



Tribhuvan University

Institute of Science and Technology

**Comparative Study of CAST and TWOFISH algorithm using
various Modes of Operations**

Thesis

Submitted to

Central Department of Computer Science and Technology

Kirtipur, Kathmandu, Nepal

In partial fulfillment of the requirements

For the Master's Degree in Computer Science and Information Technology

By

Sabita Maharjan

Roll No.:118/2070

T.U. Regd. No.: 5-2-33-481-2007

Feb, 2020

Supervisor

Mr. Jagdish Bhatta



Tribhuvan University

Institute of Science and Technology

Central Department of Computer Science and Information Technology

Student's Declaration

I hereby declare that I am the only author of this work and that no sources other than listed here have been used in this work.

Sabita Maharjan

Feb, 2020



Tribhuvan University

Institute of Science and Technology

Central Department of Computer Science and Information Technology

Supervisor's Recommendation

I hereby recommend that this thesis prepared under my supervision by **Ms. Sabita Maharjan** titled “**Comparative Study of CAST and TWOFISH algorithm using various Modes of Operations**” in partial fulfillment of the requirements for the degree of MSc in Computer Science and Information Technology be processed for the evaluation.

Asst. Prof. Jagdish Bhatta

Central Department of Computer Science and Information Technology,

Tribhuvan University,

Kathmandu, Nepal

(Supervisor)

Feb, 2020



Tribhuvan University

Institute of Science and Technology

Central Department of Computer Science and Information Technology

LETTER OF APPROVAL

We certify that, we have read this thesis and in our opinion it is satisfactory in the scope and quality as a thesis in partial fulfillment for the requirement of Master's Degree in Computer Science and Information Technology.

Evaluation Committee

Asst. Prof. Jagdish Bhatta
Central Department of CSIT
Tribhuvan University
Kirtipur, Kathmandu, Nepal
(Supervisor)

Asst. Prof. Nawaraj Poudel
Central Department of CSIT
Tribhuvan University
Kirtipur, Kathmandu, Nepal
(Head of Department)

External Examiner

Internal Examiner

Acknowledgement

I am deeply thankful to my supervisor **Asst. Prof. Jagdish Bhatta** (Tribhuvan University) for his generous advice, inspiring guidance and encouragement throughout my research for this thesis. Without his kind and patient review of this work, it would have been impossible to complete this study.

I would like to extend my gratitude to **Asst. Prof. Nawaraj Paudel** (Head of Department, CDCSIT) and faculties for their guidance and help throughout my Masters Study and help for the completion of my thesis.

Last but not the least; I would like to express my gratitude to all my family members, friends and all other people who have helped me directly or indirectly in the completion of this thesis.

Yours Obediently

Sabita Maharjan

(Msc. Computer Science and Information Technology)

Abstract

Encryption is a process to encode a message, file, image or video data (intelligent data) to convert it into a cipher data (i.e. non-intelligent data). This study is about to encrypt data (text and image file) to analyze the cipher text which was produced by the given algorithms (CAST and Twofish) in the scenario of memory consumption and time taken to encrypt it and vice-versa. Here file encryption is used to encrypt the data file while storing data in local drive for the security purposes which is achieved by converting data into cipher data by implementing different versions of CAST and Twofish encryption algorithm, which is then analyzed through their performance (time and memory) analysis. CAST encryption and Twofish encryption algorithm are used with block cipher modes of operations to implement and analyze. ECB (Electronic Code Book), CBC (Cipher Block Chaining), CFB (Cipher Feedback Mode), OFB (Output Feedback Mode and CTR (Counter Mode) are the modes of operations used in this study. Text and image files of different sizes are input for this study and different performance parameters like PSNR, NPCR, UACI, Histogram, and encryption / decryption time are used to measure the strength of the algorithms. Based on the analysis done during this study, it is found that for the image and text data, CAST-128 algorithm is found to be approximately three times better in comparison with other algorithm on encryption and decryption time analysis. Similarly Twofish algorithm is found to be better for the case of throughput analysis for both text and image data. In terms of memory utilization, Twofish-128 is found to be consuming less memory compared to other algorithms. Similarly, visual assessment and differential analysis for image data analysis, all the algorithm seems to be performing best since no difference in original and retrieved image data. And for the case of statistical analysis, there seems to be no difference in original and retrieved image histogram.

Keywords: encryption, CAST, Twofish, block modes of operation.

TABLE OF CONTENTS

CHAPTER 1	1
INTRODUCTION	1
1.1. Introduction	1
1.2. Statement of Problem	3
1.3. Objective.....	3
1.4. Report Organization	4
CHAPTER 2	5
BACKGROUND STUDY AND LITERATURE REVIEW	5
2.1. Background Study	5
2.1.1. Encryption and decryption	5
2.1.2. File Encryption	6
2.1.3. Modes Of Operation.....	7
2.1.4. Block Encryption Modes.....	8
2.1.5. Stream Encryption Modes	9
2.2. Literature Review	12
CHAPTER 3	18
METHODOLOGY	18
3.1. Methodology.....	18
3.2. CAST encryption algorithm	19
3.3. TWOFISH encryption algorithm.....	22
3.4. Performance Evaluation Parameters.....	28
3.4.1. Computational analysis	28

3.4.2.	Differential analysis	29
3.4.3.	Visual assessment analysis.....	30
3.4.4.	Statistical Analysis	30
CHAPTER 4		31
IMPLEMENTATION AND ANALYSIS		31
4.1.	Implementation.....	31
4.1.1.	C# programming language	31
4.1.2.	Microsoft .NET Framework.....	31
4.1.3.	Matlab R2018a Overview	32
4.2.	Test Data Description	32
4.3.	Analysis	33
4.3.1.	COMPUTATIONAL ANALYSIS.....	33
4.3.2.	Differential analysis	50
4.3.3.	Statistical analysis	51
4.3.4.	Visual assessment analysis.....	56
4.4.	Result	58
CHAPTER 5		60
CONCLUSION LIMITATIONS AND FUTURE RECOMMENDATION		60
5.1.	Conclusion	60
5.2.	Limitations.....	61
5.3.	Future Recommendation.....	61
References.....		62

LIST OF TABLES

Table 1: Algorithms' Settings	19
Table 2: NPCR and UACI Measures.....	51
Table 3: Histogram Analysis	52
Table 4: PSNR Measures	57

LIST OF FIGURES

Figure 1: Conventional Encryption Model.....	5
Figure 2: Electronic Codebook (ECB) mode encryption	8
Figure 3: Cipher Block Chaining (CBC) mode encryption	9
Figure 4: Output Feedback (OFB) mode encryption	10
Figure 5: Counter (CTR) mode encryption	11
Figure 6: Cipher Feedback (CFB) mode encryption	12
Figure 7: Encryption algorithm process	18
Figure 8: Flowchart of CAST-128 Encryption.....	20
Figure 9: Flowchart of Twofish Algorithm	23
Figure 10: Encryption time analysis for ECB mode for text file.....	33
Figure 11: Encryption time analysis for CBC mode for text file.....	34
Figure 12: Encryption time analysis for CFB mode for text file	34
Figure 13: Encryption time analysis for OFB mode for text file.....	34
Figure 14: Encryption time analysis for CTR mode for text file.....	35
Figure 15: Encryption time analysis for ECB mode for image file	35
Figure 16: Encryption time analysis for CBC mode for image file	36
Figure 17: Encryption time analysis for CFB mode for image file	36
Figure 18: Encryption time analysis for OFB mode for image file	36
Figure 19: CAST-128 image encryption and decryption time for CTR mode	37
Figure 20: Decryption time analysis for ECB mode for text file.....	37
Figure 21: Decryption time analysis for CBC mode for text file	38
Figure 22: Decryption time analysis for CFB mode for text file.....	38
Figure 23: Decryption time analysis for OFB mode for text file.....	38
Figure 24: Decryption time analysis for CTR mode for text file.....	39
Figure 25: Decryption time analysis for ECB mode for image file	39
Figure 26: Decryption time analysis for CBC mode for image file.....	40
Figure 27: Decryption time analysis for CFB mode for image file	40
Figure 28: Decryption time analysis for OFB mode for image file	40

Figure 29: Decryption time analysis for CTR mode for image file	41
Figure 30: Throughput for encryption and decryption for ECB mode (Text)	41
Figure 31: Throughput for encryption and decryption for CBC mode (Text)	42
Figure 32: Throughput for encryption and decryption for CFB mode (Text)	42
Figure 33: Throughput for encryption and decryption for OFB mode (Text)	42
Figure 34: Throughput for encryption and decryption for CTR mode (Text)	43
Figure 35: Throughput for encryption and decryption for ECB mode (Image).....	43
Figure 36: Throughput for encryption and decryption for CBC mode (Image)	44
Figure 37: Throughput for encryption and decryption for CFB mode (Image).....	44
Figure 38: Throughput for encryption and decryption for OFB mode (Image).....	44
Figure 39: Throughput for encryption and decryption for CTR mode (Image).....	45
Figure 40: Memory Utilization Graph for text data in ECB mode	45
Figure 41: Memory Utilization Graph for text data in CBC mode	46
Figure 42: Memory Utilization Graph for text data in CFB mode	46
Figure 43: Memory Utilization Graph for text data in OFB mode	46
Figure 44: Memory Utilization Graph for text data in CTR mode	47
Figure 45: Memory Utilization Graph for image data in ECB mode	47
Figure 46: Memory Utilization Graph for image data in CBC mode	48
Figure 47: Memory Utilization Graph for image data in CFB mode	48
Figure 48: Memory Utilization Graph for image data in OFB mode	49
Figure 49: Memory Utilization Graph for image data in CTR mode	49

List of Aberrations

ECB	:	Electronic Codebook Mode
CBC	:	Cipher Block Chaining Mode
CFB	:	Cipher Feedback Mode
OFB	:	Output Feedback Mode
CTR	:	Counter Mode
NPCR	^:	Number of Pixel Change Rate
UACI	:	Unified Average Change Intensity
PSNR	:	Peak Signal to Noise Ratio
MSE	:	Mean Square Error
XOR	:	Exclusive-OR
USB	:	Universal Serial Bus
IoT	:	Internet of Things
AES	:	Advanced Encryption Standard
DES	:	Data Encryption Standard
3-DES	:	Triple- DES
RC2	:	Rivest Cipher2
RC4	:	Rivest Cipher4
RC5	:	Rivest Cipher5
RC6	:	Rivest Cipher6
GPU	:	Graphics Processing Unit
DDoS	:	Distributed Denial of Service
NVM	:	Non Volatile Memory
XTS	:	Cipher Text Stealing
BCM	:	Block Cipher Mode of Operation

OPC	:	Output Protection Chain
VHDL	:	Very High speed integrated circuit hardware Description Language
MDS	:	Maximum Distance Separable
PHT	:	Pseudo-Hadamard Transform
MATLAB	:	Matrix Laboratory
TIF	:	Tagged Image File
BMP	:	Bit map
PNG	:	Portable Network Graphics
JPG/JPEG	:	Joint Photographic Group / Joint Photographic Experts Group
CPU	:	Central Processing Unit
NASA	:	National Aeronautics and Space Administration
CAST	:	Carlisle Adams and Stafford Tavares
.NET	:	.Network Enabled Technologies
RAM	:	Random Access Memory
ECMA	:	European Computer Manufacturer's Association
ISO	:	International Standards Organization
XML	:	extensible markup language
VB.NET	:	Visual Basic .NET
XEX	:	XOR Encrypt XOR

CHAPTER 1

INTRODUCTION

1.1. Introduction

The development of digital information and telecommunication systems has opened a wide range of new possibilities which were seized to improve the efficiency of different sorts of processes. The success of these new technologies can be attributed to a number of intrinsic advantages of digital systems: digital information is nearly insensitive to noise, it can be sent over long distances, copied or modified without any loss of quality. However, the same properties which make digital information systems so attractive render them particularly vulnerable to a broad range of abuses. Securing digital data is needed to protect the confidentiality, integrity, authenticity, and availability of data only to the intended recipient. The only way to secure digital systems without sacrificing their advantages, is to transform the information in such a way that it protects itself, independently of how it is transferred or stored. This is more common in banking and other sectors as well where companies don't want to disclose their sensitive information to be hacked or leaked to unauthorized users. So there is a need to encrypt files on a computer to resist the adversary's attempts to read the contents of the file. File encryption provides security for files. This is useful for particularly sensitive files, but is also useful for application-level transfer of files across an insecure channel such as email. With the increase in file size, along with security strength, the file encryption should be of better computational efficiency. Any suitably secure modern symmetric cipher can be used as part of a file encryption mechanism. File encryption usually uses block ciphers. [1].

The protection of digital information typically involves at least two distinct problems: secrecy protection (preventing information from being disclosed to unintended recipients) and authentication (ensuring that received messages originate from the intended sender, and were not modified on their way). In cryptology, intended senders and recipients are distinguished from unintended ones by assuming that they know some secret pieces of information, called keys. These keys can be shared between the sender and the receiver, or they can be different, in which case the sender and receiver

are also prevented from impersonating each other. In this thesis, we will concentrate on the first case, called symmetric cryptography. [2]

Symmetric cryptography addresses the problem of secrecy protection by using the shared secret key to transform the message in such a way that it cannot be recovered any more without this key. This process is called symmetric encryption. Symmetric encryption is a form of computerized cryptography using a singular encryption key to disguise an electronic message. Algorithms which perform symmetric encryption are known as ciphers. The trust in a cipher is merely based on the fact that no weaknesses have been found after a long and thorough evaluation phase. This explains the importance of a strong interaction between cryptography, the field which studies techniques to protect information, and cryptanalysis, which focuses on methods to defeat this protection. Symmetric encryption is also known as private-key encryption and secure-key encryption. Due to the better performance and faster speed of symmetric encryption, symmetric cryptography is typically used for bulk encryption / encrypting large amounts of data [3].

Based on the paradigm used to process the message, ciphers are typically categorized into one of two classes: block ciphers and stream ciphers. A block cipher processes the data blocks of fixed size. Usually, the size of a message is larger than the block size. Hence, the long message is divided into a series of sequential message blocks, and the cipher operates on these blocks one at a time. Most block ciphers can work with different keys and data size. It uses a symmetric key to encrypt data of fixed and very short length (*the block size*). In order to cope with data of arbitrary length, the cipher must be combined with a *mode of operation*. The mode of operation may also provide application of the block cipher on a stream of plaintext and make the algorithm more efficient. On the other hand, the mode of operation may convert the block cipher into a stream cipher and also to strengthen the effect of the encryption algorithm. Each mode of operation has its own parameters which are important to provide the necessary security of the algorithm. [2]

1.2. Statement of Problem

Security of information in transmission medium is most prime issue in the research field. Various security mechanisms like authentication, digital signatures, and cryptographic algorithms are used to protect messages from unauthorized attacks. Achieving security with low cost computation is integral part of cryptography. Maintaining the reliability, security, and discretion of secret information is a critical issue.

Without use of any block modes of operation, there is high chance of encrypting identical block of text in the same way which is prone to cryptanalysis. In addition, an encryption process with less computational efficiency, despite of having higher degree of security features would be impractical for real time processes. There are many modes of operation for specific purposes, including network traffic protection, hard drive encryption, etc. Using modes of operation with symmetric block cipher encryption provide facilities of encryption of large messages by dividing the message into fixed block length so that similar block of input will even result varied cipher text. This also reduces cryptanalysis. This is why they are often used in situations where there is a lot of data that needs to be encrypted. This is to mask the patterns which exist in encrypted data. At the same time, determining the level efficiency of particular mode on particular encryption is significant aspect.

1.3. Objective

The objective of this study is:

- To implement CAST and TWOFISH with ECB, CBC, CFB, OFB and CTR block modes for file encryption.
- To perform differential, statistical, visual assessment and computational analyses using the parameters NPCR (Number of Pixel Change Rate), UACI (Unified Average Change Intensity), Histogram, PSNR (Peak Signal to Noise Ratio) and encryption/decryption time respectively.

1.4. Report Organization

The organization of this thesis is as follows:

Chapter 1 consists of introduction, problem statement and objectives.

Chapter 2 describes about the background study for the research and literature review of the related work by different authors.

Chapter 3 describes the overview of the methodology of CAST and Twofish algorithm encryption with different modes of operation.

Chapter 4 describes the implementation of the algorithms and the data set description together with the experimental result of different techniques and comparison using different measures.

Chapter 5 contains the conclusions of this research work and the directions for the future works.

CHAPTER 2

BACKGROUND STUDY AND LITERATURE REVIEW

2.1. Background Study

Encryption and decryption, file encryption and block cipher modes of operation are discussed in background study which are given below:

2.1.1. Encryption and decryption

Encryption is the process of converting normal text to unreadable form. Decryption is the process of converting encrypted text to normal text in the readable form. [4]

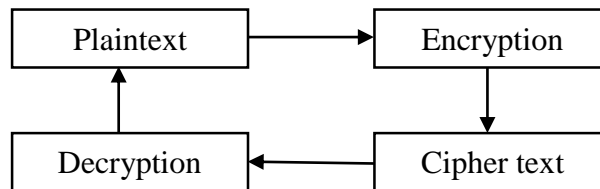


Figure 1: Conventional Encryption Model

Encryption is one of the most reliable methods used to protect data confidentiality and integrity even since the old days. Data encryption is the process of converting data in plain text format into a meaningless cipher text by means of a suitable algorithm. Data decryption is the process of converting the meaningless cipher text into the original information using keys generated by the encryption algorithms. The process of encryption and decryption of information by using a single key is known as secret key cryptography or symmetric key cryptography. In symmetric key cryptography, the same key is used to encrypt as well as decrypt the data. A secure channel is also required between the sender and the receiver to exchange the secret key. Two ciphers modes are adopted by symmetric algorithms: Block ciphers and Stream ciphers. A block cipher is functioning on fixed-length groups of bits, called blocks, with an unvarying transformation that is specified by a symmetric key. Feistel structure is adopted by many block ciphers. Such a structure consists of a number of identical rounds of processing. In each round, a substitution is performed on one half of the data being processed, followed by a permutation that interchanges the two halves. The original key is expanded so that different key is used for each round.

In Asymmetric key cryptography different keys are used for encryption and decryption. Asymmetric cryptography refers to a cryptographic algorithm which requires two separate keys, one of which is secret (or private) and one of which is public. Although different, the two parts of this key pair are mathematically linked. The public key is used to encrypt plaintext or to verify a digital signature; whereas the private key is used to decrypt cipher text or to create a digital signature. Asymmetric encryption techniques are known to be slower than Symmetric encryption which makes it impractical when trying to encrypt large amounts of data. Also to get the same security strength as symmetric, asymmetric must use a stronger key than symmetric encryption technique. [5]

Symmetric encryption techniques are further classified into Block Ciphers and Stream Ciphers.

Stream Ciphers

Stream Cipher algorithms peruse the entire intelligible message and convert each symbol of the plain text directly into a symbol of cipher text. The symbol is generally a bit, and the transformation performed is generally exclusive-OR (XOR). Due to bit by bit encoding, they are lighter and quicker schemes relying solely on confusion concepts. They also have statistically random structures and are easier to implement on hardware.

Block Ciphers

Block Cipher cryptographic schemes convert an entire block of plain text into a block of cipher text at a time. These are bulkier and slower ciphers as they involve the division of plain text into blocks and rely on both diffusion and confusion concepts. They have a simpler software implementation and also have distinct modes of operations. [6]

2.1.2. File Encryption

File Encryption means providing security for files that reside on media or in a stored state. Those are files that are resting on our hard drives, USB drives or any other type of digital media storage. Those are files that are usually not meant to be sent through network, they are stored locally, being encrypted and temporarily decrypted while being

used and then encrypted again after we finished using them. Encrypting stored files prevents others from reading, copying, or deleting encrypted files. Most often, those encrypted files can be seen in a file listing (such as in file explorer), but they cannot be accessed for reading by unauthorized persons. In this thesis, text file and different image file formats are taken for file encryption. [7]

2.1.3. Modes Of Operation

The different ways in which encryption can be achieved are called modes of operation. The purpose of a mode of operation is to extend the cryptographic properties of a block cipher to larger messages. The property which this thesis mainly focuses on is confidentiality, but modes providing message integrity and authenticity, possibly in addition to confidentiality, exist as well. Although security obviously remains the primary criterion, other (non-cryptographic) considerations often play an equally important role in the selection of a mode of operation [2]:

Data expansion: Some constructions require the plaintext length to be an exact multiple of the block length. This implies that the original message may have to be expanded with extra padding bits, which is usually undesirable.

Error propagation: Single bit transmission errors may have different effects on the decrypted cipher text. Either the error only affects a single bit or block of the recovered plain text, or it might propagate to one, a few or all subsequent blocks.

Random access: A number of modes allow cipher text blocks to be decrypted (or even modified) at arbitrary positions without first having to process all preceding blocks. This is particularly useful for storage encryption.

Parallel processing: Some modes allow different blocks to be processed simultaneously, which may be an interesting way to increase the throughput in certain applications.

2.1.4. Block Encryption Modes

Electronic Codebook Mode (ECB): The ECB mode is the most straight forward way to encrypt messages whose length exceed the block length: the message is simply partitioned into n-bit blocks, each of which is encrypted independently [8].

Encryption: $C_i = E_K(P_i)$ Eq (2.1)

Decryption: $P_i = D_K(C_i)$ Eq (2.2)

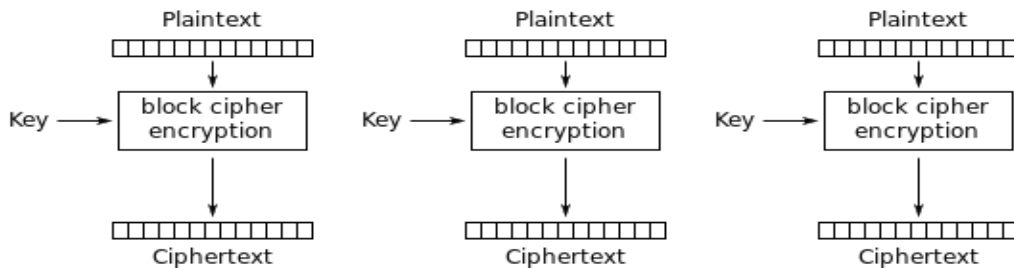


Figure 2: Electronic Codebook (ECB) mode encryption [8]

The advantages of this mode are its simplicity and its suitability for parallel processing. Blocks at arbitrary positions can be encrypted or decrypted separately and errors do not propagate from one block to another. However, the major problem of this approach is that it does not hide all patterns in the plain text: i.e., whenever the plain text contains identical blocks, so will the cipher text. This limits the applications of the ECB mode to those (rare) cases where all blocks encrypted with a single key are guaranteed to be different.

Cipher Block Chaining Mode (CBC): The CBC mode, which is presently the most widely used mode of operation, masks each plain text block with the previous cipher text block before applying the block cipher [8].

Encryption: $C_0 = IV$,Eq (2.3)

$C_i = E_K(C_{i-1} \oplus P_i)$ Eq (2.4)

Decryption: $C_0 = IV$,Eq (2.5)

$P_i = D_K(C_i) \oplus C_{i-1}$ Eq (2.6)

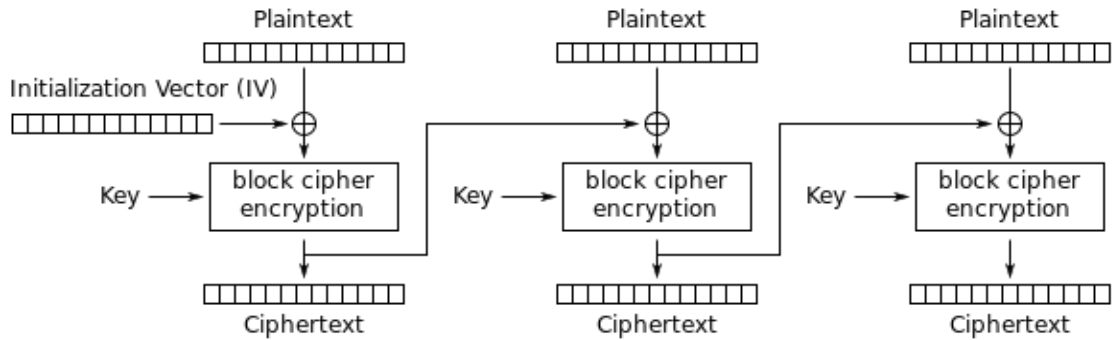


Figure 3: Cipher Block Chaining (CBC) mode encryption [8]

Since the output of a good block cipher is supposed to be completely unpredictable for anyone who does not know the key, all consecutive values of $C_{i-1} \oplus P_i$ will appear to be independent and uniformly distributed, and this regardless of the plain text (assuming that the text itself does not depend on the key). Repetitions at the input of the block cipher are therefore unlikely to occur, which cures the main short coming of the ECB mode. The cost of masking the plain text in CBC is that the cipher text feedback in the encryption part prevents the blocks from being processed in parallel. The decryption, on the other hand, depends only on two consecutive cipher text blocks, and can still be performed independently for each block. This has the additional benefit that a bit error in the cipher text can only affect the decryption of two blocks.

2.1.5. Stream Encryption Modes

Block ciphers can also be used to perform stream encryption, as illustrated by the three modes below. A noteworthy feature of these modes is that they only use the encryption function of the block cipher [2].

Output Feedback Mode (OFB): The OFB mode, encrypts plain text blocks by combining them with a stream of blocks called key stream, which is generated by iterating the block cipher:

$$\text{Encryption: } Z_0 = IV, \dots \text{Eq (2.7)}$$

$$Z_i = EK(Z_{i-1}) \dots \text{Eq (2.8)}$$

$$C_i = P_i \oplus Z_i \dots \text{Eq (2.9)}$$

$$\text{Decryption: } Z_0 = IV, \dots \text{Eq (2.10)}$$

$$Z_i = EK(Z_{i-1}) \dots \dots \dots \text{Eq (2.11)}$$

$$P_i = C_i \oplus Z_i \dots \dots \dots \text{Eq (2.12)}$$

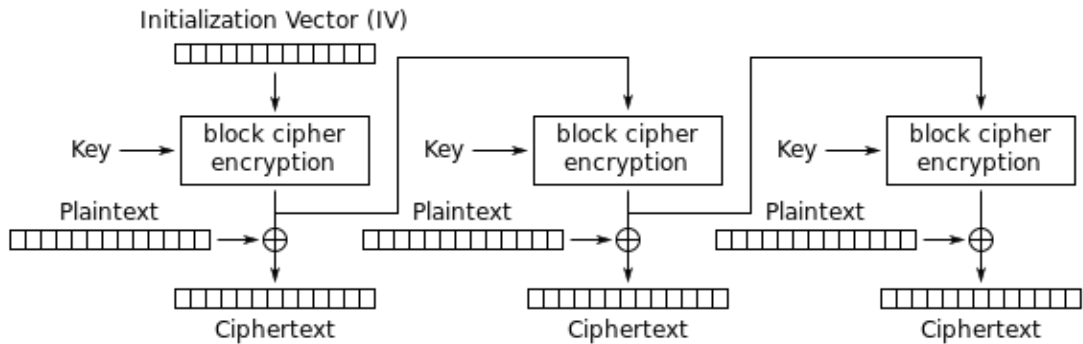


Figure 4: Output Feedback (OFB) mode encryption [8]

The generation of key stream blocks in OFB is independent of the plain text. This means that the stream can be pre-computed as soon as the IV is known, a feature which may be useful in real-time applications. The mode is strictly sequential, though: the decryption of a single block at an arbitrary position in the cipher text requires all preceding key stream blocks to be computed first. Owing to the invertibility of EK, all Z_i will necessarily be different, until one of them hits the value of Z_0 again, at which point the sequence will start repeating itself. A secure n -bit block cipher is not expected to cycle in much less than $2^n - 1$ blocks, which implies that this periodicity has no practical consequences for a typical 128-bit block cipher. The mere fact that all Z_i within a cycle are different leaks some information as well, though. As a consequence, it is not recommended to encrypt much more than $2^{n/2}$ blocks with a single key. [2]

Counter Mode (CTR): The CTR mode takes a similar approach as the OFB mode, but this time the key stream is generated by encrypting a counter:

$$\text{Encryption: } Z_0 = IV, \dots \dots \dots \text{Eq (2.13)}$$

$$C_i = P_i \oplus EK(Z_i) \dots \dots \dots \text{Eq (2.14)}$$

$$Z_{i+1} = Z_i + 1 \dots \dots \dots \text{Eq (2.15)}$$

$$\text{Decryption: } Z_0 = IV, \dots \dots \dots \text{Eq (2.16)}$$

$$P_i = C_i \oplus EK(Z_i) \dots \dots \dots \text{Eq (2.17)}$$

$$Z_{i+1}=Z_{i+1} \dots\dots\dots \text{Eq (2.18)}$$

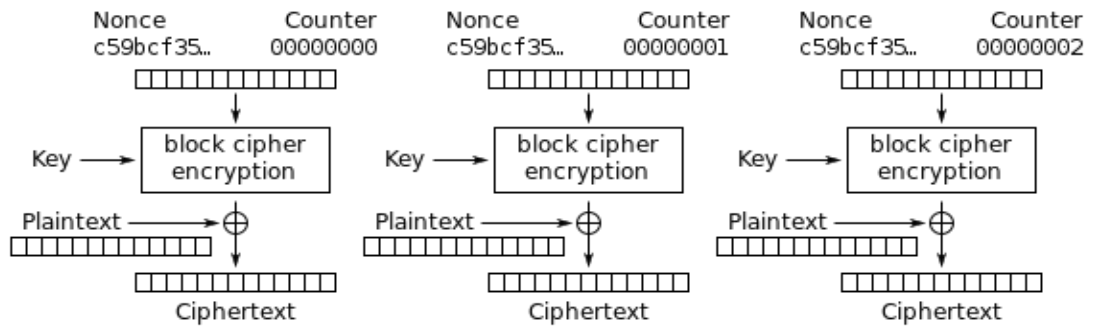


Figure 5: Counter (CTR) mode encryption [8]

As opposed to the OFB mode, the CTR mode allows data blocks at arbitrary positions to be processed independently, both during encryption and decryption. This also allows pipelining in hardware, which can result in significant efficiency gains. Apart from this feature, the OFB and the CTR mode have very similar properties [8].

Cipher Feedback Mode (CFB): Both OFB and CTR require perfect synchronization during decryption, i.e., in order to decrypt a cipher text block, the receiver needs to know the block’s exact position in the stream. The CFB mode eliminates this requirement, and is similar to CBC in this respect. The CFB mode is designed to process messages in r-bit segments, with $1 \leq r \leq n$ (typically $r=1, r=8, r=n$). The encryption mode consists in shifting successive r-bit cipher text segments back into an internal state block S_i , and combining the left most bits of $EK(S_i)$ with the plaintext:

$$\text{Encryption: } S_1=IV, \dots\dots\dots \text{Eq (2.19)}$$

$$C_i=P_i \oplus EK(S_i)[1 \dots r] \dots\dots\dots \text{Eq (2.20)}$$

$$S_{i+1}=(S_i \ll r)+C_i \dots\dots\dots \text{Eq (2.21)}$$

$$\text{Decryption: } S_1=IV, \dots\dots\dots \text{Eq (2.22)}$$

$$P_i=C_i \oplus EK(S_i)[1 \dots r] \dots\dots\dots \text{Eq (2.23)}$$

$$S_{i+1}=(S_i \ll r)+C_i \dots\dots\dots \text{Eq (2.24)}$$

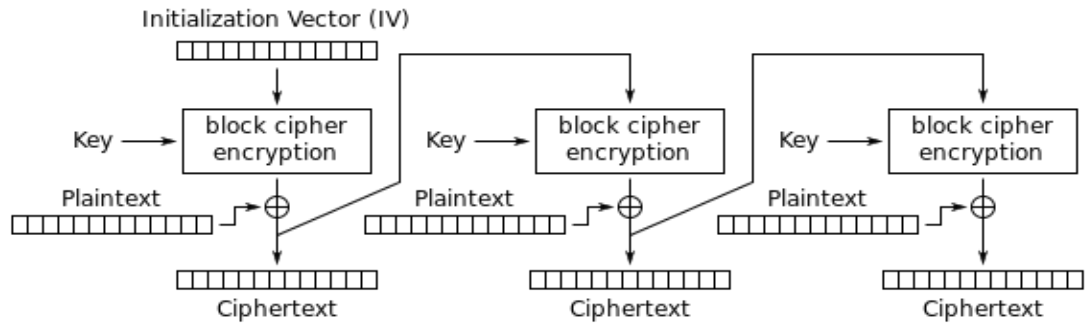


Figure 6: Cipher Feedback (CFB) mode encryption [8]

The feedback in CFB prevents the parallel encryption of plain text blocks. Still, arbitrary cipher text blocks can be decrypted independently, provided that the $[n/r]$ preceding blocks are available. As a direct consequence, single bit errors in the cipher text cannot propagate over more than $[n/r]$ successive blocks. Again, and for similar reasons as in CBC, a single key should not be used to encrypt more than $2n/2$ blocks. For small values of r , additional precautions should be taken in order to avoid weak IV values. In particular, if the bits of the IV were to form a periodic sequence, then this would considerably increase the probability of repeated values at the input of the block cipher [9].

2.2. Literature Review

Different authors carried out extensive study in this regard. The two most competitive IoT devices, the Raspberry Pi 3 and Beagle Bone Black processors were tested in [6]. Authors compared different techniques such as Twofish, Blowfish, DES, Triple-DES, AES, RC2, RC4 and ChaCha20 to test their effects on the Raspberry Pi 3 and Beagle Bone Black processors and compared different parameters like quickness and capability. Due to the processing quickness on the Beagle Bone Black being lower than that of the Raspberry Pi 3, the execution time of these ciphers nearly doubles on it. The power and memory consumption was also found to be lower on the Raspberry Pi 3. As a result, for quick, capable, secure and quick data transmission the Raspberry Pi 3 performs better than the Beagle Bone Black.

A system with high reliability and dynamic GPU encryption system for large multimedia IoT educational Big data was developed and implemented in [10] resisting real time attacks like DDoS, brutal force attacks and other tampering attacks. The

algorithm was compared with other symmetric cryptographic algorithms like AES, DES, 3-DES, RC6 and MARS based on architecture, flexibility, scalability, security level and also based on computational running time, throughput for both encryption and decryption process. Observation was done on flipping the bits in resultant process and also in algorithm execution process with substitution and permutation of S-boxes in encryption and decryption process resulting in high avalanche effect. This was considered to be a novel encryption system combining two symmetric cryptographic algorithm used for large scale data management to be highly secured.

Author in [11] recommended two methods, called FF1 and FF3-1, for format-preserving encryption. Both of these methods are modes of operation for an underlying, approved symmetric- key block cipher algorithm. These two implementations can only interoperate when they support common values for the base.

Authors from [12] performed the identification of 5 frequently used block ciphers, AES, DES, 3DES, RC5 and Blowfish. Authors have successfully identified AES from DES, 3DES, RC5 and Blowfish with a high identification rate. However, one to one identification between any two ciphers of DES, 3DES, RC5 and Blowfish could not be conducted yet, which was a hard nut to be cracked.

The author in [13] presented an overview of the concepts of and motivation for the OCB block cipher mode of operation. OCB is well suited for IoT, wireless, and other constrained devices where processing time and energy consumption are design issues. The article described two versions of the OCB algorithm (OCB1 and OCB3) that have been widely accepted. Because of its streamlined design, OCB is well suited for IoT devices, wireless sensors, and other constrained devices where processing power and energy consumption are concerns. As well, for larger multicore devices that have the ability to perform parallel processing, OCB excels at speed of execution. Thus, OCB is a versatile AE technique for a wide range of applications.

Performance of AES encryption algorithm was evaluated in [14] with different block cipher modes to find the most efficient mode for NVM storage encryption. It was found that CTR mode had performance improvement with lower latency. The study also illustrated CTR mode outperforms CBC mode due to the support of parallel encryption and decryption operations. There was only a negligible difference between CTR and

XTS-AES modes. They reached almost the same percentage and number. Therefore, XTS-AES was found to be the most suitable block cipher mode for NVM storage given the efficiency and protection.

Twofish cryptographic algorithm was implemented using library Chilkat Encryption ActiveX Ms. Visual Basic in [15]. To facilitate the implementation of the coding in Ms. Visual Basic authors have used Chilkat Encryption ActiveX. The program was implemented to maintain the confidentiality of the data when transmitted over the Internet. The speed encryption process needed 3 times longer than the decryption.

In article [16] authors described the security drawbacks of the standard BCMO (Block Cipher Mode of Operation), and propose the OPC (Output Protection Chain) to improve the security level of a block ciphering system by protecting the outputs of its BCE unit. The purpose is avoiding the security system from being attacked by known or chosen-plaintext/cipher text attacks. However, in the OPC-2, the BCE unit must be invertible, e.g., DES, 3-DES, or AES. Since the encryption speeds of non-invertible algorithms are often short, and their encryption keys are difficult to crack, if one replaces the BCE unit of the CFB, OFB, CTR or OPC-1 with a non-invertible algorithm, the security levels and the processing performance of these BCMOs will be then higher than before.

The applications of Advanced Encryption Standard (AES) in plain audio or video signal using five operation modes was discussed in [17] among which ECB is the most popular one. Due to the feature of each block cipher input being independent to the previous cipher block output in ECB, it is easy to have patterns appeared in the encrypted signal which might leave clue to the original signal. Therefore, CBC, CFB, and OFB are the better choices for the pattern-free encryption. However, if encryption is done after compressing, ECB is a good choice in terms of pattern-free as well as high speed parallel operation.

Implementation on VHDL (very high speed integrated circuit hardware description language) using Xilinx – 6.1 xst software has been done in [18] taking delay as main constraint. Twofish algorithm was studied and some modules had been modified keeping delay as main constraint. VHDL description of twofish, had been verified by functional simulation, using Xilinx xst-6.1, and Model-Sim Simulator for the waveform generation. The modules MDS and PHT had been modified and implemented for the

modified algorithms. All the modules and functions are interrelated hence, after modifying MDS and PHT function g and function F also got modified. The results showed the delay of twofish algorithm of 128-bit key and modified twofish of 128-bit key, and compared their delay results. The analysis showed that modified algorithm has less delay than the conventional one. After that the delay results of twofish algorithm with 192-bit key and modified twofish with 192-bit key had been compared. According to the results it was clear that modified 192-bit key twofish algorithm has less delay than 192-bit twofish.

Encryption and decryption of images was performed in [19] using a secret-key block cipher called 64-bits Blowfish designed to increase security and to improve performance. This algorithm was used as a variable key size up to 448 bits. This employed Feistel network which iterates simple function 16 times. The blowfish algorithm is safe against unauthorized attack and runs faster than the popular existing algorithms. The proposed algorithm was designed and realized using MATLAB. Both colour and black & white image of any size saved in tagged image file format (TIF), Bit map (bmp), Portable network graphics (PNG), Joint Photographic Experts group (jpg), etc. can be encrypted & decrypted using blowfish algorithm. Histogram of encrypted image was found to be less dynamic and significantly different from the respective histograms of the original image. Blowfish cannot be broken until an attacker tries 2^{8r+1} combinations where r is the number of rounds. Hence if the number of rounds are been increased then the blowfish algorithm becomes stronger. Since Blowfish has not any known security weak points so far it can be considered as an excellent standard encryption algorithm.

Authors from [4] provided a fair comparison between three most common symmetric key cryptography algorithms: DES, AES, and Blowfish. Since main concern was the performance of algorithms under different settings, the comparison took into consideration the behavior and the performance of the algorithm when different data loads were used. The comparison was made on the basis of these parameters: speed, block size, and key size. Simulation program was implemented using Java programming. The simulation results showed that Blowfish had a better performance than other common encryption algorithms used. Since Blowfish had not any known security weak points so far, that made it an excellent candidate to be considered as a

standard encryption algorithm. AES showed poor performance results compared to other algorithms since it required more processing power. Using CBC mode had added extra processing time, but overall it was relatively negligible especially for certain application that requires more secure encryption to a relatively large data blocks. OFB showed better performance than ECB and CBC but required more processing time than CFB. Overall time differences between all modes were negligible.

RC4 was found to be fast and energy efficient for encryption and decryption with compared to AES algorithm in [20] with different modes of operation (block cipher) and RC4 algorithm (stream cipher). The performance metrics were CPU process time, memory utilization, encryption and decryption time and throughput at different settings like variable key size and variable data packet size.

The structure and design of Rijndael cipher (new AES) have been analyzed in [21] remarking its main advantages and limitations, as well as its similarities and dissimilarities with DES. The analysis was performed following three criteria: a) resistance against all known attacks; b) speed and code compactness on a wide range of platforms; and c) design simplicity; as well as its similarities and dissimilarities with other symmetric ciphers. Thus, the fact that the new cipher and its inverse used different components, which practically eliminated the possibility for weak and semi-weak keys, was one of the principal advantages of this new cipher algorithm, compared to DES. Also, the nonlinearity of the key expansion, which practically eliminates the possibility of equivalent keys, is another big advantage. The importance of the Advanced Encryption Standard and the high security of the Rijndael algorithm had been examined. It was learnt that Rijndael AES, at that moment was an unbreakable algorithm. AES had been implemented in a large variety of languages and software tools. Some code optimizations were suggested for creation of S-box and inverse mix columns transformation. It was found that the simple transformations of AES can quite comfortably implemented in any high level or low level languages and software tools. Finally, a performance comparison among new AES and DES for different microcontrollers had been carried out, showing that new AES have a computer cost of the same order.

In reference [22] author talks about the performance evaluation of the popular block cipher algorithms such as AES, Serpent, Camellia, CAST5, and MARS on 8-bit Atmel

microcontroller. The performance of the chosen block ciphers were evaluated in terms of the code/data memory requirement, execution time, and throughput criteria. According to the results obtained from target device, it was observed that AES and Serpent are the most efficient algorithms and Mars is the most inefficient algorithm in terms of code and data memory usage. In term of execution time, CAST5 can perform the encryption/decryption procedures faster than the others. However, CAST5 takes 64-bit block of plaintext and AES take 128-bit block of plaintext. AES outperforms CAST block cipher when this situation is considered. In term of throughput criteria, AES is the fastest algorithm among the chosen block ciphers. CAST5 and Camellia can be considered an alternative block cipher for AES. Although Serpent is the slowest algorithm, it is quite efficient in terms of memory usage. Therefore, it can be used on an application, which the speed is not important, but the memory size is limited. Mars has a very high memory requirement. Thus, it is not suitable for the microcontroller applications.

CHAPTER 3

METHODOLOGY

3.1. Methodology

This study includes implementing and analysis of the block modes of operation ECB, CBC, CFB, OFB and CTR with CAST and Twofish algorithms and testing of these algorithms with various size of text file and image file. The different sized text file and image file with different file formats are fed to the each of the modules and each module is analyzed using various key size and block size of files. Test data are taken from secondary source (Sample Videos, Satellite Images and NASA Visible Earth) [23] [24] [25]. Computational analysis is done for text and image file using encryption time and decryption time. And all of the image results are evaluated in terms of statistical analysis, differential analysis and visual assessment. The methodology is depicted by following flowcharts:

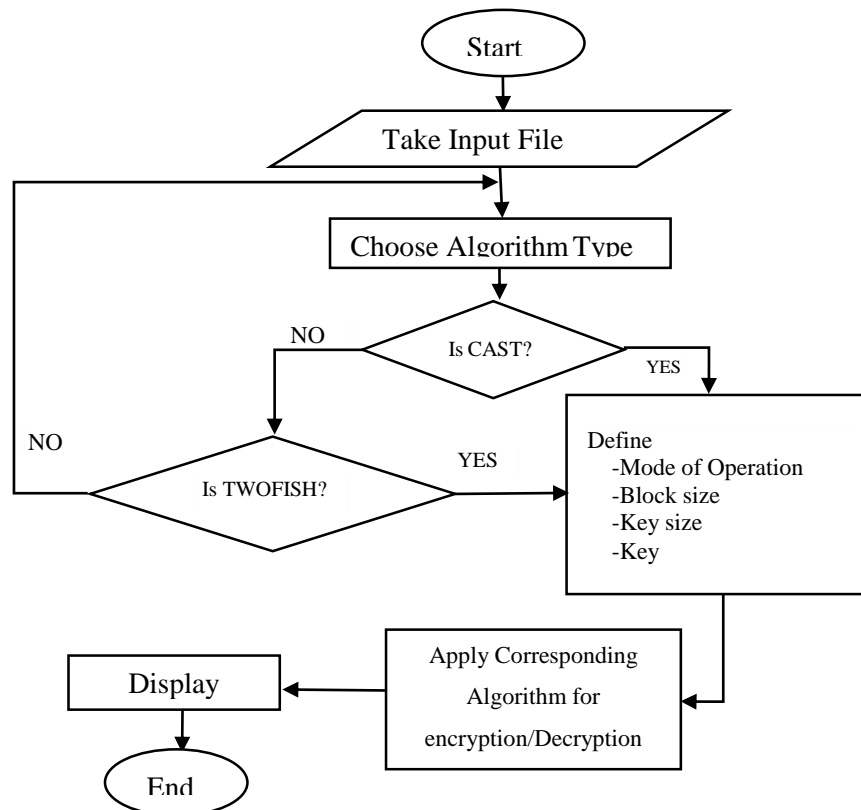


Figure 7: Encryption algorithm process

The algorithms are implemented on the different settings. The parameters chosen for the algorithm testing are of following nature.

Table 1: Algorithms' Settings

Algorithm	Key size (Bits)	Block size(Bits)	Modes
CAST	128	64	ECB, CBC, CFB, OFB, CTR
TWOFISH	128, 192, 256	128	ECB, CBC, CFB, OFB, CTR

The evaluation is meant to evaluate the results by using block ciphers. Hence, the load data (plaintext) is divided into smaller block size as per algorithm settings given in Table 1 above.

3.2. CAST encryption algorithm

CAST-128 is a symmetric block cipher with a block-size of 64-bit and a variable key-size of up to 128 bits. The algorithm was developed in 1996 by Carlisle Adams and Stafford Tavares. It is available worldwide on a royalty-free basis for commercial and non-commercial uses. CAST-128 is a 12 or 16-round Feistel network with a 64-bits block size and a key size of between 40 to 128 bits (but only in 8-bit increments). The full 16 rounds are used when the key size is longer than 80 bits. Components include large 8×32 -bits s-boxes based on bent functions, key-dependent rotations, and modular addition/subtraction and XOR operations. S-boxes $S_1, S_2, S_3,$ and S_4 are round function used for encryption and decryption and S-boxes; $S_5, S_6, S_7,$ and S_8 are key schedule s-boxes used are used in the process of key generation. [22]

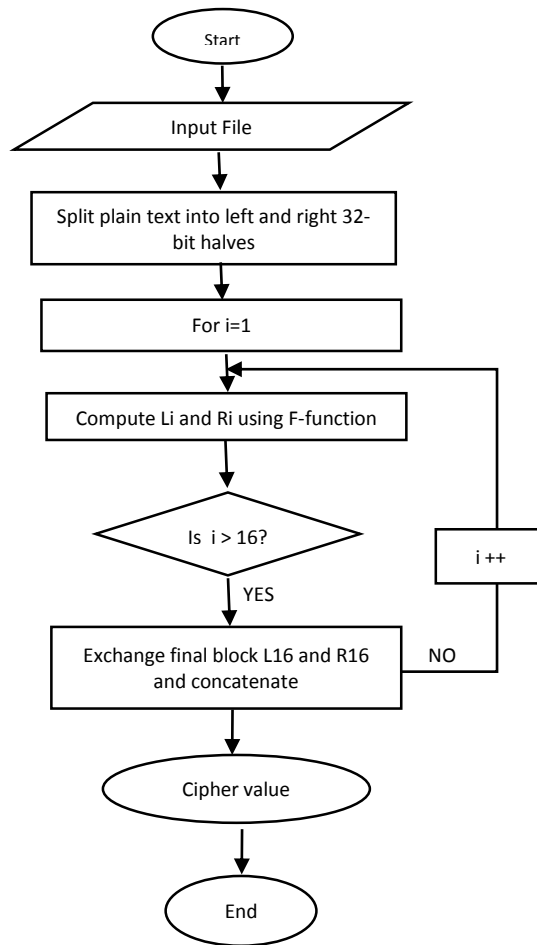


Figure 8: Flowchart of CAST-128 Encryption

The CAST full encryption process is described as below: [26]

INPUT: Plaintext => $m_1 m_2 \dots m_{64}$; and key => $K = k_1 k_2 \dots k_{128}$Eq (3.1)

OUTPUT: Ciphertext => $c_1 c_2 \dots c_{64}$Eq (3.2)

Step 1: Compute 16 pairs of sub keys $\{K_{mi}, K_{ri}\}$ from K (key schedule)

Step 2: Split the plain text into left and right 32-bit halves (L_0, R_0) such that:

$$L_0 = m_1 m_2 m_3 \dots m_{32} \dots \text{Eq (3.3)}$$

$$R_0 = m_{33} m_{34} m_{35} \dots m_{64} \dots \text{Eq (3.4)}$$

Step 3: For i from 1 to 16 (16 rounds), compute L_i and R_i as follows:

$$L_i = R_{i-1}; \dots \text{Eq (3.5)}$$

$$R_i = L_{i-1} \text{ XOR } F_i [R_{i-1}, K_{mi}, K_{ri}] \dots \text{Eq (3.6)}$$

Where F is the round function (F is of Type 1, Type 2, or Type 3, depending on i).

Step 4: Exchange final blocks L_{16} , R_{16} and concatenate to form the ciphertext as:

$$c_1 \dots c_{64} \leftarrow (R_{16}, L_{16}) \dots \dots \dots \text{Eq (3.7)}$$

$$\text{CipherText} = R_{16} || L_{16} \dots \dots \dots \text{Eq (3.8)}$$

Decryption is identical to the encryption algorithm given above, except that the rounds (and the sub key pairs) are used in reverse order to compute (L_0, R_0) from (R_{16}, L_{16})

Pairs of Round Keys

CAST-128 uses a pair of sub keys per round: a 32-bit quantity K_m is used as a "masking" key and a 5-bit quantity K_r is used as a "rotation" key. [26]

Substitution Boxes

CAST-128 uses eight substitution boxes: S-boxes $S_1, S_2, S_3,$ and S_4 are round function S-boxes; $S_5, S_6, S_7,$ and S_8 are key schedule S-boxes. Although 8 S-boxes require a total of 8 KBytes of storage, only 4 KBytes are required during actual encryption / decryption since sub key generation is typically done prior to any data input. [26]

Function F

Function F uses four S-box substitutions, each of size 8×32 , the left circular rotation operation, mod 2 addition and subtraction, exclusive OR operations four operation functions that vary depending on the round number. The strength of the F function is based primarily on the strength of the S-boxes. There are three alternating types of round function, but they are similar in structure and differ only in the choice of the exact operation (addition, subtraction or XOR) at various points. We use I to refer to the intermediate 32-bit value after the left circular rotation function and the labels I_a, I_b, I_c and I_d to refer to the 4 bytes of I. With these conventions, function F is defined as follows: [26]

Rounds 1,4,7,10,13,16	$I = ((K_{mi} + R_{i-1}) \lll K_{ri})$ $F = ((S_1 [I_a] \oplus S_2 [I_b]) - (S_3 [I_c])) + S_4 [I_d]$
Rounds 2,5,8,11,14	$I = ((K_{mi} \oplus R_{i-1}) \lll K_{ri})$ $F = ((S_1 [I_a] - S_2 [I_b]) + (S_3 [I_c])) \oplus S_4 [I_d]$
Rounds 3,6,9,12,15	$I = ((K_{mi} - R_{i-1}) \lll K_{ri})$ $F = ((S_1 [I_a] + S_2 [I_b]) \oplus (S_3 [I_c])) - S_4 [I_d]$

Masking Subkeys And Rotate Subkeys

Let K_{m1}, \dots, K_{m16} be 32-bit masking subkeys (one per round). Let K_{r1}, \dots, K_{r16} be 32-bit rotate subkeys (one per round); only the least significant 5 bits are used in each round. [26]

for $(i=1; i \leq 16; i++) \{ K_{mi} = K_i; K_{ri} = K_{16+i}; \}$

3.3. TWOFISH encryption algorithm

Twofish is a 128 bit block cipher that accepts variable key up to 256 bits. Generally Twofish algorithm is used for encryption process that means hiding information within one information. Following are some parameters which need to be taken care always for a safe and secure data encryption process i.e.

Imperceptibility: Imperceptibility is the property in which a person should be unable to distinguish the original and the embedded data.

Robustness: refers to the degree of difficulty required to destroy embedded information without destroying the cover data.

Embedding Capacity: Refers to the amount of secret information that can be embedded without degradation to the quality of the data. [27]

In Twofish algorithm, the F-function consists of five kinds of component operations: fixed left rotation by 8 bits, key dependent S-boxes, Maximum Distance Separable (MDS) matrices, Pseudo-Hadamard Transform (PHT), and two subkey additions modulo 2^{32} . There are four kinds of key dependent S-boxes together with the MDS matrix form and g-function. This g-function appears two times in the cipher structure,

which causes significant redundancy. There are total 16-rounds in twofish algorithm [15].

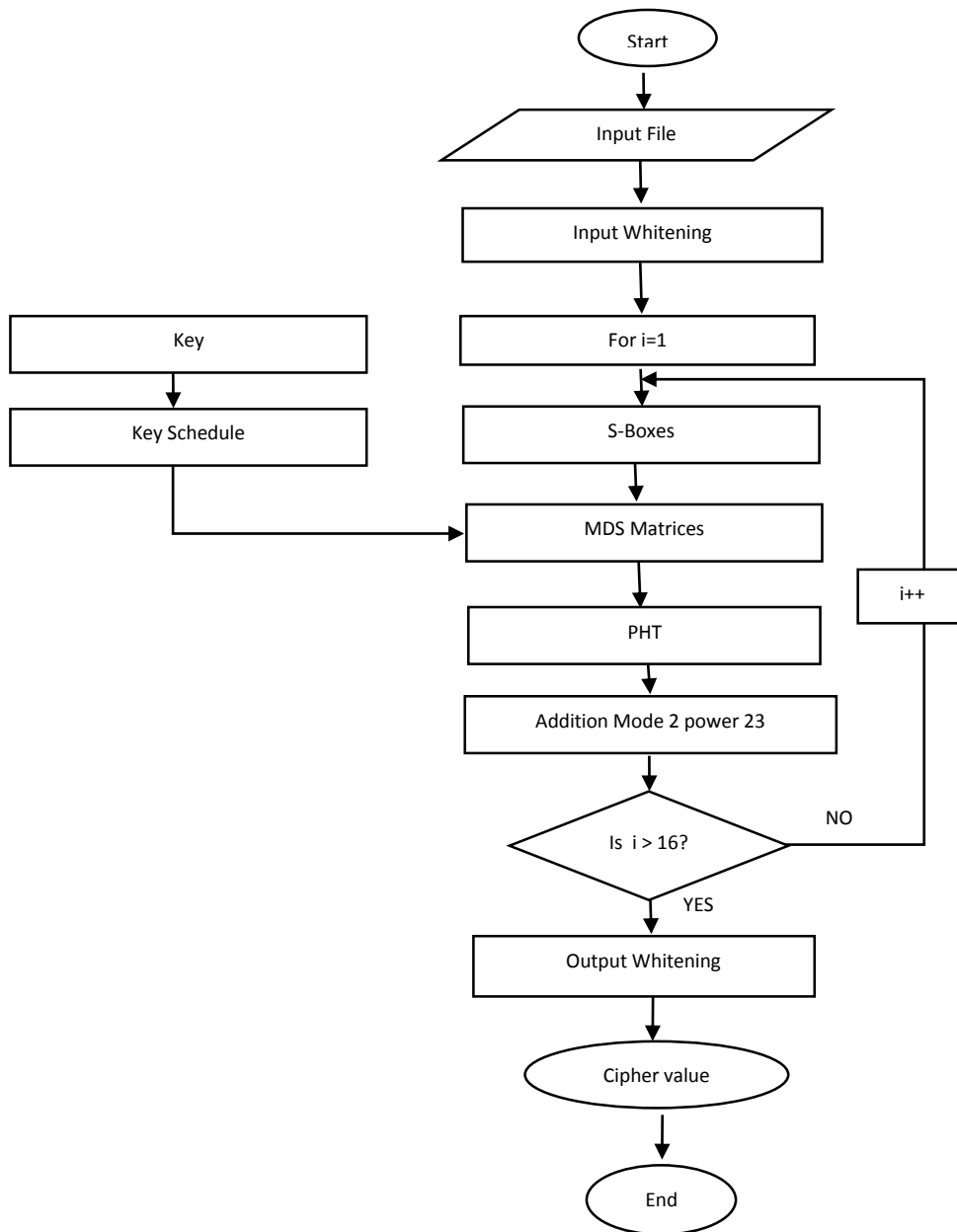


Figure 9: Flowchart of Twofish Algorithm

The algorithm is described in following steps:

Step 1: Bit input as much as 128 bits would be divided into four sections, each for 32 bits. Two parts of the bit will be the right part, the two parts of the other bits will be left.

Step 2: Bit-XOR input in advance with the four key parts (whitening).

$$R_{0,i} = P \oplus K_i; \quad i=0, \dots, 3 \dots\dots\dots \text{Eq (3.9)}$$

Where K is the key, K_i means the sub key where $i=0, \dots, 3$.

Input and output data are XOR-ed with eight sub-keys $K_0 \dots K_7$. These XOR operations are called input and output whitening. [15]

TWOFISH FUNCTIONS AND MODULES:

Whitening: Whitening includes the Xoring of input and output data with eight sub-keys ($K_0 - K_7$). Thus this operation is performed only at the output level, i.e., 1st round, hence called input whitening and at the output level, i.e., after 16th round, hence called output whitening. The whitening operation is actually used to increase the difficulty for attackers, to search for key, by hiding the inputs to 1st and last round. The sub-keys ($K_0 - K_7$) used in this operation are also calculated in the same manner as the other round sub-keys, and are not used in other operations. [28]

S-boxes: An S-box is a table-driven substitution operation used in most block ciphers. S-boxes vary in both input size and output size, and can be created either randomly or algorithmically. Twofish uses four different, bijective, key-dependent, 8-by-8-bit S-boxes. These S-boxes are built using two fixed 8-by-8-bit permutations and key material. [27]

Function F: The Feistel function F is a key-dependent permutation on 64 bit values. It takes three arguments, two input words R_0 and R_1 , and the round number r used to select the appropriate sub keys. R_0 is passed through the g function, which yields T_0 . R_1 is rotated left by 8 bits and then passed through the g function to yield T_1 . The results T_0 and T_1 are then combined in a PHT and two words of the expanded key are added. The following set of equations describes the details of F function: [29]

$$T_0 = g(R_0) \dots\dots\dots \text{Eq (3.10)}$$

$$T_1 = g(\text{ROL}(R_1; 8)) \dots\dots\dots \text{Eq (3.11)}$$

$$F_0 = (T_0 + T_1 + K_{2r+8}) \text{ mod } 2^{32} \dots\dots\dots \text{Eq (3.12)}$$

$$F_1 = (T_0 + 2T_1 + K_{2r+9}) \bmod 2^{32} \dots\dots\dots \text{Eq (3.13)}$$

Function g: The function g forms the heart of twofish. The input word X is split into four bytes. Each byte is run through its own key dependent S-box. Each S-box is bijective, takes 8 bits of input, and produces 8 bits of output. The four results are interpreted as a vector of length 4 over GF (2⁸), and multiplied by the 4x4 MDS matrix (using the field GF (2⁸) for the computations). The resulting vector is interpreted as a 32-bit word which is the result of g. [29]

$$x_i = \left\lfloor X / 2^{8i} \right\rfloor \bmod 2^8 \quad i = 0, \dots, 3 \dots\dots\dots \text{Eq (3.14)}$$

$$y_i = s_i[x_i] \quad i = 0, \dots, 3 \dots\dots\dots \text{Eq (3.15)}$$

$$\begin{pmatrix} z_0 \\ z_1 \\ z_2 \end{pmatrix} = \begin{pmatrix} \dots & \dots & \dots \\ \vdots & MDS & \vdots \\ \dots & \dots & \dots \end{pmatrix} \cdot \begin{pmatrix} y_0 \\ y_1 \\ y_2 \end{pmatrix} \dots\dots\dots \text{Eq (3.16)}$$

$$Z = \sum_{i=0}^3 z_i \cdot 2^{8i} \dots\dots\dots \text{Eq (3.17)}$$

where s_i are the key-dependent S-boxes and Z is the result of g.

MDS Matrices: MDS is maximum separable matrix. It is a matrix of bytes that multiplies a vector of four bytes. Multiplications are carried out in the Galois Field GF (2⁸) with the primitive polynomial x⁸ + x⁶ + x⁵ + x³ + 1. Each byte is converted into a polynomial in which each power p of x is present only if the pth bit is 1. A multiplication in GF amounts to a multiplication of polynomials followed by a division by the primitive polynomial. [29]

The MDS matrix is given by:

$$\text{MDS} = \begin{pmatrix} 01 & EF & 5B & 5B \\ 5B & EF & EF & 01 \\ EF & 5B & 01 & EF \\ EF & 01 & EF & 5B \end{pmatrix} \dots\dots\dots \text{Eq (3.18)}$$

PHT: PHT is a reversible transformation of a bit string that provides cryptographic diffusion. Pseudo-hadamard transform consists of two additions. Twofish uses a 32-bit

PHT to mix the outputs from its two parallel 32-bit g functions. Given two inputs, a and b, the 32-bit PHT is:

$$A' = a + b \text{ mod } 2^{32} \dots\dots\dots \text{Eq (3.19)}$$

$$B' = a + 2b \text{ mod } 2^{32} \dots\dots\dots \text{Eq (3.20)}$$

Both additions are implemented in the same way as ordinary addition modulo 2^{32} . Twofish uses a 32-bit PHT to mix the outputs from its two parallel 32-bit g functions. PHT Using shift operation, in this method of PHT two 32-bit inputs are given, say in_1 and in_2 . Here for the operations of equations shown below, are performed using the shifting. The function can be easily explained with the help of the following equations [29]:

$$\text{For out}_1 \qquad \qquad \qquad \text{out}_1 = in_1 + in_2 \dots\dots\dots \text{Eq (3.21)}$$

$$\text{For out}_2 \qquad \qquad \qquad \text{out}_2 = in_1 + in_2x(i) \dots\dots\dots \text{Eq (3.22)}$$

$$\text{Where:} \qquad \qquad \qquad \text{In}_2x(i) = in_2(i - 1)$$

For $i = 1$ to 31

$$in_2x(0) = 0 \dots\dots\dots \text{Eq (3.23)}$$

For $i = 0$

The Key Schedule: The key schedule has to provide 40 words of expanded key K_0, \dots, K_{39} , and the 4 key-dependent S-boxes used in the g function. Twofish is defined for keys of length $N = 128$, $N = 192$, and $N = 256$. Keys of any length shorter than 256 bits can be used by padding them with zeroes until the next larger defined key length. We define $k = N/64$. The key M consists of $8k$ bytes m_0, \dots, m_{8k-1} . The bytes are first converted into $2k$ words of 32 bits each [29]

$$M_i = \sum_{j=0}^3 m_{(4i+j)} \cdot 2^{8j} \dots\dots\dots \text{Eq (3.24)}$$

$i = 0, \dots, 2k-1$

and then into two word vectors of length k.

$$M_e = (M_0, M_2, \dots, M_{2k-2}) \dots \text{Eq (3.25)}$$

$$M_o = (M_1, M_3, \dots, M_{2k-1}) \dots \text{Eq (3.26)}$$

A third word vector of length k is also derived from the key. This is done by taking the key bytes in groups of 8, interpreting them as a vector over GF (2⁸), and multiplying them by a 4×8 matrix derived from an RS code. Each result of 4 bytes is then interpreted as a 32-bit word. These words make up the third vector.

$$\begin{pmatrix} S_{i,0} \\ S_{i,1} \\ S_{i,2} \\ S_{i,3} \end{pmatrix} = \begin{pmatrix} \cdot & \dots & \cdot \\ \vdots & RS & \vdots \\ \cdot & \dots & \cdot \end{pmatrix} \cdot \begin{pmatrix} m_{8i} \\ m_{8i+1} \\ m_{8i+2} \\ m_{8i+3} \\ m_{8i+4} \\ m_{8i+5} \\ m_{8i+6} \\ m_{8i+7} \end{pmatrix} \dots \text{Eq (3.27)}$$

$$S_i = \sum_{j=0}^3 S_{i,j} \cdot 2^{8j} \dots \text{Eq (3.28)}$$

for i = 0, ..., k-1, and S = (S_{k-1}, S_{k-2}, ..., S₀)

Note that S lists the words in “reverse” order. For the RS matrix multiply, GF (2⁸) is represented by GF(2)[x]/w(x), where w(x) = x⁸+x⁶+x³+x²+1 is another primitive polynomial of degree 8 over GF(2). The mapping between byte values and elements of GF (2⁸) uses the same definition as used for the MDS matrix multiply. Using this mapping, the RS matrix is given by: [29]

$$RS = \begin{pmatrix} 01 & A4 & 55 & 87 & 5A & 58 & DB & 9E \\ A4 & 56 & 82 & F3 & 1E & C6 & 68 & E5 \\ 02 & A1 & FC & C1 & 47 & AE & 3D & 19 \\ A4 & 55 & 87 & 5A & 58 & DB & 9E & 03 \end{pmatrix} \dots \text{Eq (3.29)}$$

The three vectors M_e, M_o, and S form the basis of the key schedule.

Additional Key Lengths: Twofish can accept keys of any byte length up to 256 bits. For key sizes that are not defined above, the key is padded at the end with zero bytes to the next larger length that is defined. For example, an 80-bit key m₀, ..., m₉ would be extended by setting m_i = 0 for i = 10, ..., 15 and treating it as a 128-bit key. [29]

The Function h: This is a function that takes two inputs—a 32-bit word X and a list L = (L₀,..., L_{k-1}) of 32-bit words of length k—and produces one word of output. This function works in k stages. In each stage, the four bytes are each passed through a fixed S-box, and Xored with a byte derived from the list. Finally, the bytes are once again passed through a fixed S-box, and the four bytes are multiplied by the MDS matrix just as in g. More formally: we split the words into bytes. [29]

$$l_{i,j} = \left\lfloor L_i / 2^{8j} \right\rfloor \text{ mod } 2^8 \dots\dots\dots \text{Eq (3.30)}$$

$$x_j = \left\lfloor X / 2^{8j} \right\rfloor \text{ mod } 2^8 \dots\dots\dots \text{Eq (3.31)}$$

for i = 0,...,k -1 and j = 0,...,3. Then the sequence of substitutions and Xors is applied.

$$y_{k,j} = x_j \qquad j = 0,\dots,3$$

3.4. Performance Evaluation Parameters

The algorithms implemented during this study are analyzed from various dimensions such as:

3.4.1. Computational analysis

Computational analysis in this thesis is based on encryption and decryption time, memory utilization and throughput.

Encryption time- The encryption time is the time that an encryption algorithm takes to produce a cipher text from a plaintext.

Decryption time- The decryption time is the time that a decryption algorithm takes to produce a plaintext from a cipher text.

Throughput- The throughput of an encryption scheme define the speed of encryption/decryption. The throughput is calculated as the sum of total plaintext encrypted and total cipher text decrypted in Kilobytes / sum of encryption and decryption time (KB/sec). As the throughput increases, power consumption decreases.

Memory Utilization-The Memory Utilization defines how much memory is being consumed while doing the encryption or decryption. [20]

3.4.2. Differential analysis

Differential analysis is a technique which observes how difference in input affects differences on the output. It is done by using Number of Pixels Change Rate (NPCR), and Unified Average Changing Intensity (UACI) with the original image and decrypted image. Differential analysis is done for security measures.

NPCR (Number of Pixels Change Rate)

NPCR concentrates on the absolute number of pixels which changes value in differential attacks. The NPCR measures the percentage of different pixel numbers between the plain image and encrypted image. [30]

NPCR is defined as:

$$NPCR = \frac{\sum_{i,j} D(i,j)}{M \times N} \times 100\% \dots\dots\dots \text{Eq (3.32)}$$

$$\text{Where } D(i,j) = \begin{cases} 1, & \text{if } I(i,j) \neq I'(i,j) \\ 0, & \text{else} \end{cases}$$

UACI (Unified Average Changing Intensity)

UACI measured the average intensity of differences between two paired cipher images.

UACI is given by: [30]

$$UACI = \frac{1}{M \times N} \left(\sum_{i,j} \frac{|I(i,j) - I'(i,j)|}{255} \right) \times 100\% \dots\dots\dots \text{Eq (3.33)}$$

For a better system, the value of UACI should be low, and NPCR should be high.

3.4.3. Visual assessment analysis

Visual assessment analysis is done to measure the performance of the decryption procedure. Visual assessment analysis is done by using Peak Signal to Noise Ratio (PSNR) of input images and result images to measure the security and quality of encrypted images.

PSNR (Peak Signal to Noise Ratio)

The peak signal-to-noise ratio (PSNR) is the ratio between a signal's maximum power and the power of the signal's noise. It is the ratio of mean square difference of the component for the two images to the maximum mean square difference that can exist between any two images. PSNR is commonly used to measure the quality of reconstructed images that have been compressed. [31]

$$PSNR = 10 \times \lg \left(\frac{255^2}{MSE} \right) \dots\dots\dots \text{Eq (3.34)}$$

$$MSE = \frac{1}{M \times N} \sum_{i=1}^N \sum_{j=1}^M [I(i, j) - I'(i, j)]^2 \dots\dots\dots \text{Eq (3.35)}$$

The MSE can be described as the mean of the square of the differences in the pixel values between the corresponding pixels of the two images. [31]

3.4.4. Statistical Analysis

Statistical analysis is done by using Histogram analysis. The histogram gives the distribution of pixel in the images. This histogram is a graph showing the number of pixels in an image at each different intensity value found in that image. [32]

CHAPTER 4

IMPLEMENTATION AND ANALYSIS

4.1. Implementation

The two encryption algorithm CAST-128 and Twofish are implemented in C# language as programming language in .NET framework version 4.7.03056 in Microsoft Visual Studio Enterprise 2017 version 15.7.3. Microsoft Visual Studio .NET is an application-development tool for writing applications; the .NET Framework provides the infrastructure required to run those applications. For differential, statistical and visual assessment analysis of different images, MATLAB R2018a is used. The implementation of the algorithm is done in Acer Reliability Travelmate 8572 with Intel^(R) coreTM i7 CPU @2.67 GHz core processor with Installed RAM of 8GB and usable 7.68GB and system type of 64-bit Operating System, x64 based processor.

4.1.1. C# programming language

C# is a general-purpose, modern and object-oriented programming language. It is a hybrid of C and C++, it is a Microsoft programming language developed to compete with Sun's Java language. It was developed around 2000 by Microsoft as part of its .NET initiative, and later approved as an international standard by Ecma (ECMA-334) and ISO (ISO/IEC 23270:2018). It is an object-oriented programming language used with XML-based Web services on the .NET platform and designed for improving productivity in the development of Web applications. C# boasts type-safety, garbage collection, simplified type declarations, versioning and scalability support, and other features that make developing solutions faster and easier, especially for COM+ and Web services. Microsoft critics have pointed to the similarities between C# and Java.

4.1.2. Microsoft .NET Framework

.NET is a software framework which is designed and developed by Microsoft. In easy words, it is a virtual machine for compiling and executing programs written in different languages like C#, VB.NET, etc. It is used to develop Form-based applications, Web-based applications, and Web services. There is a variety of programming languages

available on the .Net platform, VB.Net and C# are the most common ones. It is used to build applications for Windows, phone, web etc. It provides a lot of functionalities and also supports industry standards. .NET Framework supports more than 60 programming languages in which 11 programming languages are designed and developed by Microsoft. The remaining Non-Microsoft Languages which are supported by .NET Framework but not designed and developed by Microsoft.

4.1.3. Matlab R2018a Overview

MATLAB (matrix laboratory) is a multi-paradigm numerical computing environment and proprietary programming language developed by MathWorks. MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages, including C, C++, C#, Java, Fortran and Python.

Although MATLAB is intended primarily for numerical computing, an optional toolbox uses the MuPAD symbolic engine allowing access to symbolic computing abilities. An additional package, Simulink, adds graphical multi-domain simulation and model-based design for dynamic and embedded systems. MATLAB R2018a was released on March 2018, with two new products, Predictive Maintenance Toolbox for designing and testing condition monitoring and predictive maintenance algorithms, and Vehicle Dynamics Blockset for modeling and simulating vehicle dynamics in a virtual 3D environment.

4.2. Test Data Description

Test data is taken for the experiment analysis are the different text and image files. The input text file are collected with different size. The input images types are of .jpg, .png and .tif types. Text data are randomly generated and image file are secondary data sets. The secondary image sets are collected from secondary source (Sample Videos, Satellite Images and NASA Visible Earth) [23] [24] [25]. The size of text files collected varies from 50 kb to 500 mb and image files varies from 50 kb to 198 mb with maximum dimension of 12000×12000 .

4.3. Analysis

The algorithms implemented during this study are analyzed from various dimensions. Computational analysis based on encryption and decryption time, throughput and memory utilization, differential analysis based on Number of Pixels Change Rate (NPCR), and Unified Average Changing Intensity (UACI), statistical analysis based on histogram analysis and visual assessment analysis based on Peak Signal to Noise Ratio (PSNR) and Mean Square Error (MSE) of input images and enciphered images has been done.

4.3.1. COMPUTATIONAL ANALYSIS

Encryption time analysis for text file

One of the most important performance criteria of the algorithms is encryption time and decryption time. In this analysis, we used CAST-128, Twofish-128, Twofish-192 and Twofish-256 algorithms with different operation modes to encrypt and decrypt different size of text files and image files. The execution time and decryption time is shown in Figure below.

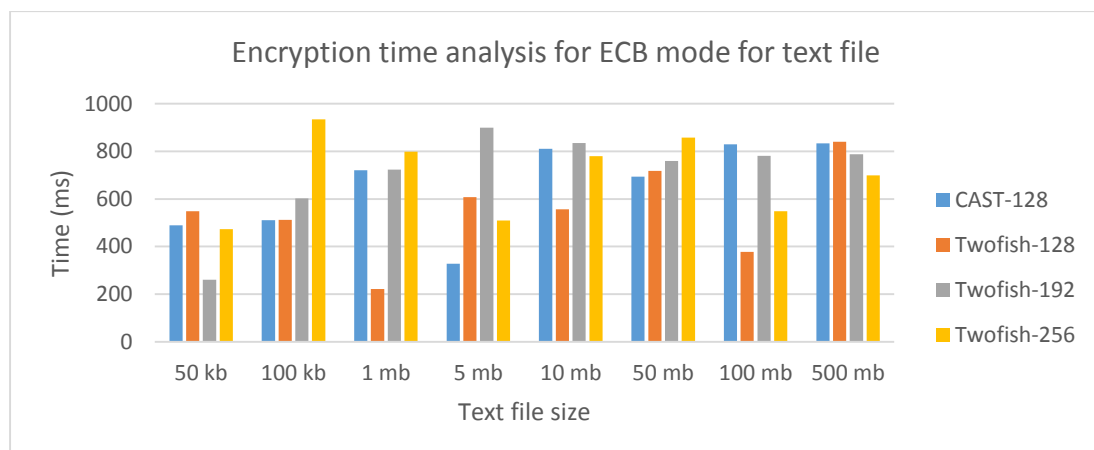


Figure 10: Encryption time analysis for ECB mode for text file

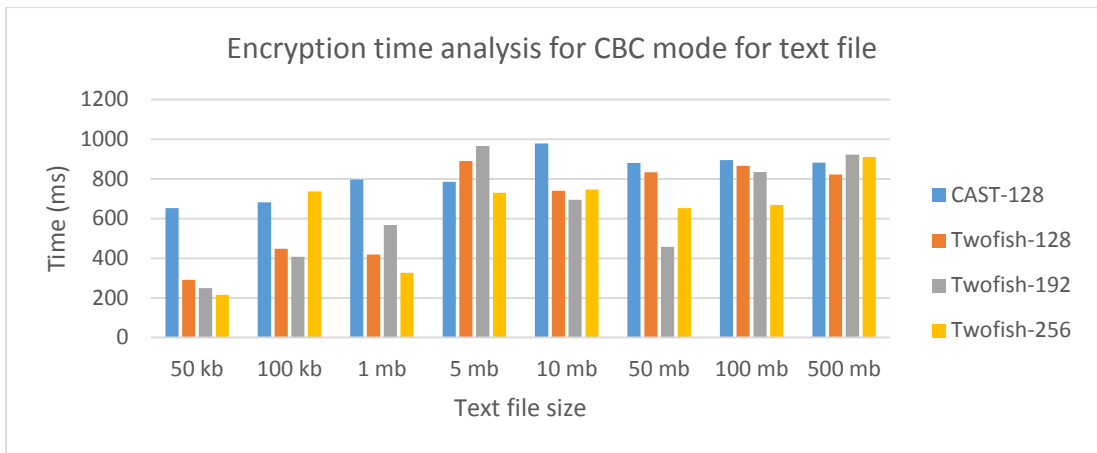


Figure 11: Encryption time analysis for CBC mode for text file

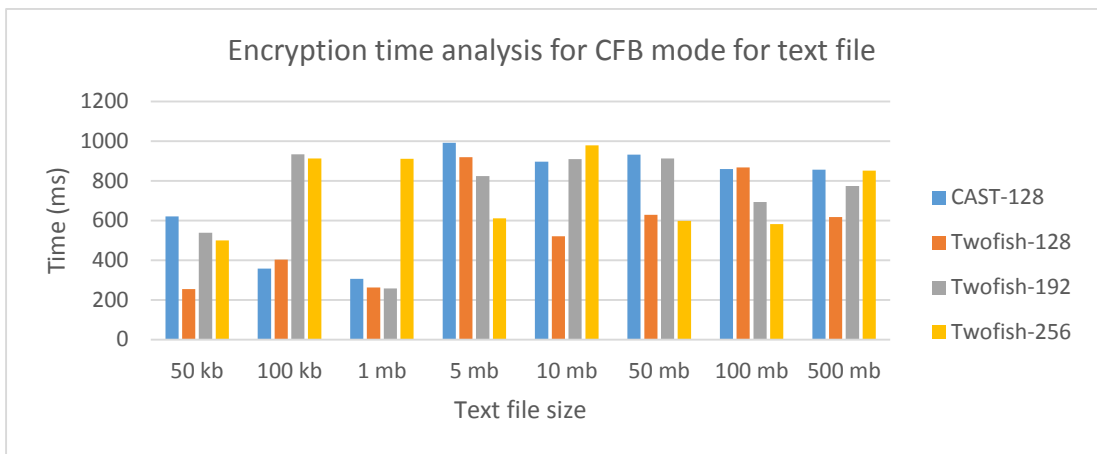


Figure 12: Encryption time analysis for CFB mode for text file

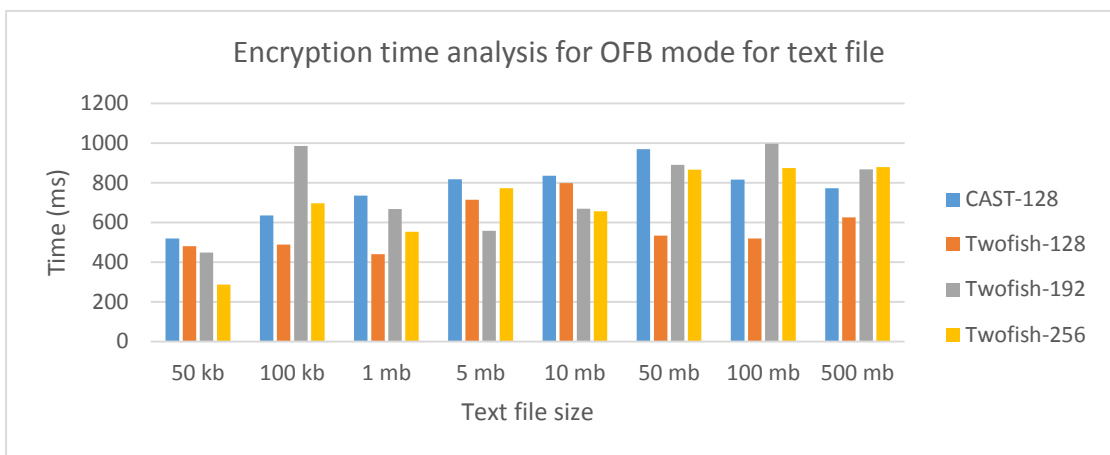


Figure 13: Encryption time analysis for OFB mode for text file

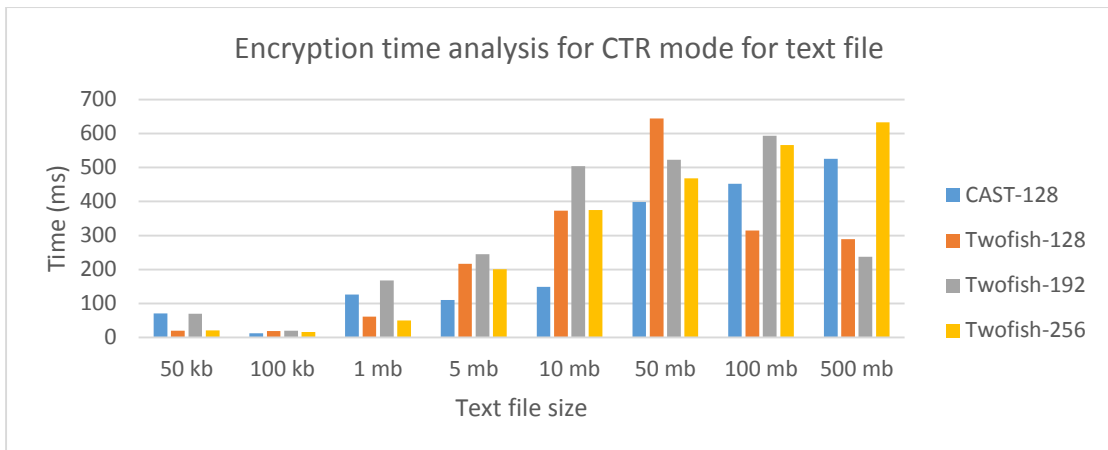


Figure 14: Encryption time analysis for CTR mode for text file

By analyzing all of the outcomes above for encryption time for text file for CAST-128, Twofish-128, Twofish-192 and Twofish-256 with ECB, CBC, CFB, OFB and CTR modes, on an average CAST-128 algorithm performed better (it took 231ms, which is smaller as compare to others) for encrypting text file in CTR mode of operation but for CBC mode, it performed worst (took 818ms, which is larger) compared with other modes of operation.

Encryption time analysis for image file

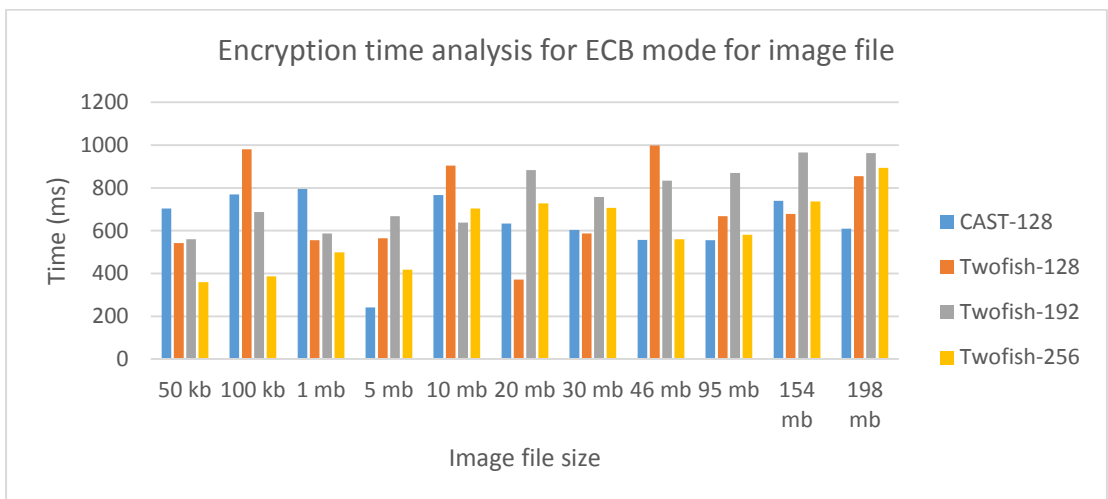


Figure 15: Encryption time analysis for ECB mode for image file

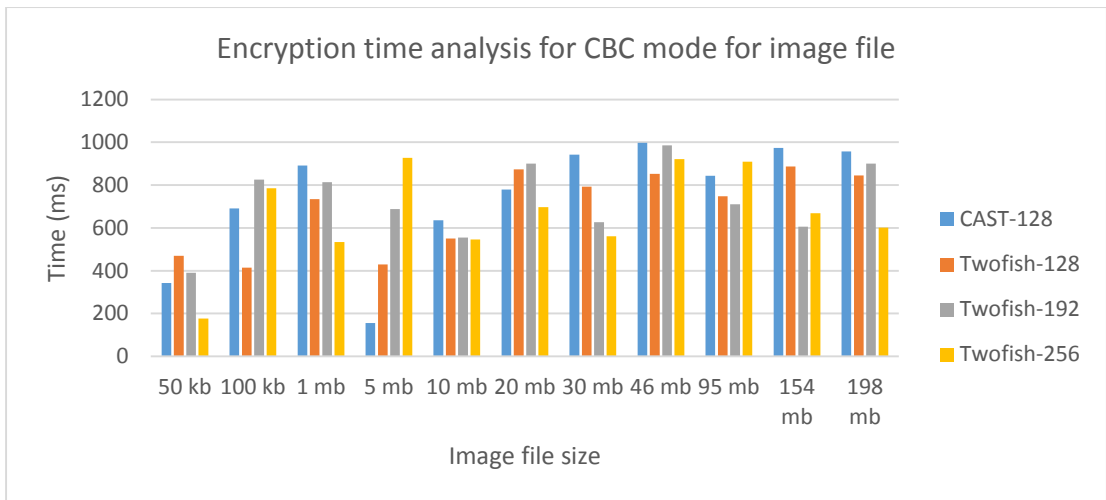


Figure 16: Encryption time analysis for CBC mode for image file

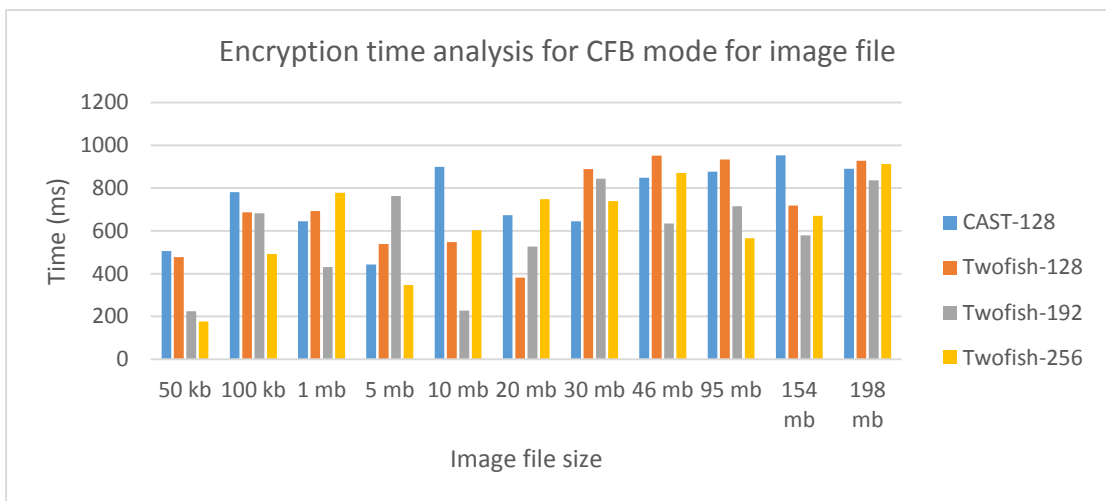


Figure 17: Encryption time analysis for CFB mode for image file

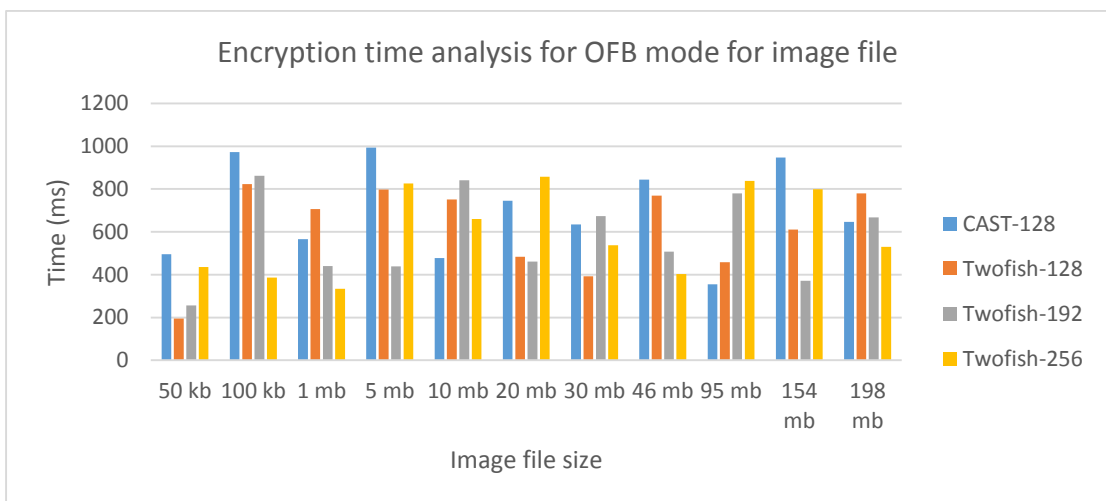


Figure 18: Encryption time analysis for OFB mode for image file

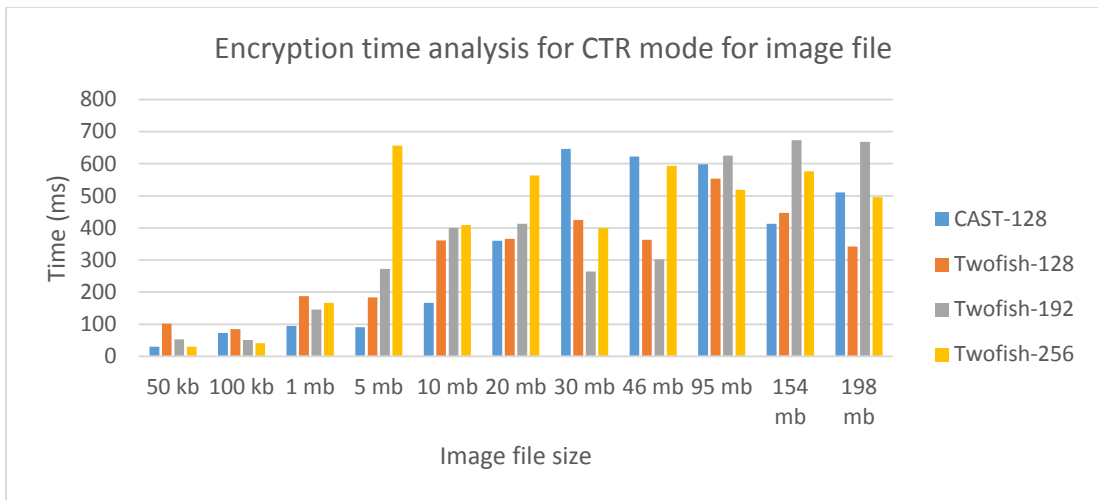


Figure 19: CAST-128 image encryption and decryption time for CTR mode

From the above, encryption time for image file was analyzed for CAST-128, Twofish-128, Twofish-192 and Twofish-256 with ECB, CBC, CFB, OFB and CTR modes, on an average, Twofish-128 algorithm performed better for encrypting image file in CTR mode of operation. It took approximately 310ms in CTR mode, which is much smaller as compare to other modes of operation. And Twofish-192 algorithm performed worst in ECB mode i.e., it took 764ms in ECB mode, which is much larger than others.

Decryption time analysis for text file

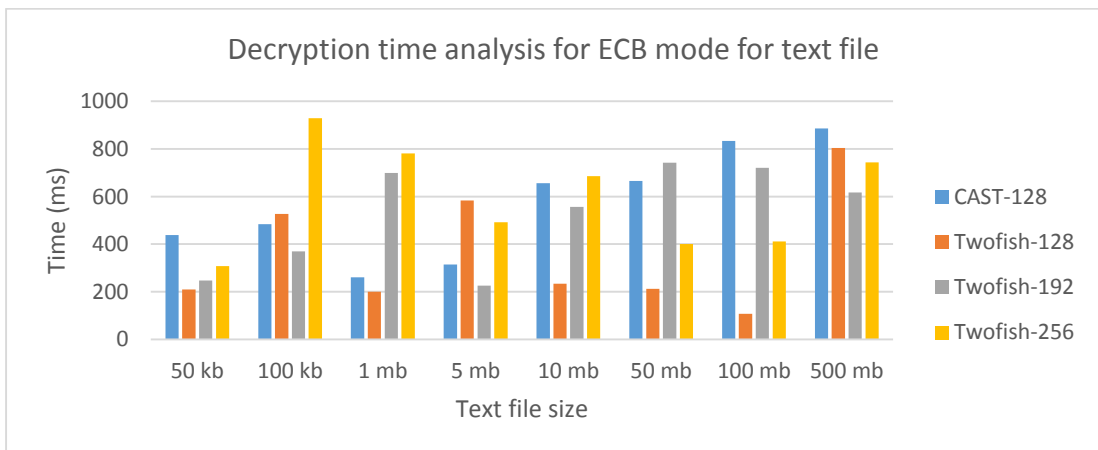


Figure 20: Decryption time analysis for ECB mode for text file

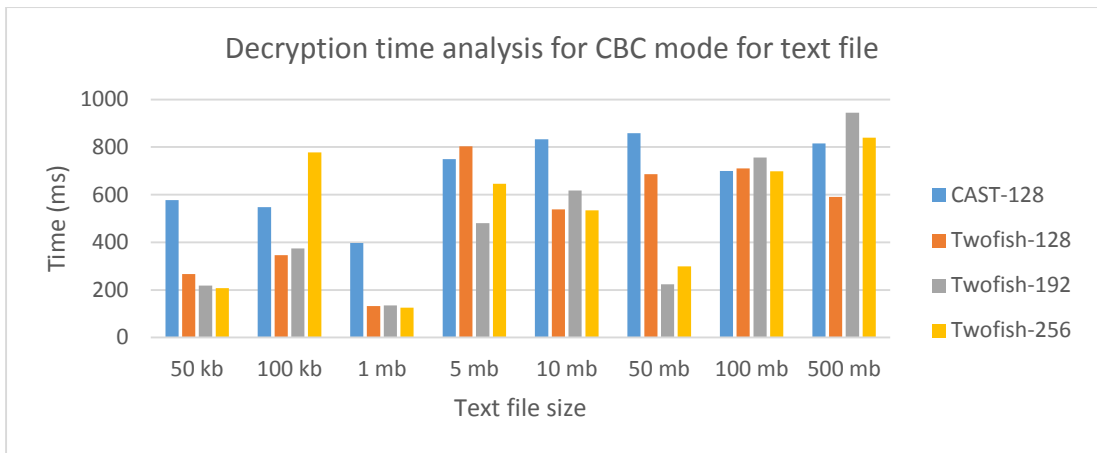


Figure 21: Decryption time analysis for CBC mode for text file

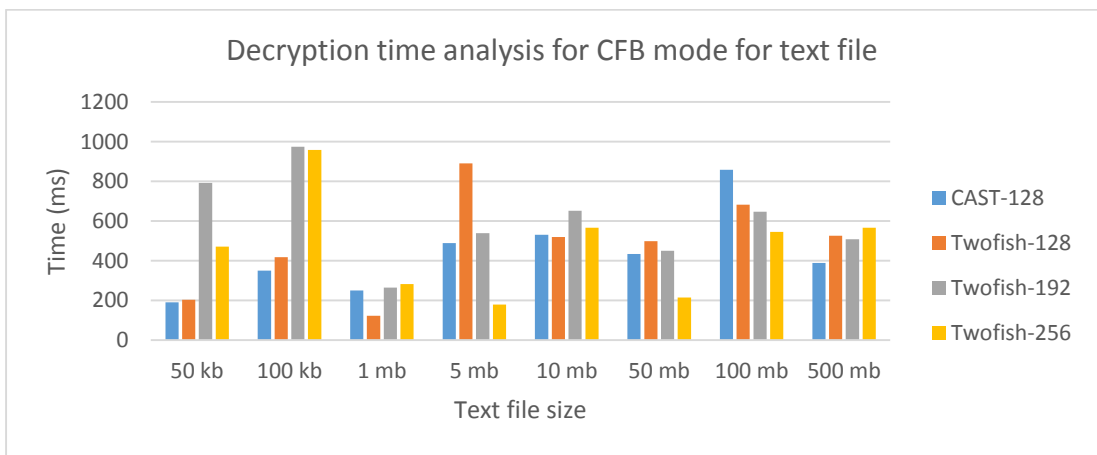


Figure 22: Decryption time analysis for CFB mode for text file

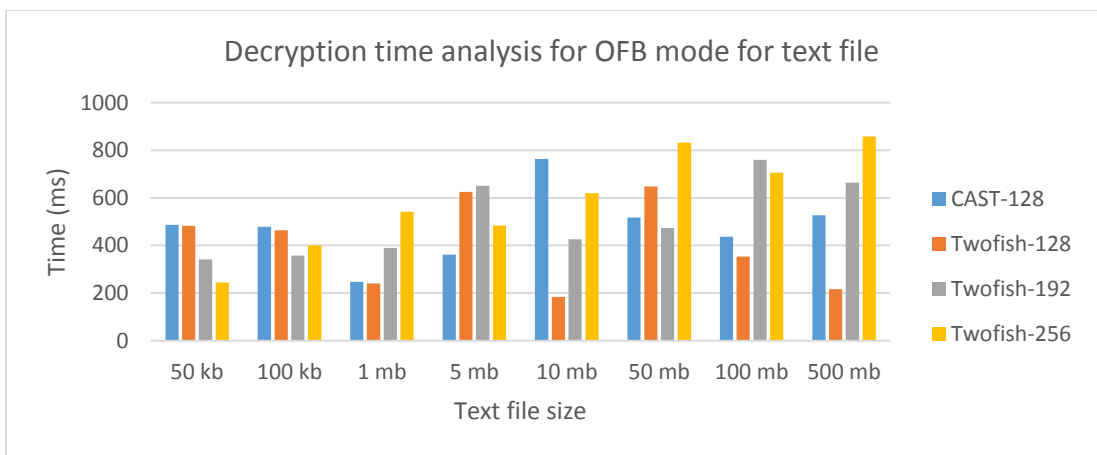


Figure 23: Decryption time analysis for OFB mode for text file

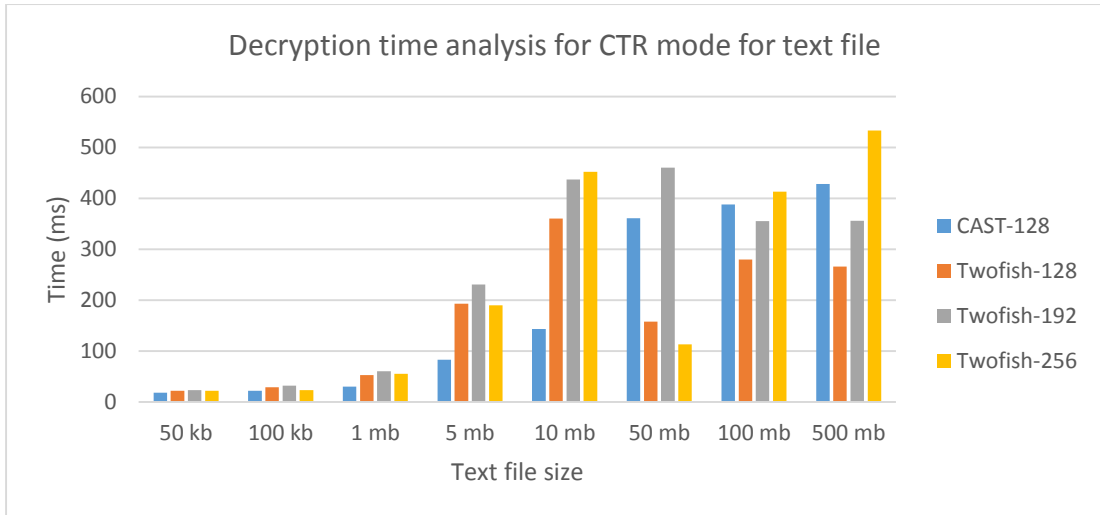


Figure 24: Decryption time analysis for CTR mode for text file

From the above, analyzing decryption time for text file in ECB, CBC, CFB, OFB and CTR mode of operation with CAST-128, Twofish-128, Twofish-192 and Twofish-256 algorithms, on an average, Twofish-128 algorithm performed better for decrypting text file in CTR mode of operation. It took approximately 170ms in CTR mode, which is much smaller as compare to other modes of operation. And CAST-128 algorithm performed worst in CBC mode i.e., it took 685ms in CBC mode, which is much larger than others.

Decryption time analysis for image file

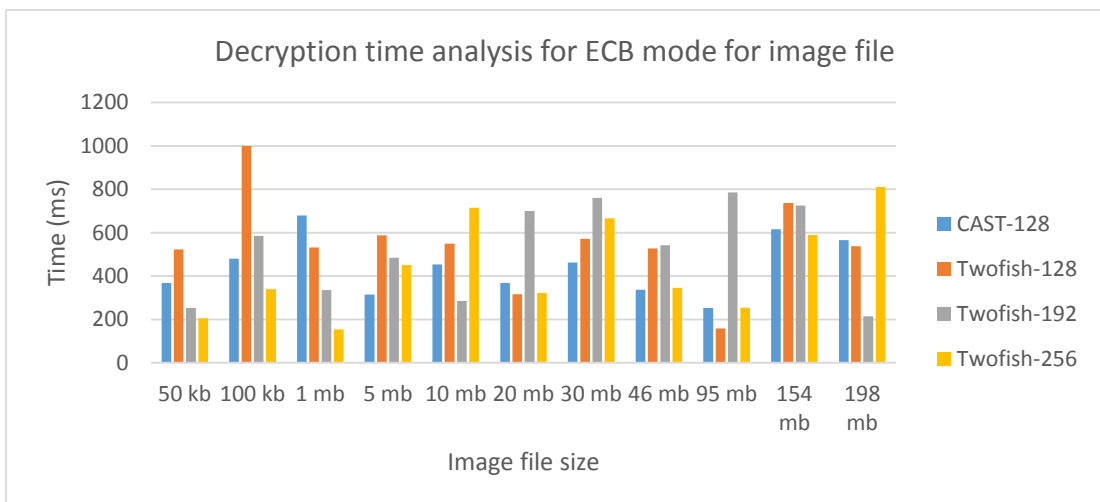


Figure 25: Decryption time analysis for ECB mode for image file

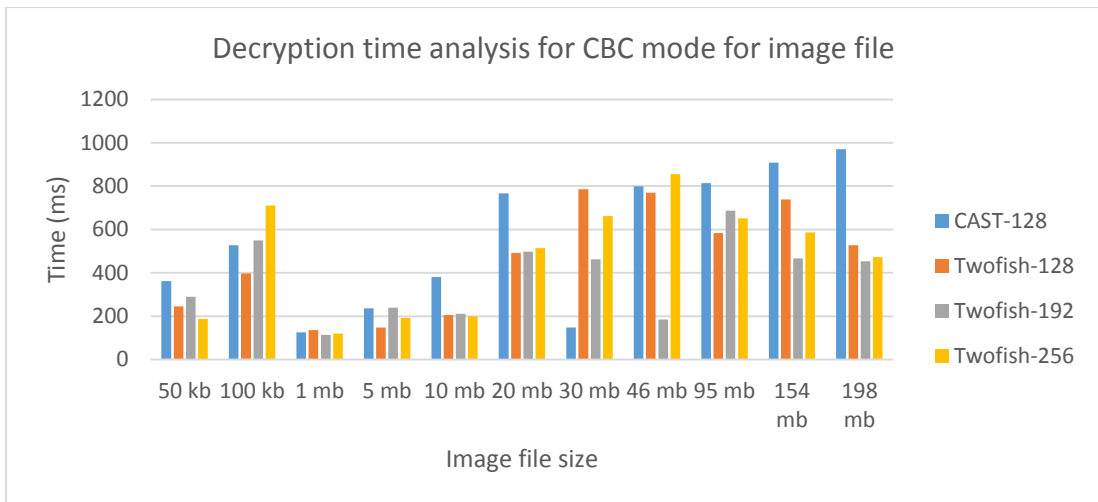


Figure 26: Decryption time analysis for CBC mode for image file

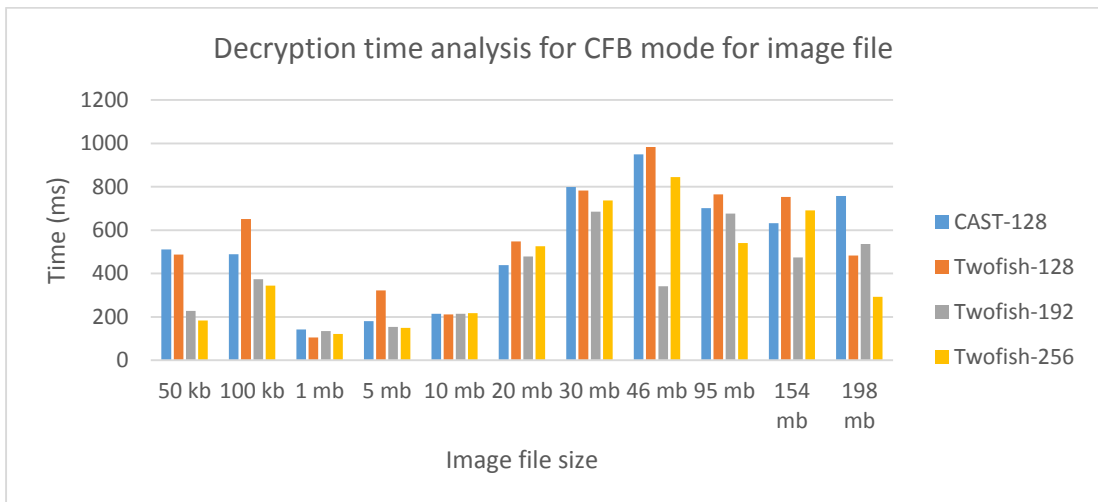


Figure 27: Decryption time analysis for CFB mode for image file

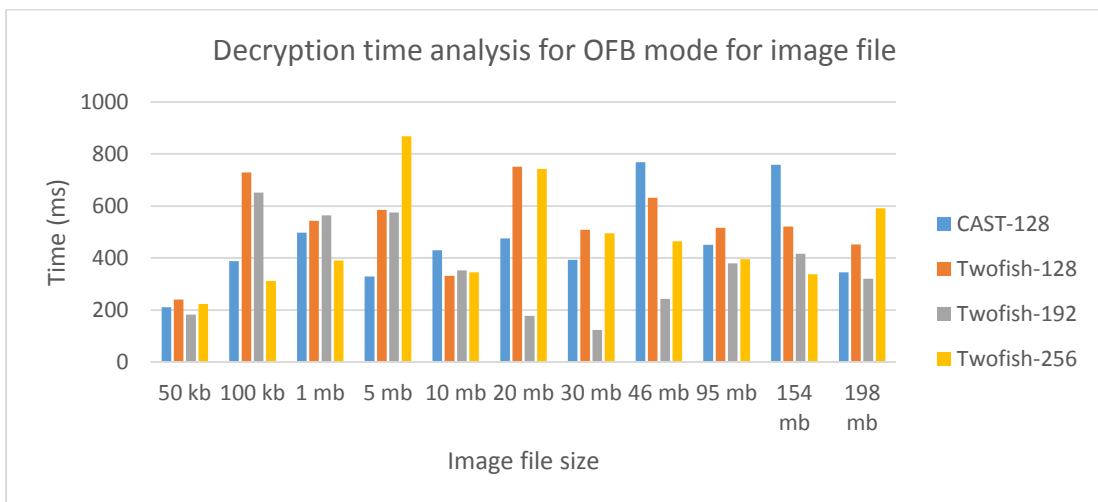


Figure 28: Decryption time analysis for OFB mode for image file

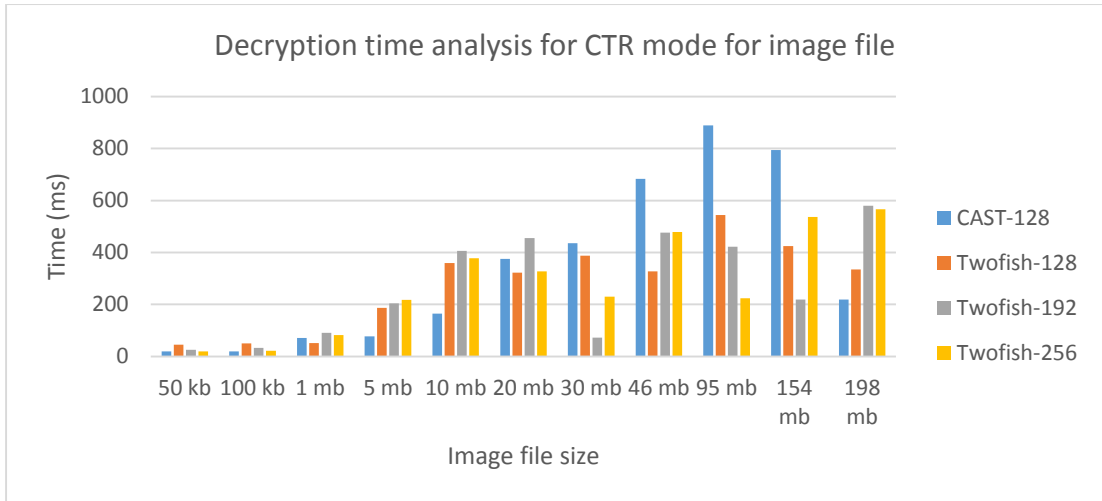


Figure 29: Decryption time analysis for CTR mode for image file

Analyzing decryption time for image file in ECB, CBC, CFB, OFB and CTR mode of operation with CAST-128, Twofish-128, Twofish-192 and Twofish-256 algorithms, on an average, Twofish-192 algorithm performed better for decrypting text file in CTR mode of operation. It took approximately 271ms in CTR mode, which is much smaller as compare to other modes of operation. And Twofish-128 algorithm performed worst in CFB mode i.e., it took 553ms in CFB mode, which is much larger than others.

Performance Results with throughput for text data

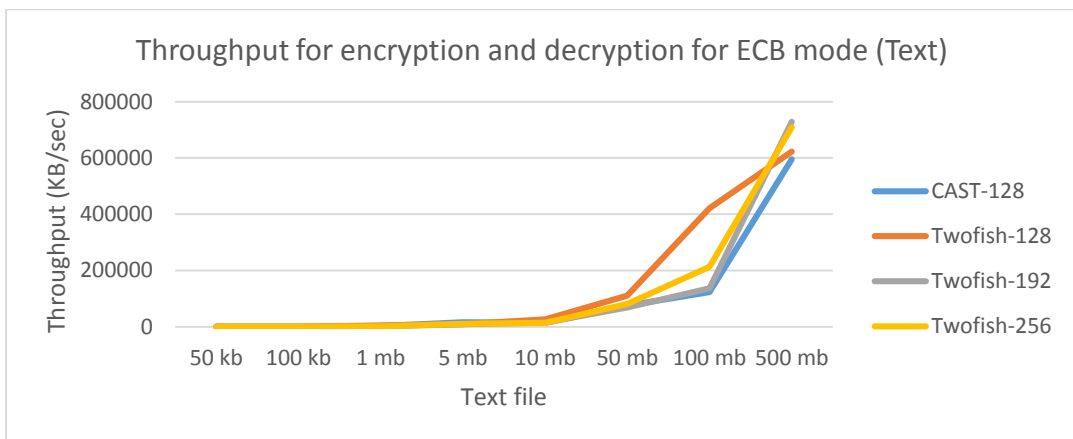


Figure 30: Throughput for encryption and decryption for ECB mode (Text)

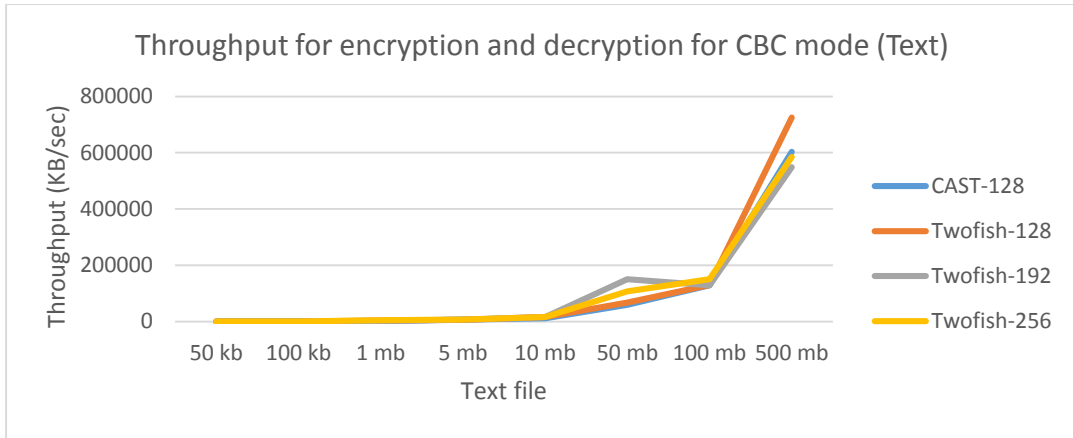


Figure 31: Throughput for encryption and decryption for CBC mode (Text)

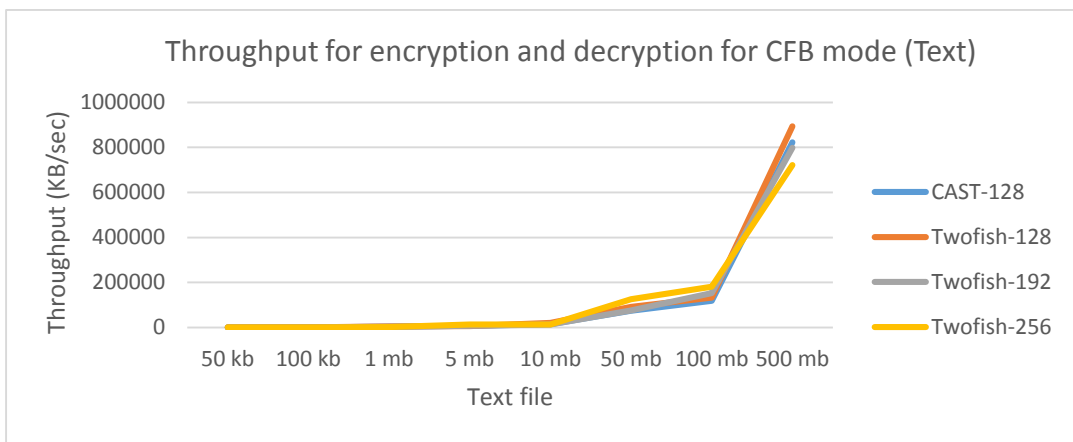


Figure 32: Throughput for encryption and decryption for CFB mode (Text)

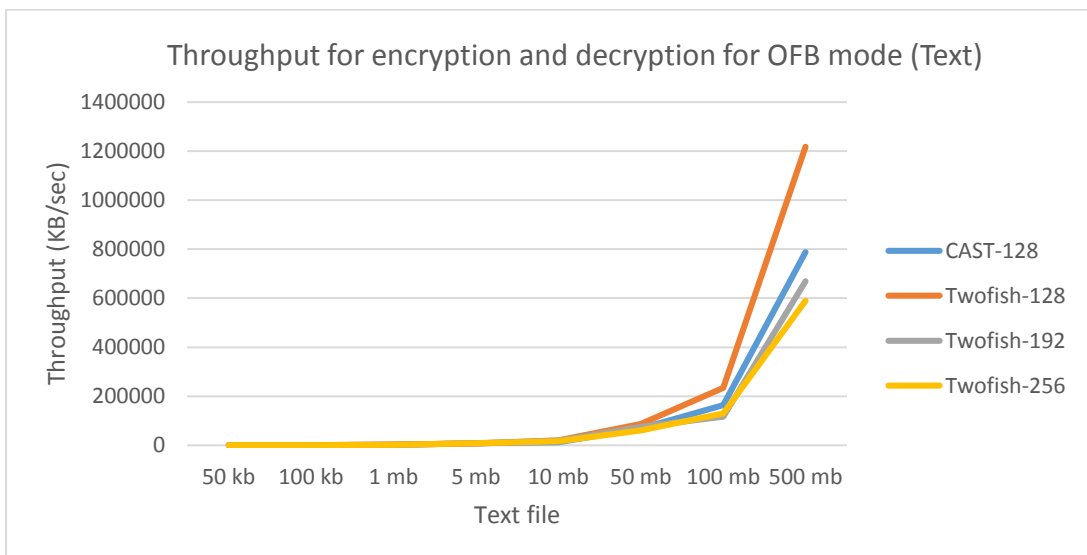


Figure 33: Throughput for encryption and decryption for OFB mode (Text)

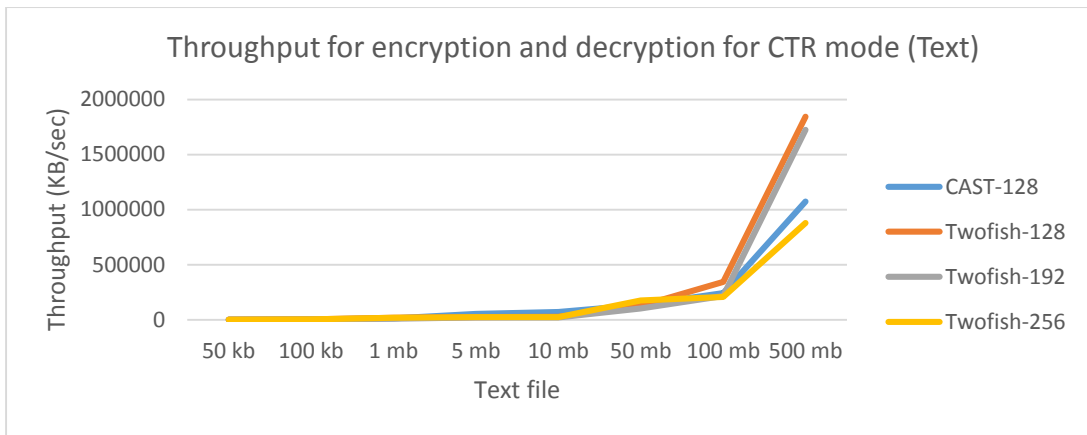


Figure 34: Throughput for encryption and decryption for CTR mode (Text)

Analyzing the above throughput graphs for encryption time and decryption time for text file in ECB, CBC, CFB, OFB and CTR mode, it is found that for all of the modes, there is no significant variation in throughput for text size below 10mb of size. But for greater than 10mb text file size, it is found that with gradual increase in file size, Twofish (Twofish-128, Twofish-192 and Twofish-256) algorithm is found to be 3 times better than CAST-128 algorithm with an average throughput value of 418906 KB/sec for Twofish and 199719 for CAST-128 algorithm.

Performance Results with throughput for image data

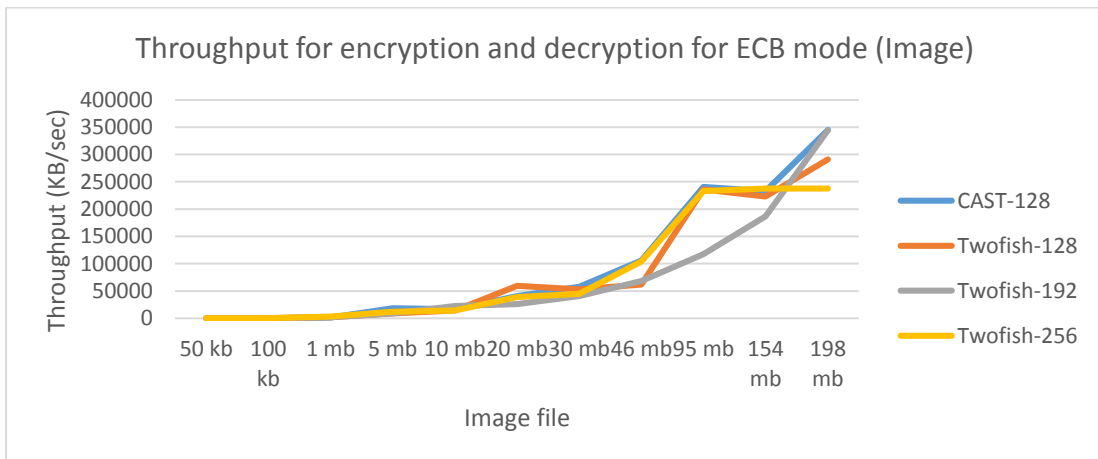


Figure 35: Throughput for encryption and decryption for ECB mode (Image)

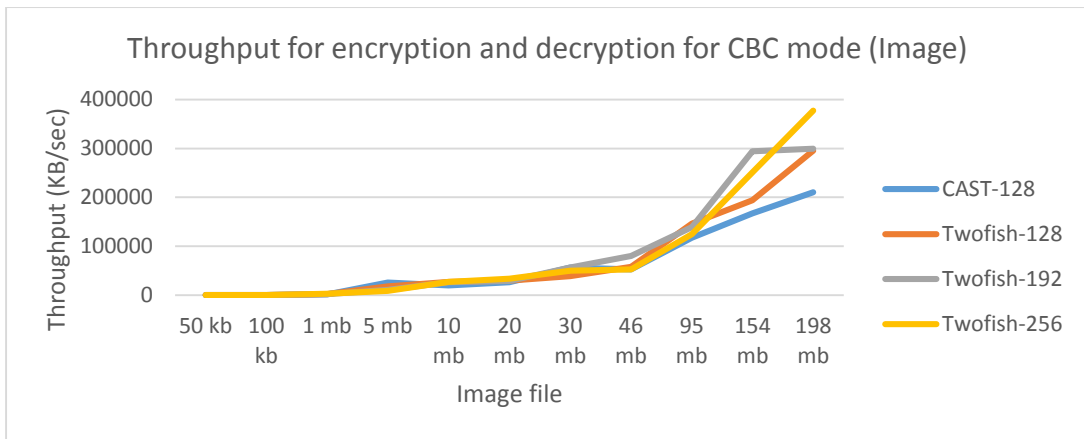


Figure 36: Throughput for encryption and decryption for CBC mode (Image)

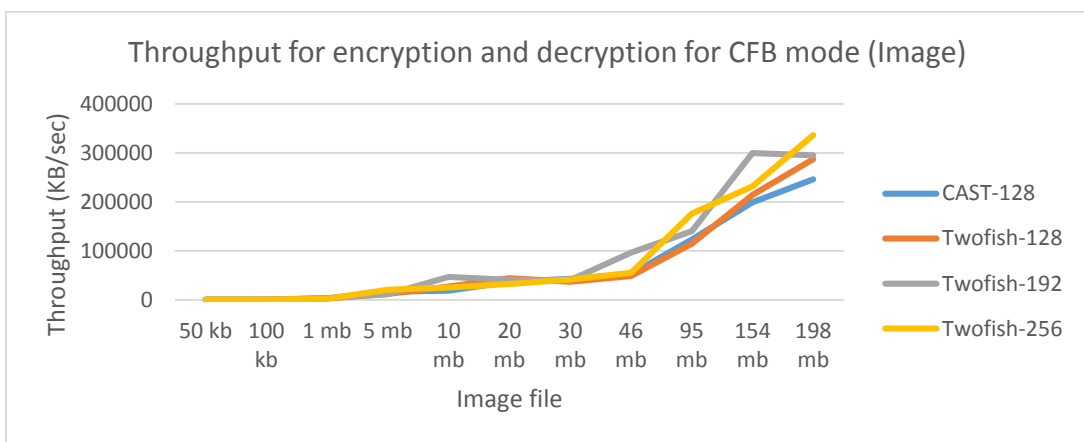


Figure 37: Throughput for encryption and decryption for CFB mode (Image)

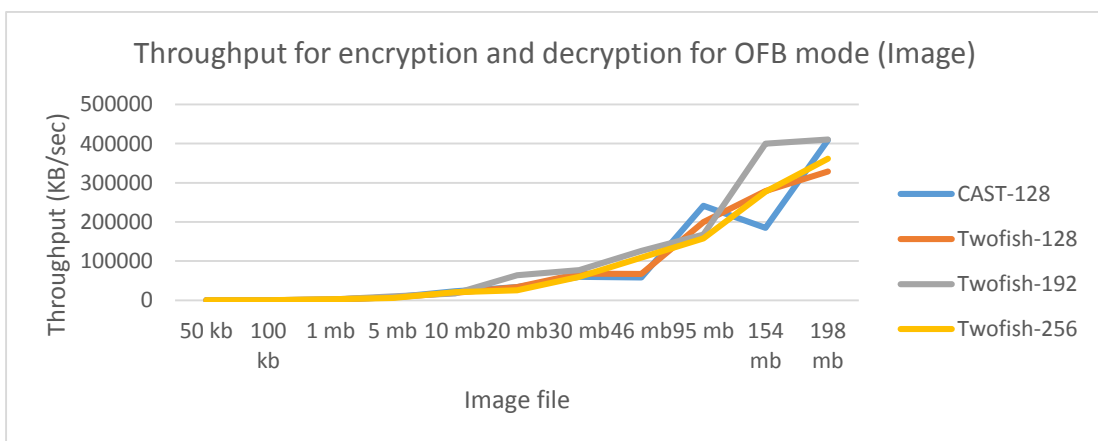


Figure 38: Throughput for encryption and decryption for OFB mode (Image)

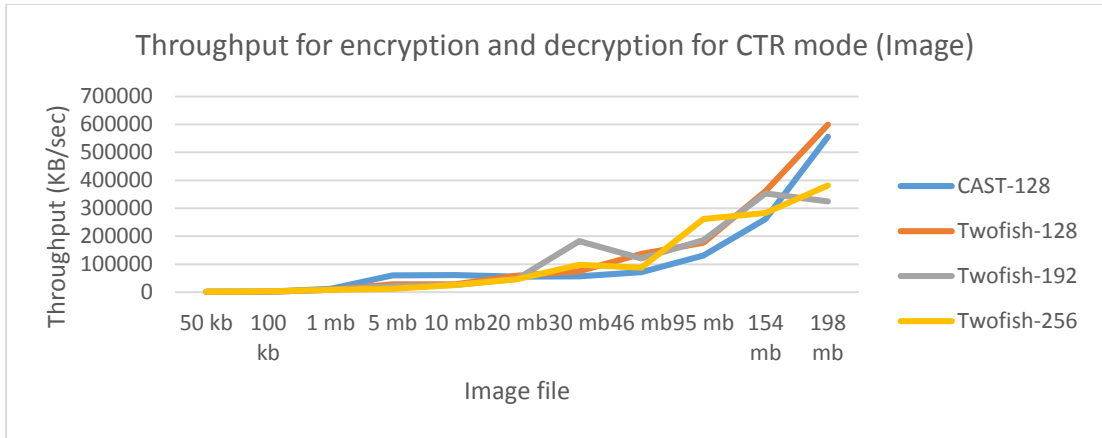


Figure 39: Throughput for encryption and decryption for CTR mode (Image)

Analyzing the above throughput graphs for encryption time and decryption time for image file in ECB, CBC, CFB, OFB and CTR mode, it is found that for all of the modes, there is no significant variation in throughput for image file size below 20mb of size. But for greater than 20mb image file size, it is found that with gradual increase in file size, Twofish algorithm is found to be 3 times better than CAST-128 algorithm with an average throughput value of 194623 KB/sec for Twofish and 114813 for CAST-128 algorithm. But for some cases CAST-128 is found better than Twofish algorithm, but this can be negligible because majority of case with high throughput yielded by Twofish algorithm.

Performance Results with memory utilization for text file

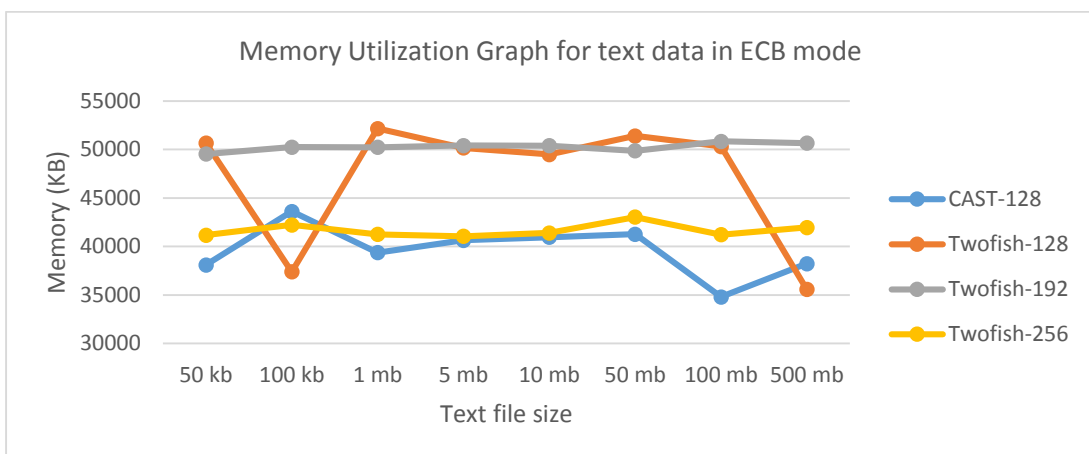


Figure 40: Memory Utilization Graph for text data in ECB mode

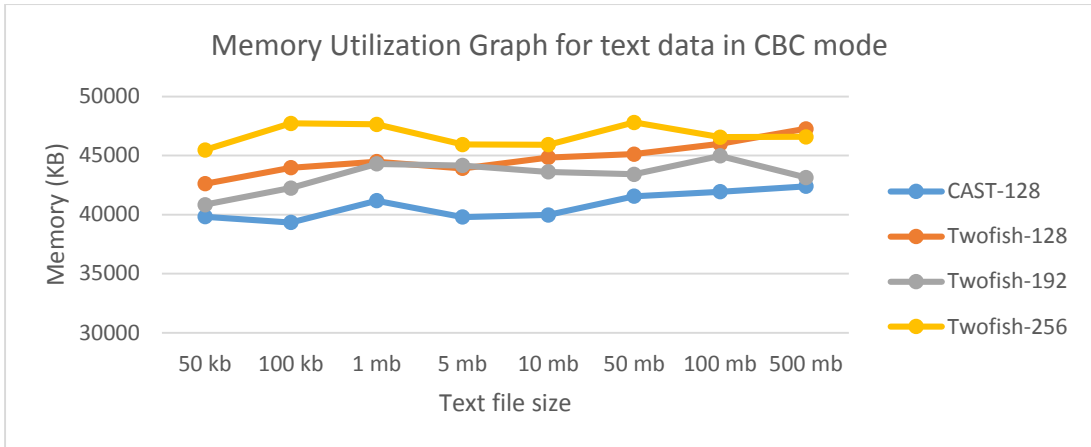


Figure 41: Memory Utilization Graph for text data in CBC mode

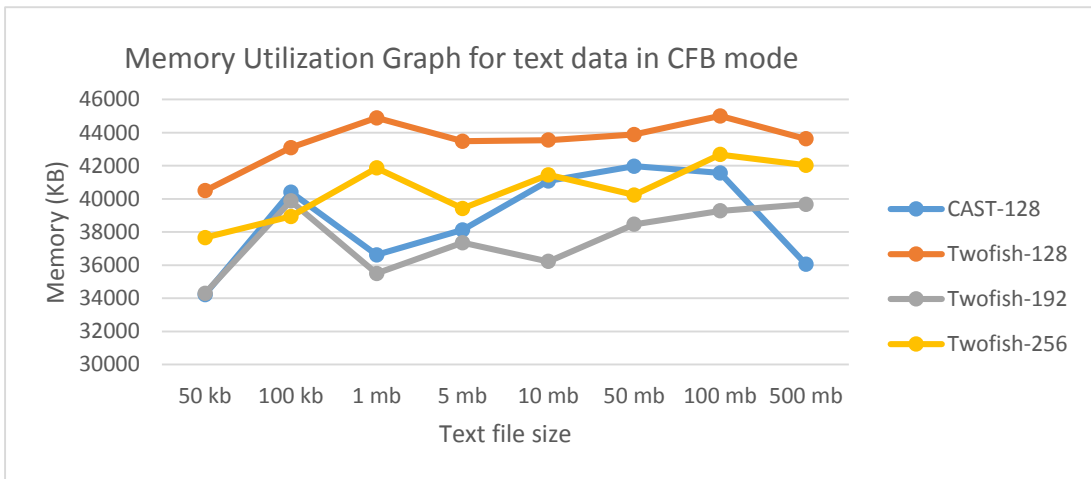


Figure 42: Memory Utilization Graph for text data in CFB mode

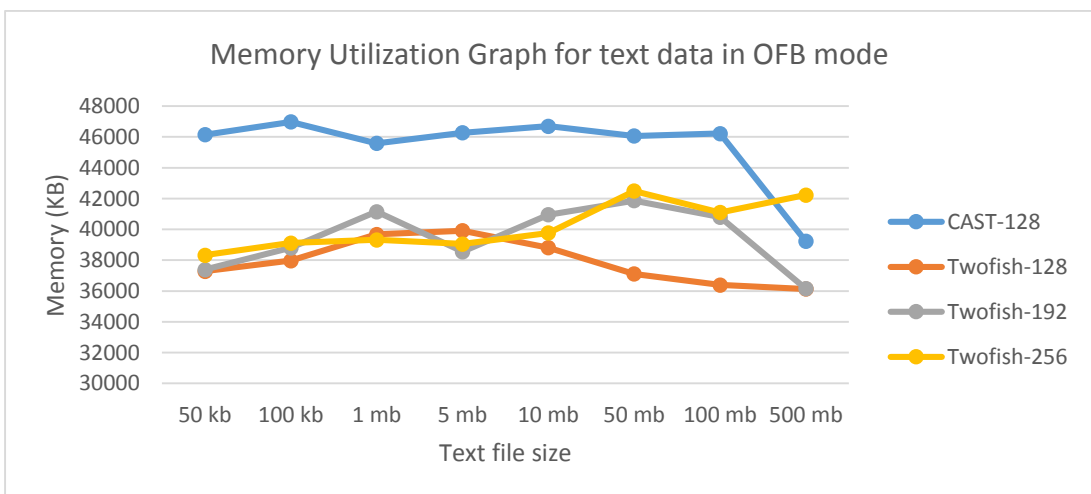


Figure 43: Memory Utilization Graph for text data in OFB mode

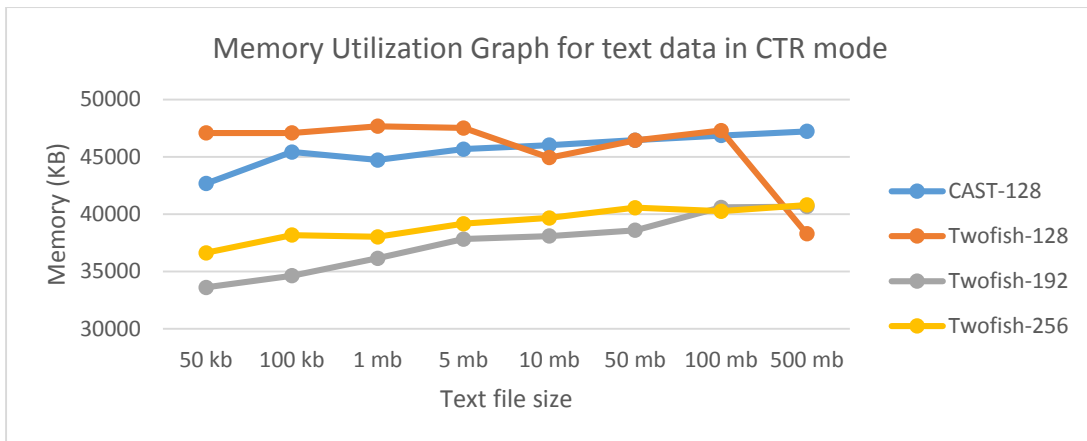


Figure 44: Memory Utilization Graph for text data in CTR mode

From the above memory utilization graph, it is found that none of the algorithm has constant memory utilization. But out of that randomness, CAST-128 and Twofish-192 took minimum memory utilization as compared to other in most of the cases. But for some case, CAST-128 algorithm exhibited maximum memory than other algorithm such as in OCB mode. But as compared to its numeric value (45402 KB), it is less than other algorithms in other cases.

Performance Results with memory utilization for image file

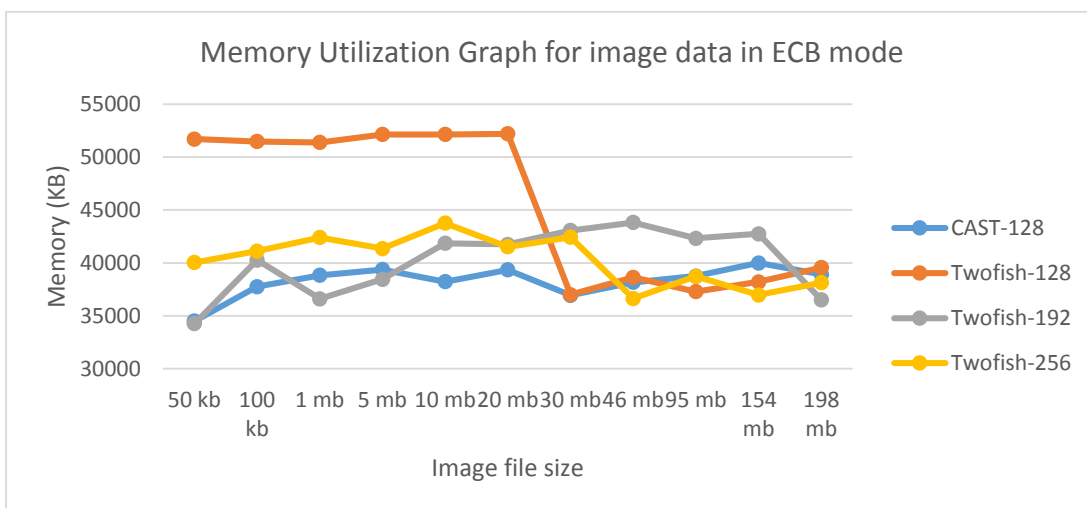


Figure 45: Memory Utilization Graph for image data in ECB mode

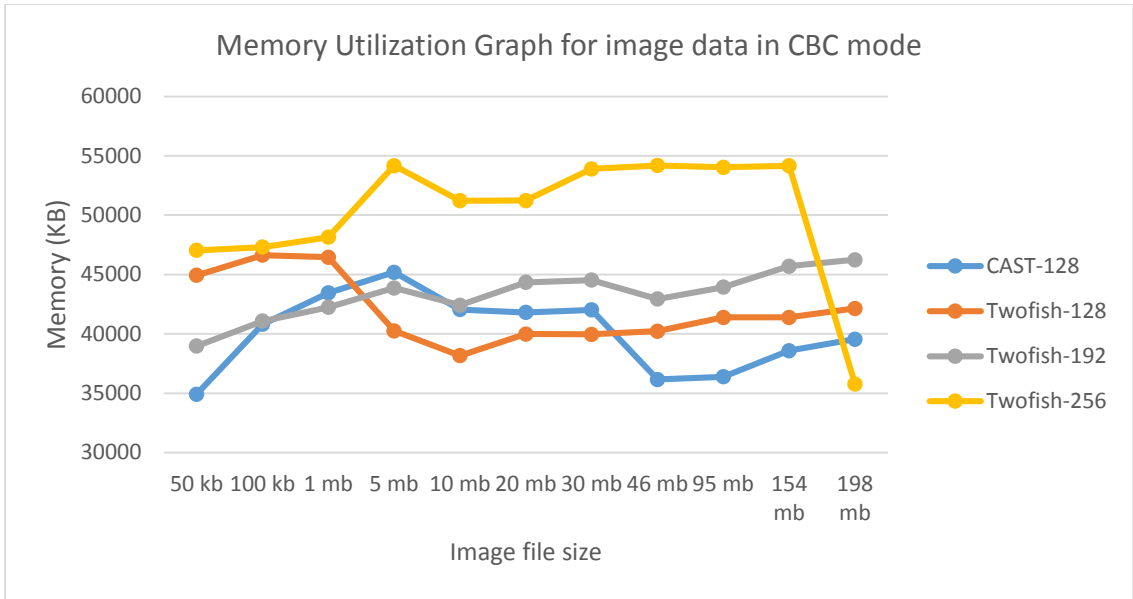


Figure 46: Memory Utilization Graph for image data in CBC mode

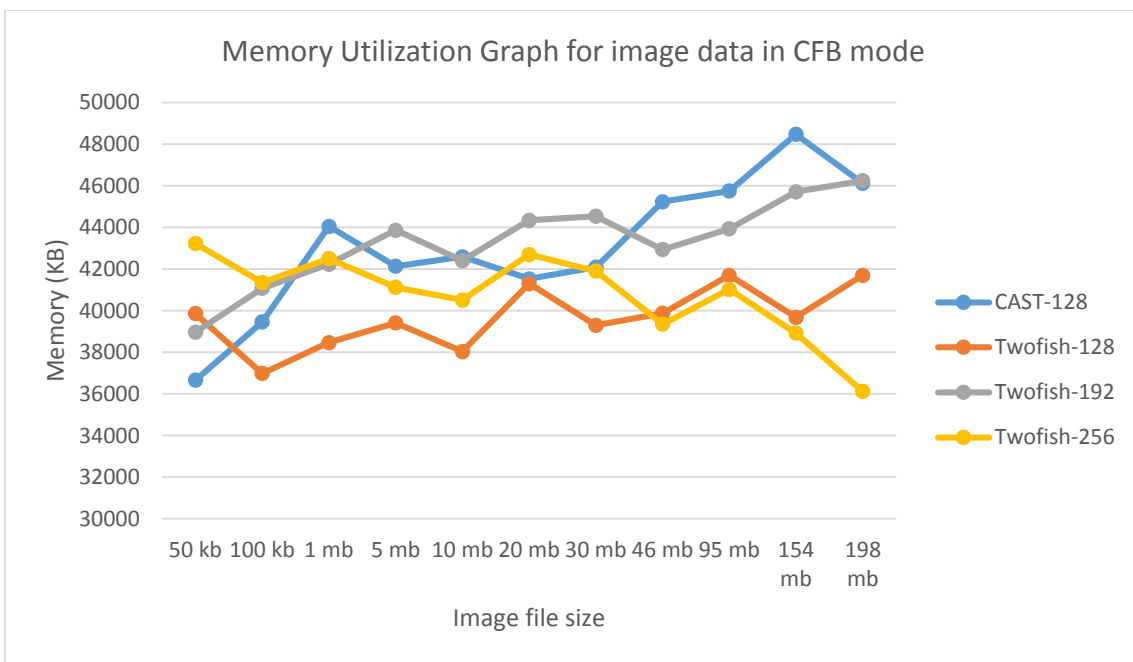


Figure 47: Memory Utilization Graph for image data in CFB mode

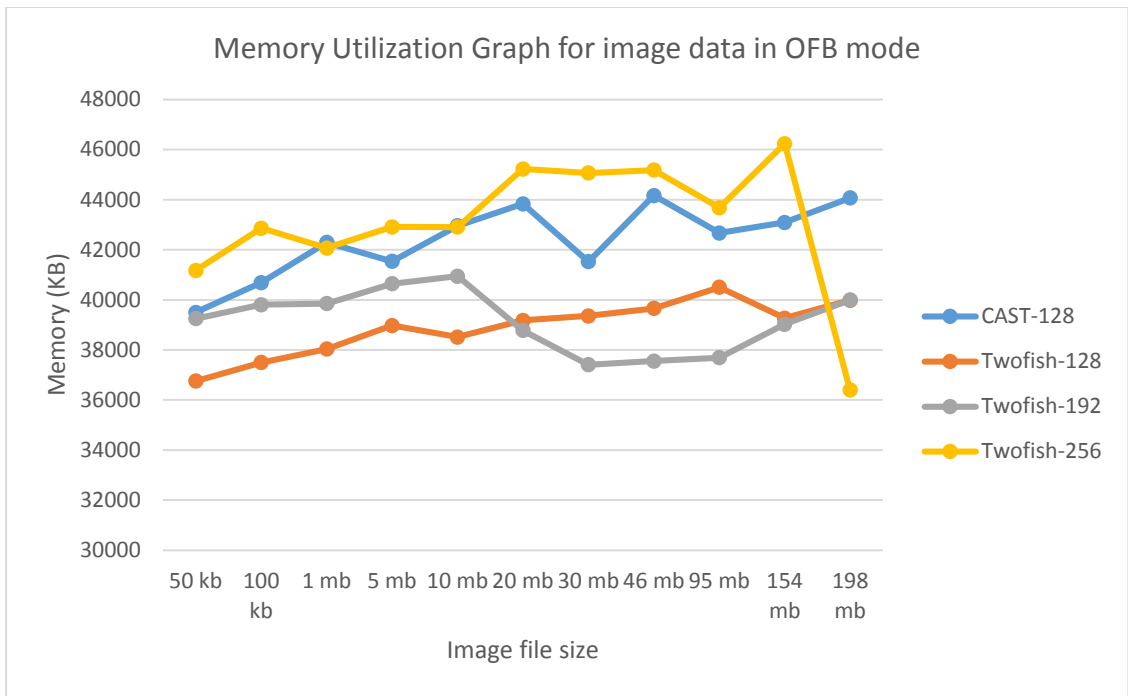


Figure 48: Memory Utilization Graph for image data in OFB mode

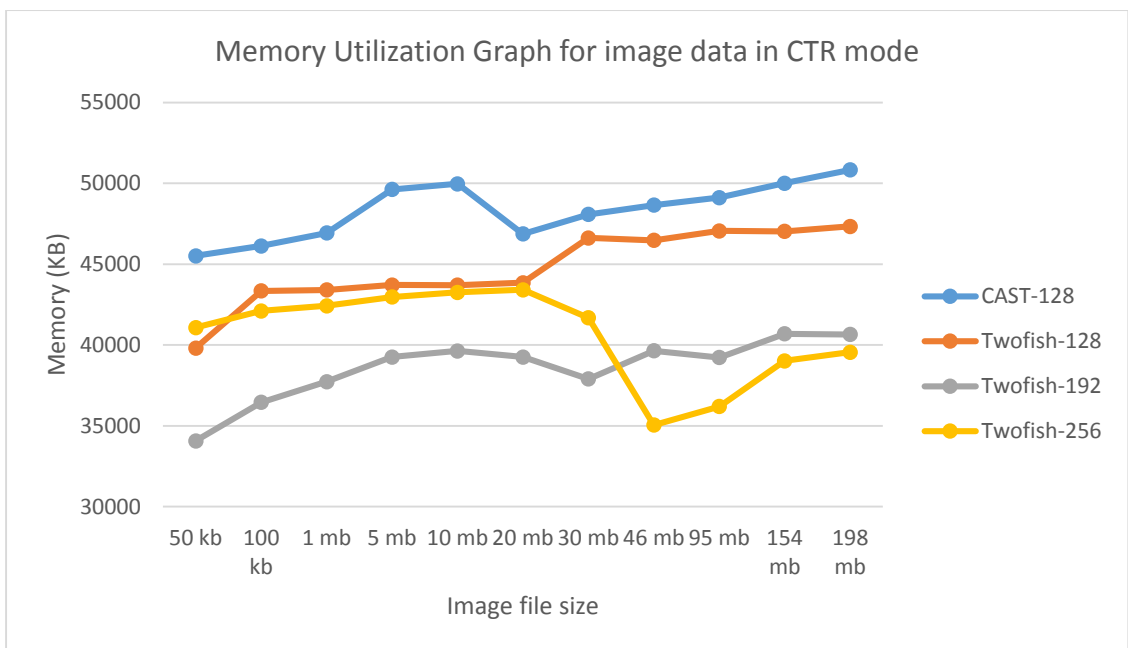


Figure 49: Memory Utilization Graph for image data in CTR mode

From the above memory utilization graph, it is found that none of the algorithm has constant memory utilization. But out of that randomness, Twofish-128 and Twofish-256 took minimum memory utilization as compare to other in most of the cases. But it also not guarantying for better memory consumption (minimum memory consumption) as compare to other in every cases.

4.3.2. Differential analysis

Differential analysis is done for security measures. It is a technique which observes how difference in input affects differences on the output. NPCR (Number of pixel change) and UACI (Unified Average Change Intensity) are the two widely used security analyses in image encryption community for differential analysis. NPCR concentrates on the absolute number of pixels which changes value in differential attacks while the UACI focuses on the averaged difference between two paired images (original image and decrypted image).

The different types of images are taken for experiments to analyze the algorithm performance. The images taken here are the same images which were used in encryption and decryption process described in test data description. The difference in NPCR and UACI of different algorithm is as follows:

Table 2: NPCR and UACI Measures


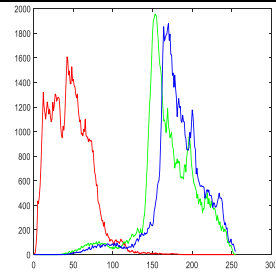
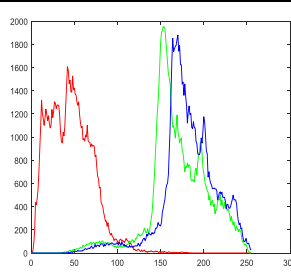
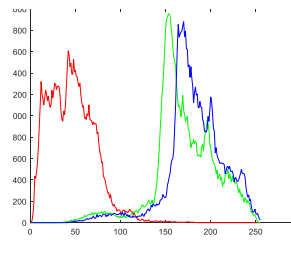
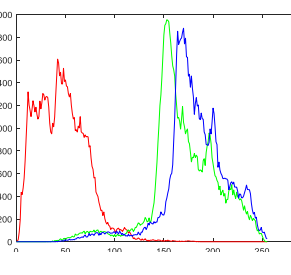
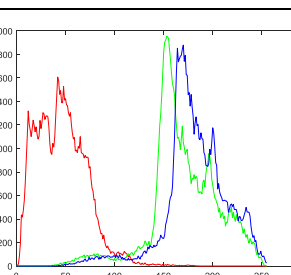

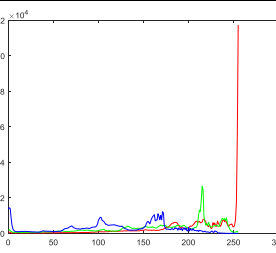
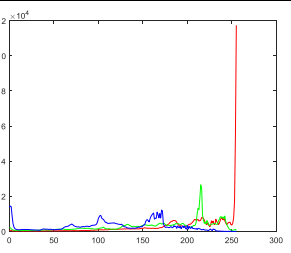
S. N	Image	Dimension	Method	NPCR (%)	UACI (%)
1	butterfly.jpg	300*300 (50 KB)	CAST-128 / TWOFISH-128 / TWOFISH-192 / TWOFISH-256	0	0
2	laptop.jpg	1050*700 (100 KB)	CAST-128 / TWOFISH-128 / TWOFISH-192 / TWOFISH-256	0	0
3	wing.jpg	2192*2921 (1 MB)	CAST-128 / TWOFISH-128 / TWOFISH-192 / TWOFISH-256	0	0
4	GeoEye.jpg	11846*9945 (46 MB)	CAST-128 / TWOFISH-128 / TWOFISH-192 / TWOFISH-256	0	0
5	Airbus-Spot.tif	5181*4828 (95 MB)	CAST-128 / TWOFISH-128 / TWOFISH-192 / TWOFISH-256	0	0
6	world.png	21600*21600 (154 MB)	CAST-128 / TWOFISH-128 / TWOFISH-192 / TWOFISH-256	0	0

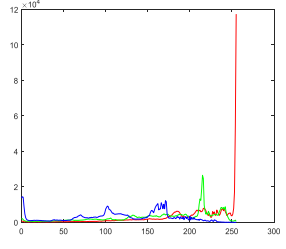
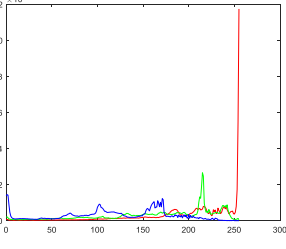
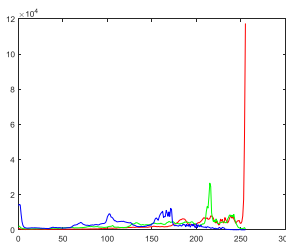

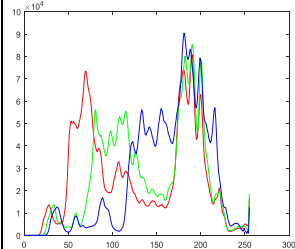
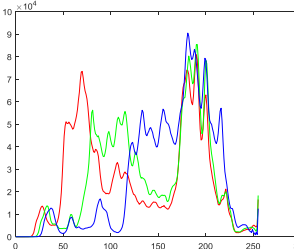
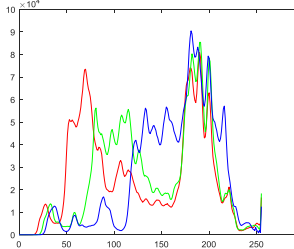
The above results shows the different technique in differential analysis for the image. It is observed that all of the four techniques give no difference in between two paired images (original image and decrypted image). That means the decrypted image has no change in pixel and pixel intensity. So the decrypted images retrieved are of best quality.

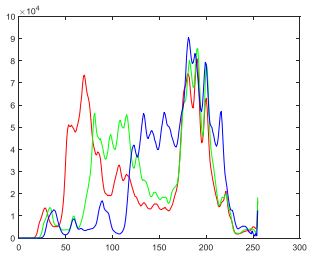
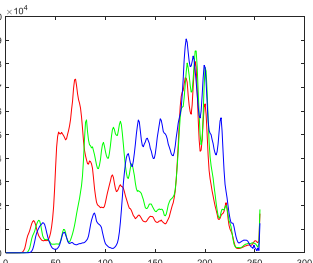

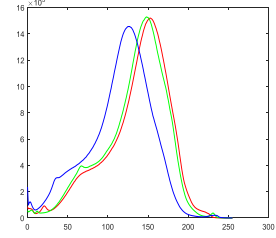
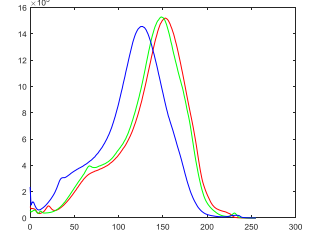
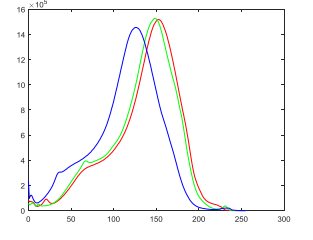
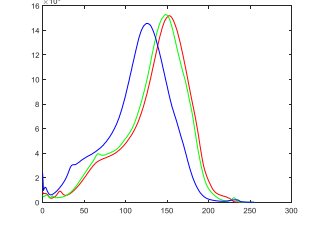
4.3.3. Statistical analysis

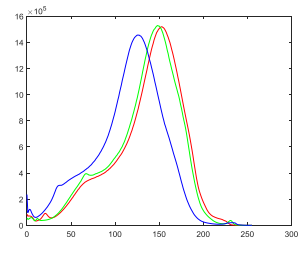

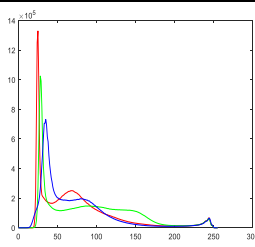
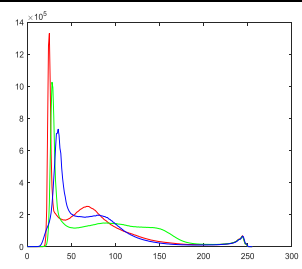
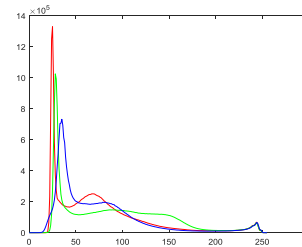
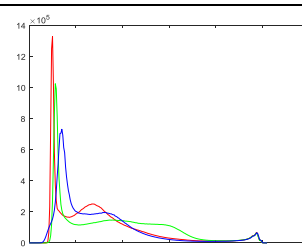
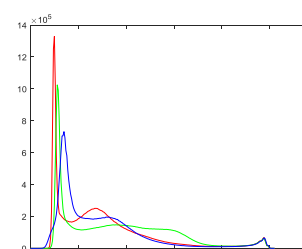
Statistical analysis has been carried out using histogram analysis. The histogram of an image normally refers to a histogram of the pixel intensity values. This histogram is a graph showing the number of pixels in an image at each different intensity value found in that image. The images taken here are the same images which were used in encryption and decryption process described in test data description. Histogram of input image and decrypted image is analyzed graphically as follows:

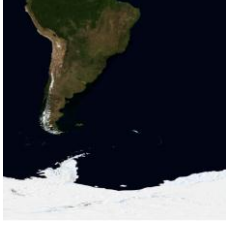
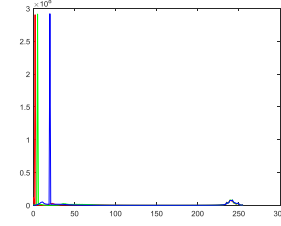
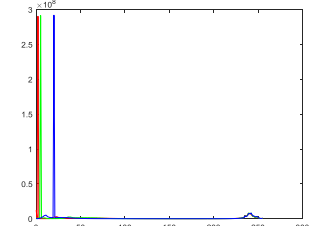
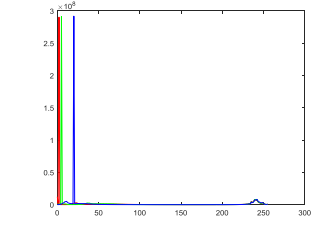
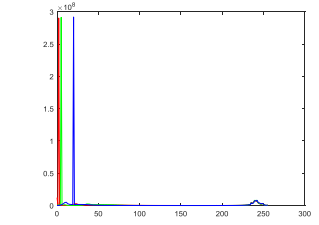
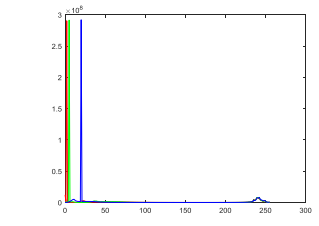
Table 3: Histogram Analysis

S. N	Image	Original Image Histogram	Method	Decrypted Image Histogram
1	 <p>butterfly.jpg</p>		CAST-128	
			TWOFISH-128	
			TWOFISH-192	
			TWOFISH-256	
2	 <p>laptop.jpg</p>		CAST-128	

			TWOFISH -128	
			TWOFISH -192	
			TWOFISH -256	
3	 wing.jpg		CAST-128	
			TWOFISH -128	

			TWOFISH -192	
			TWOFISH -256	
4	 GeoEye.jpg		CAST-128	
			TWOFISH -128	
			TWOFISH -192	

			TWOFISH -256	
5	 Airbus-Spot.tif		CAST-128	
			TWOFISH -128	
			TWOFISH -192	
			TWOFISH -256	

6	 world.png		CAST-128	
			TWOFISH -128	
			TWOFISH -192	
			TWOFISH -256	

From the above resultant histogram of original image and decrypted cipher image using all four techniques, it is observed that histograms are hardly distinguishable. The curves that are obtained from CAST-128, Twofish-128, Twofish-192 and Twofish-256 are almost identical as the difference during encryption and decryption is extremely small.

4.3.4. Visual assessment analysis

Visual assessment analysis is done to measure the performance of the decryption procedure. For that the PSNR value and MSE value will be calculated. It is the ratio of mean square difference of the component for the two images to the maximum mean square difference that can exist between any two images. Greater the value of PSNR higher the image quality. The images taken here are the same images which were used

in encryption and decryption process described in test data description. PSNR and MSE for the sample test data image are as show in the table below:

Table 4: PSNR Measures

S. N	Image	Dimension	Method	MSE (dB)	PSNR (dB)
1	butterfly.jpg	300*300 (50 KB)	CAST-128 / TWOFISH-128 / TWOFISH-192 / TWOFISH-256	0	Inf
2	laptop.jpg	1050*700 (100 KB)	CAST-128 / TWOFISH-128 / TWOFISH-192 / TWOFISH-256	0	Inf
3	wing.jpg	2192*2921 (1 MB)	CAST-128 / TWOFISH-128 / TWOFISH-192 / TWOFISH-256	0	Inf
4	GeoEye.jpg	11846*9945 (46 MB)	CAST-128 / TWOFISH-128 / TWOFISH-192 / TWOFISH-256	0	Inf
5	Airbus-Spot.tif	5181*4828 (95 MB)	CAST-128 / TWOFISH-128 / TWOFISH-192 / TWOFISH-256	0	Inf
6	world.png	21600*21600 (154 MB)	CAST-128 / TWOFISH-128 / TWOFISH-192 / TWOFISH-256	0	Inf

The PSNR value of the original image with decrypted image is calculated and it is observed that the value of MSE is 0 in all the cases and therefore PSNR value as infinity. This depicts that all of the algorithm has good visual assessment. Higher the PSNR value, lesser destruction of image properties hence more improvement in the decrypted image is obtained.

4.4. Result

The overall analysis of text and image data analysis shows that CAST-128 algorithm is found to be effective while encrypting text file in CTR mode of operation but CBC mode, this algorithm performed worst. In case of image file encryption, Twofish-128 algorithm performed best 3 times than Twofish-192 algorithm. Similarly for decrypting text file, Twofish-128 algorithm performed 4 times best than CAST-128 algorithm whereas for image file, Twofish-192 algorithm performed better by 2 times than Twofish-128 algorithm in CTR mode. So in overall, Twofish algorithm is found to be more effective and, CTR mode is found to be more effective among all other modes of operation.

In terms of throughput analysis, Twofish-128 algorithm is found to be more efficient in text data analysis with respect to each text file throughput. For image data analysis, Twofish-256 algorithm performed efficiently in most of the image file encryption whereas CAST-128 algorithm is found to be incompetent. Average throughput for text file analysis by Twofish algorithm is 418906 KB/sec and for image file analysis is 194623 KB/sec. Here, Twofish algorithm is found to be 3 times effective than CAST-128 algorithm.

In terms of memory utilization analysis, in overall, Twofish algorithm has performed well consuming less memory compared to other algorithms for text and image data analysis. CAST-128 algorithm is found to be effective of some cases but as compared with other algorithms in other cases, it is still ineffective. And it is found that all of the algorithm has uniform memory utilization irrespective of data size. With the increase of file size, no significant change in memory utilization is observed.

Differential analysis of image data shows that all of the four algorithmic methods give good result in differential analysis. Results show that no any difference in between original image and decrypted image. It proves that all of the algorithm performed best in differential analysis so there is no change in recovered image.

Statistical analysis of images shows that histograms of original image and decrypted cipher image are hardly distinguishable. The curves that are obtained from CAST-128,

Twofish-128, Twofish-192 and Twofish-256 are almost identical as the difference during encryption and decryption is extremely negligible.

Visual analysis of images depicts that the value of MSE is 0 in all the cases and therefore, PSNR value as infinity. This shows that all of the algorithm has good visual assessment. Higher the PSNR value, lesser destruction of image properties hence more improvement in the decrypted image is obtained.

CHAPTER 5

CONCLUSION LIMITATIONS AND FUTURE RECOMMENDATION

5.1. Conclusion

Number of approach has been invented for the secure encryption mechanism. In this study CAST-128, Twofish-128, Twofish-192 and Twofish-256 algorithms have been implemented. ECB, CBC, CFB, OFB and CTR modes of operation have been configured with all the algorithm techniques. The text data and image data sets of different sizes and different types were taken into account. All of the data sets were tested with the algorithm to measure the strength of algorithm. Overall analysis and result from the above discussion conclude that the variants of Twofish algorithms has performed best and among the five modes of operation, CTR mode of operation has outperformed in terms of execution process. Twofish algorithm also outperforms in throughput analysis. Similarly, this algorithm performed good in terms of memory consumption as well. In terms of image quality analysis, all of the algorithms performed outstanding work. This algorithm is found to be effective approximately three times than CAST-128 algorithm. However for text data encryption time analysis, CAST-128 algorithm performed better than Twofish variants. From these analysis we can also conclude that bigger block and keys can improve the security of encryption technique. But this also decreases encryption and decryption speed. For image files with high graphics and plain image, it is also found that these factors also affect the processing speed such as gradual increase and gradual decrease in encryption and decryption time. This also affect in memory consumption which has hampered in bar graph, memory graph and as well as throughput analysis in the result.

5.2. Limitations

There are several notable limitations in this research:

- The overall image data analysis (bigger size of image data) could not be accomplished as expected such as differential, statistical and visual assessment analysis due to the limitation of computing resource. For this we required high powerful laptop which could not be performed as expected.
- Comparison between original image and encrypted image could be performed since encrypted image data could not be retrieved (read), so most of the image data analysis has been done by comparing original and retrieved images.

5.3. Future Recommendation

First and foremost future recommendation of this research is to overcome with my research limitation. And, after using text and image data, my future work will also include experiments on audio and video data. Also, my future work will be including XEX-based tweaked-codebook mode with cipher text stealing (XTS) mode of operation which is a block cipher mode of operation used for full disk encryption. Also, the other modules of algorithm can be added or modified to get variation in experiment result. And the experiments to be carried out in better simulators to get better result. In addition, this kind of research can be helpful in those areas such as network data transmission where hardware level of encryption is needed. This research is more focused on time and performance analysis, so in future, this can be upgraded with comparing strength of algorithms or computing security levels of encryption algorithms.

References

- [1] ScienceDirect topics, "File encryption - An overview," ScienceDirect topics, [Online]. Available: <https://www.sciencedirect.com/topics/computer-science/file-encryption>.
- [2] C. D. CANNIERE, "ANALYSIS AND DESIGN OF SYMMETRIC ENCRYPTION ALGORITHMS," 2007.
- [3] D. S. Abd Elminaam, H. M. Abdual Kader and M. M. Hadhoud, "Evaluating The Performance of Symmetric Encryption Algorithms," *International Journal of Network Security*, vol. 10, no. 3, p. 213–219, 2009.
- [4] J. Thakur and N. Kumar, "DES, AES and Blowfish: Symmetric Key Cryptography Algorithms Simulation Based Performance Analysis," *International Journal of Emerging Technology and Advanced Engineering*, 2011.
- [5] C. RIMAN and P. E. ABI-CHAR, "Comparative Analysis of Block Cipher-Based Encryption Algorithms: A Survey," *Information Security and Computer Fraud*, vol. 3, no. 1, pp. 1-7, 2015.
- [6] N. K. B and N. R. O, "ESTIMATION OF CRYPTOGRAPHIC APPROACH ON IoT DEVICES," *International Journal of Recent Scientific Research* , vol. 10, no. 7, pp. 33664-33669, 2019.
- [7] "What is File Encryption," [Online]. Available: http://www.file-encryption.net/file_encryption.php.
- [8] W. Stallings, *Cryptography and Network Security Principles and Practices*, Prentice Hall, 2005.

- [9] "Computer Network | Block cipher modes of operation," Computer Network, [Online]. Available: <https://www.geeksforgeeks.org/computer-network-block-cipher-modes-of-operation/>.
- [10] S. A. Dass and J. Prabhu, "Comparative Analysis of a Systematic Coherent Encryption Scheme for Large-Scale Data Management Using Cryptographic Encryption Technique," *Smart Intelligent Computing and Applications*, 2019.
- [11] M. Dworkin, "Recommendation for Block Cipher Methods for Format-Preserving Encryption," *Draft NIST Special Publication 800-38G*, 2019.
- [12] C. Tan, X. Deng and L. Zhang, "Identification of Block Ciphers under CBC Mode," *8th International Congress of Information and Communication Technology (ICICT-2018)*, 2018.
- [13] W. Stallings, "The offset codebook (OCB) block cipher mode of operation for authenticated encryption," *Cryptologia*, 2018.
- [14] A. Khodjanov, F. Rustamov and J. Yun, "Efficient block cipher mode for NVM," *Journal of Physics: Conference Series*, 2018.
- [15] M. A. MUSLIM, B. PRASETIYO and ALAMSYAH, "IMPLEMENTATION TWOFISH ALGORITHM FOR DATA SECURITY IN A COMMUNICATION NETWORK USING LIBRARY CHILKAT ENCRYPTION ACTIVEX," *Journal of Theoretical and Applied Information Technology*, vol. 84, no. 3, 2016.
- [16] Y.-L. Huang, F.-Y. Leu, J.-C. Liu and J.-H. Yang, "A Block Cipher Mode of Operation with Two Keys".
- [17] C.-W. Huang, C.-L. Yen, C.-H. Chian, K.-H. Chang and C.-J. Chang, "The Five Modes AES Applications in Sounds and Images," in *Sixth International Conference on Assurance and Security*, Taiwan, 2010.

- [18] P. Gehlot, S. R. Biradar and B. P. Singh, "Implementation of Modified Twofish Algorithm using 128 and 192-bit keys on VHDL," *International Journal of Computer Applications* (0975 – 8887), vol. 70, no. 13, 2013.
- [19] P. Singh and P. K. Singh, "IMAGE ENCRYPTION AND DECRYPTION USING BLOWFISH ALGORITHM IN MATLAB," *International Journal of Scientific & Engineering Research*, vol. 4, no. 7, 2013.
- [20] N. Singhal and J. Raina, "Comparative Analysis of AES and RC4 Algorithms for Better Utilization," *International Journal of Computer Trends and Technology*, 2011.
- [21] P. N. Penchalaiah and D. R. Seshadri, "Effective Comparison and Evaluation of DES and Rijndael Algorithm (AES)," *International Journal on Computer Science and Engineering*, vol. Vol. 02, no. 05, 2010.
- [22] M. Cakiroglu, "Software implementation and performance comparison of popular block ciphers on 8-bit low-cost microcontroller," *International Journal of the Physical Sciences*, vol. 5, no. 9, pp. 1338-1343, 2010.
- [23] "Free High-Resolution Satellite Images Samples," Effigis, [Online]. Available: <https://www.effigis.com/en/solutions/satellite-images/satellite-image-samples/>.
- [24] "NASA Visible Earth: June, Blue Marble Next Generation w/," Topography and Bathymetry, [Online]. Available: <https://visibleearth.nasa.gov/view.php?id=73726>.
- [25] Sample Videos, "Download Sample Videos," Dummy Videos of demo use, [Online]. Available: <https://sample-videos.com>.
- [26] C. Adams, "RFC 2144 - The CAST-128 Encryption Algorithm," *Network Working Group*, May 1997.

- [27] A. K, J. Solomon, H. M and I. V, "A Study of Twofish Algorithm," *International Journal of Engineering Development and Research (www.ijedr.org)*, vol. 4, no. 2, 2016.
- [28] A. Singh, "FPGA Implementation and Analysis of DES and TWOFISH Encryption Algorithms," INDIA, 2010.
- [29] B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall and N. Ferguson, "Twofish: A 128-Bit Block Cipher," 15 June 1998.
- [30] X. Chen and C.-J. hu, "Adaptive medical image encryption algorithm based on multiple chaotic mapping," *Saudi Journal of Biological Sciences*, vol. 24, no. 8, Dec 2017.
- [31] A. S and N. B S, "Quality Assessment of Resultant Images after Processing," *Computer Engineering and Intelligent Systems*, vol. 3, no. 7, 2012.
- [32] "Intensity Histogram," [Online]. Available: <https://homepages.inf.ed.ac.uk/rbf/HIPR2/histogram.htm>.
- [33] D. Blazhevski, A. Bozhinovski, B. Stojchevska and V. Pachovski, "MODES OF OPERATION OF THE AES ALGORITHM," in *The 10th Conference for Informatics and Information Technology*, 2013.
- [34] K. Aggarwal, J. K. Saini and H. K. Verma, "Performance Evaluation of RC6, Blowfish, DES, IDEA, CAST-128 Block Ciphers," *International Journal of Computer Applications (0975 – 8887)*, vol. 68, no. 25, 2013.