Efficient Selection of Compression-Encryption Algorithms for Securing Data Based on Various Parameters

Dissertation

Submitted in partial fulfillment of the requirement for the degree of

Master of Technology in Computer Engineering

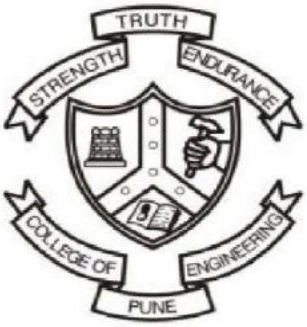
By

Ranjeet D. Masram

MIS No: 121222010

Under the guidance of

Prof. Dr. Jibi Abraham



Department of Computer Engineering and Information Technology

College of Engineering, Pune

Pune – 411005

June, 2014

DEPARTMENT OF COMPUTER ENGINEERING AND

INFORMATION TECHNOLOGY,

COLLEGE OF ENGINEERING, PUNE

CERTIFICATE

This is to certify that the dissertation titled

Efficient Selection of Compression-Encryption Algorithms for Securing Data Based on Various Parameters

has been successfully completed

By

Ranjeet D. Masram

MIS No: 121222010

and is approved for the partial fulfillment of the requirements for the degree of

Master of Technology, Computer Engineering

Dr. Jibi Abraham
Project Guide,
Department of Computer Engineering
and Information Technology,
College of Engineering, Pune,
Shivaji Nagar, Pune-411005.

Dr. J. V. Aghav
Head,
Department of Computer Engineering
and Information Technology,
College of Engineering, Pune,
Shivaji Nagar, Pune-411005.

June 2014

Acknowledgments

I express my sincere gratitude towards my guide Professor Dr. Jibi Abraham for her constant help, encouragement and inspiration throughout the project work also for providing me infrastructural facilities to work in. I would also like to thanks Mrs. Rajni Moona, for providing directions to make progress in this work. Without their invaluable guidance, this work would never have been a successful one. We are immensely grateful to C-DAC, Pune for their support and providing funding to the project. I also like to convey my sincere gratitude to Dr. J. V. Aghav (HOD), all faculty members and staff of Department of Computer Engineering and Information Technology, College of Engineering, Pune for all necessary cooperation in the accomplishment of dissertation. Last but not least, I would like to thank my family and friends, who have been a source of encouragement and inspiration throughout the duration of the project.

Ranjeet Devidas Masram

College of Engineering, Pune

Abstract

For faster communication and exchange of data, most of the information comes in the form of electronic data. Most of the computer applications related to health are not secure and these applications exchange lot of confidential health data having different file formats like HL7, DICOM images and other audio, image, textual and video data formats etc. These types of confidential data need to be transmitted securely and stored efficiently. Therefore, along with security, factors like implementation cost and performance of different cryptographic algorithms also needs to be considered for practical implementation.

Parameters like different data types, data density, data size, key sizes and block cipher modes of operation need to be analyzed for comparative performance analysis of various cryptographic ciphers. Data compression is a method of reducing the size of the data file so that the file should take less disk space for storage. Compression of a file depends upon encoding of file. In lossless data compression algorithm there is no data loss while compressing a file, therefore confidential data can be reproduce if it is compressed using lossless data compression. Compression reduces the redundancy and if a compressed file is encrypted it is having a better security and faster transfer rate across the network than encrypting and transferring uncompressed file. But in some cases, compression increases the overhead like size of file and processing time etc.

This work provides the comparative performance analysis of seven cryptographic algorithms (RC4, Blowfish, Skipjack, RC2, AES, DES, and Triple DES) for various parameters like different, data types, data density, data size, key sizes and block cipher modes. It proposes with a performance matrix for different levels of security based on the analysis. Compression reduces the size of the file and encryption provides security to data. Therefore this work also proposes a learning compression-encryption model for identifying the files that should be compressed before encrypting and the files that should be encrypted without compressing them. Learning model also identifies a suitable compression algorithm for compressing these files for increasing performance of the system for transferring and storing file efficiently.

Contents

Certificate i

Acknowledgement ii

Abstract iii

List of Figures vi

List of Tables viii

1. Introduction 1

    1.1 Concept of Cryptography 1

    1.2 Concept of Compression 4

    1.3 Motivation 5

    1.4 Problem Definition 6

    1.5 Object and Scope of Project 7

2 Literature Survey 8

    2.1 Overview of Cryptographic Algorithms 8

    2.2 Overview of Data Compression Algorithms 10

    2.3 Overview of File Formats 12

    2.4 Overview of Work done field of Cryptography and Compression 13

    2.5 JCA 16

3 Architectural Models and Design 18

3.1 Problem Analysis 18

3.2 Symmetric Key Cryptographic Model using JCA 19

3.3 Design of Model of Compression Algorithms for Data Type Analysis 23

3.4 Compression-Encryption Learning Model 25

4 Testing and Results 29

4.1 Experimental Setup 29

4.2 Test case Analysis and Observation for Symmetric Key Cryptography 30

4.3 Proposed Security Performance Matrix 41

4.4 Analysis of Data Types for Compression Algorithms 43

4.5 Compression-Encryption Learning Model 44

5 Conclusion and Future Scope 49

5.1 Conclusion 49

5.2 Future Scope 50

Appendix 51

Bibliography 52

List of Figures

1.1	Concept of Cryptography	1
1.2	Classification of Ciphers	2
1.3	Data Loss with Lossy Compression	4
1.4	Maintaining quality with Lossless Compression Algorithm	5
2.1	Difference between Lossy and Lossless Compression Algorithm	11
2.2	Compression Coding Techniques	11
3.1	Working of Symmetric Key Cryptography	19
3.2	Symmetric Key Cryptography architectural model	20
3.3	Symmetric Key Cryptography flowchart	22
3.4	Compression Analysis flowchart	24
4.1	Encryption time Vs Cipher Algorithm for files of different data type of size 50MB	31
4.2	Encryption time Vs Cipher Algorithm for files of different data type of size 100MB	32
4.3	File size Vs Encryption Time for AIFF file of different sizes	33
4.4	File size Vs Encryption Time for AVI file of different sizes	34
4.5	File size Vs Encryption Time for DICOM file of different sizes	34
4.6	Variation of key sizes for different cipher Algorithms	37

4.7	Encryption Rate variation for key variation of cipher Algorithms on 50.5 DICOM File	37
4.8	Block Mode Variation of AES 128 for 10MB files	40
4.9	Variation of cipher block modes on 50.5MB DICOM file	40
4.10	Encryption time for different modes and keys of AES	42
4.11	CompressionScore Database Schema	45
4.12	CompressionRank Database Schema	46

List of Tables

2.1	Details of selected cryptographic algorithms	10
3.1	CompressionScore Table	27
3.2	CompressionRank Table	27
4.1	Execution parameters for files of different size	33
4.2	Encryption rate for files of different size	35
4.3	Encryption rate for sparse and dense data file	36
4.4	Encryption rate for key variation of different cipher algorithms on 50.5MB DICOM file	38
4.5	Encryption rate for variation of cipher block modes on 50.5MB DICOM file	41
4.6	Encryption rate for different modes and keys of AES	43
4.7	Security performance matrix	43
4.8	Compression Ratio of Different file formats	44
4.9	Best Compression Algorithm for different data type	47

Chapter 1

Introduction

1.1 Concept of Cryptography

Today Cryptography is a powerful tool used to protect the information in computer systems. Cryptography actually means secret writing, even the ancient human desired to keep and store secrets [8] [9] [12]. In ancient days, cryptography was available only to generals and Emperors, but today it is nearly used by everyone, everyday. Every time when a credit card transaction is done, phone call is made, secure website is used; there is use of cryptography.

In cryptography original message is basically encoded in some non-readable format. This process is called encryption. The only person who knows how to decode the message can get the original information. This process is called decryption [8] [13]. The Figure 1 shows the concept of cryptography.

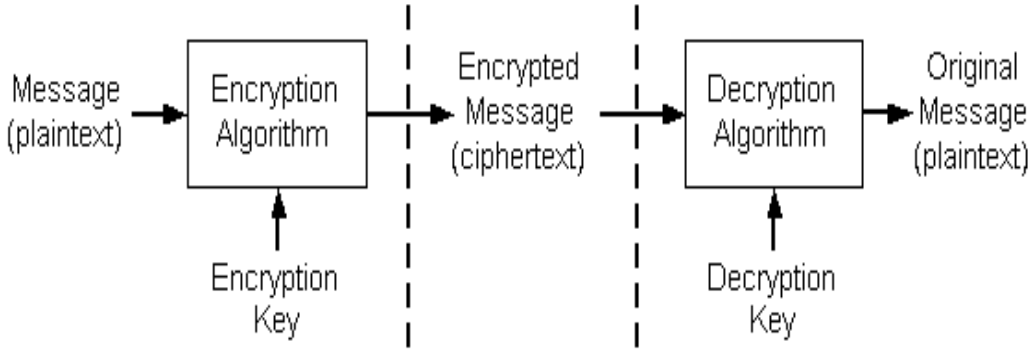


Fig. 1.1: Concept of Cryptography

There are main services provided by Symmetric Cryptography [8] [12] [14]. All these four services deal with storing or transmitting of data. These services are as follows:

- *Confidentiality:* keeping the data secret.
- *Integrity:* keeping the data unaltered.
- *Authentication:* to be certain where the data came from.

On the basis of key used, cipher algorithms are classified as [12]:

- Asymmetric key algorithms (Public-key cryptography), where two different keys are used for encryption and decryption.
- Symmetric key algorithms (Private-key cryptography), where the same key is used for encryption and decryption.

On the basis of input data, ciphers are classified as [12]:

- Block ciphers, which encrypt block of data of fixed size, and
- Stream ciphers, which encrypt continuous streams of data.

Classical ciphers used substitution and transposition for encryption and decryption [14]. The rotor machine is a device that is used to encrypt and decrypt secret messages. It is a stream cipher device and electro-mechanical in nature. Figure 1.2 shows the classification of ciphers.

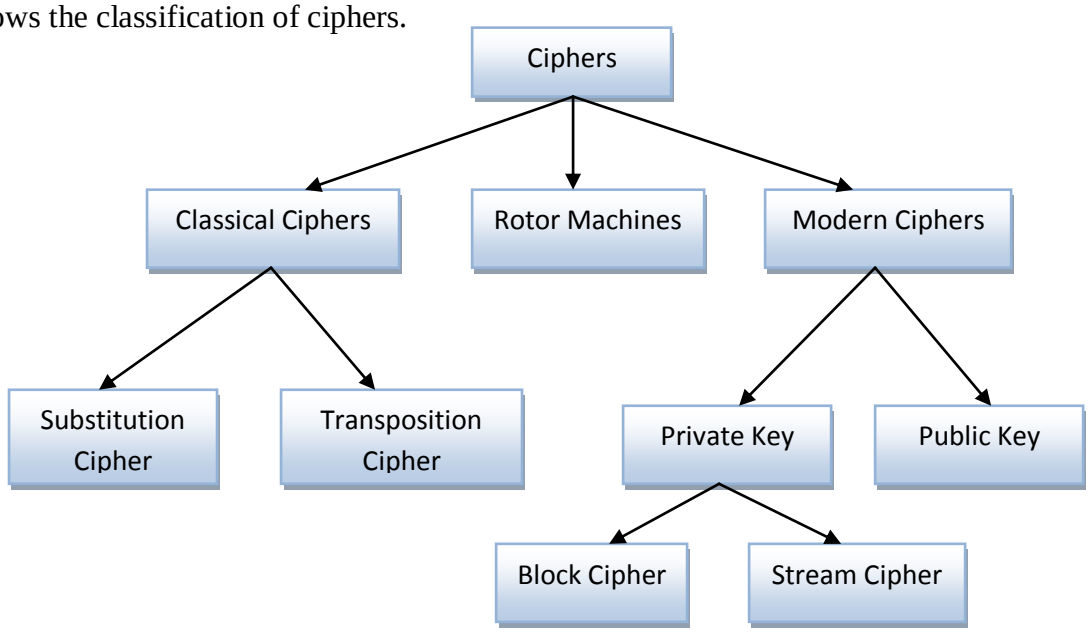


Fig. 1.2: Classification of Ciphers

Performance of any encryption algorithms depends upon the mainly two things that are security and time required for encryption. Following are some of the parameters that can have effect on encryption time of the ciphers algorithms:

- Data type
- Data size
- Data Density
- Key size of cipher algorithm
- Cipher block modes

Data types: There are various data types' files. Data type represents the encoding of the files. Common examples are as follows:

- Text: ANSI, UNICODE (16 & 32 bit little and Big Endian), UTF-8.
- Audio: WAV, AIFF, M4A, MP3, WMA, MP4.
- Video: AVI, MOV, MP4, MPEG, WMV, GIF.
- Images: JPEG, TIFF, GIF, BMP, PNG.
- Others: Medical Informatics Standard i.e. DICOM (Images/Binary + Text), HL7 etc.

Data Size: Data size is the space occupied by a file on a disk. Audio and video files take more space on disk than textual files as they contain multimedia information.

Data Density: Density of data represents the amount of different information present in the data. The more the information, the dense is the data and lesser the information, sparse is the data. For example if there are two files, X and Y, both contain 2000 words and having size 50kb and 200kb respectively, then file X is denser. The more the information, the dense is the data and lesser the information, sparse is the data. Sparse file is a file that contains most of the empty spaces and attempts to use the computer space more effectively.

Key size: Key size is the size of the key measured in bits and will depend on algorithm. For example AES is having key sizes 128, 192 and 256 bits.

Cipher Block Modes: In cryptography, block cipher mode for a block cipher algorithm indicates how ciphertext blocks are encrypted from plaintext blocks and vice versa. Commonly used block cipher modes are ECB, CBC, CFB, OFB, PCBC and CTR etc.

1.2 Concept of Compression

Data compression is a method of reducing the size of the data file so that the file should take less disk space for storage [11]. The file that contains redundancy gets reduced by compression. Data compression algorithms can be classified as:

a. Lossy compression algorithms
b. Lossless data compression algorithms

Lossy compression algorithms

In lossy data compression algorithms there is loss of original data while performing compression [20]. In computer science and Information technology, a data encoding method in which the data is compressed by losing some amount of data is lossy compression. The Figure 1.3 shows data loss that occurs when Lossy Compression algorithm is used.

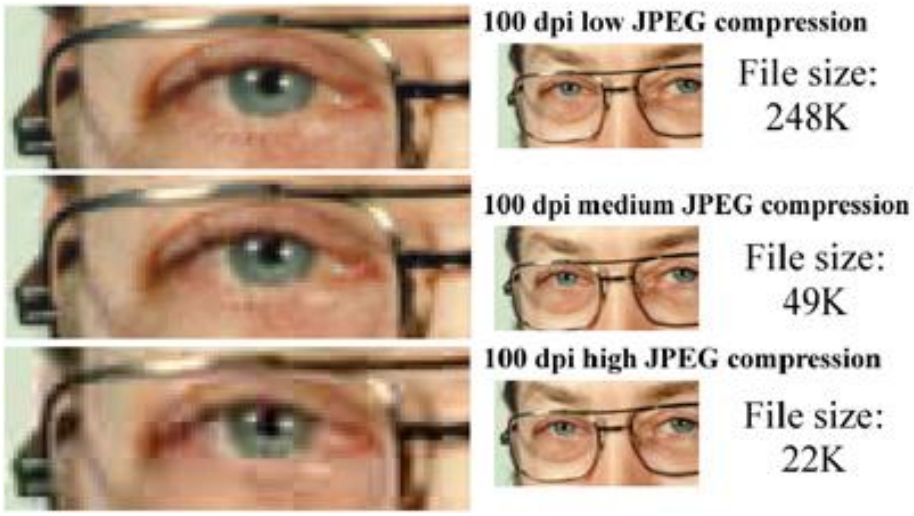


Fig. 1.3: Data Loss with Lossy Compression

The different photos in Figure 1.3 demonstrate that with the increases in the compression the original image coarsens progressively, as the data of the original image is discarded.

Lossless data compression algorithms

In case of lossless data compression algorithm there is no data loss while compressing a file, it guarantees to reproduce the exactly same data as input. If data loss is not desirable the lossless data compression algorithms should be used. Some of the

peculiar examples include executable text documents, programs and source codes etc. Some of the image file formats also uses lossless compression. Peculiar examples include PNG or GIF image files [21]. Figure 1.4 shows that the quality of the file is maintained when lossless compression algorithm is used.

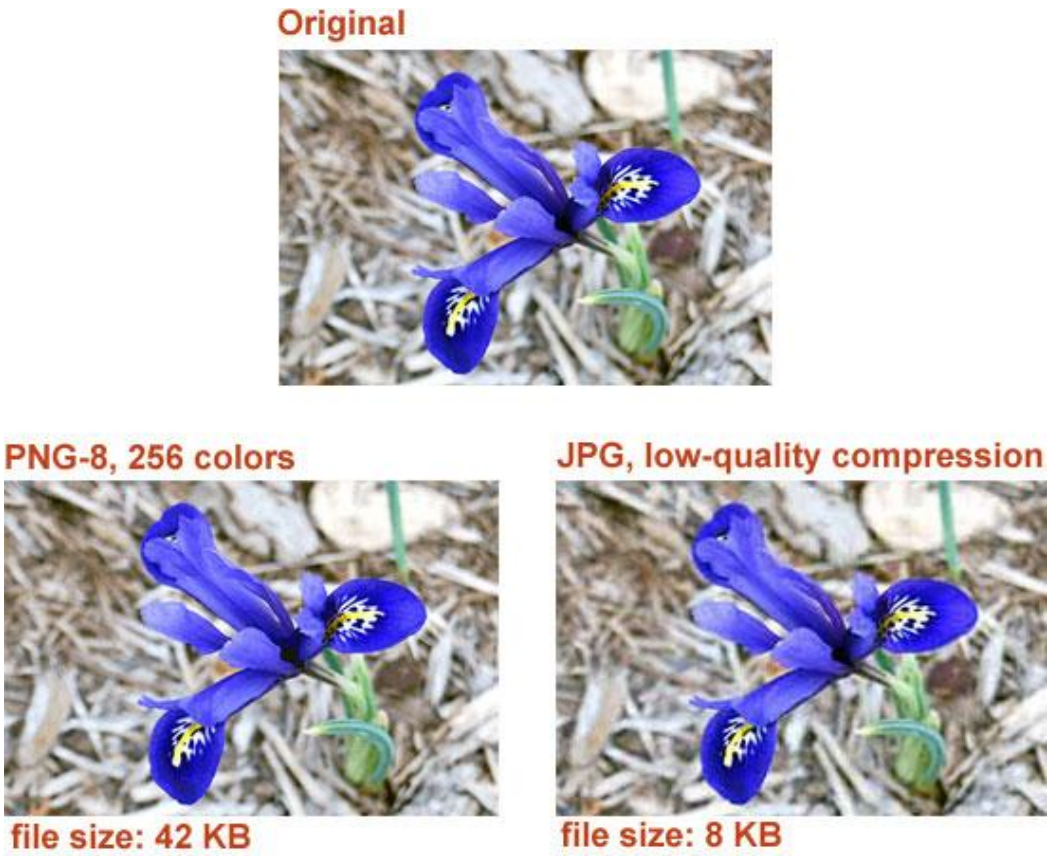


Fig. 1.4: Maintaining quality with Lossless Compression Algorithm

1.3 Motivation

To ensure the security of electronic data while transferring through networks, cryptographic techniques are used. Like every process, encryption and decryption processes involve use of CPU recourses like CPU cycle memory. These processes require good amount of time for I/O, encryption and decryption operation. Hence these cryptographic algorithms consume a good amount of resources for encrypting and decrypting the data. So it is essential for an encryption algorithm to have good performance along with the security. To prevent data loss during transmission and to promote faster transmission, many different compression algorithms are used to reduce the size of the data during transmission. Usually lossless compression algorithms are used if data that is being transferred is important and if data loss is not

affordable. If a compressed file is encrypted, it has better security and faster transfer rate across the network than encrypting and transferring uncompressed file. But in some cases, compression increases the overhead like size of file and processing time etc. Hence there is a need to analyze different symmetric key cryptographic algorithms for various parameters so as to understand the factors that can affect the performance of the cryptographic algorithms. Also identify whether the file that has to be compressed before encrypting or not. If compression is needed then identify the best suitable compression algorithm that should be used for compressing the file according to data type and data size to reduce the overhead of time for compression and increase the efficiency and security to data that is being transferred.

1.4 Problem Definition

For achieving faster communication most of confidential data is circulated through networks as electronic data. Many different computer applications in different domains exchange a lot of confidential data. The field of health care and electronic health care records is one such example. Some of the computer applications related to field of health care are not secure. Health related data has different data formats like HL7, DICOM, audio, image, textual and video data formats etc. Most of this data is confidential and needs to be transmitted securely. Cryptographic ciphers play an important role in securing this confidential data against unauthorized attacks. Therefore, along with security, factors like implementation cost and performance of different cryptographic algorithms also need to be considered for practical implementation. Parameters like data type, data size, data density, key size, key strength, encryption time, decryption time affects the selection of encryption algorithm and therefore we need some benchmark for selection of cryptographic algorithm. If the file size is small, encryption and transfer of such files can improve the performance of the system to some extent. For this, a file may be first compressed and then encrypted. Data compression is a method of reducing the size of the data file. Compression of a file depends upon encoding of file. Using lossless data compression algorithms, confidential data can be reproduced as it was before compression.

1.5 Objective and Scope of Project

In a practical scenario, along with security, performance and cost are important factors for implementation of cryptographic algorithms. The main focus of our work is on comparing the encryption algorithms on the basis of their performance.

1. To analyze performance of various cryptographic algorithms on following parameters
    - Data type
    - Data size
    - Data density
    - Key size
    - Cipher block modes
    - Encryption Time
2. To bring out a performance matrix for different levels of security based on the analysis.
3. To develop learning model for compressing files before encrypting them.

Following are the cryptographic and compression algorithms that have been considered for comparative performance evaluation:

Cryptographic Algorithm: DES, 3-DES, AES, Blowfish, Skipjack, Rivest Ciphers (RC2 and RC4).

Compression Algorithms: LZ4, LZF, Huffman, Deflate, ZIP, Gzip etc.

Chapter 2

Literature Survey

The proposed system requires selecting a suitable compression algorithm for compressing the files followed by selecting the right encrypting algorithm based on the various parameters. So this chapter will give an overview of symmetric key cryptographic algorithms, compression Algorithms and the prior work and literature survey in the field cryptography and compression Algorithms.

2.1 Overview of Cryptographic Algorithms

Symmetric key cryptographic ciphers come in two types, stream and block ciphers. Stream ciphers works on bits stream or bytes stream. Stream ciphers are used for securing data of terminal and wireless applications. Block ciphers performs encryption or decryption on fixed size block of data. The plaintext is not always in multiple of block size, therefore padding bits are needed to compensate partially filled block. The padding scheme defines how the plaintext is filled with data for last block. In network applications block ciphers are used for transmission of files of huge sizes which require high security. Deciphering ciphertext without knowing the key is called cryptanalysis. Cryptanalysis of block ciphers is difficult compared to stream ciphers [9]. Hence in most of the applications, block ciphers are used for providing better security than stream ciphers.

Structure of the cipher algorithms describes the construction of blocks for the ciphers. Mathematically linked series of operations are used in Substitution-permutation network in cryptography to construct the block of the Symmetric key cryptographic

block cipher. In SP network key and plain text are taken as an input and number of alternating rounds of S-Box substitution and permutation are applied to get a single ciphertext block. The reverse process is done for decryption of the blocks. Symmetric key cryptographic ciphers have different structures that are used to construct the block of the different Symmetric key block ciphers. There are symmetric key structures like Feistel network, Substitution-permutation network etc. In the case of Fiestel network, the encryption and decryption process of the block are almost similar to each other, except it requires the reversal of key schedule. Iteration is a characteristic feature of Fiestel network cipher as an internal function knows as round function.

Block ciphers come in various block modes. Block mode for cipher algorithm determines how ciphertext blocks are created by encryption from plaintext blocks and vice versa. ECB, CBC, CFB, OFB, PCBC and CTR etc are commonly used block modes [9]. ECB has poor security properties since encryption of a block with a fixed size always yields the same result; hence susceptible to dictionary attacks, replay attacks etc. All other modes require an initialization vector while encrypting the first block and are considered to be more secure. If we are not using random initialization vector, then CBC is also susceptible to dictionary attacks. Parallelization in encryption cannot be achieved with CBC mode. PCBC mode is similar to CBC mode with a little variation. In case of CBC first plaintext block is XORed with IV and remaining all plaintext blocks are XORed with previous ciphertext blocks; while in case of PCBC operation on first plaintext block is similar to CBC but remaining all plaintext blocks are XORed with previous plaintext as well as previous ciphertext block [9] [12]. CTR acts as stream cipher and does not require additional padding bits. CTR provides better security with 128 bit block, but its security is inadequate if 64 bit block is used without random nonce (initialization vector). Parallelization in encryption can be achieved with CTR mode. CFB and OFB are also streaming modes like CTR. In OFB mode, last block of key stream is continually encrypted to produce key stream; while in CFB mode, last block of ciphertext is always encrypted to produce key stream to encrypt next plaintext block. Streaming modes are more prone to bit-flipping attacks [13].

There are different symmetric cryptographic algorithms in the literature [8] [9]. Out of them, the algorithms listed in the Table 2.1 are selected for detailed study on health

care data in particular. Except RC4, all the selected algorithms are of block type, since most of the health related data are to be provided with high security.

Table 2.1: Details of selected cryptographic algorithms

Algorithm Name	**Structure**	**Rounds**	**Key Size (In bits)**	**Block Size (In bytes)**	**Block Modes**
AES	Substitution-permutation network	10, 12, 14	128, 192, 256	16	ECB, CBC, CFB, OFB, CTR, PCBC.
DES	Balanced Feistel network	16	56	8	
Triple DES	Feistel network	48	112, 168	8	
RC2	Source-heavy Feistel network	18	40 to 1024	8	
Blowfish	Feistel network	16	32 to 448	8	
Skipjack	Unbalanced Feistel network	32	80	8	
RC4	----	256	40 to 2048	----	Stream Cipher

2.2 Overview of Data Compression Algorithms

The main use of compression algorithm is to reduce the size of data [11]. But in some cases, data loss is not desirable. Though lossy compression reduces the size to a greater extent, it also incurs data loss. The loss of the data is proportional to the compression of the file. The Figure 2.1 shows the difference between lossless and lossy compression.

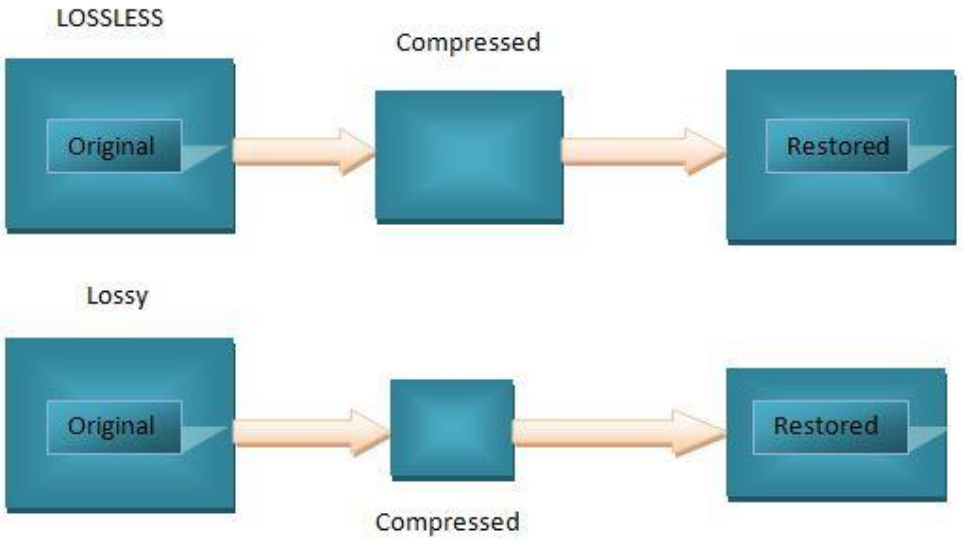


Fig. 2.1: Difference between Lossy and Lossless Compression Algorithms

From the Figure 2.1, it can be understood that if data loss is not desirable lossless compression algorithms should be chosen for data compression. The Figure 2.2 shows the various coding techniques used for compression of data file.

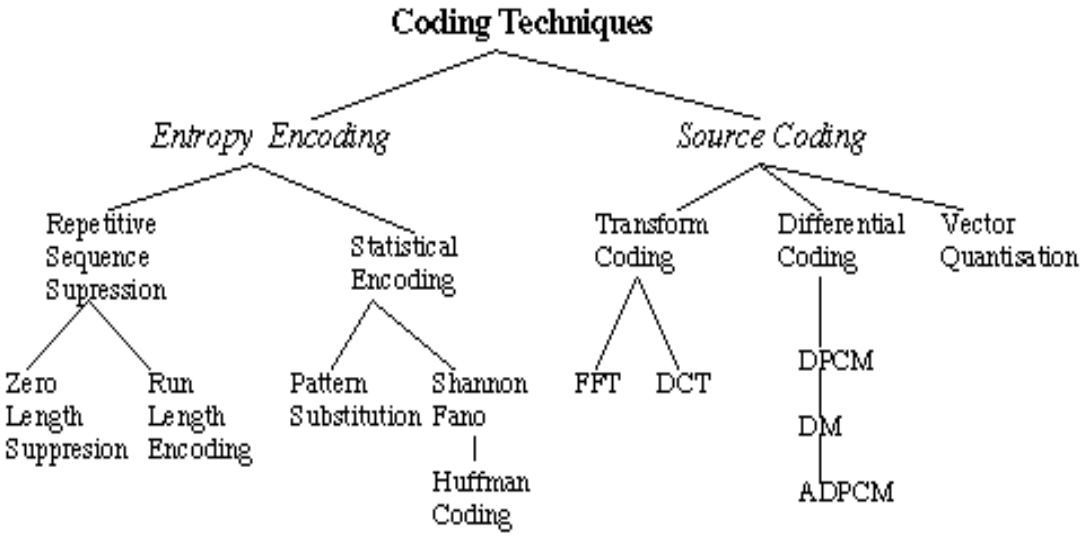


Fig. 2.2: Compression Coding Techniques

Entropy is typically used for determination of uncertainty that is associated for any given random variable. Similarly in compression it is used to determine the probability of getting the redundancy in a file. Source Coding is used in case of signal processing.

Lossless compression is used in many applications. Lossless compression can be broadly classified into two type's entropy encoding and dictionary based encoding. Following are some of the compression algorithms that uses dictionary based encoding.

- Huffman Code-It assign more bits to symbols appear less and fewer bits to the symbols that occur more frequently. Every Huffman code having the same average code length. It optimizes the single byte at time.
- Deflate- It combines the LZ77 and Huffman code for compression in which LZ77 optimizes sequence of bytes whereas Huffman works on single byte.
- LZ4-It lossless data compression algorithm that is primarily focused on compression and decompression speed. The algorithm gives a slightly worse compression ratio than others.
- LZF- It is a fast compression algorithm that takes very little working memory and code space.
- Zip- Zip is a lossless compression Algorithm. Zip compress as well as archive the file. Content of the .Zip file can be a single file of group of files enclosed in a folder.

Compression factor or compression ratio is a good metric for determining the amount of compression done on a file.

Compression ratio in terms of disk utilization is given as

$$Compression\ ratio\ in\ term\ of\ disk\ utilization = \left(1 - \frac{compressed\ file\ Size}{uncompressed\ file\ Size}\right) * 100 \quad (1)$$

2.3 Overview of File Formats

Textual data formats are ANSII and UNICODE (16 and 32 bit little and big Endian and UTF-8). ANSII is encoding scheme for 128 characters basically made for English alphabets. It contains numbers 0-9, character alphabets A-Z and a-z, and some special characters. In Unicode standard, unique numbers are provided for every character independent of platform. Image file formats may be TIFF, GIF,

BMP and PNG, JPEG [11]. JPEG is image file format for digital images that uses lossy compression method. TIFF and BMP are image file format that are used to store images of raster graphics. GIF image is similar to image format of bitmap images. GIF uses LZW (Lempel-Ziv-Welch) technique of compression and for each image it can support up to 8 bits/pixel. PNG is alternative to GIF image file format and allows more compression than GIF [11]. Audio file formats are WAV, AIFF, M4A, MP3 and WMA. WAV and AIFF are usually uncompressed audio file format. M4A (audio) uses Apple Lossless compression format but often it is compressed with Advance audio coding (lossy). MP3 and WMA are lossy compression audio formats. Video file formats are AVI, M4V, MPEG and WMV etc. AVI format contains the data (audio as well as video data) file container; which allows audio with video playback synchronously. M4V and MP4 are very similar format, but M4v can be protected by Digital Rights Management copy protection. MPEG contains compressed audio and visual digital data. WMV is compressed video format developed by Microsoft. DICOM is basically medical imaging standard for storing, transmitting, printing, and handling information. DICOM is implemented in almost every medical imaging device like radiotherapy device, cardiology imaging and radiology (ultrasound, X-ray, MRI, CT etc.). DICOM files are generally having varying data densities. Density of data represents the amount of different information present in the data file. A file is said to be dense, if file size is less and the content is more.

2.4 Overview of Work done in field of Cryptography and Compression

A lot of analysis has been done on the symmetric cryptographic algorithm to analyze the performance of different cryptography algorithms. This section covers the overview of some cryptographic algorithm analysis.

In paper [6], the author compared four symmetric key cryptographic algorithms AES, DES, 3-DES and Blowfish for performance evaluation on two different machine to check which of the above algorithm performs better as compared to other. The author has performed four simulations to test the performance. The first simulation was taken on Pentium II, 266 MHz machine in ECB mode. The second simulation was taken on Pentium 4, 2.4 GHz machine in ECB mode. The third simulation was taken on Pentium II, 266 MHz machine in CFB mode. And the final simulation was taken on

Pentium 4, 2.4 GHz machine in ECB mode. The simulation was done on relatively small file size. It was observed that for all the four simulations Blowfish algorithm took relatively less time than other symmetric key cryptographic algorithms for encryption. It was also concluded that performance of AES is better than DES and 3-DES.

In paper [15], the author has simulated different symmetric key cryptographic algorithms like AES, DES, 3-DES and Blowfish. The simulation was done on 0.5 to 20MB data blocks. The simulation results show that the Blowfish yields better results than other symmetric key cryptographic algorithms when it comes to processing power. AES yield poor results as it requires high processing power. Initially all the simulations were taken in ECB mode and it was observed that Blowfish takes comparatively less processing time than others. AES takes relatively higher time when the block size is high. It was also concluded that 3-DES will always take more time as compared to 3-DES as it involves 3 phases of encryption. Another simulation was done on all the symmetric key algorithms in CBC mode. It was concluded that CBC took more time for performing encryption than ECB mode.

In paper [2], the author compared AES and DES algorithms on image file format. The performance metric chosen by the author was encryption time. MATLAB software platform was used for implementation of these two symmetric key cryptographic cipher algorithms. AES uses three keys of 128, 192 and 256 bits. AES took less encryption and decryption time than DES for all the three keys used for evaluation.

In paper [3], the author compared two symmetric key cryptographic algorithm AES, which is a block cipher and RC4, a stream cipher to check which algorithm yield good results. To determine the performance different performance metrics where chosen. Performance metric like throughput to determine the speed of encryption, CPU process time to calculate the amount of time taken by CPU to process the file, memory utilization to keep the check on usage of memory and key size where chosen. From the experiment it was concluded that AES is not energy efficient for performing encryption and decryption on a file as RC4. Encryption time of AES and RC4 was also compared for different data packets. Encryption time of RC4 was found to be better for different size data packets. Key size of AES was varied from 128,192 to 256. Similarly key size of RC4 was also varied. The time for encryption increase for

AES with increase in key size. The RC4 yield nearly same results for different key size. In terms of key size again RC4 was better than AES. Based on the research RC4 was found to be better than AES.

In paper [7], the author compared the three symmetric key cryptographic algorithms Blowfish, DES and 3-DES. The author used Java and JCA for implementation of these cryptographic algorithms. The objective was to evaluate the three cryptographic algorithms for CPU execution time. Two different platforms like SunOS and Linux were chosen for performance evaluation. The CPU execution time was evaluated for generating the secret key on 10 MB file along with the encryption and decryption. It was concluded that for all evaluation blowfish was fastest followed by DES and 3-DES.

In paper [4], the author has compared the three symmetric key cryptographic algorithms Blowfish, DES and AES for different block cipher modes, like ECB, OFB and CBC. The processor used for the experiment was AMD Sempron having 2GB RAM. The author used java programming platform. All the programs were simulated using jdk1.7 on file of data block of nearly 200KB. All the results were taken couple of times for analysis. The first phase results were taken for all above mentioned algorithms using ECB Mode. The second phase results were taken for all above mentioned algorithms using CBC Mode. The third phase results were taken for all above mentioned algorithms using OFB Mode. It was found that for all the three phases Blowfish algorithm was giving best execution time for encrypting a file among all the algorithms. Performance of AES was poor and it required much processing power than the other two algorithms. The difference of execution time for all the modes was almost negligible. Some extra processing time was required in case of CBC mode. It was proposed to use CBC mode for more security. OFB mode gave better performance followed by ECB and CBC modes respectively.

In paper [1], the author has compared Blowfish, Two fish, IB_mRSA, RSA, RC2 for performance evaluation. Pentium Core2Duo processor was used by the author having 2GB RAM and CPU speed of 2.20 GHz. Windows Operating systems was used for taking all the simulation results. All the results were taken on the files having file size ranging from 10KB to 70KB. The metrics chosen for performance evaluation were throughput and encryption and decryption time. C#.NET software platform was used

for implementing cryptographic algorithms. Microsoft Excel was used for making graphs of simulation results. From all the simulation results it was concluded that, encryption time of IB_mRSA was better than rest of the selected cryptographic algorithms. Blowfish was second best performing algorithm. It was concluded that except Blowfish, IB_mRSA has an advantage over all the remaining algorithms for processing time and throughput. It was also concluded that RSA algorithm has least performance.

In paper [20], the author compared different lossless data compression algorithms such as Arithmetic encoding, Huffman coding, Run Length encoding algorithms. The performance measure used was compression ratio. It was concluded that the compression ratio of Huffman was poor compared to Arithmetic encoding, but the speed of compression and decompression required for Huffman was better than arithmetic encoding. It was also concluded that Huffman requires less memory for performing compression.

In paper [18], the author compared different lossless data compression algorithms for text files. In this paper the author compared Run Length encoding, Adaptive Huffman Algorithm, LZW algorithm, Shannon Fano Algorithm and Huffman Encoding etc. The comparison measures where compression, decompression time and compression ratio. It was observed that the Shannon Fano was the most effective compression algorithm for text files.

In paper [19], the author compared different lossless data compression algorithms for text files. The compression algorithms selected for analysis were Shannon Fano, Huffman and LZW. The comparison measures where compression ratio, compression and decompression time. It was concluded that LZW takes more time for compression than the other two algorithms but relatively less time for decompression. LZW was found better at decompression than other two algorithms. Compression of file was more for LZW followed by Huffman and Shannon Fano.

2.5 JCA

The java platform (openjdk1.6.0_14) is used for implementation. JCA and JCE are used for cipher algorithm implementation [12] [13]. JCA platform contains “provider” based architecture. This architecture of JCA contains some set of APIs for performing

encryption (symmetric ciphers, asymmetric ciphers, block ciphers, stream ciphers), secure random number generation, message digests (hash), certificates and certificate validation, key generation and digital signatures. The design of JCA architecture is based on following principles.

- Independence of Implementation.

Security algorithms are not needed to be implemented for applications. Java platform itself provides security services for it. These services are implemented by the provider interface. There can be multiple providers for providing security functionality.

- Interoperability of Implementation.

Across application the providers are interoperable. Specifically, no provider is bound to any of the specific application and vice versa.

- Extensibility of algorithm

Built-in providers are in large number in java platform which are used for implementation of security services. However, there can be some application that can rely on proprietary services. Such services can be implemented into java platform

Chapter 3

Architectural Models and Design

3.1 Problem Analysis

Computationally cryptographic algorithms are very much intensive. Substantial resources are consumed by cryptographic algorithm for encrypting and decrypting the data. While processing, cryptographic algorithm takes CPU memory, performs I/O operations on file and requires considerable amount of processing time. During the designing of any cryptographic algorithm the primary focus is on the security against the various types of attacks. The reliability of any cryptographic algorithm depends upon the security provided by the cryptographic algorithm for any type of attack and also on the time required for cracking the security of cryptographic algorithm by different types of attacks. But along with security there are other factors that are equally important as security against the attacks. The factors like implementation cost and the performance of cryptographic algorithm play vital role while practically implementing any algorithm for security. Factors like data type, data size, data density of a file, key size variation and block cipher modes variation need to be benchmarked to determine the performance of algorithms in a particular environment.

For transferring a file from one system to another considerable amount of time is required. Smaller files require less time for transferring across the network. With the increase in the size of file, the time to transfer also increases. Compression reduces the size of data file. In lossless data compression, complete data can be recovered without any loss. Therefore compressing data with lossless compression algorithm is a good choice before transferring file. But it is sometime observed that for some files, the file size increases during compression by lossless compression. These are the files

that cannot be compressed. It is also observed that different lossless compression algorithms have different compression ratios for different types of files. So it is important to find out which file to compress by a particular compression algorithm.

Symmetric key cryptography is generally used to encrypt and send a large size data. It is observed that small size file get transfer quickly and securely across network. As the file size increases the possibility of loss of data and security issues also increases. So there is a need to analyze, the type of data that should be compressed before encryption and the type of data that should be send without being compressed. Therefore encryption algorithms need to be analyzed for performance for various parameters and compression algorithm need to be analyzed according to the type of file. The compression-encryption model can considerably increase the performance of the system.

3.2 Symmetric Key Cryptography Model using JCA

Processing time of symmetric key cryptographic algorithms is far less than that of asymmetric key cryptographic algorithms. Therefore symmetric key cryptography is generally used to encrypt the data having large sizes. In symmetric cryptography, there is a single key (called secret key or private key) that is used to encrypt as well as decrypt the data. The parties that need to communicate with each other must have same secret key. The Figure 3.1 shows the working of symmetric key cryptography.

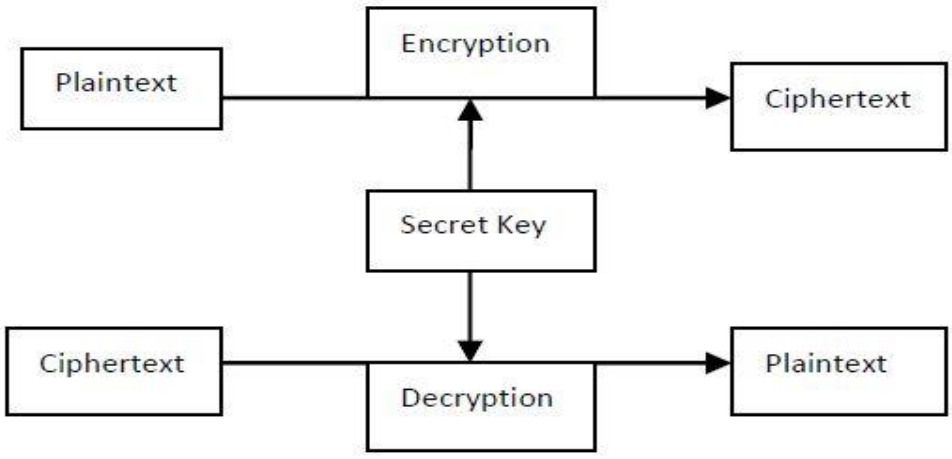


Fig. 3.1: Working of Symmetric Key Cryptography

According to the working of Symmetric key cryptography a model is designed for encrypting the file for performance analysis using JCA. Figure 3.2 shows the Symmetric key Cryptography architectural model used for performance analysis for different mentioned parameters.

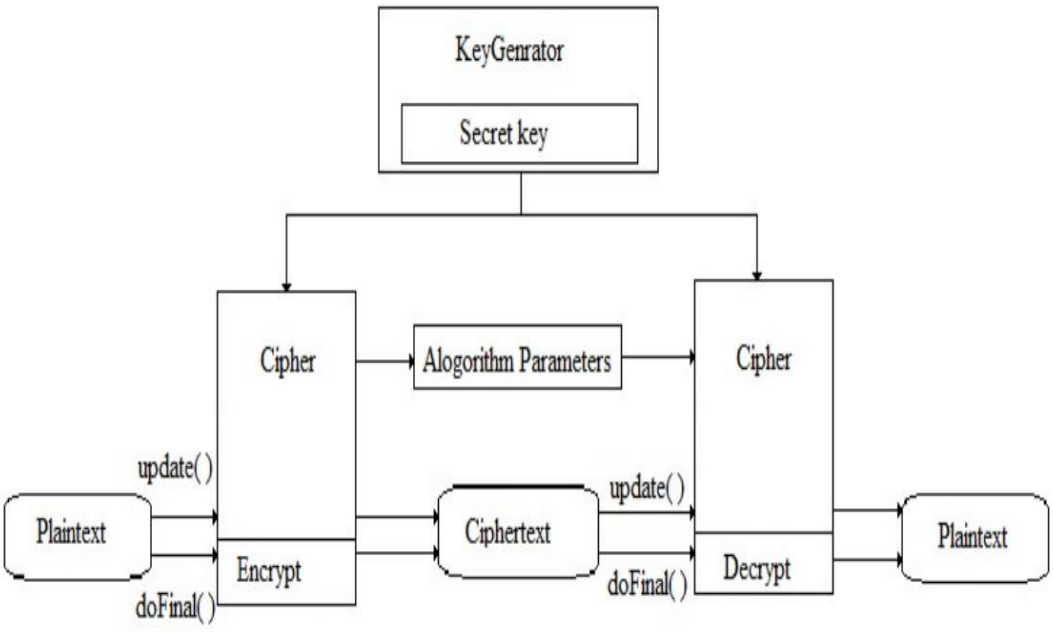


Fig. 3.2: Symmetric key Cryptography architectural model.

The Cryptography architecture of Java, JCA contains a class named "Cipher". This class contains number of functionalities of cryptographic ciphers that can be used to encrypt and decrypt any data file. KeyGenerator class is used for generating secret key for encrypting and decrypting the data file. For generating key of any symmetric key ciphers, it is important to have knowledge of Cryptography algorithms and key size supported by them. The algorithm for generating the secret key is as follows:

Procedure Name: ***GenerateTheKey***

Input Parameters: *AlgoName: Name of Cipher Algorithm*

KeyBits: Size of the key in bits,

Path: path to store the key

Start Procedure

Obtain security provider for cipher Algorithm

Use keyGenerator function provided by security provider

Create file for storing key

Key = keyGenerator (AlgoName, KeyBit)

Path = Write secret key to the file

End Procedure

The key generated by this procedure can be used for analysis of different parameter. The algorithm for analysis of different parameters is as follows:

Procedure Name: ***Crypt***

Input Parameters: *inputFile: Name of input file for encrypting/decrypting,*
outputFile: Name of outputFile,
AlgoName: Name of Cipher Algorithm,
Mode: specific cipher mode,
BlockSize: block size of cipher,
SecretKey: path of secret key file,
Operation: Encryption/Decryption

Start Procedure

Obtain security provider for cipher Algorithm;

Get the instance of the cipher Algorithm;

If (NOT ECB)

Initialize IV parameter;

Read the input file;

Initialize cipher Algorithm with parameters;

Check the operation to be performed on a file;

Read secret key;

Start the timer for calculating encryption/Decryption time;

Encrypt/Decrypt the file;

Stop the timer;

Write the outputFile to specified path;

End Procedure

Crypt procedure is used for both decryption and encryption of a data file. In the procedure "Crypt" the cipher algorithm parameters like algorithm name, input file, block mode, secret key are passed. A specific security provider is used for a particular cipher algorithm, for example "sun" provider for AES. The flow chart Figure 3.3 explains the complete analysis procedure.

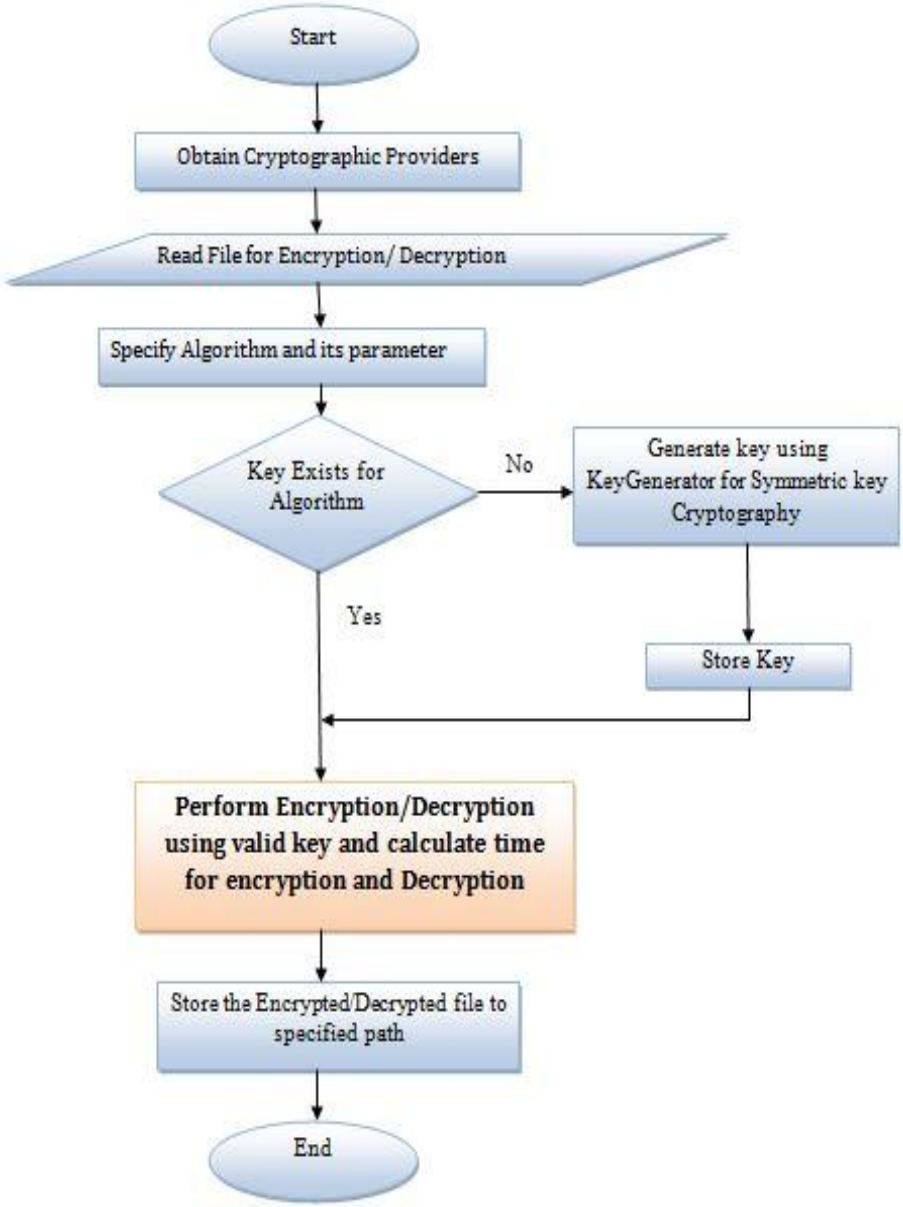


Fig. 3.3: Symmetric key cryptography flowchart

The flow chart gives the brief idea about the execution of the program flow control. Initially obtain providers *e.g.* sun, bouncy castle, cryptix etc. Read the input file for encryption or decryption. The program checks that the key is present for algorithm or not. If the key is not present, GenerateTheKey procedure is used to generate the key

for performing the cipher operation. The generated key is stored to the specified path. If the key is already present at the specified path, that key is used. A timer is started before encrypting or decrypting a file. After encryption or decryption is performed, the timer is stopped. The output file is then written to specified location and the results are written to result sheet. During any of the cipher operation the I/O time is not taken. Due to this we get pure time for evaluating the performance of a particular cipher algorithm.

3.3 Design of Model of Compression Algorithm for Data Type Analysis

Every compression algorithm has different level of compression for different files. For example, there are two compression algorithms A and B. Suppose compression algorithm A has compression ratio of 30% for X file. It is not necessary that compression algorithm B will have same compression Ratio for X file. Compression Algorithm B can give more, less or no compression at all. This sub section provides information on calculating the compression ratio for different compression algorithms. The following algorithm describes the process for calculating compression ratio for a file.

```
    Procedure Name:      Compress
    Input Parameters:    inputFile: Name of input file for compression

Start Procedure
    Array = Read all the compression algorithm in Array
    Read input file for compression
    Determine the data type from file extension
    Find uncompressed size of file
    for i = 1 to ArrayLength
            Compress file by Compression Algo[i]
            Calculate compressed file size;
            Calculate compression Ratio by eq. 1
    End for
    Write the compression ratio to database
End Procedure
```

Figure 3.4 shows the flow chart for analyzing the compression ratio of different compression algorithms.

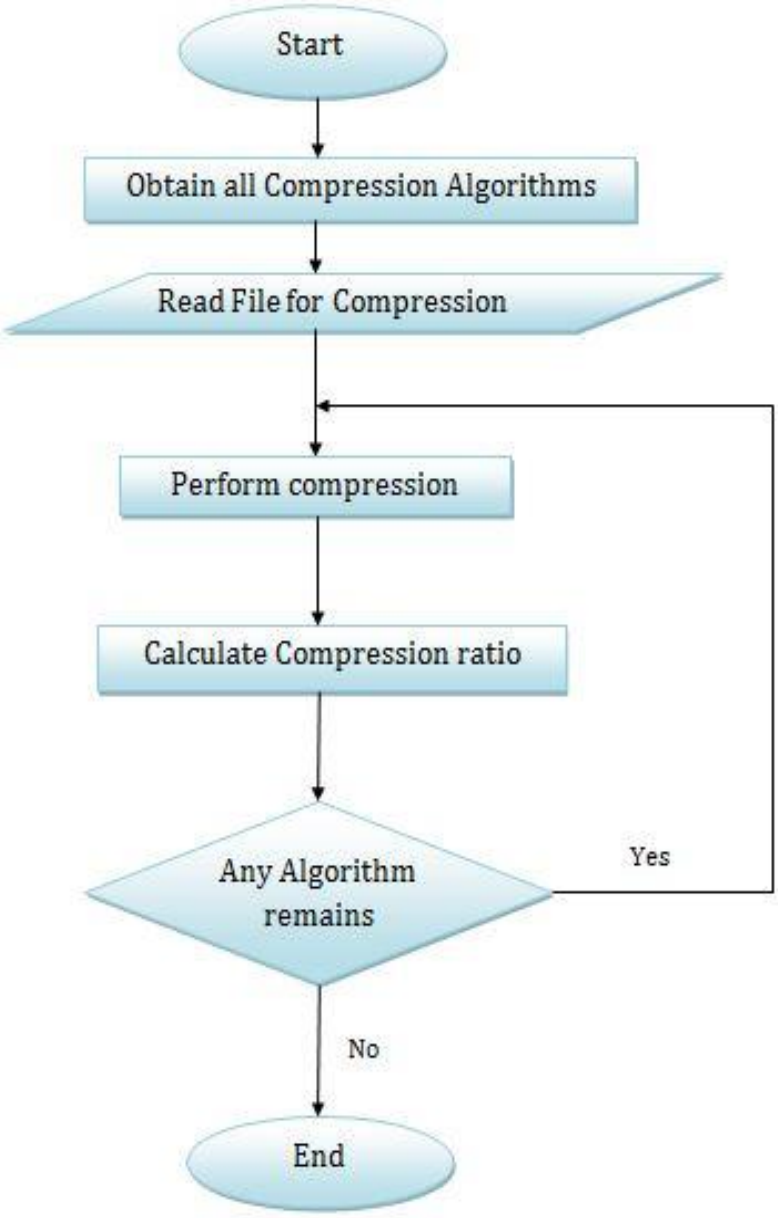


Fig. 3.4: Compression Analysis flowchart

The algorithm and flow chart describes the methodology used for calculating the compression ratio of the file. Compression ratio of each compression algorithm can be analyzed for files of different data types and different sizes. First the file is read and the data type of file is determined from the extension of the file. The uncompressed file size is noted. Then the file is compressed using each selected compression algorithm and compression ratio corresponding to each compression algorithm is

calculated using eq. 1. The result are then stored in the database. These results can be used to analyze the compression algorithms for different files.

3.4 Compression-Encryption Learning Model

Encryption provides security and compression reduces the size of data file and reduces disk utilization. A learning model for compression of the file is proposed, for selecting a file of a specific type for compression, by a suitable compression algorithm, before encryption is performed on it. The learning model will help to identify, on which file compression should be done before encrypting it, to decrease the disk utilization and to increase the efficiency of the system while transferring of the file. The learning model is divided into two components as Supervised Learning component and Background learning component.

a. Supervised Learning

Supervised learning is the type of learning in which the system in trained by giving some input data, so that it can create some knowledge base to decide what operation should be done when the system is actually put into the work. In supervised learning, numbers of files of different types are tested by each compression algorithm, for training the system. Data loss is not affordable, as the files may contain important information, therefore only lossless compression algorithms are chosen for testing.

In supervised learning, the input file is first compressed by all the compression algorithms, then encrypted by using specific cipher algorithm, decrypted and decompressed. Time required for each of these operations is calculated. For each compression algorithm the compression ratio is calculated. For ranking the system for selecting the compression algorithm according to data type, each compression algorithm is given some score on the basis of time taken by it and its compression ratio.

For developing a ranking system, a formula for calculating compression algorithm score is determined.

$$Compression\ Score = \frac{Compression\ Ratio}{\log_{10}(Compression\ Time\ +\ Encryption\ Time + Decryption\ Time\ +\ Decompression\ Time)} \quad (2)$$

Time is taken in milliseconds. The variation in compression ratio is not as large as time. To reduce the time difference in a proper scale, logarithmic time is taken. The formula is devised empirically, by testing it on number of files of different type.

The ranking system will be helpful for deciding which compression algorithm should be chosen for compression for a specific data type. Each compression algorithm is ranked for a file, according to the compression score obtained for it.

Higher the compression score, higher is the compression ranking. The ranking logic of compression algorithm for each file is as follows:

Procedure Name: ***RankingLogic***
Input Parameters: *Compression Algo A, B, C;*
Cipher Algo: z;

Start Procedure
For each input file
$\forall(Compression\ Algo\ A, B, C)$
Calculate:
Compression time;
Encryption time of z;
Decryption time of z;
Decompression time;
Total time for all four operations;
Compression Score;
Sort score in descending order for A, B, C.
Compression Rank for A, B, C =
For (i = 1; i < = no. of Compression Algo; i++)
Rank[i] = Sort[i];
End Procedure

Each compression algorithm gets rank by the above algorithm for files of different data type and different sizes. The algorithm with highest compression score gets first rank followed by others according to compression score.

The results of learning model such as compression time, encryption time, decryption time, decompression time, compression ratio, and compression score of each compression algorithm are stored in the database table "compressionScore" for creating a ranking system. Table 3.1 shows the design of the database schema of table compressionScore used for storing the result of simulation.

Table 3.1: CompressionScore Table

File Name	File Type	File Size	Compression Algo 1			Compression Algo 2			----	Comp Algo N
			Comp Ratio	Total Time	Comp Score	Comp Ratio	Total Time	Comp Score	---	---

This ranking system allocates the rank to each file in training system. These ranks are maintained in the training database "compressionRank". The Table 3.2 shows the design of the database schema of table "compressionRank" used for storing the rank of each compression algorithm.

Table 3.2: CompressionRank Table

File Name	File Type	File Size	Rank_Algo1	Rank_Algo2	-----	Rank_AlgoN

b. Background Learning

Background learning is the type of learning in which the system keeps on learning even when the system is put to work and keep on increasing its knowledge base. Due to background learning the knowledge base is continuously increasing with each single work done by the system. The background learning helps to improve the efficiency of the system. Though during training lot of data is tested, sometime the

training data falls short for training the system. The resulting system may have some wrong results. Background learning keeps on updating the knowledge base, hence if some mistake has been introduced during training the system, it can be solved automatically by ever increasing knowledge base. Thus due to background learning, efficiency of the system can be improved. Once the system is put to work, the system keep on learning by new input files and updates the database tables for various input files. The background learning component runs all the compression algorithms for each type of file. The algorithm with highest compression score always gets first rank. For each input, the compression Score and compression Rank are added in the compressionScore and compressionRank database table. The size of these tables keeps of increasing continuously for each input.

The best compression Algorithm is chosen for compression of a specific type of file as follows:

For Compression Algo A, B, and C:
For total files of specific type in "compressionRank" Table
Best Compression Algo = Max Rank 1st of specific type of file compressed by (A, B, C).

The above algorithm decides the compression algorithm that should be chosen. This best compression algorithm is then chosen for compression of file of specific type, then encrypt it and transfer it across the network.

Chapter 4

Testing and Results

4.1 Experimental Setup

The java platform (openjdk1.6.0_14) is selected for implementation. JCA and JCE are used for cipher algorithm implementation. JCA platform contains "provider" based architecture. This architecture of JCA contains some set of APIs for performing encryption (symmetric ciphers, asymmetric ciphers, block ciphers, stream ciphers), secure random number generation, message digests (hash), certificates and certificate validation, key generation and digital signatures. Sun and Bouncy Castel provider are used for implementing cryptographic algorithms listed in Table 2.1. All cipher algorithms are implemented using sun provider except skipjack, which is implemented using Bouncy Castel provider. Following is the piece of code used for analysis:

```
// BC = Bouncy Castel Provider
cipher = Cipher.getInstance(algorithmName, "BC");
// for encryption
operation = Cipher.ENCRYPT_MODE;
//Initializing cipher
cipher.init(operation,  secretKey);
//Performing Encryption
encryptOutLength = cipher.update(inputBytes, 0, bufferSize, outputBytes);
```

```
encryptOutLength = cipher.doFinal(inputBytes, 0, inLength);
// for decryption
operation = Cipher.DECRYPT_MODE;
//Initializing cipher
cipher.init(operation,  secretKey);
//Performing Decryption
decryptOutLength = cipher.update(inputBytes, 0, bufferSize, outputBytes);
decryptOutLength = cipher.doFinal(inputBytes, 0, inLength);
```

All execution results are benchmarked on a particular machine. The execution results are taken on a machine having Intel® Core™ i7-2600 (4 cores, 3.40 GHz) processor with Intel® Q65 Express 4 GB 1333 MHz DDR3 (RAM) and Ubuntu 12.04 LTS operating System (Kernel Version: 3.2.0-23-generic-pae i386).

4.2 Test case analysis and observation for Symmetric key Cryptography

From the related works, it is realized that none of the work did a very detailed analysis of the performance of various symmetric algorithms, on various parameters of different type of files. In order to select the most suitable cryptographic algorithm for encryption, following test cases are considered to analyze the time taken for encryption by various cryptographic algorithms.

1. Files of different data types having same sizes.
2. Files having same data type but different sizes.
3. Files with different data densities.
4. Key variation for various encryption algorithms.
5. Block mode variation for various block ciphers.

Case study 1: Files of different data types having same sizes.

A data file format represents the standard for encoding the information to be stored in computer file. This case study is taken to check whether the encryption has dependency on type of data or not. Different data type files like audio, image, textual, video and health data file format like DICOM of nearly 50MB, and 100MB in size are

chosen and encryption time of different cipher algorithms is calculated for these data types.

For all executions of a specific cipher algorithm, varying parameter is data type and constant parameters are key size and block cipher mode. Key size and block mode are at kept at bare minimal parameters. The key size of AES, DES, 3-DES, RC2, Blowfish, Skipjack, and RC4 are kept at minimum values as 128, 56, 112, 40, 32, 80 and 40 bits respectively. Block cipher mode used is ECB with PKCS#5 padding scheme. Figure 4.1 and 4.2 shows the execution time of the algorithms for different data type files.

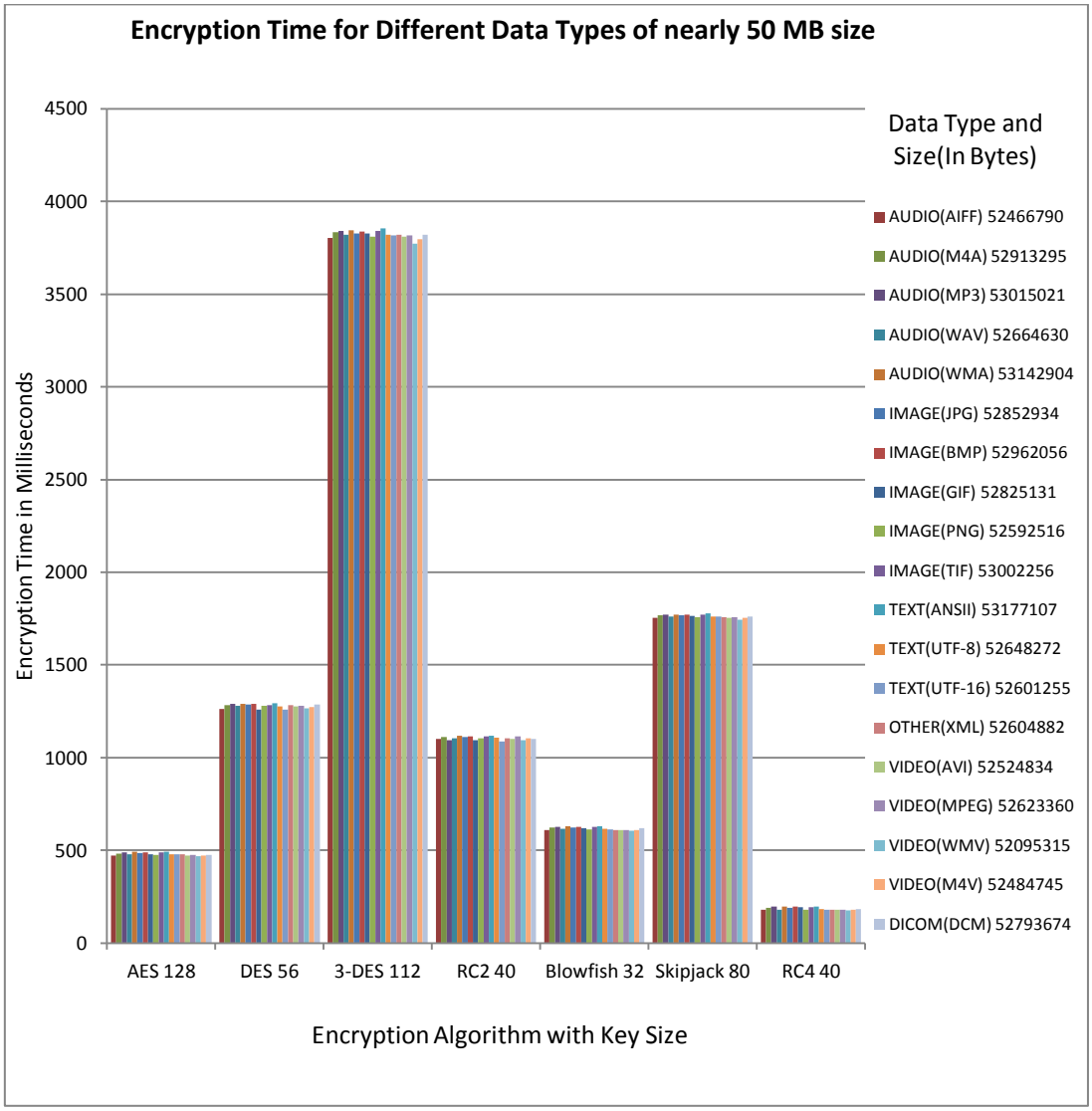


Fig. 4.1: Encryption time Vs Cipher Algorithm for files of different data type of size 50MB.

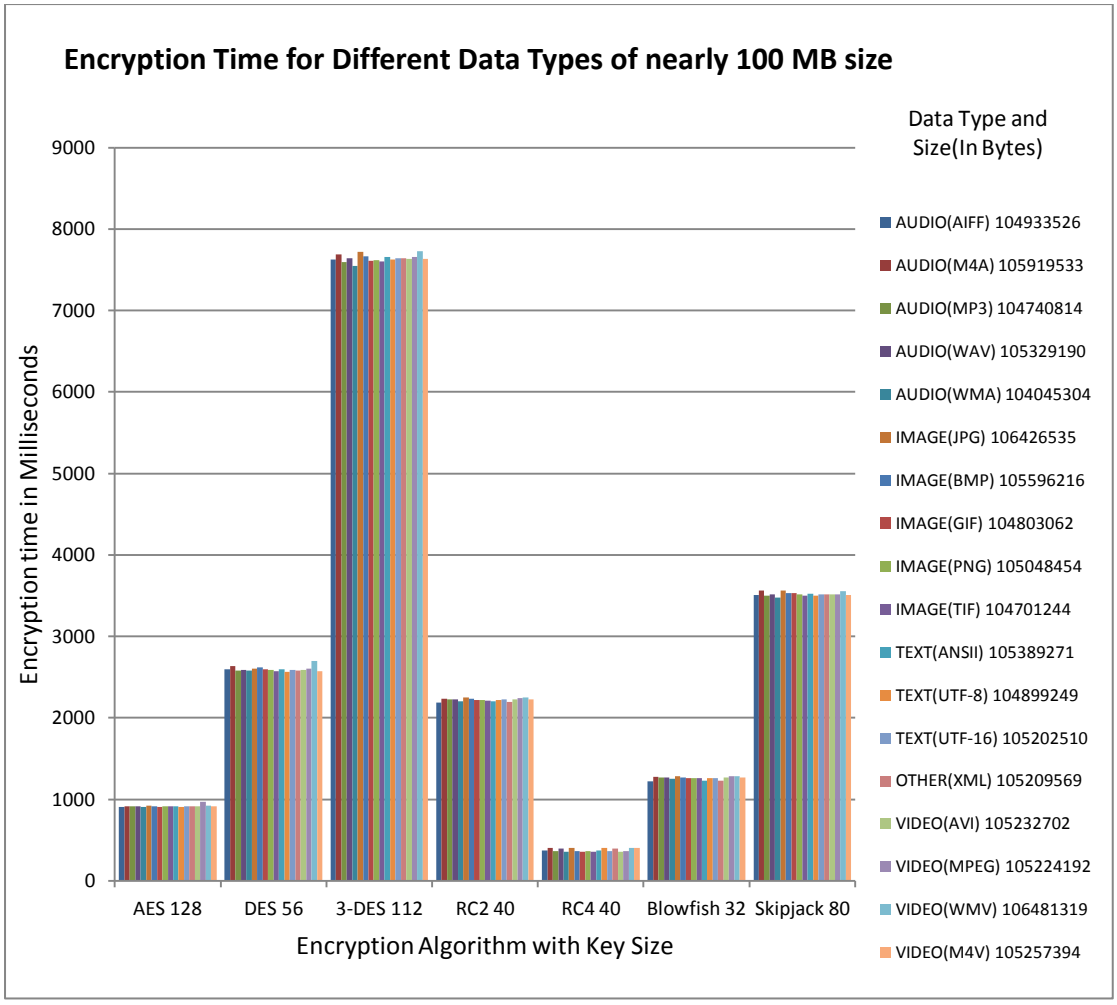


Fig. 4.2: Encryption time Vs Cipher Algorithm for files of different data type of size 50MB.

Observation:

The result shows that the encryption time does not vary according to the type of the data. Files of different formats have nearly same encryption time for a particular encryption algorithm. For example for file size of 50MB, AES encryption algorithm with key size 128 bits takes nearly 0.45 seconds for all the tested data types. Similarly all other tested encryption algorithms took same time for different file types of same type. Therefore, encryption depends only on the number of bytes in the file and not on the type of file. AES works faster than all the block ciphers that were simulated at bare minimal parameters. RC4 with key size 40 is fastest among the algorithms tested.

Case Study 2: Data files of same type with different sizes.

This case study is taken to ensure once again the observations obtained from case study 1, that encryption time depends on number of bytes in the file. In this study is, different files of same types but different sizes are given for encryption and estimated

the encryption time. For all executions, key size and block mode are kept at bare minimal parameters. Table 4.1 gives the details about the files used for all executions and Figure 4.3, 4.4 and 4.5 show the execution results.

Table 4.1: Execution parameters for files of different sizes

File Type	Varying Parameters (Data Size)	Constant Parameters
AIFF	10.7MB, 50MB, 100MB	Data Type, Key size, Block Mode
AVI	50MB, 100MB, 482MB	
DICOM	14.9MB, 50.3MB, 115MB, 151MB	

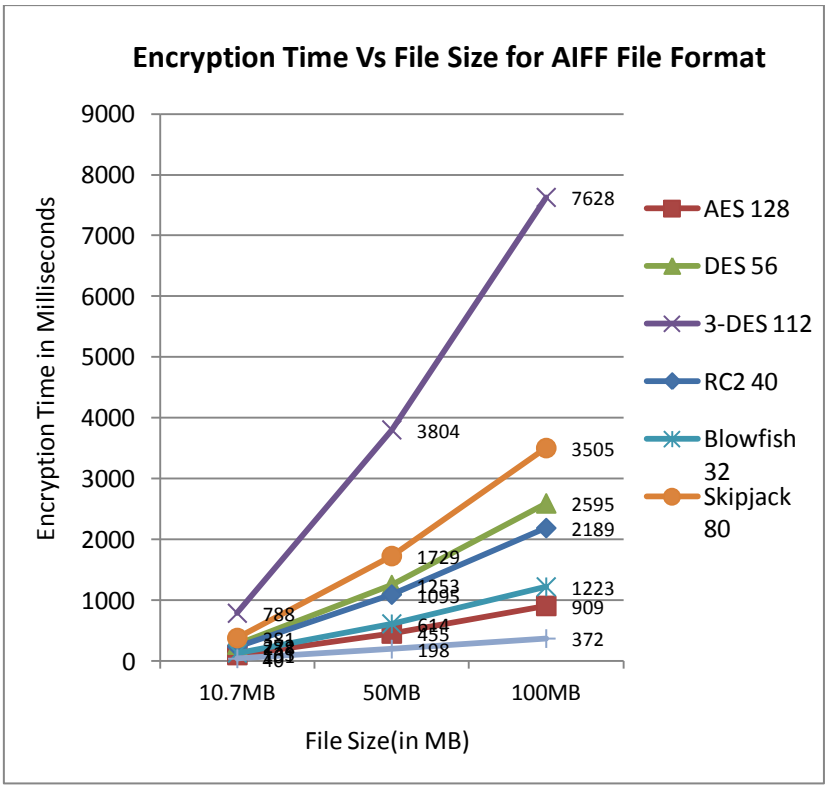


Fig. 4.3: File size Vs Encryption time for AIFF file of different sizes

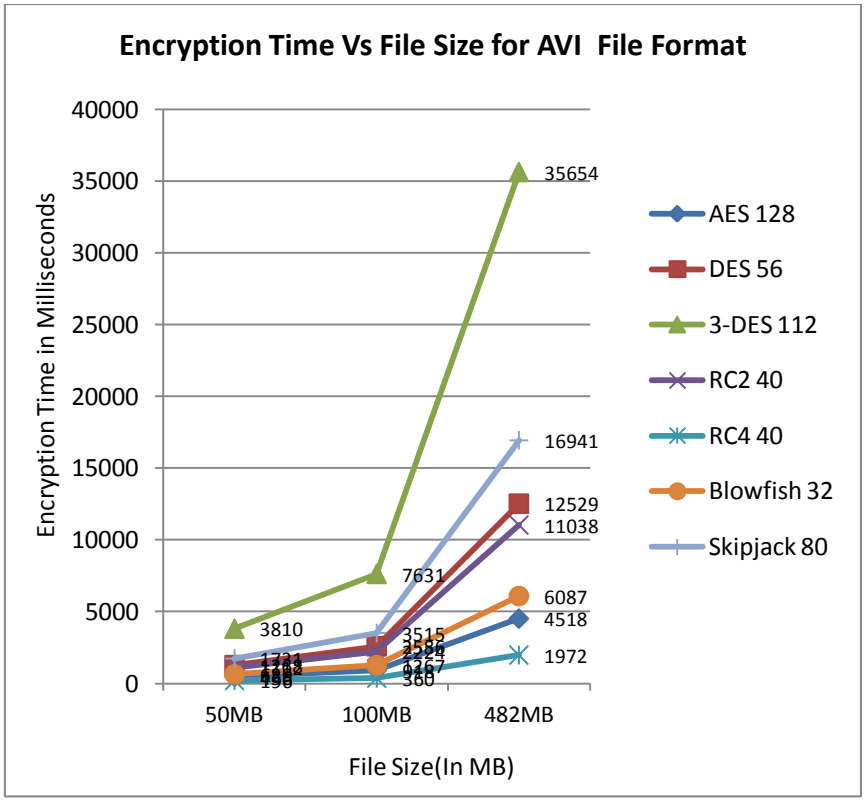


Fig. 4.4: File size Vs Encryption time for AVI file of different sizes

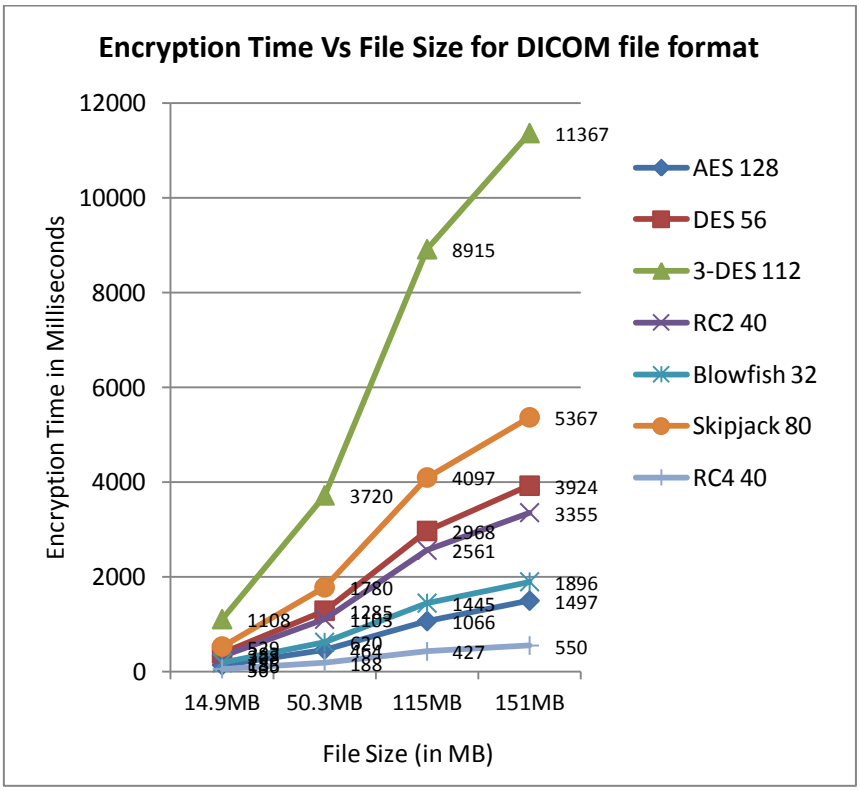


Fig. 4.5: File size Vs Encryption time for DICOM file of different sizes.

Encryption rate can be defined as the number of bytes encrypted per unit time.

$$Encryption\ Rate = \frac{File\ Size}{Encryption\ Time} \tag{3}$$

The encryption rate for different files is also calculated by Eq. 3 to see the effect of varying the size of file on the encryption time, and is given in Table 4.2.

Table 4.2: Encryption rate for files of different sizes

File Type	Size (In MB)	Encryption Rate In (MB/sec)						
		AES 128	DES 56	3-DES 112	RC2 40	Blowfish 32	Skipjack 80	RC4 40
AIFF	10.7	109.18	39.43	13.61	45.06	80.63	28.14	268.12
	50	109.95	39.92	13.15	45.68	81.47	28.93	270.66
	100	110.07	38.55	13.11	45.71	81.81	28.54	268.96
AVI	50	107.83	39.49	13.14	45.04	79.62	28.93	265.52
	100	109.3	38.8	13.14	45.11	79.19	28.54	270.72
	482	108.74	38.49	13.52	45.09	79.22	28.46	265.55
DICOM	14.9	107.65	38.93	13.45	45.46	80.61	28.19	266.31
	50.3	108.48	39.17	13.53	45.63	81.19	28.28	267.76
	115	108.69	39.03	12.99	45.24	80.18	28.28	271.34
	151	108.21	38.61	13.32	45.15	79.91	28.23	270.47
Average		**108.8**	**39.04**	**13.3**	**45.32**	**80.383**	**28.452**	**268.5**

Observation:

From the results in Table 4.2 and Figure 4.3, 4.4 and 4.5, we can find that the result for different size of data varies proportional to the size of data file. Encryption time increases as file size increases in multiples of data size. For each encryption algorithm, same parameters are used for files of different sizes. It is observed that the encryption rate of a specific algorithm is almost constant for every file. Encryption rate of AES is around 108 MB/s, which highest among all the block ciphers tested at bare minimal parameters. Encryption rate of RC4 is around 268MB/s, which is highest among all the ciphers.

Case Study 3: Files with different densities of data.

This case study is taken to check whether encryption rate depends on density of data. Encryption rate is evaluated for different files, a sparse AIFF file of 69MB and a dense AIFF file of 58.5MB. For a cipher algorithm, key size and block mode are kept at bare minimal parameters. The results of execution are shown in Table 4.3.

Table 4.3: Encryption rate for sparse and dense data file

Algorithm Name	Sparse (72000118 Bytes) AIFF file		Dense (61392454 Bytes) AIFF file	
	Encrypt Time(ms)	Encryption Rate(MB/s)	Encrypt Time(ms)	Encryption Rate(MB/s)
AES 128	634	108.28	540	108.40
DES 56	1801	38.11	1537	38.08
3-DES 112	5076	13.52	4365	13.41
RC2 128	1520	45.16	1285	45.55
Blowfish 128	854	80.38	723	80.96
Skipjack 128	2386	28.77	2042	28.66
RC4 128	253	271.35	216	271.01

Observation:

Encryption rate for sparse and dense file has been calculated by using Eq. 3. Table 4.3 shows that the encryption time is not affected by density of data in a file. The variation in time with respect to different algorithms follows the same pattern for both sparse and dense files. The encryption rate for a particular cipher algorithm remains the same, even if the file is sparse or dense. It depends on only the number of bytes in the file.

Case Study 4: Encryption Algorithms with different key sizes

The security of an encryption algorithm depends upon its key size. The larger the key size is, harder it is to crack the encryption algorithm. Therefore it is required to check whether there is any overhead in encryption time with increase in key size. This case study is to analyze the effect of changing the size of encryption key on encryption

time. DICOM file of 50.5MB is taken and different cipher algorithms are executed for different size of keys supported by them in ECB mode with PKCS#5 padding scheme. The various key sizes mentioned in Table 4.4 are used during experimentation. Figure 4.6 shows the result of execution for key size variation.

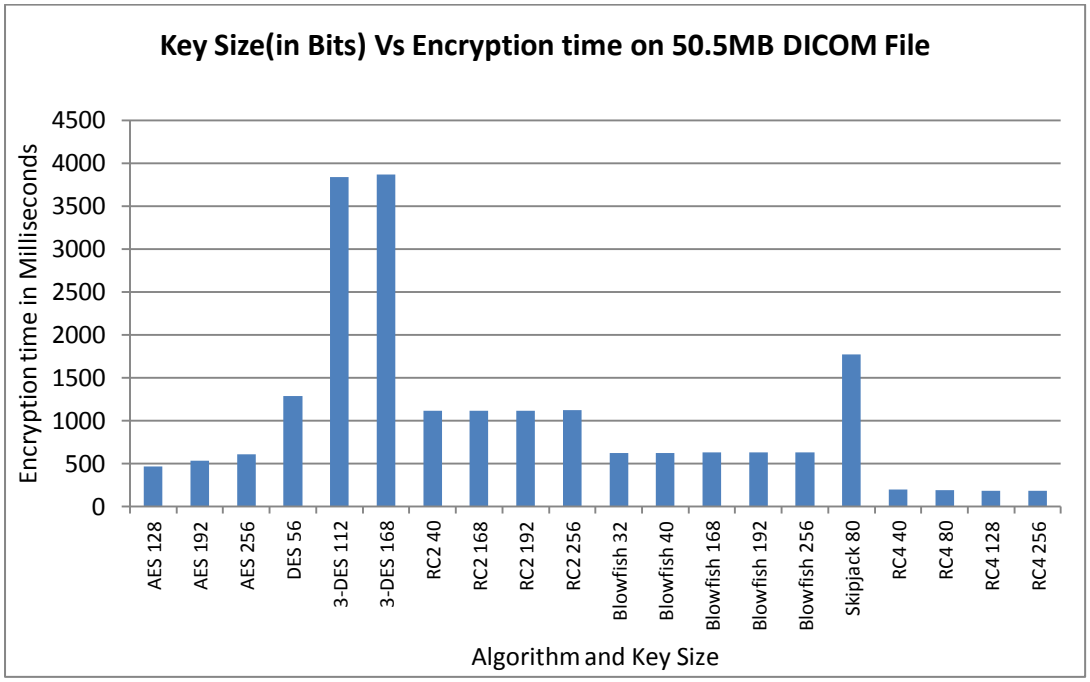


Fig. 4.6: Variation of key sizes for different cipher Algorithms

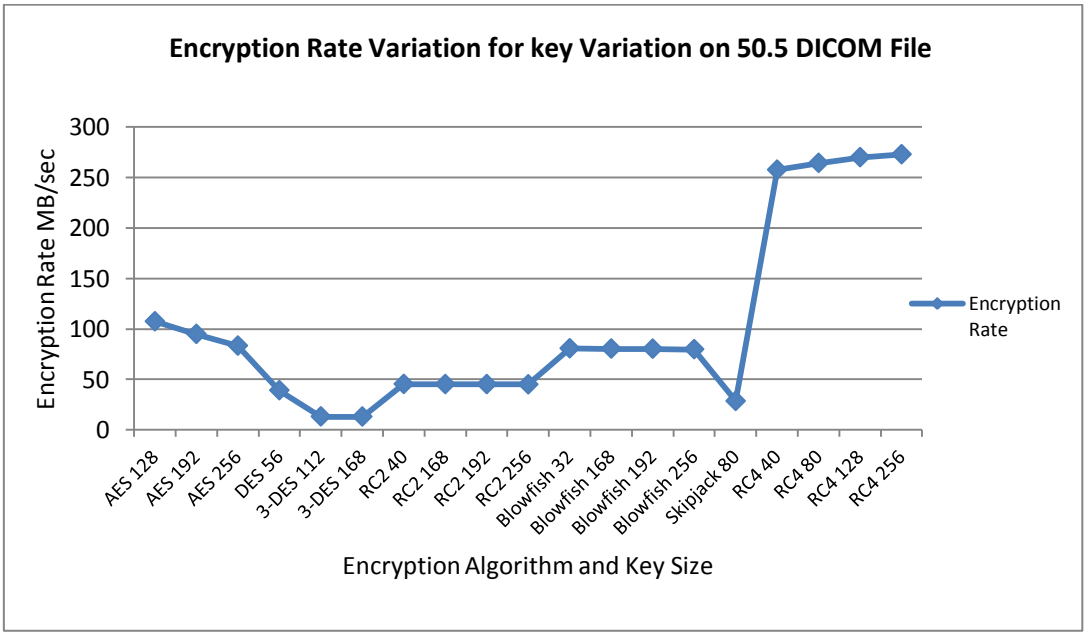


Fig. 4.7: Encryption Rate variation for key variation of cipher Algorithms on 50.5 DICOM File

Table 4.4: Encryption rate for key variation of different cipher algorithms on 50.5MB DICOM file

Algorithm	Key Size	Encryption Rate(in MB/s)
AES	128	107.4466
	192	94.92459
	256	83.33314
DES	56	39.1472
3-DES	112	13.16129
	168	13.06257
RC2	40	45.33203
	168	45.21028
	192	45.16984
	256	44.96873
Blowfish	32	80.67074
	40	80.41383
	168	80.35385
	192	80.28598
	256	79.77865
Skipjack	80	28.5149
RC4	40	257.6525
	80	264.3973
	128	270.0529
	256	272.9723

Observation:

The execution results show that for all ciphers algorithms, the encryption time varies with the change in the size of the key. Encryption time increases with increase in key size for block ciphers. The variation in time is very small. Using Eq. 1 encryption rate for different algorithms is calculated. Table 4.4 shows the evaluation results for different cipher algorithms. With the change in size of key the change in rate of encryption rate is quiet small. For better security, even if higher key size is applied,

still the overhead in encryption time is not too large. AES dominates in the block cipher. RC4 is fastest among all algorithms tested.

Case Study 5: Cipher Algorithms with different block modes.

Security of cipher algorithm also varies according to block cipher modes. Different block cipher modes are used for different applications. For example PCBC is used in WASTE and Kerberos v4. Security levels may differ according to type of application and can be classified as:

- High: There are some applications where security matters the most than speed of encryption, like data related to health.
- Medium: The applications where speed of encryption and security both are important. For example chatting applications.
- Low: In some applications speed of encryption matters most than security, like securing data on personal systems or data transfer within an organization which will have no adverse effect on operation of organization if leaked.

This case study is to check the encryption time variation with respect to block cipher modes. All block cipher algorithms have been executed for different block modes with PKCS#5 padding scheme on 50.5MB DICOM file. The key size for particular block cipher algorithm is kept at the bare minimal value. The various block modes mentioned in Table 2.1 are used for evaluation. The Figure 4.8 shows the block cipher variation for AES 128 and Figure 4.9 shows the result of execution for all encryption algorithms.

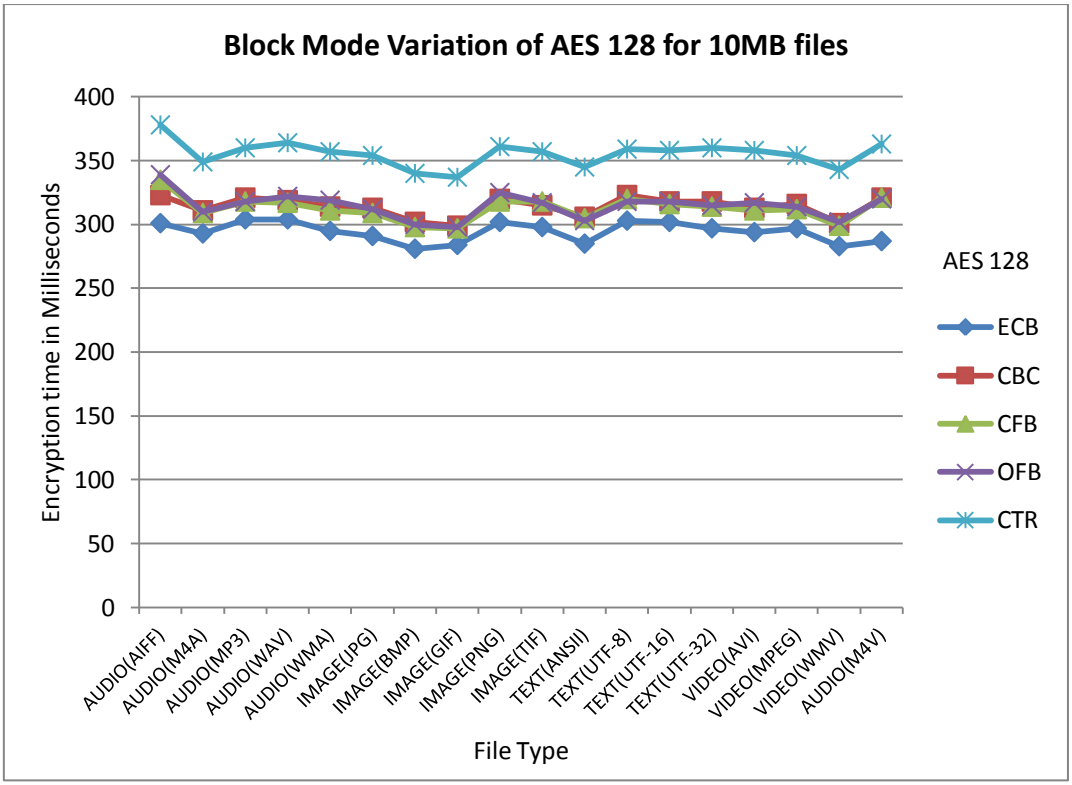


Fig. 4.8: Block Mode Variation of AES 128 for 10MB files

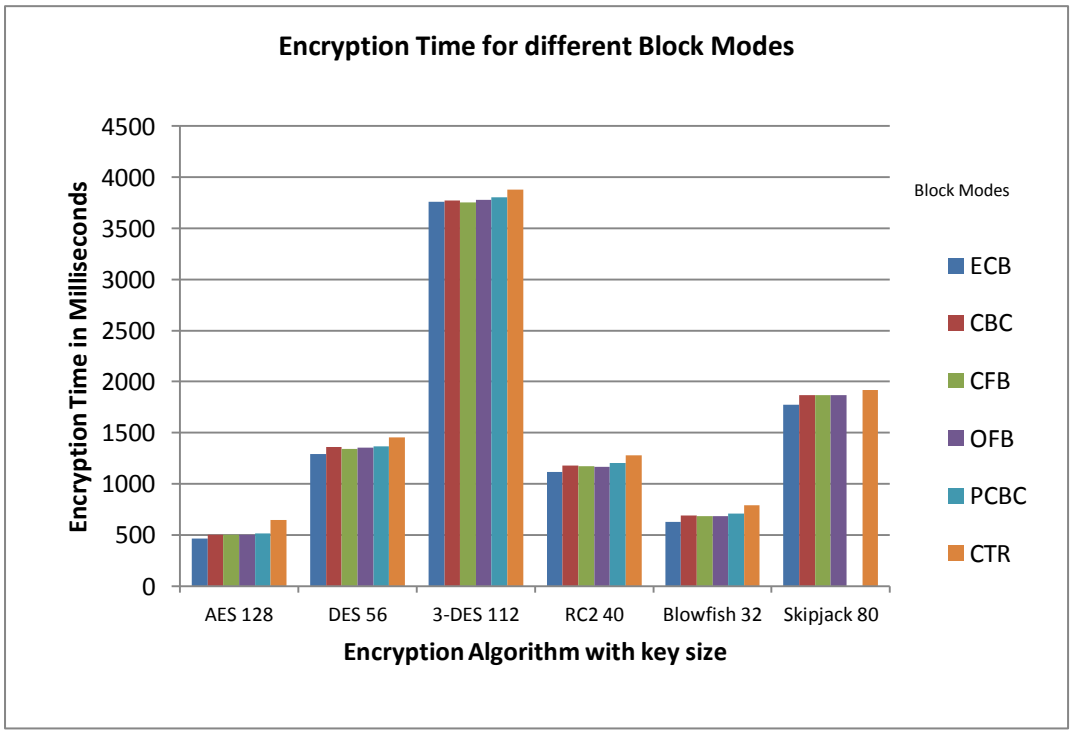


Fig. 4.9: Variation of cipher block modes on 50.5MB DICOM file

Table 4.5: Encryption rate for variation of cipher block modes on 50.5MB DICOM file

	Encryption Rate(in MB/s) for Different Block modes on 50.5MB DICOM file					
Modes	ECB	CBC	CFB	OFB	PCBC	CTR
AES 128	108.6	101.2	100.6	100.4	98.24	78.54
DES 56	39.14	37.16	37.6	37.37	36.94	34.7
3-DES 112	13.44	13.39	13.44	13.37	13.27	13.02
RC2 40	45.33	42.8	43.08	43.27	41.90	39.70
Blowfish 32	80.67	73.19	73.61	74.04	71.52	64.00
Skipjack 80	28.51	27.00	27.06	27.00	---	26.30

Observation:

The result of execution shows that a different block cipher algorithm follows almost the same pattern for each block modes. Table 4.5 gives the encryption rate for different cipher algorithm evaluated using Eq. 1. Evaluated results show that encryption rate differs considerably according to type of cipher block mode. For all encryption algorithms, ECB takes lesser time for encryption and CTR highest among all.

4.3 Proposed Security Performance Matrix

NIST (National Institute of standard and Technology) has recommended various block cipher modes of operation for confidentiality [14]. All the block cipher modes mentioned in Table 1 are modes for maintaining confidentiality of messages. The block modes are trivial to break by different types of attacks. On the basis of security levels, we propose a model of block cipher modes consisting of 3 levels of security.

1. Low: for applications dealing with data which requires low security but faster processing.
2. Medium: for application dealing with data which requires medium security.
3. High: for applications dealing with data which require high security like health data.

AES performs best among all the block ciphers analyzed for various cases considered for research. A detailed study is made on the security strength of various block mode operations [12] [14]. ECB mode is fastest but less semantically secure with respect to a chosen-plaintext attack (SemCPA) attacks. CBC, CFB, OFB, PCBC are SemCPA secure with random IV, but SemCPA insecure if IV is nonce. Cipher modes ECB, CBC, CFB, OFB, and PCBC are not semantically secure with respect to a chosen-ciphertext attack (SemCCA) attacks. CTR is SemCPA secure. CTR can be cracked by SemCCA attack but it is more secure than all other cipher modes.

As AES comes out to be the best in case of performance as well as security among all the block ciphers. Encryption time and encryption rate for all keys and modes supported by AES are analyzed. The Figure 4.10 shows the encryption time for different modes and keys of AES algorithm. Table 4.6 shows the encryption rate for all modes and keys supported by AES.

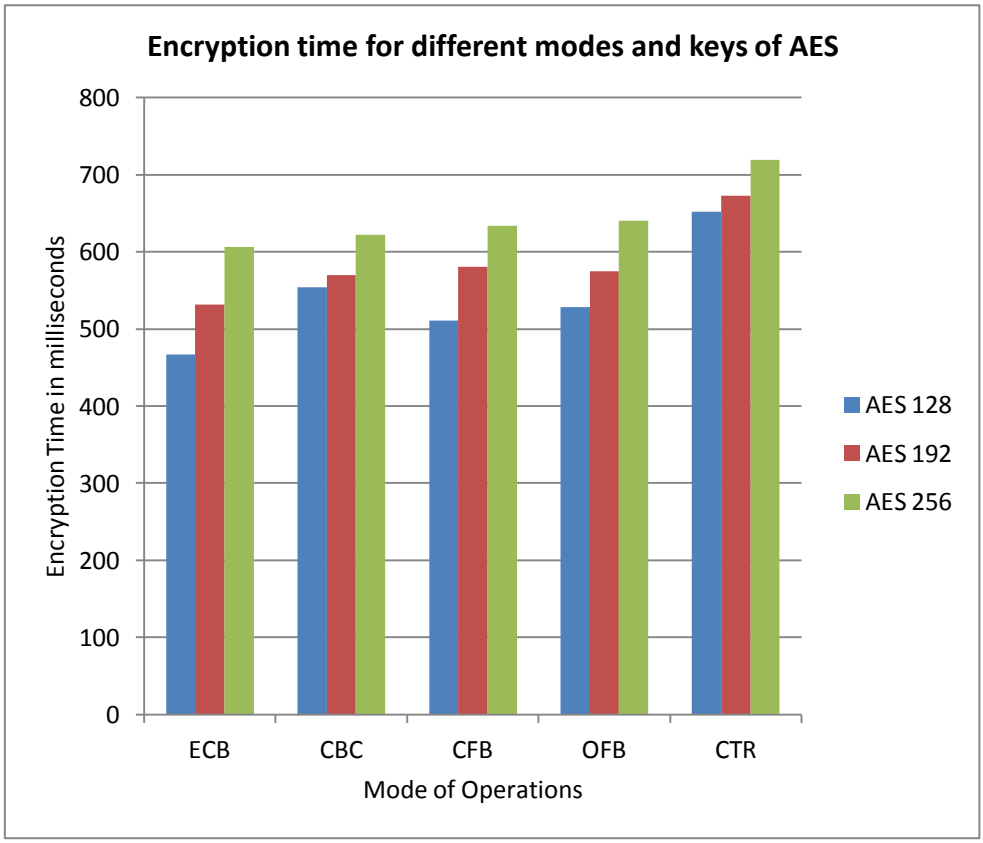


Fig 4.10: Encryption time for different modes and keys of AES

Table 4.6: Encryption rate for different modes and keys of AES

Encryption Rate for AES					
	ECB	CBC	CFB	OFB	CTR
AES 128	108.15	101.2	100.6	100.4	78.54
AES 192	94.94	89.39	86.93	87.84	75.04
AES 256	83.34	81.20	79.66	78.91	70.24

From the Figure 4.10 and Table 4.6 the performance of AES for different modes and keys can be determined for proposing the security performance matrix. From the analysis of all algorithms a performance matrix is proposed for encrypting a file at proper security and suitable rate. Table 4.7 shows the proposed performance matrix for AES algorithm.

Table 4.7: Security performance matrix

Security	**AES**		
	Key size	Mode	Encryption Rate (MB/s)
Low	128	ECB	108
Medium	192	CBC, CFB, OFB,	89, 87, 88,
High	256	CTR	70

4.4 Analysis of Data Types for Compression Algorithms.

LZ4, LZF, Huffman, Deflate and Gzip compression algorithm were selected for analysis of compression ratio of different files. A data file format represents the standard for encoding the information to be stored in computer file. Like encryption, analysis is done on different data type files for compression Algorithms. Different data type files like audio, image, textual, video and health data file format like DICOM of 100MB in size is chosen and encryption time of different cipher algorithms is calculated for these data types. Gzip, Deflate, Huffman compression algorithm are implemented using standard “sun” library. LZ4, LZF are implemented

using “Apache” Library. The compression ratio and compression time of different compression algorithm are shown in the Table 4.8.

Table 4.8: Compression Ratio of Different File Formats

File format	LZ4		Huffman		Deflate		LZF		Gzip	
	CT	CR	CT	CR	CT	CR	CT	CR	CT	CR
AIFF	82	2.47	3541	19.86	3380	19.67	1023	2.87	3440	19.80
WAV	94	2.06	3900	23.37	3743	23.17	1017	2.63	3849	23.11
WMA	87	38.04	2941	39.3	2338	39.12	743	36.49	2530	39.2
BMP	249	24.4	3135	45.89	2583	44.52	855	25.7	2782	44.92
TIF	218	22.68	2577	39.94	2750	39.01	917	22.06	2812	39.86
TXT	57	94.18	1393	85.44	844	84.88	580	43.06	856	84.99
DCM	56	55.97	1237	62.33	978	60.56	207	58.72	1279	61.09
AVI	60	6.64	3387	6.94	3176	6.94	1063	5.74	3206	6.94

These were some of the files compatible with compression. Here

CT – compression Time in milliseconds

CR – compression ratio

4.5 Compression-Encryption Learning Model

A compression-encryption learning model is made for compressing a file using suitable compression algorithm, before encrypting it. Analyzing compression algorithm for different data type showed that compression of file is different for different algorithm. Though some compression algorithms, compress the file to greater extent but the time require for compression is too high. Due to this system is not that much efficient. Score and rank of each compression algorithm is calculated for each file type. MYSQL data base is used to store the testing reading. Figure 4.11 and Figure 4.12 shows the database schema used for storing compression score and compression rank of the selected compression Algorithm.

```
mysql> describe compressionScore;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| Name   | varchar(20) | NO   | PRI | NULL    |       |
| Type   | varchar(20) | NO   | PRI | NULL    |       |
| Size   | bigint(20)  | NO   | PRI | NULL    |       |
| CR_LZ4 | double      | NO   |     | NULL    |       |
| TT_LZ4 | double      | NO   |     | NULL    |       |
| S_LZ4  | double      | NO   |     | NULL    |       |
| CR_LZF | double      | NO   |     | NULL    |       |
| TT_LZF | double      | NO   |     | NULL    |       |
| S_LZF  | double      | NO   |     | NULL    |       |
| CR_HF  | double      | NO   |     | NULL    |       |
| TT_HF  | double      | NO   |     | NULL    |       |
| S_HF   | double      | NO   |     | NULL    |       |
| CR_DF  | double      | NO   |     | NULL    |       |
| TT_DF  | double      | NO   |     | NULL    |       |
| S_DF   | double      | NO   |     | NULL    |       |
| CR_ZIP | double      | NO   |     | NULL    |       |
| TT_ZIP | double      | NO   |     | NULL    |       |
| S_ZIP  | double      | NO   |     | NULL    |       |
| CR_GZ  | double      | NO   |     | NULL    |       |
| TT_GZ  | double      | NO   |     | NULL    |       |
| S_GZ   | double      | NO   |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+
21 rows in set (0.00 sec)
```

Fig. 4.11: CompressionScore Database Schema

Notations

- Name - indicate name of the input file
- Type - data type of the input file
- Size – size of input file
- CR – compression ratio
- TT – total time for compression, encryption, decryption, decompression
- S – Algorithm score
- LZ4 – LZ4 compression algorithm
- LZF – LZF compression algorithm
- HF – Huffman compression
- DF – Deflate compression
- ZIP – Zip compression
- GZ – Gzip Compression

```
mysql> describe compressionRank;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| Name         | varchar(20) | NO   | PRI | NULL    |       |
| Type         | varchar(20) | NO   | PRI | NULL    |       |
| Size         | bigint(20)  | NO   | PRI | NULL    |       |
| Rank_LZ4     | int(11)     | NO   |     | NULL    |       |
| Rank_LZF     | int(11)     | NO   |     | NULL    |       |
| Rank_Huffman | int(11)     | NO   |     | NULL    |       |
| Rank_Deflate | int(11)     | NO   |     | NULL    |       |
| Rank_ZIP     | int(11)     | NO   |     | NULL    |       |
| Rank_GZIP    | int(11)     | NO   |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
9 rows in set (0.00 sec)
```

Fig. 4.12: CompressionRank Database Schema

Following is the sample piece of code of learning model used for analyzing different files.

```
// creating object for different compression algorithms
CompressionAlgorithms selectCompAlgo = new CompressionAlgorithms();
// apply compression Algorithms to input file
compressDuration = selectCompAlgo.compress(inputFile, compressFile);
//apply encryption and decryption and calculate their duration
// apply decompression
 decompressDuration = selectCompAlgo.decompress(compressFile, decompFile);
// calculate compression Ratio and Score
// calculate Rank and store in database
```

The model was tested on some 670 different files of different file formats. Different compression algorithm like Huffman, Deflate, LZF, LZ4 and Zip were used to analyze these files. All compression algorithms get some rank on the basis of compression ratio and time required for the compression. From the analysis of the encryption algorithms it was found that AES was best performing and secure symmetric cipher algorithm. Therefore for all the analysis AES encryption algorithm was used having key size of 128 bits with ECB mode.

While analyzing the system was, it was observed that the compression ratio for a particular file format using a specific compression algorithm differs. For example bmp files of 10MB, 25MB and 30MB shows compression ratio 35%, 29% and 32 % respectively for LZ4. The best compression algorithm is selected from the rank of compression Algorithm.

Table 4.9: Best Compression Algorithm for different data type

File Type	Best Compression Algorithm	Compression Ratio
DICOM	LZ4	20%
BMP	Huffman	41.58%
WMA	LZ4	33.52%
AIFF	Deflate	15.88%
WAV	Deflate	8.72%

The Table 4.9 shows the best compression algorithm for different files. These compression algorithms can be used for compressing the file before it is encrypted for transferring across the network. As the model is self learning it can improve itself with each input.

Chapter 5

Conclusion and Future Scope

5.1 Conclusion

Different Symmetric key algorithm have been analyzed for performance evaluation for various parameters like different data type, data size, data density, key size and cipher block modes, and tested how the encryption time varies for different symmetric cipher algorithms. From the simulation results following thing are concluded:

- After analyzing files of different formats have nearly same size it was concluded that encryption time does not vary according to the type of the data.
- The encryption rate for a particular cipher algorithm remains the same, even if the file is sparse or dense. Encryption rate is independent of data density.
- Encryption depends only on the number of bytes in the file and not on the type of file or file density. Encryption rate of AES was found to be 108MB/sec in ECB mode which was highest among all block cipher selected for analysis.
- Analysis of files of different sizes leads to the conclusion that, different size of data varies proportional to the size of data file. Encryption time increases as file size increases in multiples of data size. Encryption rate of a specific algorithm is remains almost constant for every file

- The encryption time varies with the change in the size of the key. Encryption time increases with increase in key size for block ciphers but reduces for stream cipher like RC4 with increase in key size.
- All the block cipher modes follow the same pattern, *i.e.* all cipher algorithms takes less encryption time using ECB mode and high encryption time using CTR mode.

After analysis of all parameters, AES was found to be most suitable encryption algorithm having encryption rate of 108MB/sec in ECB mode. AES was used in the proposed compression-encryption model. A compression-encryption model was proposed for identifying the files that should be compressed before encrypting and the files that should be encrypted without compressing them. A formula was derived imperially to determine best suitable compression algorithm that should be used for compressing the file according to data type and data size to reduce the overhead of time for compression and to increase the efficiency and security to data that is being transferred.

For different data type files, different compression algorithms appeared to be most suitable for compression of file, for example LZ4 for DICOM and WMA, Huffman for BMP, Deflate for AIFF and WAV etc. The compression-encryption model was helpful for increasing the security as well as the performance of the system while transferring the file across the network.

5.2 Future Scope

The future work can be compromise of implementing this work for cloud environment and the efficiency of the system can be improved even more by implementing it in Hadoop environment.

Appendix

Paper Publication

The following lists the publication out of the research work.

- "Dynamic Selection of Symmetric Key Cryptographic Algorithms for Securing Data Based on various Parameters". Published in Fifth International Conference on Communications Security & Information Assurance (CSIA 2014) May 24~25, 2014, Delhi, India. The conference was conducted by Academy & Industry Research Collaboration Center (AIRCC)
  web: http://airccse.org/.
- "Analysis and Comparison of Symmetric Key Cryptographic Algorithms based on Various File Features". Selected for publication in International Journal of Network Security & Its Applications (IJNSA), 2014.

Bibliography

[1] AL.Jeeva1, Dr.V.Palanisamy and K.Kanagaram, "Comparative Analysis of Performance Efficiency and Security Measures of Some Encryption Algorithms", International Journal of Engineering Research and Applications (IJERA), Vol. 2, Issue 3, May-Jun 2012, pp.3033-3037.

[2] S. Soni, H. Agrawal, M. Sharma, "Analysis and comparison between AES and DES Cryptographic Algorithm", International Journal of Engineering and Innovative Technology, Vol 2, Issue 6, December 2012, pp.362-365.

[3] Nidhi Singhal and J.P.S.Raina, "Comparative Analysis of AES and RC4 Algorithms for Better Utilization", International Journal of Computer Trends and Technology, Vol 2, Issue 6, July-Aug 2011, pp.177-181.

[4] Jawahar Thakur and Nagesh Kumar, "DES, AES and Blowfish: Symmetric Key Cryptography Algorithms Simulation Based Performance Analysis", International Journal of Emerging Technology and Advanced Engineering, Vol 1, Issue 2, December 2011, pp.6-12.

[5] Allam Mousa and Ahmad Hamad, "Evaluation of the RC4 Algorithm for Data Encryption", International Journal of Computer Science & Applications, Vol 3,Issue 2, June 2006, pp.44-56.

[6] Aamer Nadeem, Dr M. Younus Javed, "A Performance Comparison of Data Encryption Algorithms", First International Conference on IEEE Information and Communication Technologies (ICICT), 27-28 Aug. 2005, pp 84-89.

[7] Kofahi, N.A, Turki Al-Somani, Khalid Al-Zamil, “Performance evaluation of three Encryption/Decryption Algorithms”, IEEE 46th Midwest Symposium on Circuits and Systems, 30-30 Dec. 2003, pp. 790-793.

[8] Jonathan Knudsen, “Java Cryptography”, 2nd Edition, O’Reilly, 2011.

[9] William Stallings, “Cryptography and Network Security”, 5th Edition, Pearson, 2012.

[10] O.S. Pianykh, "Digital Imaging and Communications in Medicine (DICOM)", 1st Edition, Springer, 2008.

[11] John Miano, “Compressed Image File Formats”, 1st Edition, Addison Wesley Longman, Inc, 1999.

[12] Atul Kahate, “Cryptography and Network Security”, 2nd Edition, Tata McGraw Hill, 2012.

[13] V.K. Pachgare, “Cryptography and Information Security”, 1st Edition, PHI Learning PVT LTD, 2008.

[14] Dworkin, M. NIST Special Publication 800-38A. Recommendation for block cipher Modes of operation: Modes and techniques, Dec.2001.

[15] Simar Preet Singh and Raman Maini, “Comparison of Data Encryption Algorithms”, International Journal of Computer Science and Communication (IJCSC), Vol. 2, Issue 1, January-June 2011, pp. 125-127.

[16] Lalit Singh and R.K. Bharti, “Comparative Performance Analysis of Cryptographic Algorithms”, International Journal of Advanced Research in Computer Science and Software Engineering (IJARCSSE), Vol. 3, Issue 11, November 2013, pp. 563-568.

[17] Boqiang Liu, Minghui Zhu, Zhenwang Zhang, Cong Yin, Zhongguo Liu and Jason Gu, “Medical Image Conversion with DICOM”, IEEE Canadian Conference on Electrical and Computer Engineering (CCECE), 22-26 April 2007, pp. 36-39.

[18] S.R. Kodituwakku and U. S.Amarasinghe, "Comparison of Lossless Data Compression Algorithms for Text Data", Indian Journal of Computer Science and Engineering, Vol. 1, Issue 4, 2007, pp. 416-425.

[19] A. K. Bhattacharjee, T. Bej and S. Agarwal, "Comparison Study of Lossless Data Compression Algorithms for Text Data", IOSR Journal of Computer Engineering (IOSR-JCE), Vol. 11, Issue 6, June 2013, pp. 15-19.

[20] Shrusti Porwal, Yashi Chaudhary, Jitendra Joshi and Manish Jain, "Data Compression Methodologies for Lossless Data and Comparison between Algorithms", International Journal of Engineering Science and Innovative Technology (IJESIT) Vol. 2, Issue 2, March 2013, pp. 142-147.

[21] Li Yinfeng and Zhang Jian, "An Effective Lossless Compression Algorithm for Medical Image Set Based On Denoise Improved Mmp Method", IEEE 46th Midwest Symposium on Circuits and Systems, Vol 2, 27-30 Dec. 2003, pp. 729-732