=double(L$), and L[j]=double(L[j-1]) for all j ≥ 1. Given the above, define Inci (Δ) = Δ ⊕L[j] where j = ntz(i) is the number of trailing zero bits in the binary representation of i. Likewise define Inc∗(Δ) = Δ⊕L∗. and Inc$(Δ) = Δ⊕L$.

We call the increment function we have described "table-based" because, to implement them, it is natural to precompute a table of 128-bit L∗, L$, and L[j]. Then, for each kind of increment, we just xor in the needed value from the table.

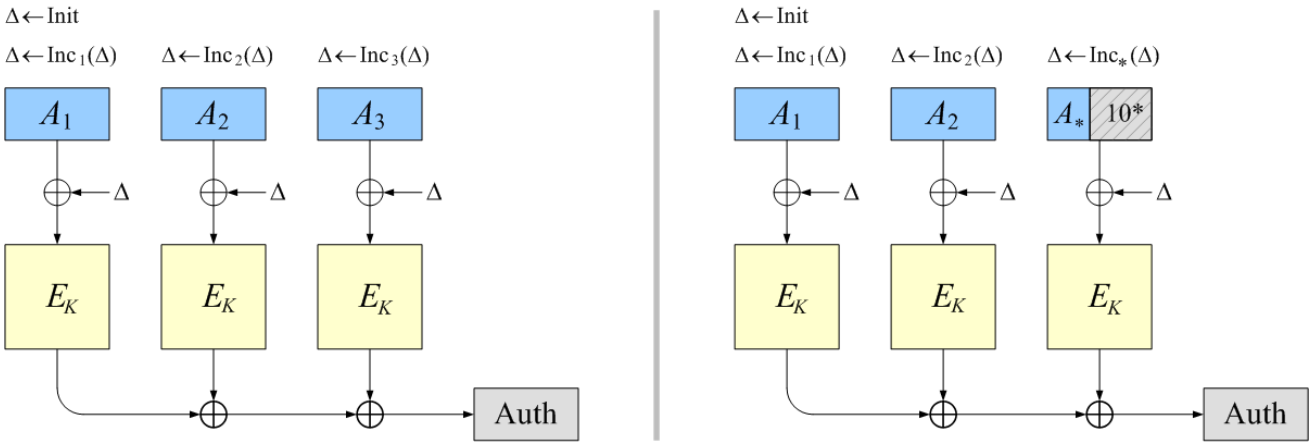



**Figure 1.1.7.2: Decryption in OCB Mode**

Decryption under OCB works in the expected way: given K, N, and CT = C1 C2 … Cm T, recover M in the natural way and recompute the tag T* that "should" be at the end of CT. Regard M as the (authentic) underlying plaintext for CT if T = T*. Regard the ciphertext as invalid if T ≠ T*.

When |M| is not a positive multiple of 128 bits we process the final block a little differently. This time we break the plaintext M into blocks M = M1 ... Mm M∗ where |Mi| = 128 and 0 ≤ |M∗| < 128. Computation works as in the following picture. We redefine Checksum as Checksum = M1 ⊕…⊕ Mm ⊕ M∗ 10* where the M∗ 10* notation means to append a single 1-bit and then the minimum number of 0-bits to get the string to be 128 bits.

### 1.1.8. Galois Counter (GCM) Mode

The Galois/Counter Mode (GCM) is a typical block cipher modes of operation using block cipher algorithm. In this version, we provide Advanced Encryption Standard (AES) processing ability, the cipherkey length for AES should be 128/192/256 bits.

The algorithm flow chart of encryption part of GCM mode is shown as follow:

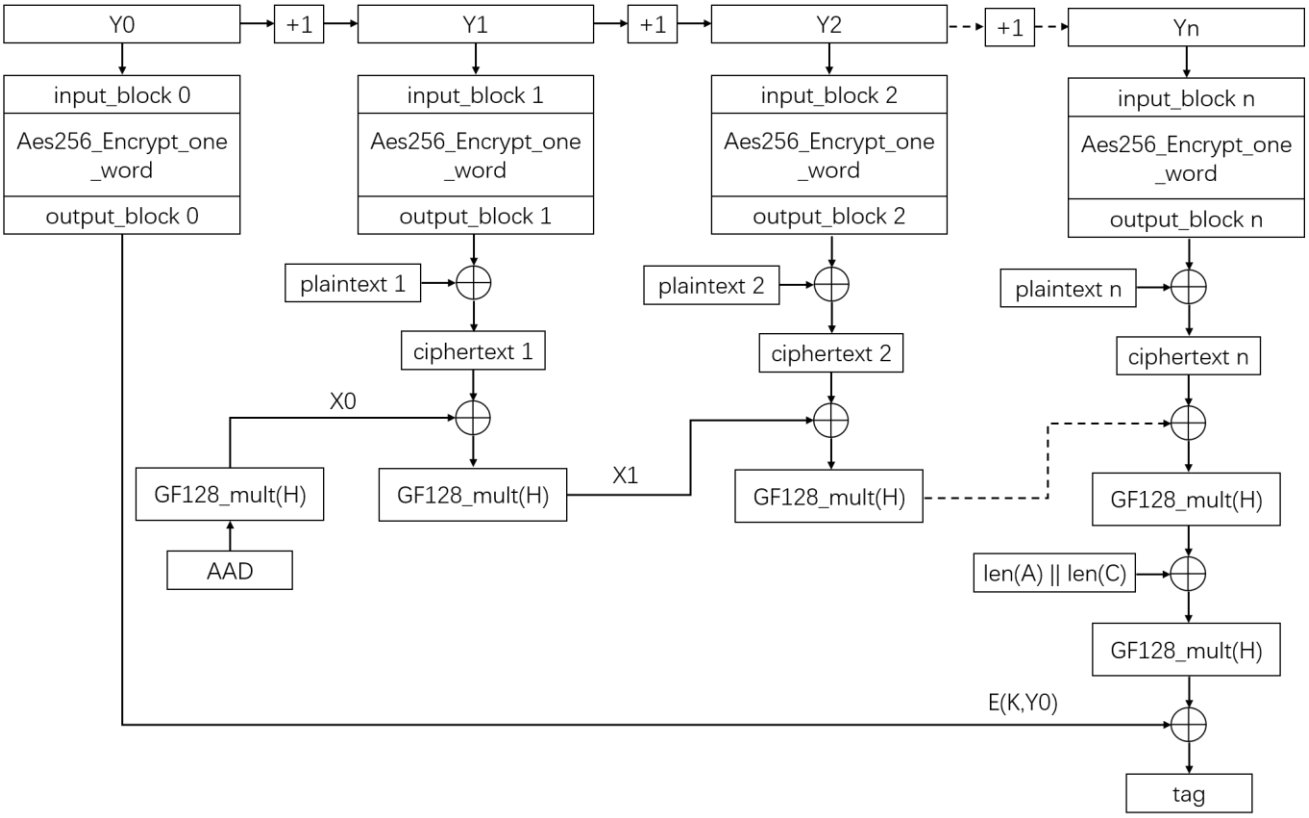



**Figure 1.1.8.1: Encryption in GCM Mode**

As we can see from the chart, the GCM encryption part can be divided into two individual parts: The Counter Mode (CTR) and The Galois Message Authentication Code (GMAC). GCM is used to encrypt the plaintext to ciphertext, and GMAC is used to generate the MAC.

The algorithm flow chart of decryption part of GCM mode is given below. The decryption part is very similar with the encryption part of GCM mode. The only difference is that we decrypt the ciphertext to plaintext in the decryption part.

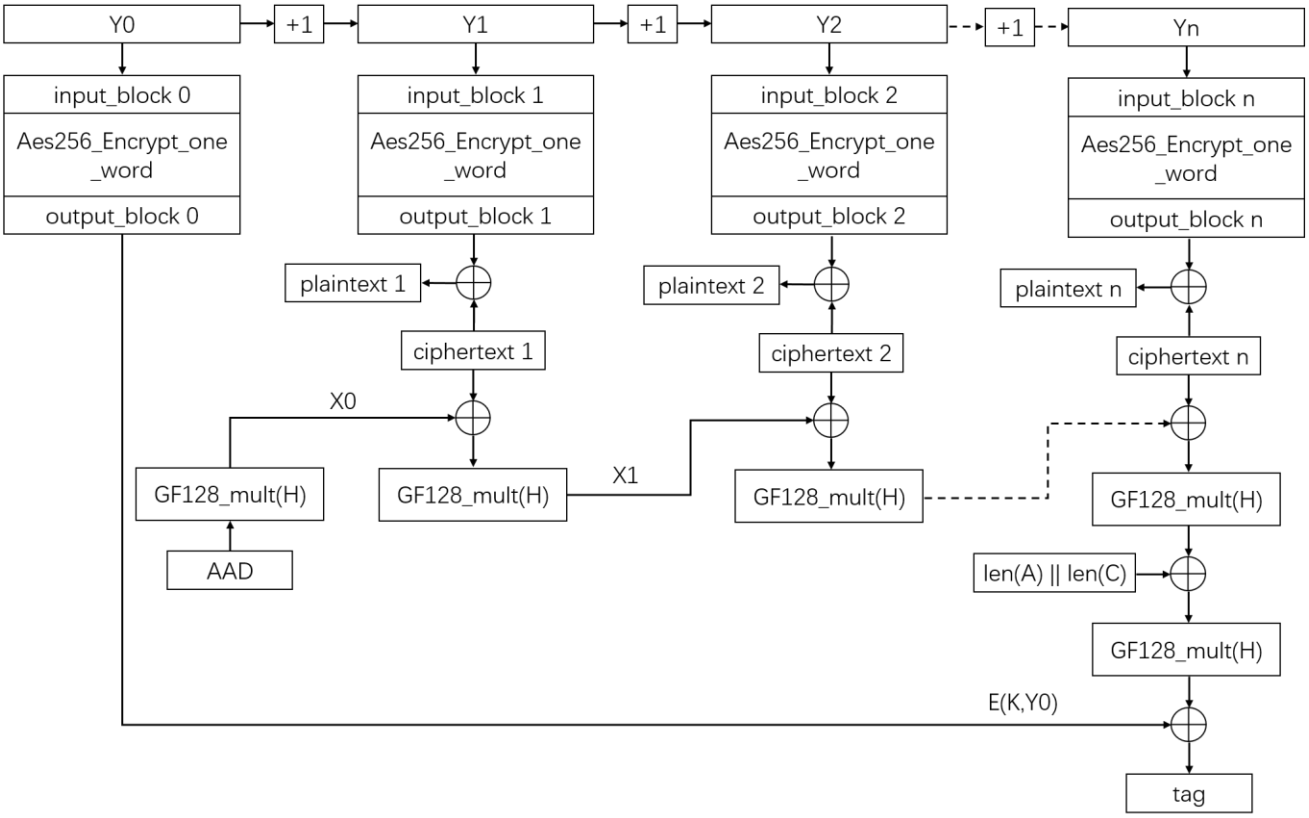



**Figure 1.1.8.2: Decryption in GCM Mode**

### 1.1.9. Cipher Block Chaining – Message Authentication Code (CCM) Mode

CCM stands for Counter with CBC-MAC mode. CCM is a generic authenticate-and-encrypt block cipher mode. CBC-MAC is utilized to generate an authentication string while CTR mode is used to encrypt. The CCM mode is a typical block cipher mode of operation using block cipher algorithm.

The algorithm flow chart of encryption part of CCM mode is shown as follow:

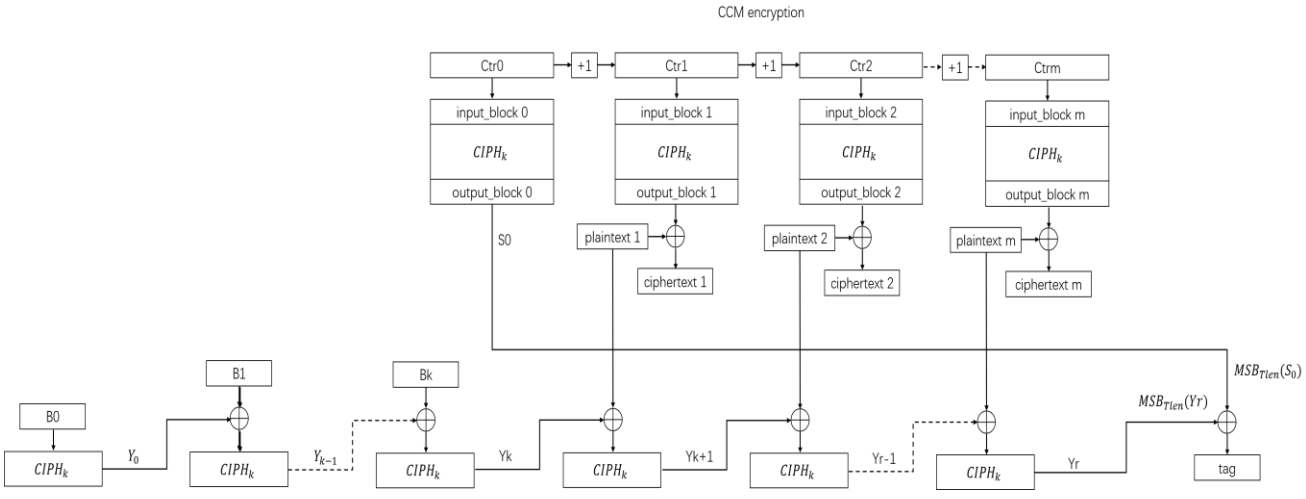



**Figure 1.1.9.1: Encryption in CCM Mode**

As we can see from the chart, the CCM encryption part can be divided into two individual parts: The Counter Mode (CTR) and The Cipher Block Chaining-Message Authentication Code (CBC-MAC). CTR is used to encrypt the plaintext to ciphertext, and CBC-MAC is used to generate the data tag (MAC).

The algorithm flow chart of decryption part of CCM mode is shown as follow:

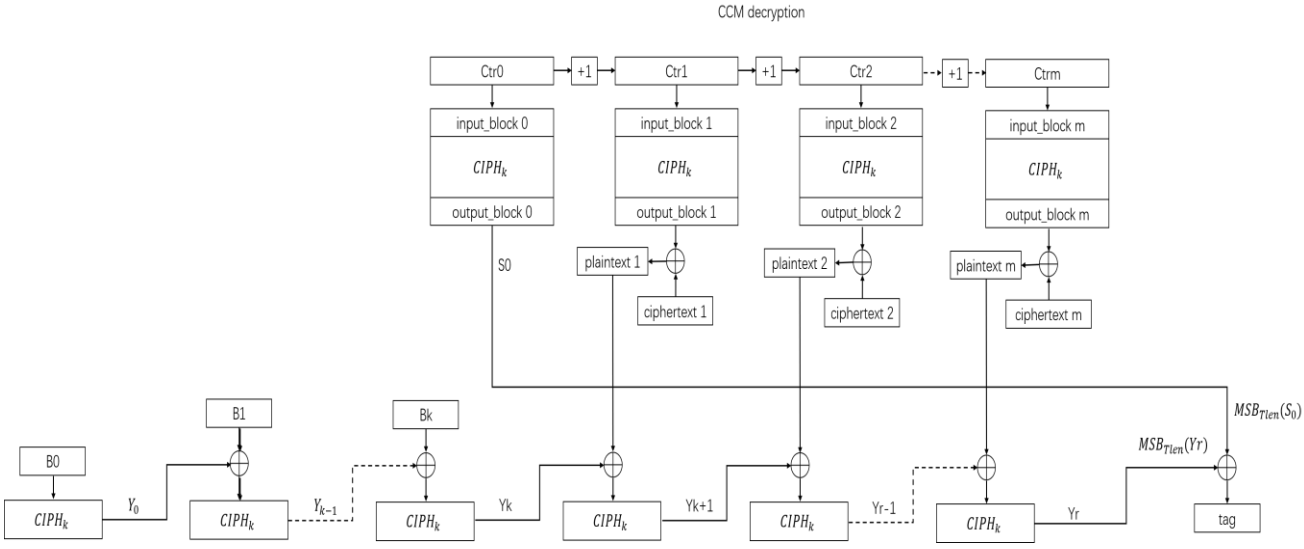



**Figure 1.1.9.2: Decryption in CCM Mode**

The decryption part is very similar with the encryption part of CCM mode. The only difference is that we decrypt the ciphertext to plaintext in the decryption part. In decryption part of CCM mode, we don't provide a bool flag to indicate whether the data is authentic or not. You should compare the tag which the decryption part gives with the tag from CCM encryption part to judge the authenticity of the data. If the data is authentic, then the tags should be equal.

## 2.2. Statement of the Problem

Many researchers have published comparative analysis on the performance of different modes of operation. For example, the authors compared six different symmetric key encryption algorithms using ECB mode, namely: DES, Triple DES, AES, RC2, RC6 and Blowfish. They compared the algorithms in terms of CPU clock cycle, CPU time, encryption time, and battery power consumption.

But there are no detailed studies of various modes of operation of AES when encrypting an image. In order to fill this void, there is need to conduct a comparative analysis of AES modes for image encryption. With large amount of image data being used in everyday life, it is necessary to study about various image encryption techniques possible. Since AES is considered as the standard by NIST, there is need to find, which mode of operation AES is better for encrypting images.

## 2.3. Motivation

In case of AES algorithm, there are also few modes which are not considered for comparative analysis. Offset Codebook mode (OCB mode) which is considered as the fastest way to provide authenticity and confidentiality is one among them. The main reason behind this might be that the OCB is patented for commercial use in the USA. Due to this, it is not widely used and has not become a standard. Apparently, all patents have now been abandoned. Therefore, there is a need to conduct a comparative analysis of various modes of AES, including the OCB mode and find out, which algorithm is better for encrypting image data.

## 2.4. Advantage

The comparative analysis of the various modes of operation of AES algorithm will help to choose the most appropriate mode for encrypting data. The analysis would also help to understand, which type of data can be encrypted optimally using which mode of AES. Also, the analysis would help to understand the advantages and disadvantages of various modes used in AES algorithm.

## 2.5. Challenges

It is very much essential to make maintain the security of image data. So, while choosing an encryption technique, one must be aware of the advantages and disadvantages of those techniques. The key point of an encryption technique is the time taken and resources it uses. So, there is need to find various factors regarding image encryption techniques.

## 2.6. Objectives

Many published comparative analysis on various modes of operation of AES algorithms have been already published. But they have missed out few important modes of operation. There are various studies regarding the five most common modes of operation, we targeted in one study the most common five modes of operation (ECB, CBC, CFB, OFB and CRT) in terms of encryption time, decryption time, and throughput. We conduct our study on the most popular encryption algorithm AES. This study fills the gaps in the previous studies on AES.

## 2.7. Application

Image data is one of the crucial data. With the advancements in the fields of Artificial Intelligence and Machine Learning, the Computer Vision domain has got a boom. From facial data used for face recognition, fingerprints used for authentication till Medical Imaging, everywhere, image data is used. These data has to be protected from unauthorized access. So, these images need to be encrypted. A detailed study on various modes of AES algorithm in image encryption would thus help to find which mode would be better for which type of image. The results would also help to choose efficient and less resource consuming technique to encrypt and decrypt image data.

## 2.8. Organization of Reports

In order to study about the algorithms, a windows application using C# is developed, so that it is easier for selecting the files and finding the results. The selected image or the file and the encryption mode will be passed to the Python script and the results are passed back to the windows form. The time taken for encryption and decryption is displayed by the windows form along with the peak memory used during the process of encryption and decryption is also displayed. The interface provided by the windows form helps it easier to select the files and analyse the results.

# CHAPTER 2

# LITERATURE SURVEY

# 2. LITERATURE SURVEY

In the past few years the security and integrity of data is the main concern. In the present scenario almost all the data is transferred over computer networks due to which it is vulnerable to various kinds of attacks. To make the data secure from various attacks and for the integrity of data we must encrypt the data before it is transmitted or stored. Cryptography is a method of storing and transmitting data in a form that only those it is intended for can read and process. It is a science of protecting information by encoding it into an unreadable format. It is an effective way of protecting sensitive information as it is stored on media or transmitted through network communication paths.

The purpose of Cryptography include authentication, integrity and privacy or confidentiality. Authentication is the process of proving one's identity. It is another part of data security that we encounter with everyday computer usage. Just think when you log into your email, or blog account. The simple sign-in process is a form of authentication that allows you to log into applications, files, folders and even an entire computer system. Integrity means the data being updated only by the authenticated people. Confidentiality is ensuring that no one can read the message except the intended receiver. Encryption is the process of obscuring information to make it unreadable without special knowledge. Encryption has been used to protect communications for centuries, but only organizations and individuals with an extraordinary need for secrecy had made use of it. In the mid-1970s, strong encryption emerged from the sole preserve of secretive government agencies into the public domain, and is now used in protecting widely-used systems, such as Internet e-commerce, mobile telephone networks and bank automatic teller machines.

Data Security is primary concern for every communication system. The relentless growth of Internet and communication technologies has made the extensive use of images unavoidable. There are many ways to provide security to data that is being communicated. In January, 1997 NIST began its effort to develop the AES, a symmetric key encryption algorithm, and made a worldwide public call for the algorithm to succeed DES. Initially 15 algorithms were selected, which was then reduced down to 4 algorithms, RC6, Rijndael, Serpent and Two-fish, all of which were iterated block ciphers. The four finalists were all determined to be qualified as the AES. The algorithm had to be suitable across a wide range of hardware and software systems. The algorithm had to be relatively simple as well. After extensive review the Rijndael algorithm was chosen to be the AES algorithm. The Advanced Encryption Standard can be programmed in software or built with pure hardware.

For Rijndael, the length of both the block to be encrypted and the encryption key are not fixed. They can be independently specified to 128, 192 or 256 bits. The number of rounds, however, varies according to the key length. It can be equal to 10, 12 and 14 when the key length is 128bits, 192 bits and 256 bits, respectively. The basic components of Rijndael are simple mathematical, logical, and table lookup operations. The latter is actually a composite function of an inversion over Galois Field (GF) with an affine mapping. Such structure makes Rijndael suitable for hardware implementation.

The block ciphers are schemes for encryption or decryption where a block of plaintext is treated as a single block and is used to obtain a block of ciphertext with the same size. Today, AES (Advanced Encryption Standard) is one of the most used algorithms for block encryption. It has been standardized by the NIST (National Institute of Standards and Technology) in 2001, in order to replace DES and 3DES which were used for encryption in that period. The size of an AES block is 128 bits, whereas the size of the encryption key can be 128, 192 or 256 bits. Please note this, there is three length in the key, but the size of the encryption block always is 128 bits. Block cipher algorithms should enable

encryption of the plaintext with size which is different from the defined size of one block as well. We can use some algorithms for padding block when the plaintext is not enough a block, like PKCS5 or PKCS7, it also can defend against PA attack, if we use ECB or CBC mode. Or we can use the mode of AES which support a stream of plaintext, like CFB, OFB, CTR mode.

# CHAPTER 3

# METHODOLOGY

# 3. METHODOLOGY

## 3.1. Overview

In our experiment, we used available python classes to do the performance evaluation of the operation modes. The implementation uses managed wrappers for AES available in Crypto.Cipher module of the PyCryptodome package. The functionality of a cryptographic cipher which is used for encryption and decryption is provided by the Cipher class. The various modes used in the experiment, namely ECB, CBC, CFB, OFB, CTR, OCB, CCM and GCM are used from the modes available in the AES class in python.

A computer with 2.30 GHz CPU and 16 GB RAM is used to conduct the experiment. The computer encrypts files with different sizes in the ranges 500 KB to 200 MB. For the CFB and OFB modes, we used the value of s = 16 bits. The performance evaluation is conducted in terms of encryption time, decryption time and throughput, which are defined as follows:

1) Encryption time: is the time consumed to generate a ciphertext from a plaintext by an encryption algorithm.

2) Decryption time: is the time consumed by a decryption algorithm to reproduce a plaintext from a ciphertext.

3) Throughput: is calculated as the whole encrypted plaintext in Kilobytes divided by encryption time (KB/sec). For encryption scheme, the throughput indicates the speed of encryption. When the throughput increases, the power consumption decreases.

## 3.2. Architecture of Proposed System

For this experiment, a windows form is developed using the C# language. The user gets to choose various modes available for encryption. The user can select the intended file using the button present. If the file is image, it will be displayed. On clicking the Proceed button, the corresponding path and the selected encryption mode is passed to the python script, after creating a process for the execution of python script.

The python script accepts two arguments, based on which the corresponding variables will be assigned. The first argument, which consists the path to the selected file, will be assigned to the variable, which will be later used to open the file, read the data present in it and then encrypt the data present in it. The encrypted data will be later saved in a particular folder. Once the encryption is done, the time taken for encryption and the peak memory taken is printed.

After the encryption, the function corresponding to the decryption is called. The encrypted file is opened, read and based on the calculated initialization vector and key; decryption is done. Once the decryption is done, the time taken for the process of decryption and the peak memory utilized is also displayed.

It is to be noted that, based on the parameters required for the encryption and decryption modes, the

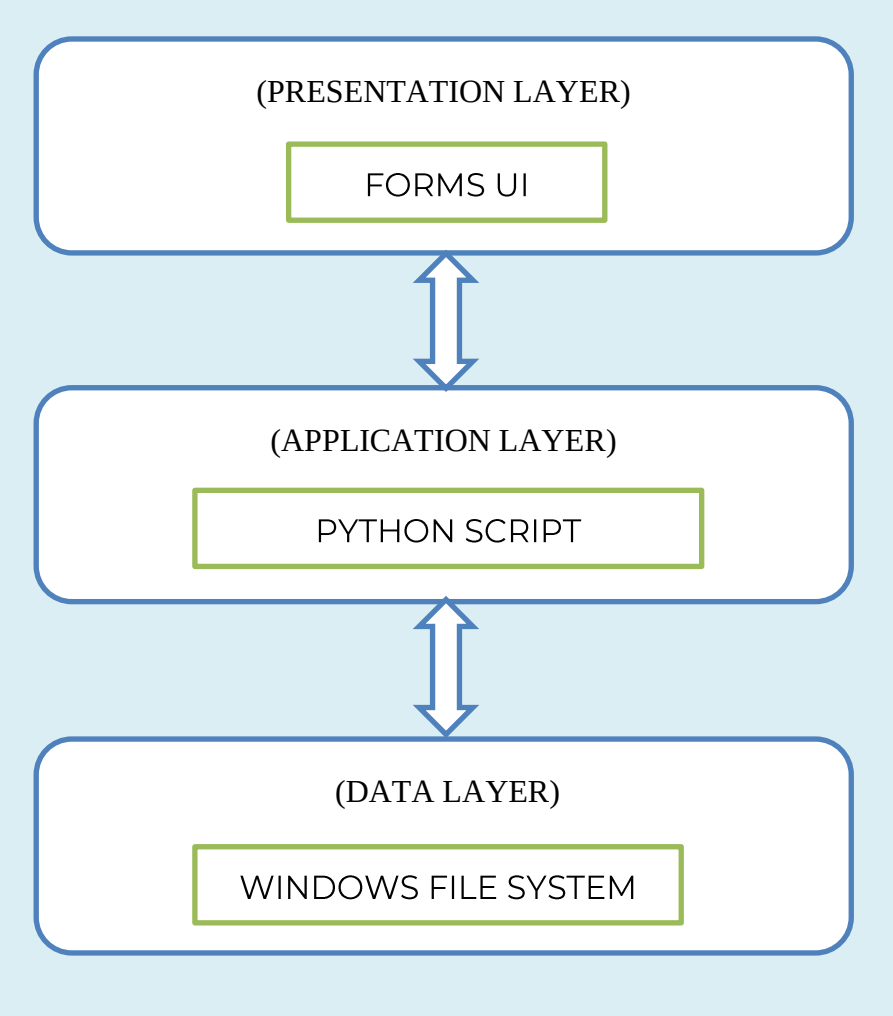



**Figure 3.1: Architecture of Proposed System**

corresponding functions will be called. Also, as per the requirements of the encryption and decryption functions of various modes, block size is assigned.

# CHAPTER 4

# EXPERIMENTS AND RESULTS

# 4. EXPERIMENTS AND RESULTS

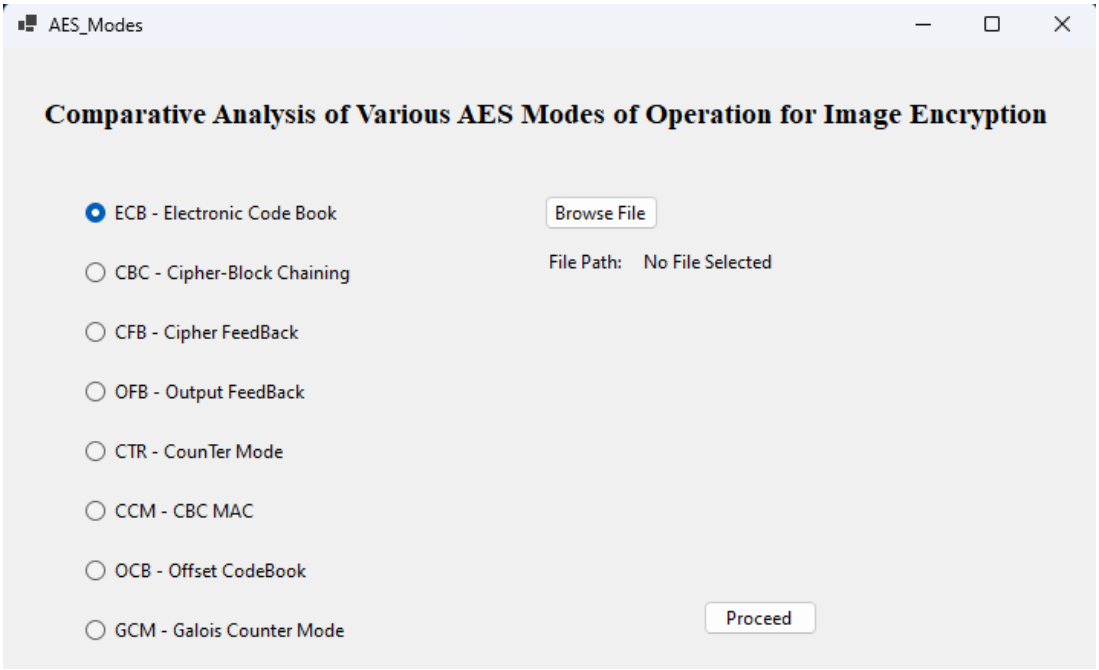



**Figure 4.1: UI of the Application**

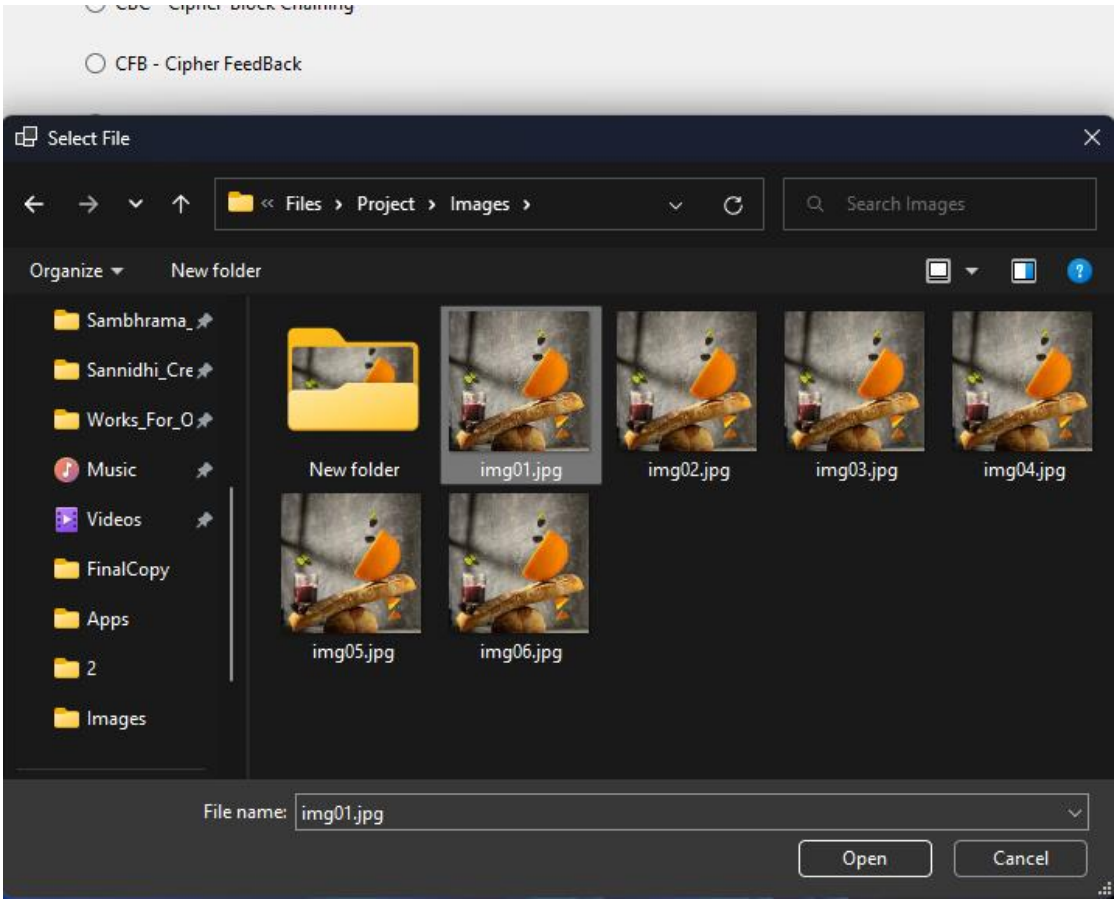



**Figure 4.2: Selection of File**

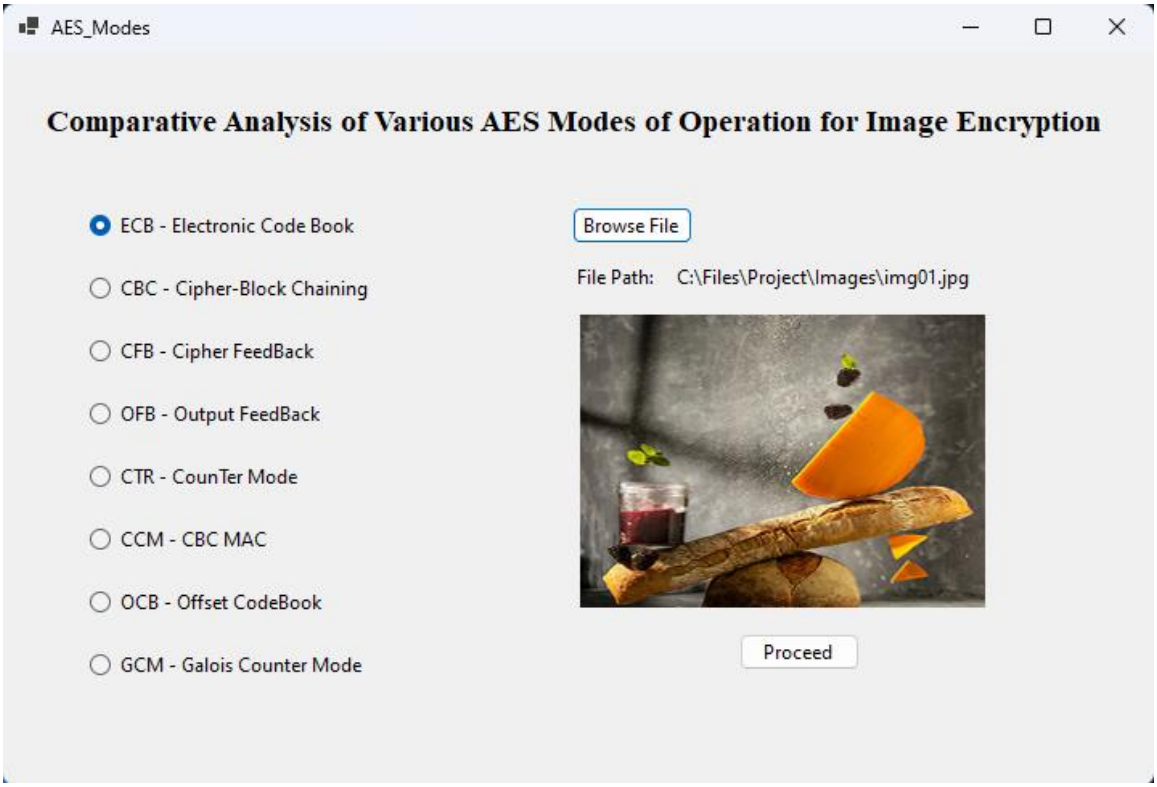



**Figure 4.3: Loading of Image**

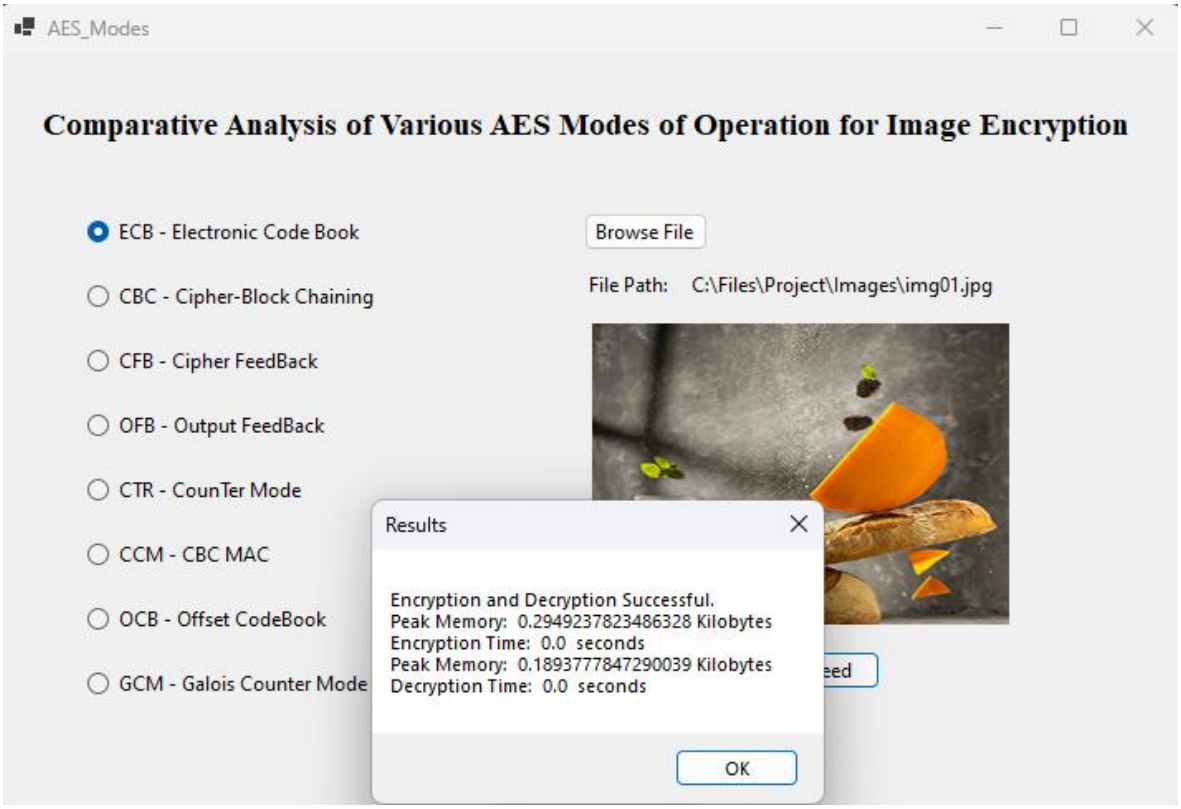



**Figure 4.4: Results of Encryption and Decryption**

The following results show the effect of changing file size on the performance in each mode.

*A. Performance Evaluation Based on Encryption Time*

Figure 4.1 shows the performance of the modes of operation in terms of encryption time. The encryption time of these modes for different file sizes is plotted. We can observe that ECB takes less time than other modes. Moreover, we can notice that the encryption time for CTR and GCM, encryption time of CBC and OFB, encryption time of CCM and OCB are almost same. The modes CCM and OCB have larger encryption time compared to the modes ECB, CTR, GCM, CFC and OCB, when the file size increase. The CFB mode has the highest encryption time. Generally, the differences between the modes are negligible in small files (less than 10 MB), except for CFB mode. Table 4.1 summarizes the results of the comparative encryption time analysis among all these modes of operation.

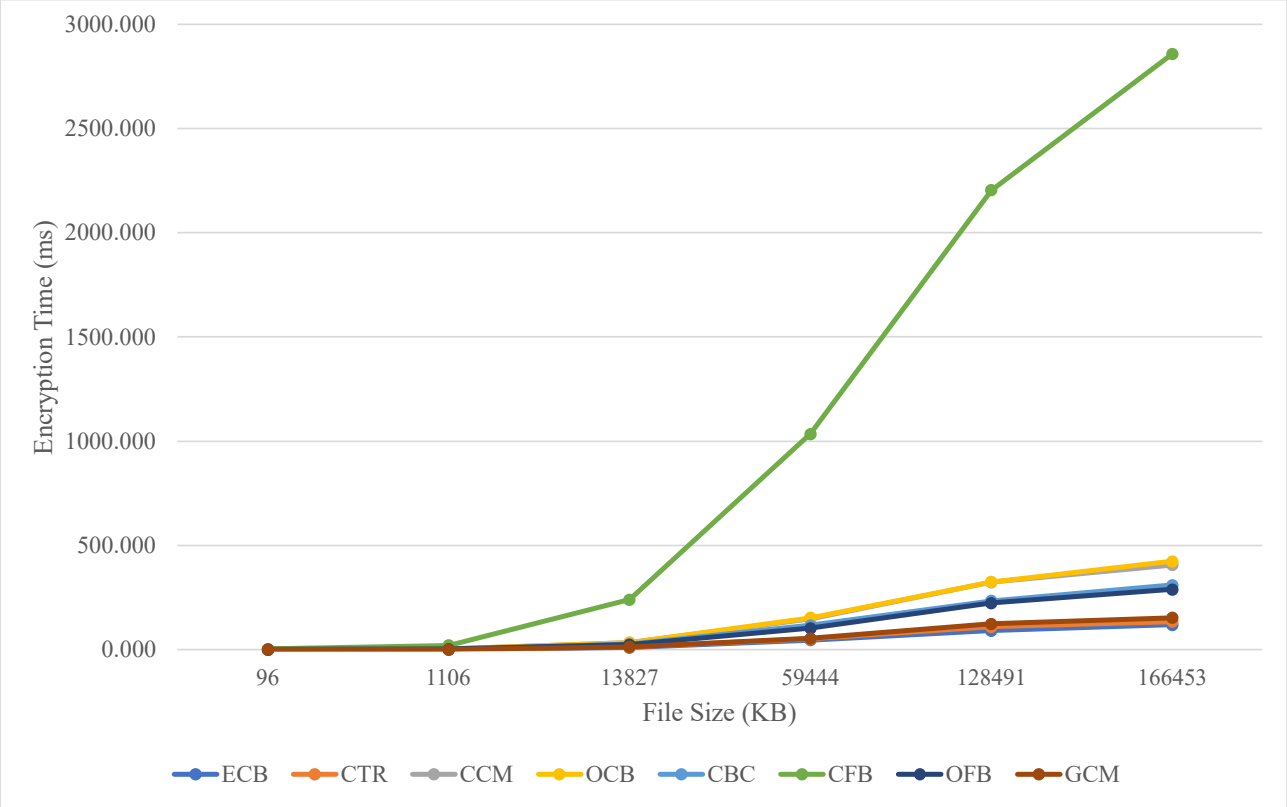



**Figure 4.1: Encryption Time Analysis**

| File Size (KB) | ECB (ms) | CTR (ms) | CCM (ms) | OCB (ms) | CBC (ms) | CFB (ms) | OFB (ms) | GCM (ms) |
|---|---|---|---|---|---|---|---|---|
| 96 | 1.080 | 1.007 | 1.012 | 3.000 | 1.097 | 3.000 | 0.995 | 1.011 |
| 1,106 | 0.991 | 2.002 | 3.036 | 2.998 | 2.000 | 20.820 | 3.000 | 1.002 |
| 13,827 | 10.000 | 11.987 | 34.068 | 35.000 | 26.000 | 240.203 | 23.940 | 14.167 |
| 59,444 | 46.732 | 52.575 | 148.246 | 153.508 | 117.738 | 1034.368 | 103.862 | 55.084 |
| 128,491 | 91.731 | 110.609 | 323.362 | 324.907 | 234.477 | 2203.607 | 223.991 | 123.783 |
| 166,453 | 119.902 | 134.650 | 407.018 | 423.605 | 309.621 | 2856.766 | 289.638 | 153.470 |

**Table 4.1: Encryption Time Comparison**

*B. Performance Evaluation Based on Decryption Time*

In Figure 4.2 decryption time for the same file sizes as we did for encryption time is plotted to show the performance of these modes in terms of decryption time. We can observe that ECB again takes less time than other modes for files of sizes above 1 MB. Moreover, we can notice that there is similar trend as in the encryption time. The decryption time for CTR and GCM, decryption time of CBC and OFB, decryption time of CCM and OCB are almost same The modes CCM and OCB have larger decryption time compared to the modes ECB, CTR, GCM, CFC and OCB, when the file size increase. The CFB mode has the highest decryption time. Generally, the differences between the modes are negligible in small files (less than 10 MB), except for CFB mode. Table 4.2 summarizes the results of the comparative decryption time analysis among all these modes of operation.

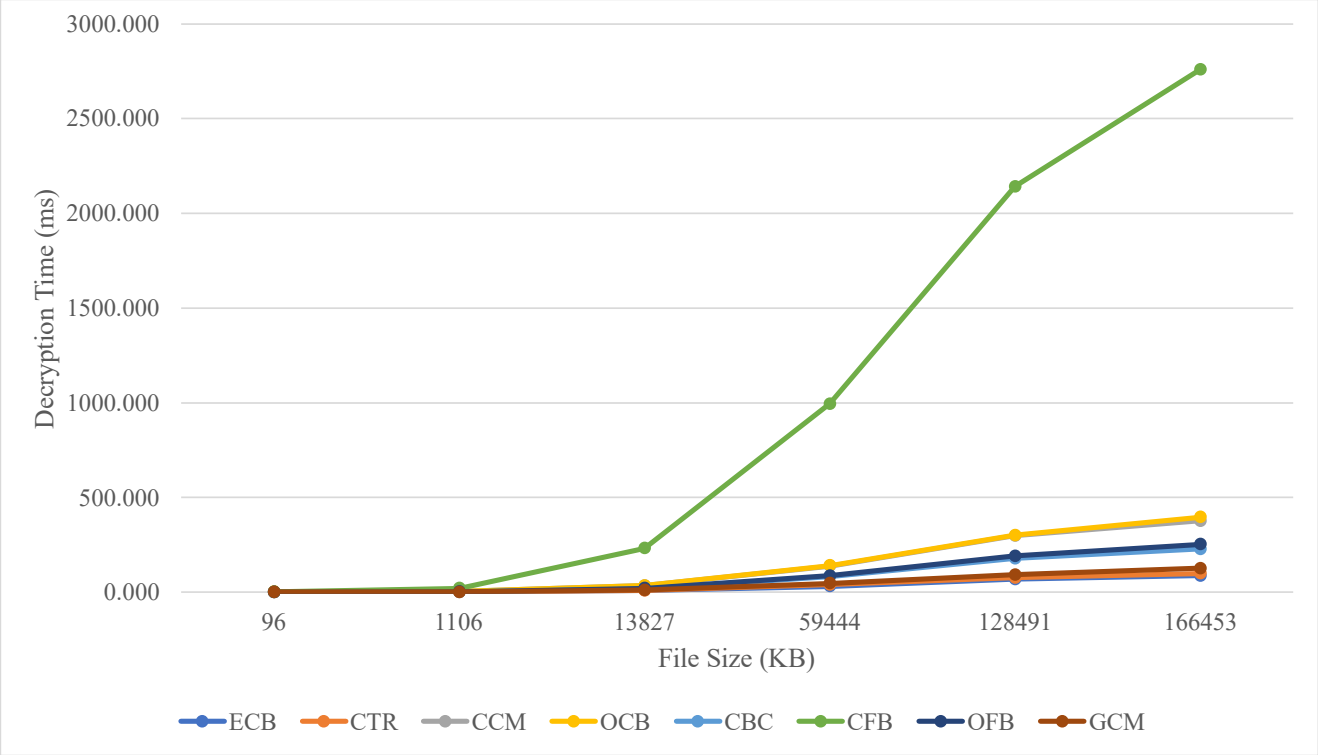



**Figure 4.2: Decryption Time Analysis**

| File Size (KB) | ECB (ms) | CTR (ms) | CCM (ms) | OCB (ms) | CBC (ms) | CFB (ms) | OFB (ms) | GCM (ms) |
|---|---|---|---|---|---|---|---|---|
| 96 | 1.180 | 0.991 | 1.507 | 2.000 | 1.003 | 1.096 | 0.100 | 1.022 |
| 1,106 | 0.999 | 1.000 | 2.978 | 3.051 | 2.000 | 20.382 | 2.000 | 1.005 |
| 13,827 | 9.905 | 7.993 | 35.397 | 34.820 | 17.991 | 231.613 | 20.128 | 10.512 |
| 59,444 | 31.027 | 40.648 | 137.619 | 140.170 | 82.607 | 994.698 | 86.167 | 46.689 |
| 128,491 | 68.308 | 76.334 | 298.550 | 301.119 | 178.003 | 2142.080 | 190.796 | 92.533 |
| 166,453 | 87.550 | 98.179 | 376.547 | 396.108 | 228.467 | 2760.766 | 252.052 | 126.299 |

**Table 4.2: Decryption Time Comparison**

*C. Performance Evaluation Based on Throughput*

In our experiment, we use AES with a fixed size key of 128 bit. The difference of the throughput for encryption and decryption among the modes of operation is relatively small in general. The encryption and decryption throughput of these modes are shown in Figure 4.3 and 4.4 respectively. We can notice that the ECB mode is the fastest among the other modes of operation, and therefore, consumes less power than the other modes.

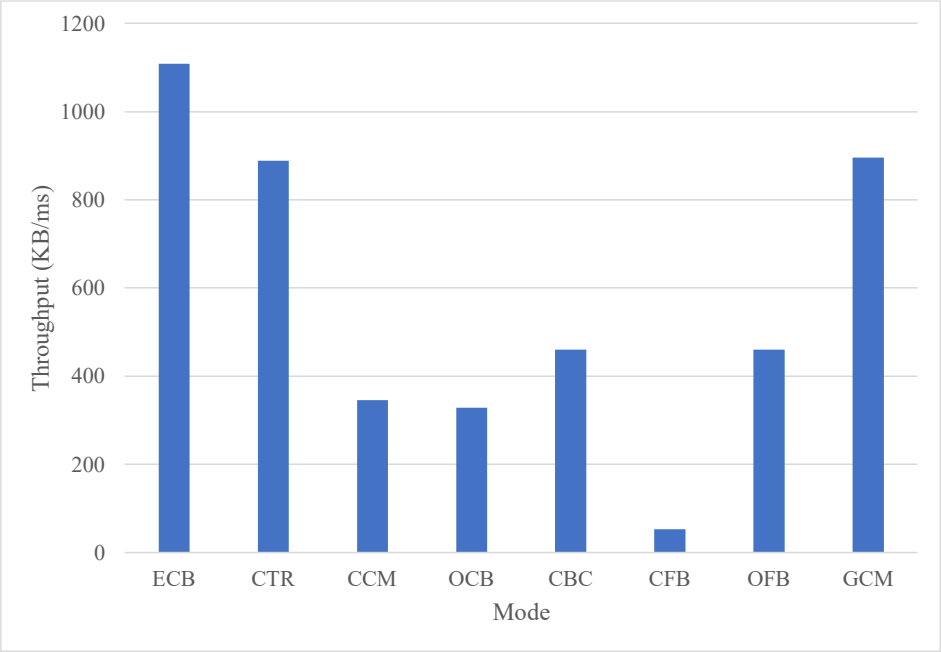



**Figure 4.3: Comparison of Encryption Throughput**

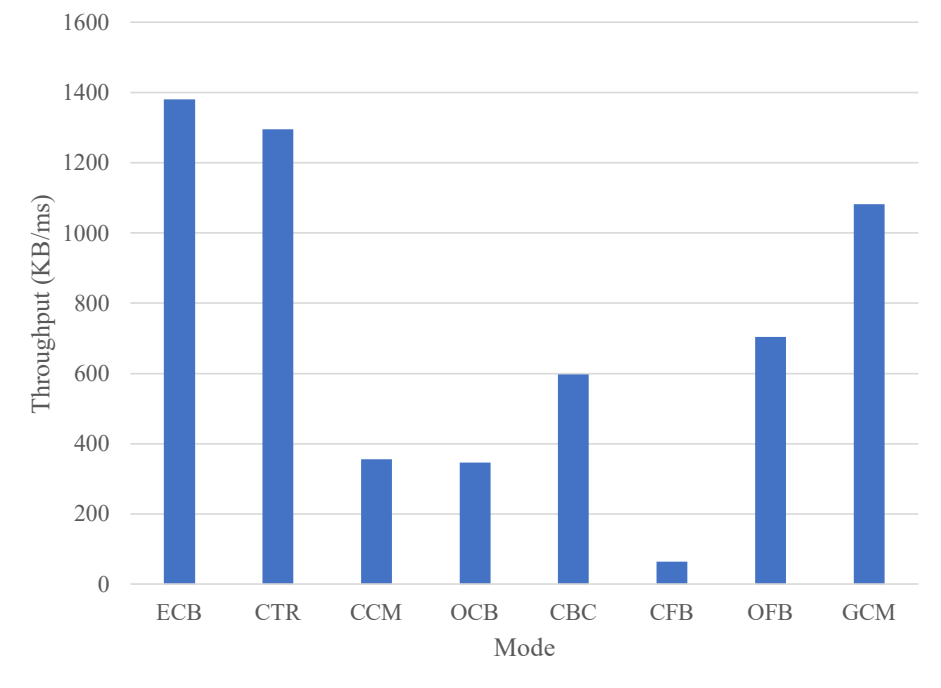



**Figure 4.4: Comparison of Decryption Throughput**

| Mode | Encryption Throughput (KB/ms) | Decryption Throughput (KB/ms) |
|---|---|---|
| ECB | 1108.13 | 1380.45 |
| CTR | 888.27 | 1295.61 |
| CCM | 345.38 | 355.03 |
| OCB | 328.60 | 346.44 |
| CBC | 460.47 | 597.89 |
| CFB | 52.79 | 63.60 |
| OFB | 460.56 | 703.69 |
| GCM | 896.12 | 1081.51 |

**Table 4.3: Throughput Numerical Analysis**

Table 4.3 summarizes the numerical results of the comparative analysis of the encryption and decryption throughput. We can observe that CCM and CFB modes have quite the same throughput or encryption and decryption.

# CHAPTER 5

# CONCLUSION AND FUTURE WORKS

# 5. CONCLUSION AND FUTURE WORKS

In this experiment, a detailed comparison of the various block cipher modes of operation on AES in terms of encryption time, decryption time and throughput are presented. The modes of operation we targeted in this study are ECB, CBC, CFB, OFB and CTR as recommended by NIST for block ciphers, along with OCB, GCM and CCM, which are also having good performance. Our study shows that ECB takes less time to encrypt and decrypt than the other modes. The difference between the modes is relatively small for small files. However, with big size files, there is noticeable difference in the performance among these modes.

In case of image encryption using the modes ECB, CBC, CFB, OFB, CTR, OCB, GCM and CCM, it is also observed that the encrypted file is of same size as that of original file. Even the decrypted file has same size as that of the original file. The peak memory usage during the encryption and decryption of image in these modes is almost same for all these modes.

The comparative analysis can be extended to other modes of operations of AES algorithm. The comparative analysis can be conducted for various modes in different algorithms such as Blowfish algorithm and so on., The results of the study can be future used to suggest the user of appropriate algorithm based on the file selected and purpose. Faster algorithms can be used for web-based applications and purpose, whereas, more safer algorithms are needed for encrypting medical images. Algorithms which depreciate the quality of images are fine to be used for social media and other web uses, whereas, no detail must be lost during encryption and decryption of documents, medical images and so on.,

The results for further analysis can be used to automate the process of choosing an appropriate algorithm and appropriate mode of operation for encrypting images. The concepts of Machine Learning can also be implemented for this purpose.

# REFERENCES


i) S. Almuhammadi and I. Al-Hejri, "A comparative analysis of AES common modes of operation," 2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE), 2017, pp. 1-4, doi: 10.1109/CCECE.2017.7946655.

ii) Dworkin, M., Barker, E., Nechvatal, J., Foti, J., Bassham, L., Roback, E. and Dray, J. (2001), Advanced Encryption Standard (AES), Federal Inf. Process. Stds. (NIST FIPS), National Institute of Standards and Technology, Gaithersburg, MD.

iii) https://pycryptodome.readthedocs.io/en/latest/src/cipher/aes.html

iv) Blazhevski, Dobre & Божиновски, Адријан & Stojcevska, Biljana & Pachovski, Veno. (2013). MODES OF OPERATION OF THE AES ALGORITHM.

v) https://csrc.nist.rip/groups/ST/toolkit/BCM/documents/proposedmodes/gcm/gcm-spec.pdf

vi) W. Stallings, Cryptography and network security: principles and practices. Pearson Education India, 2006.

vii) Karthigaikumar, P., and Soumiya Rasheed. "Simulation of image encryption using AES algorithm." IJCA special issue on “computational science-new dimensions & perspectives” NCCSE (2011): 166-172.

viii) Padate, Roshni, and Aamna Patel. "Image encryption and decryption using aes algorithm." International Journal of Electronics and Communication Engineering & Technology (2015): 23-29.

ix) https://www.techtarget.com/searchsecurity/definition/encryption

x) https://www.khanacademy.org/computing/computers-and-internet/xcae6f4a7ff015e7d:online-data-security/xcae6f4a7ff015e7d:data-encryption-techniques/a/symmetric-encryption-techniques

xi) https://www.educative.io/answers/what-is-the-aes-algorithm

xii) https://www.highgo.ca/2019/08/08/the-difference-in-five-modes-in-the-aes-encryption-algorithm/

xiii) https://infocenter.nordicsemi.com/index.jsp?topic=%2Fcom.nordic.infocenter.nrf52832.ps.v1.1%2Fccm.html

xiv) https://web.cs.ucdavis.edu/~rogaway/ocb/ocb-faq.htm