HAWASSA UNIVERSITY

INSTITUTE OF TECHNOLOGY

FACULTY OF ELECTRICAL ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

EVALUATING CRYPTOGRAPHIC ALGORITHMS OF AUDIO AND VIDEO DATA FOR SYSTEM RESOURCE CONSTRAINED OF IoT DEVICES

BY

WAGARI BERHANU

ADVISOR: DR. ING. TWOFIK JEMAL (Ph.D.)

CO-ADVISOR: ZEKARIAS HAILU (M.Sc.)

A THESIS

SUBMITTED TO THE DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

IN

COMMUNICATION ENGINEERING AND NETWORKING

HAWASSA, ETHIOPIA

APRIL, 2019

HAWASSA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

EXAMINERS' APPROVAL SHEET

As members of the board of examiners of the final master's degree open defense. we certify that we have lead evaluated the thesis prepared by Wagari Berhanu under the title Evaluating cryptographic algorithms of audio and video data for system resource constrained of IoT devices and recommend that it be accepted as fulfilling the thesis requirement for the degree of M.Sc in Communication Engineering and Networking.

Name of chairperson	signature	date
Name of internal examiner	signature	date
Name of external examiner Towfik Jemal (Dr.-Ing., PhD)	signature	date 30-06-2019
Name of principal advisor	signature	date
Name of Co-advisor	signature	date

Final approval and acceptance of the contingent upon the submission of the final copy of the thesis to the SG and DGC/SGC of the candidate department

Thesis approved by

SGS	signature	date

Declaration

I undersigned solemnly declare that the thesis report Evaluating cryptographic algorithms of audio and video data for system resource constrained of IoT devices, is based on my own work carried out during the course of my study under the supervision of Dr.-Ing. Towfik J. and Co-advisor Zekariyas Hailu

I assert the statements made and conclusions drawn are an outcome of my thesis work. I further certify that

- ✓ I followed the guidelines provided by the university in writing the report.
- ✓ Whenever I have used materials (data, theoretical analysis, and text) from other sources, I have given due credit to them in the text of the report and giving their details in the references.
- ✓ The work contained in the report is original and has been done by me under the general supervision of my supervisor.
- ✓ The work has not been submitted to any other Institution for any other any certificate in this university or any other University of Ethiopian or abroad.

Name: Wagari Berhanu

Signature _____

Date: February 9, 2019

Name of Advisor: Dr.-Ing. Twofik Jemal

Signature _____

Date: February 9, 2019

Name of Co-Advisor: Zekaris Hailu

Signature _____

Date: February 9, 2019

Abstract

We are living in an internet period in which digital data exchange in electronic way has progressed tremendously and securing information has become a challenge. Unless a creative means of securing information has been developed, the problem will be vast and limitless. In a bid to keep the pace of the era, cryptography plays an important role in information security systems. Cryptography is a process of creating information indecipherable to an unauthorized person.

It is quite a well-known fact that IoT (Internet of Things) is a system of interrelated computing devices or digital machines is the ability to transfer data over a network. the devices that used in IoT is low power devices for this reason, the system resource is one sever issues.

To do so various cryptographic algorithms used in IoT devices. Meanwhile, these algorithms consume a significant amount of computing resources such as CPU time, memory and computation time. This thesis provides evaluating AES, Blowfish, RSA, Trivium and Elliptic Curve Cryptosystems based on system resource usage to securing IoT device communications on the basis of encryption time, decryption time, throughput, CPU time, power efficiency and memory usage in audio and video data using Java as the programming language to develop algorithms.

The result obtained shows that AES has better performance with encryption time, decryption time, throughput rate, CPU time, power efficiency and memory usage with audio and video files in various format.

Key Words: AES, Blowfish, RSA, Trivium, Elliptic Curve, IoT and Cryptography.

Acknowledgement

First I would like to thank heavenly God who helped me to come to the end of my thesis. And I express my deep sense of gratitude to my respected and learned guides, Dr.Ing. Twofik Jemal and Inst. **Zekarias Hailu** for their valuable help and guidance, thankful for the encouragement they give for me to completing this thesis. I also grateful to respected Inst. **Fistum Zerfu**, PG communication thesis coordinator for permitting us to utilize all the necessary facilities of the institution. I also thankful to all the other faculty & staff members of our department for their kind co-operation and help. Lastly, I would like to express my deep apperception towards my classmates and me in debt to my parents for providing me moral support and encouragement.

Wagari Berhanu

PGCom/028/09

Table of contents

List of Figures

List of tables

Chapter one

Introduction

Information security is a perplexing issue of data communications nowadays that reaches many parties including secure communication channel, strong data encryption technique and trusted third party to maintain the database. The conventional methods of encryption can only maintain the data security. Therefore, cryptography plays an important role to secure data transmitted using various communication channels [1], [5].

Cryptography is usually referred to as “the study of secret”. Cryptography provides a method for securing and authenticating the transmission of information across insecure communication channels. It enables us to store sensitive information or transmit it over insecure communication networks so that unauthorized persons cannot read it.

The main aim of cryptography is to keep data secure from unauthorized access by unintended users. In cryptography, the original message is usually refereed as plaintext. The technique of scrambling the plaintext in such a way that conceals its substance is called encryption. Encrypting plaintext makes the information in unreadable form called cipher text. The process of converting cipher text to its original information is called decryption. A system that performs encryption and decryption is called cryptosystem. On the basis of key used, cipher algorithms are classified as asymmetric key algorithms, in which encryption and decryption is done by two different keys and symmetric key algorithms, where the same key is used for encryption and decryption. Symmetric key algorithms are much faster computationally than asymmetric algorithms as the encryption process is less complicated. Asymmetric encryption techniques are almost 1000 times slower than Symmetric techniques, because they require more computational processing power. The focus of this thesis is on symmetric and asymmetric cryptography those algorithms are implemented in IoT device [22].

Cryptography is at the heart of a vast range of daily activities, such as electronic commerce, bankcard payments and electronic building access to name a few. It is one of the cornerstones of Internet security.

Internet of Things represents [32] a general concept for the ability of network devices to sense and collect data from the world around us, and then share that data across the Internet where it can be processed and utilized for various interesting purposes.

The Internet of Things (IoT) are doing a significant protagonist aspect of our daily lives. It covers many fields including healthcare, automobiles, entertainments, industrial appliances, sports, homes, etc. The pervasiveness of IoT eases some everyday activities, enriches the way people interact with the environment and surroundings, and augments our social interactions with other people and objects. IoT security as endeavor of safeguarding connected devices and networks in the Internet of things (IoT) [29].

Itis of key importance for modern, connected organizations to have an understanding of cryptography and its applications on IoT. When used appropriately, cryptography can play an important role in securing IoT environment [19]. But incorrect applications of the technology can lead to a false sense of security and can ultimately lead to the loss or disclosure of sensitive data.

1.1 Background

The word ‘cryptography’ was coined by combining two Greek words, ‘Krypto’ meaning hidden and ‘graphene’ meaning writing.

The art of cryptography is considered to be born along with the art of writing [1], [3]. As civilizations evolved, human beings got organized in tribes, groups, and kingdoms. This led to the emergence of ideas such as power, battles, supremacy, and politics. These ideas further fueled the natural need of people to communicate secretly with selective recipient which in turn ensured the continuous evolution of cryptography as well.

The roots of cryptography are found in Roman and Egyptian civilizations.

1.2 Hieroglyph the Oldest Cryptographic Technique

The first known evidence of cryptography can be traced to the use of ‘hieroglyph’. Some 4000 years ago, the Egyptians used to communicate by messages written in hieroglyph. This code was

the secret known only to the scribes who used to transmit messages on behalf of the kings. One such hieroglyph is shown below [6,27].

Figure 1.1: Hieroglyph [6].

Later, the scholars moved on to using simple mono-alphabetic substitution ciphers during 500 to 600 BC. This involved replacing alphabets of message with other alphabets with some secret rule. This **rule** became a **key** to retrieve the message back from the garbled message [2-6].

The earlier Roman method of cryptography, popularly known as the **Caesar Shift Cipher,** relies on shifting the letters of a message by an agreed number (three was a common choice), the recipient of this message would then shift the letters back by the same number and obtain the original message [23].

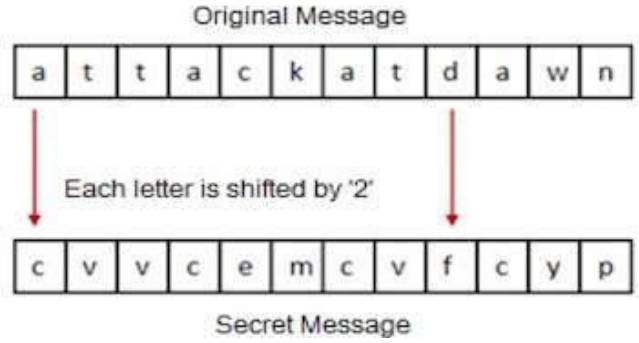


Figure 1.2: Caesar shift cipher [6].

1.2.1 Evolution of Cryptography

It is during and after the European Renaissance, various Italian and Papal states led the rapid proliferation of cryptographic techniques. Various analysis and attack techniques were researched in this era to break the secret codes [1,8,19,20,31].

- Improved coding techniques such as **Vigenere Coding** came into existence in the 15th century, which offered moving letters in the message with a number of variable places instead of moving them the same number of places.
- Only after the 19th century, cryptography evolved from the ad hoc approaches to encryption to the more sophisticated art and science of information security.
- In the early 20th century, the invention of mechanical and electromechanical machines, such as the **Enigma rotor machine,** provided more advanced and efficient means of coding the information.
- During the period of World War II, both **cryptography** and **cryptanalysis** became excessively mathematical.

With the advances taking place in this field, government organizations, military units, and some corporate houses started adopting the applications of cryptography. They used cryptography to guard their secrets from others. Now, the arrival of computers and the Internet has brought effective cryptography within the reach of common people [25,34].

1.3 Statement of the problem

We are living in reckless growing internet era where in digital data exchange is made widely in minute to minute bases. IoT giving a countless attribute for our world. As a result of this experience securing information in electronic device way has become a challenge. Hence, the role of cryptography in securing information is indispensable. Currently there are different cryptographic techniques used in IoT devices to encrypt and decrypt files so as to protect from unauthorized users. In IoT devices are low power, for this reason, the system resource is one sever issues. Besides such devices do not use algorithm's that need high resources to enhance security. audio and video data have great role in IoT industry to secure this data is the one of severe issue. On top of that, the existing comparative studies on this subject are not comprehensive Because, in the existing comparative studies, it is not clearly shown how the performance of the individual cryptography techniques for audio and video data. Moreover, some finding are not include all parameters that used in IoT device for audio and video data. As a result, this study attempts to provide the best alternative algorithms to optimize uses of system resource for IoT devices on audio and video data.

1.4 Objective

1.4.1 General objective

The general objective of this thesis is to make comparative analysis of AES, Blowfish, RSA, Trivium and Elliptic-Curve Cryptosystems and selecting efficient algorithms for securing IoT devices.

1.4.2 Specific objectives

- Encrypt the binary data and evaluate the time of encryption
- Decrypt the encrypted data and evaluate the time of decryption
- Evaluating the throughput of encryption and decryption
- Evaluating CPU time usage.
- Comparing power efficiency of the algorithms.
- Evaluating memory usage of the algorithms

1.5 Literature Review

In [8], the Improving the Security of Internet of Things Using Encryption Algorithms proposed a hybrid encryption algorithm which has been conducted in order to reduce safety risks and enhancing encryption's speed and less computational complexity. The result shows the algorithm uses less memory because of less fiscal complexity.

In the G. F. Pereira et. al. [6] they provided a detailed evaluation of symmetric cryptographic primitives providing different security services in relevant real-world platforms and operating systems, typical of IoT and WSN they provide time and energy benchmarks of reference implementations for different platforms and operating systems and analyze their experimental result show that AES uses less encryption time for small size and Trivium uses less encryption time for large size of text message data.

In the B. Nithya et. al. [36] as they implied symmetric cryptography with different size of text file DES has least encryption time and also it takes less memory for decryption but low throughput. TDES has high decryption time and also it uses more space to encrypt/decrypt. But TDES throughput is better than DES and RC4. RC4 uses less memory, high encryption/decryption time but low throughput. When compared to all algorithms AES has better throughput and it needs only less space for encryption/decryption process in .Net environment.

The author U. Kumar et. al. [7] as they implied on cryptography libraries WolfSSL, AvrCryptoLib, WiseLib, TinyECC and RelicToolKit they conclude that considered as all library due to their varying features and optimizations made for different specific platforms.

M. A. Razzaq et. al. [8] they show that on twelve different types of attacks are categorized as low-level attacks, medium-level attacks, high-level attacks, and extremely high-level attacks along with their nature/behavior as well as suggested solutions to encounter these attacks they addressed.

Z. Wang et. al. [9] they propose control transfer protocol to enable common portal devices to control large-volume IoT devices. The protocol leverages lightweight hash functions to achieve secure and efficient control transfer among resource limited IoT devices. They analyze the

privacy and security guarantee their protocol. They also conduct simulations over real IoT devices to evaluate the performance.

S. K. Kim et. al. [10] they proposed a solution to reduce overheads causing unnecessary connection delay in every initial connection to service by completing the preparation of mutual authentication data and session key while on proxy preparation task. Through the improvement of proxy preparation, they obtained experimental results that could reduce connection delay by about 2.6 times, in proportion to the size of messages. Their work shows that an approach to reduce the additional overhead that would be paid for by applying the Jini security framework to the development of sensor application based on Jini IoT.

In [11] as implied on Security the performance analysis of execution time of different algorithms in terms of Encryption time and Decryption time with different sizes of text files and image files. The results show that AES algorithm is the best and takes less time to encrypt and decrypt a file (Text or Image) as compared to other algorithms (Blowfish, DES and Triple DES). After AES, Blowfish algorithm performs better as compared to the DES and Triple DES.

In [12] analyzed that the performance of existing encryption techniques like DES, AES and Blowfish algorithms. Based on the image files used and the experimental result it was concluded that Blowfish algorithm consumes least encryption time and DES consume maximum encryption time. They also observed that Decryption of Blowfish and AES algorithms is better than DES algorithm. From their research work, they concluded that Blowfish algorithm is better than AES and DES algorithms.

B.Bharathi et. al. [13] presents various evaluation techniques and performance metrics that can be used to test any cryptographic algorithms. Their simulation results shows that SF Block Cipher has better performance than Blowfish almost all the test cases. They also identified that there is change in performance when there is a change in key size of SF Block cipher algorithm. Overall it is identified that SF can be used in circumstances where there is need for high security.

F. Maqsood et. al. [14] they used encryptiontime, decryption time and key generation time to evaluate the cryptographic schemes. The performance results show that the symmetric schemes are computationally inexpensive when compared with asymmetric schemes. The key generation time is depending on the key length of bits.

In [15] Provides evaluation of both symmetric (AES, DES, Blowfish) as well as asymmetric (RSA) cryptographic algorithms by taking different types of files like binary, text and image files. A comparison has been conducted for these encryption algorithms using evaluation parameters such as encryption time, decryption time and throughput. From the presented simulation results, it was concluded that AES has better performance than other algorithms in terms of both throughput and encryption decryption time.

1.6 Scope of the thesis

On this thesis analyzed and compared AES, Blowfish, RSA, Trivium and Elliptic-Curve Cryptosystems algorithms based on system resource usage for securing IoT environment for video and audio data on the basis of encryption time, decryption time, throughput encryption rate, throughput decryption rate, CPU time, power efficiency and memory usage cryptographic algorithms for different sizes of audio and video data alongside various format types.

1.7 Limitation of the thesis

- ✓ Does not consider the effect of the channel beyond this did not consider other parameters and algorithms.

1.8 Methodology

Generally, the methodology mainly consists of the following tasks:

- ✓ First, reviews of different research papers which are closely related to this thesis work performed.
- ✓ After reviewing relevant literatures, theoretical explanation about the algorithms.
- ✓ Then, model techniques, such as AES, Blowfish, RSA, Trivium and ECC using the respective detailed block diagrams.
- ✓ The performance evaluation done by simulating these techniques in terms of Encryption time, Decryption time, Throughput, Memory usage, CPU usage and Power efficiency using java software [36].
- ✓ Finally, the performance analysis of the simulation results of the Encryption time, Decryption time, Throughput, Memory usage, CPU usage and Power efficiency of all algorithms discussed in detail.

Tools of writing documents & soft wares

- ✓ Development → JDK 8.2
- ✓ Documentation→MS OFFICE 2013 and Latex
- ✓ Flow chart Design-→ Edraw max

1.9 Thesis outline

The rest of the thesis is organized as follows:

Chapter 2 gives an overview of selected algorithms for evaluation according with their characteristics and behavior.

Chapter 3 describe the system model of selected algorithms in detail.

Chapter 4 focuses on the simulation result and discussion of selected algorithms in the parameters of encryption time, decryption time, throughput, memory usage, power efficiency and CPU usage of different size of video and audio data's in various formats discussed in this chapter.

Chapter 5 concludes this thesis with some limitation and future research scopes.

References: The list of literature which has been cited in this study is given in this part of the thesis document.

Chapter two

Selected algorithms for evaluation.

2.1 Advanced Encryption Standard (AES)

AES is a block cipher with a block length of 128 bits. AES allows for three different key lengths: 128, 192, or 256 bits. Encryption consists of 10 rounds of processing for 128-bit keys, 12 rounds for 192-bit keys, and 14 rounds for 256-bit keys. Except for the last round in each case, all other rounds are identical Each round of processing includes one single-byte based substitution step, a row-wise permutation step, a column-wise mixing step, and the addition of the round key. The order in which these four steps are executed is different for encryption and decryption [13,18,25,33].

To appreciate the processing steps used in a single round, it is best to think of a 128-bit block as consisting of a 4 × 4 matrix of bytes, arranged as follows:

$$\begin{bmatrix} byte_0 & byte_4 & byte_8 & byte_{12} \\ byte_1 & byte_5 & byte_9 & byte_{13} \\ byte_2 & byte_6 & byte_{10} & byte_{14} \\ byte_3 & byte_7 & byte_{11} & byte_{15} \end{bmatrix}$$

Therefore, the first four bytes of a 128-bit input block occupy the first column in the 4 × 4 matrix of bytes. The next four bytes occupy the second column, and so on.

Whereas AES requires the block size to be 128 bits, the original Rijndael cipher works with any block size (and any key size) that is a multiple of 32 as long as it exceeds 128. The state array for the different block sizes still has only four rows in the Rijndael cipher. However, the number of columns depends on size of the block. For example, when the block size is 192, the Rijndael cipher requires a state array to consist of 4 rows and 6 columns [22].

Assuming a 128-bit key, the key is also arranged in the form of a matrix of 4 × 4 bytes. As with the input block, the first word from the key fills the first column of the matrix, and so on. The four column words of the key matrix are expanded into a schedule of 44 words. Each round consumes four words from the key schedule.

Figure 2.1 on the depicts the arrangement of the encryption key in the form of 4-byte words and the expansion of the key into a key schedule consisting of 44 4-byte words[19].

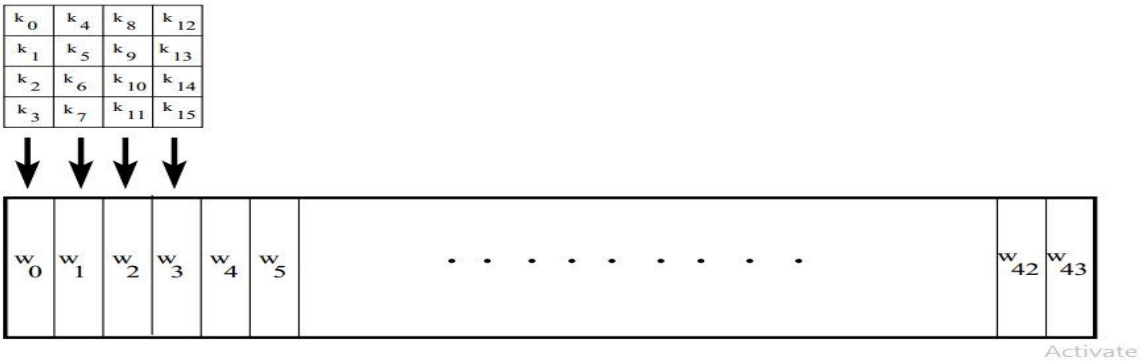


Figure 2.1: the four words of the original 128-bit key being expanded into a key schedule consisting of 44 words [2].

2.1.1 The Structure of AES

Before explanation of the structure the figure shows overall about encryption and decryption life cycle.

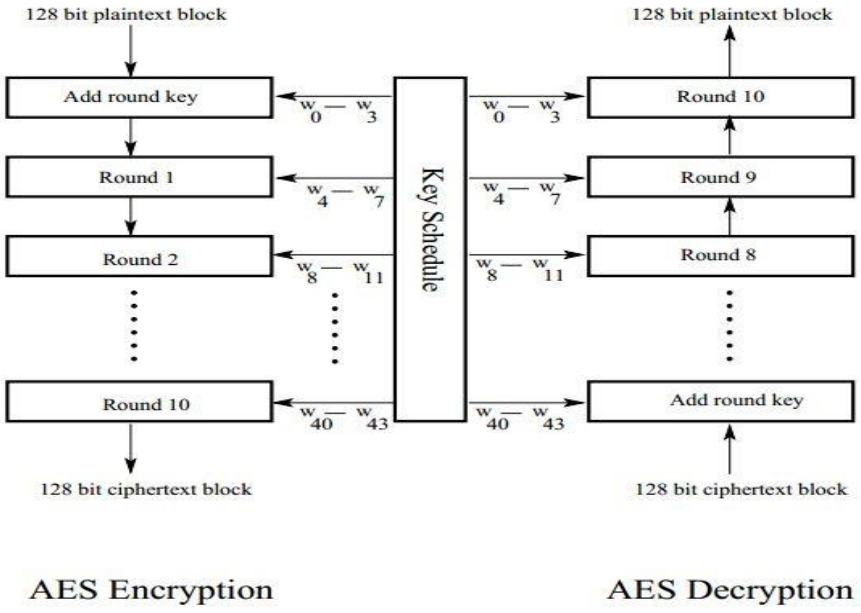


Figure 2.2: Structure of AES encryption/decryption [30].

The number of rounds shown in Figure 2, 10, is for the case when the encryption key is 128 bit long. (As mentioned earlier, the number of rounds is 12 when the key is 192 bits, and 14 when the key is 256.) Before any round-based processing for encryption can begin, the input state array is XORed with the first four words of the key schedule. The same thing happens during decryption except that now we XOR the cipher text state array with the last four words of the key-schedule [29].

For encryption, each round consists of the following four steps:

i. Substitute bytes
ii. Shift rows
iii. Mix columns
iv. Add round key.

The last step consists of XORing the output of the previous three steps with four words from the key schedule.

For decryption, each round consists of the following four steps:

i. Inverse shift rows
ii. Inverse substitute bytes
iii. Add round key
iv. Inverse mix columns.

The third step consists of XORing the output of the previous two steps with four words from the key schedule.

2.1.2 The Four Steps in Each Round of Processing

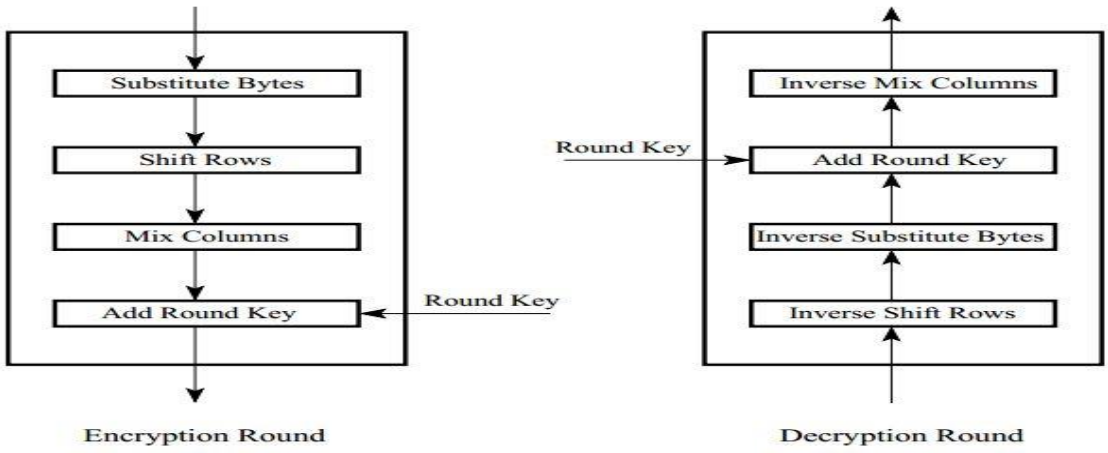


Figure 2.3: The one step round four encryption as well as decryption [2].

2.1.3 Substitute Bytes

2.1.3.1 SubBytes and InvSubBytes

Also called Sub Bytes for byte-by-byte substitution during the forward process. The corresponding substitution step used during decryption is called InvSubBytes. This step consists of using a 16 × 16 lookup table to find a replacement byte for a given byte in the input state array [2,29,30,31].

This is a byte-by-byte substitution using a rule that stays the same in all encryption rounds. The byte-by-byte substitution rule is different for the decryption chain, but again it stays the same for all the rounds.To explaining the byte substitution step that allows us to find the substitute byte for a given byte by simply looking up a pre-computed 256-element array of numbers.

In the modern way of explaining the byte substitution step for the encryption chain, let xin be a byte of the state array for which we seek a substitute byte xout. We can write xout = f(xin). The function f(xin) involves two nonlinear operations:

i. First find the multiplicative inverse x′ = xin-1 in GF (28);

ii. Then we scramble the bits of x′ by XORing x′ with four different circularly rotated versions of itself and with a special constant byte c = 0x63. The four circular rotations are through 4, 5,6, and 7 bit positions to the right.

The modern explanation of the byte substitution step as presented above applies equally well to the decryption chain, except for the fact that you first apply the bit scrambling operation to the byte and then you find its multiplicative inverse in GF (28).

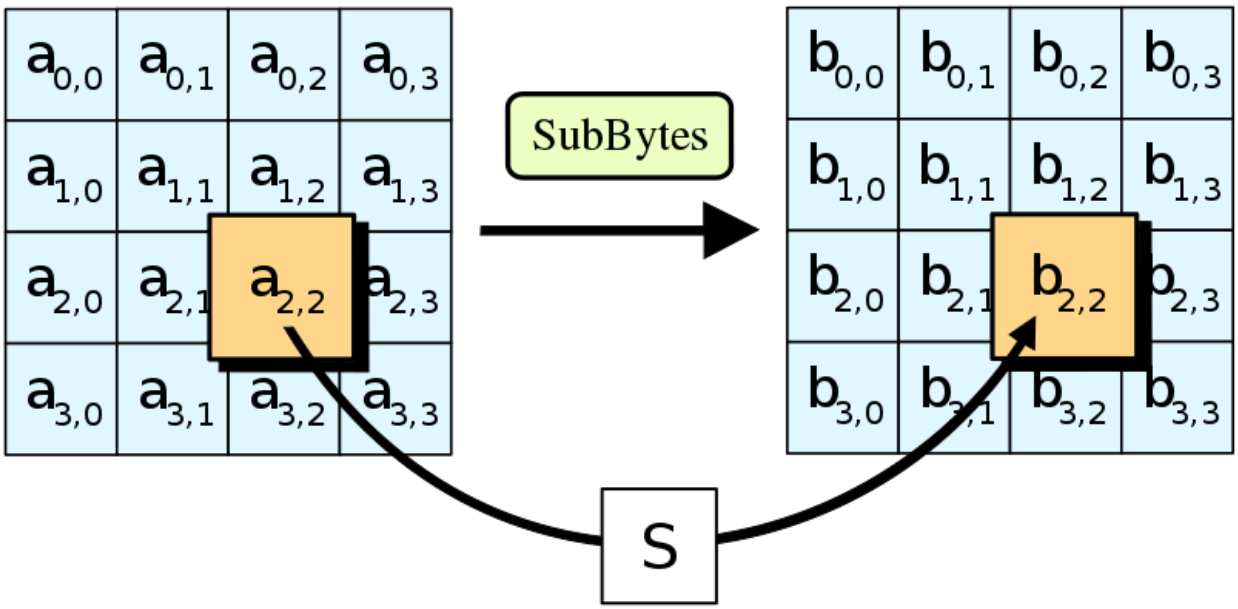


Figure 2.4: Sub bytes step, one of four stages in a round of AES [2].

2.1.4 Shift Rows Step

2.1.4.1 Add Round Key

Used for adding the round key to the output of the previous step during the forward process. The corresponding step during decryption is denoted InvAddRoundKey for inverse add round key transformation**.**

Each round has its own round key that is derived from the original 128-bit encryption key in the manner described in this section. One of the four steps of each round, for both encryption and decryption, involves XORing of the round key with the state array.The AES Key Expansion algorithm is used to derive the 128-bit round key for each round from the original 128-bit

encryption key. As you'll see, the logic of the key expansion algorithm is designed to ensure that if you change one bit of the encryption key, it should affect the round keys for several rounds. In the same manner as the 128-bit input block is arranged in the form of a state array, the algorithm first arranges the 16 bytes of the encryption key in the form of a 4 ×4 array of bytes [28].

k0	k4	k8	k12
k1	k5	k9	k13
k2	k6	k10	k14
k3	k7	k11	k15

⇓

[w0 w1 w2 w3]

The first four bytes of the encryption key constitute the word w0,the next four bytes the word w1, and so on.

The algorithm subsequently expands the words [w0, w1, w2, w3] into a 44-word key schedule that can be labeled w0, w1, w2, w3, ..., w43

Of these, the words [w0, w1, w2, w3] are bitwise XOR'ed with the input block before the round-based processing begins. The remaining 40 words of the key schedule are used four words at a time in each of the 10 rounds.

For decryption

The above two statements are also true for decryption, except for the fact that we now reverse the order of the words in the key schedule. The last four words of the key schedule are bitwise XOR'ed with the 128-bit cipher textblock before any round-based processing begins. Subsequently, each of the four words in the remaining 40 words of the key schedule are used in each of the ten rounds of processing [20].

2.1.5 ShiftRows and InvShiftRows

Is used for shifting the rows of the state array during the forward process. The corresponding transformation during decryption is denoted InvShiftRows for Inverse ShiftRow Transformation. The goal of this transformation is to scramble the byte order inside each 128-bit block [33].

This is where the matrix representation of the state array becomes important. The ShiftRows transformation consists of

i. not shifting the first row of the state array at all
ii. circularly shifting the second row by one byte to the left
iii. circularly shifting the third row by two bytes to the left
iv. Circularly shifting the last row by three bytes to the left.

This operation on the state array can be represented by

S0.0	S0,1	S0,2	S0,3
S1.0	S1,1	S1,2	S1,3
S2.0	S2,1	S2,2	S2,3
S3.0	S3,1	S3,2	S3,3

S0.0	S0,1	S0,2	S0,3
S1.1	S1,2	S1,3	S1,0
S2.2	S2,3	S2,0	S2,1
S3.3	S3,0	S3,1	S3,2

Recall again that the input block is written column-wise. That is the first four bytes of the input block fill the first column of the state array, the next four bytes the second column, etc. As a result, shifting the rows in the manner indicated scrambles up the byte order of the input block.

For decryption

- The corresponding step shifts the rows in exactly the opposite fashion. The first row is left unchanged, the second row is shifted to the right by one byte, the third row to the right by two bytes, and the last row to the right by three bytes, all shifts being circular.

-

S0.0	S0,1	S0,2	S0,3
S1.0	S1,1	S1,2	S1,3
S2.0	S2,1	S2,2	S2,3
S3.0	S3,1	S3,2	S3,3

S0.0	S0,1	S0,2	S0,3
S1.3	S1,0	S1,1	S1,2
S2.2	S2,3	S2,0	S2,1
S3.1	S3,2	S3,3	S3,0

2.1.6 Mix Columns Step

2.1.6.1 MixColumns and InvMixColumns

Is used for mixing up of the bytes in each column separately during the forward process. The corresponding transformation during decryption is denoted InvMixColumns and stands for inverse mix column transformation [21].

The goal is here being to further scramble up the 128-bit input block. The shift-rows step along with the mix-column step causes each bit of the cipher text to depend on every bit of the plaintext after 10 rounds of processing.

This step replaces each byte of a column by a function of all the bytes in the same column. More precisely, each byte in a column is replaced by two times that byte, plus three times the next byte, plus the byte that comes next, plus the byte that follows. The words 'next' and 'follow' refer to bytes in the same column, and their meaning is circular, in the sense that the byte that is next to the one in the last row is the one in the first row.

For the bytes in the first row of the state array, this operation can be stated as $s'0,j = (0x02 \times s0,j) \otimes (0x03 \times s1,j) \otimes s2,j \otimes s3,j$

For the bytes in the second row of the state array, this operation can be stated as $s'1,j = s0,j \otimes (0x02 \times s1,j) \otimes (0x03 \times s2,j) \otimes s3,j$

For the bytes in the third row of the state array, this operation can be stated as $s'2,j = s0,j \otimes s1,j \otimes (0x02 \times s2,j) \otimes (0x03 \times s3,j)$
And, for the bytes in the fourth row of the state array, this operation can be stated as $s'3,j = (0x03 \times s0,j) \otimes s1,j \otimes s2,j \otimes (0x02 \times s3,j)$

- More compactly, the column operations can be shown

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \times \begin{bmatrix} S0.0 & S0,1 & S0,2 & S0,3 \\ S1.0 & S1,1 & S1,2 & S1,3 \\ S2.0 & S2,1 & S2,2 & S2,3 \\ S3.0 & S3,1 & S3,2 & S3,3 \end{bmatrix} = \begin{bmatrix} S'0.0 & S'0,1 & S'0,2 & S'0,3 \\ S'1.0 & S'1,1 & S'1,2 & S'1,3 \\ S'2.0 & S'2,1 & S'2,2 & S'2,3 \\ S'3.0 & S'3,1 & S'3,2 & S'3,3 \end{bmatrix}$$

For Decryption

Where, on the left hand side, when a row of the leftmost matrix multiples a column of the state array matrix, additions involved are meant to be XOR operations.

The corresponding transformation during decryption is given by

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \times \begin{bmatrix} S0.0 & S0,1 & S0,2 & S0,3 \\ S1.0 & S1,1 & S1,2 & S1,3 \\ S2.0 & S2,1 & S2,2 & S2,3 \\ S3.0 & S3,1 & S3,2 & S3,3 \end{bmatrix} = \begin{bmatrix} S'0.0 & S' 0,1 & S' 0,2 & S' 0,3 \\ S'1.0 & S' 1,1 & S' 1,2 & S' 1,3 \\ S'2.0 & S' 2,1 & S' 2,2 & S' 2,3 \\ S'3.0 & S' 3,1 & S' 3,2 & S' 3,3 \end{bmatrix}$$

2.2 Blowfish

Blowfish is a keyed, symmetric cryptographic block cipher designed [29] by Bruce Schneier in 1993 and placed in the public domain. Blowfish is included in a large number of cipher suites and encryption products, including SplashID. Blowfish's security has been extensively tested and proven. As a public domain cipher, Blowfish has been subject to a significant amount of cryptanalysis, and full Blowfish encryption has never been broken. Blowfish is also one of the fastest block ciphers in public use, making it ideal for a product like SplashID that functions on a wide variety of processors found in mobile phones as well as in notebook and desktop computers.Notable features of the design include key-dependent S-boxes and a highly complex key schedule [27].

2.2.1 How it works Blowfish algorithm

Blowfish has a 64-bit block size and a key length of anywhere from 32 bits to 448 bits. It is a 16-round Feistel cipher and uses large key-dependent S-boxes. It is similar in structure to CAST-128, which uses fixed S-boxes.

The diagram to the below shows the action of Blowfish. Each line represents 32 bits. The algorithm keeps two subkey arrays: the 18-entry P-array and four 256-entry S-boxes. The S-boxes accept 8-bit input and produce 32-bit output. One entry of the P-array is used every round, and after the final round, each half of the data block is XORed with one of the two remaining unused P-entries.

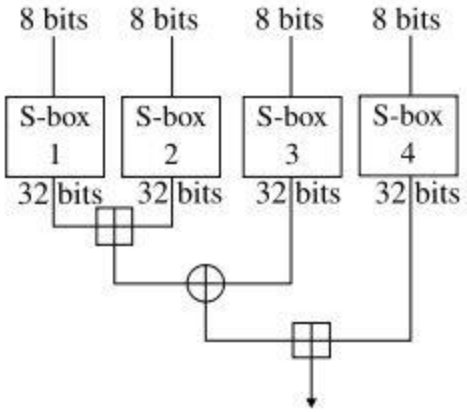


Figure 2.5: Feistel structure of blowfish [2].

The diagram to the right shows Blowfish's F-function. The function splits the 32-bit input into four eight-bit quarters, and uses the quarters as input to the S-boxes. The outputs are added modulo 232 and XORed to produce the final 32-bit output.Since Blowfish is a Feistel network, it can be inverted simply by XORing P17 and P18 to the cipher text block, then using the P-entries in reverse order. Blowfish's key schedule starts by initializing the P-array and S-boxes with values derived from the hexadecimal digits of pi, which contain no obvious pattern. The secret key is then XORed with the P-entries in order (cycling the key if necessary). A 64-bit all-zero block is then encrypted with the algorithm as it stands. The resultant cipher text replaces P1 and P2. The cipher text is then encrypted again with the new subkeys, and P3 and P4 are replaced by the new cipher text. This continues, replacing the entire P-array and all the S-box entries. In all, the Blowfish encryption algorithm run 521 times to generate all the subkeys - about 4KB of data is processed [21].

2.3 RSA

RSA [26] is a cryptosystem for public-key encryption, and is widely used for securing sensitive data, particularly when being sent over an insecure network such as the Internet.

RSA was first described in 1977 by Ron Rivest, Adi Shamir and Leonard Adleman of the Massachusetts Institute of Technology. Public-key cryptography. In RSA cryptography, both the public and the private keys can encrypt a message; the opposite key from the one used to encrypt a message is used to decrypt it. This attribute is one reason why RSA has become the most widely used asymmetric algorithm: It provides a method of assuring the confidentiality, integrity, authenticity and non-reputability of electronic communications and data storage.

RSA derives its security from the difficulty of factoring large integers that are the product of two large prime numbers. Multiplying these two numbers is easy, but determining the original prime numbers from the total -- factoring -- is considered infeasible due to the time it would take even using today's super computers.

The public and the private key-generation algorithm is the most complex part of RSA cryptography. Two large prime numbers, p and q, are generated using the Rabin-Miller primality test algorithm. A modulus n is calculated by multiplying p and q. This number is used by both the public and private keys and provides the link between them. Its length, usually expressed in bits, is called the key length. The public key consists of the modulus n,and a public exponent, e, which is normally set at 65537, as it's a prime number that is not too large. The e figure doesn't have to be a secretly selected prime number as the public key is shared with everyone. The private key consists of the modulus n and the private exponent d, which is calculated using the Extended Euclidean algorithm to find the multiplicative inverse with respect to the totient [22].

2.3.1 Security of RSA

The security of RSA relies on the computational difficulty of factoring large integers. As computing power increases and more efficient factoring algorithms are discovered, the ability to factor larger and larger numbers also increases. Encryption strength is directly tied to key size, and doubling key length delivers an exponential increase in strength, although it does impair

performance. RSA keys are typically 1024- or 2048-bits long, but experts believe that 1024-bit keys could be broken in the near future, which is why government and industry are moving to a minimum key length of 2048-bits [22].

2.3.2 RSA Algorithm Structure

.

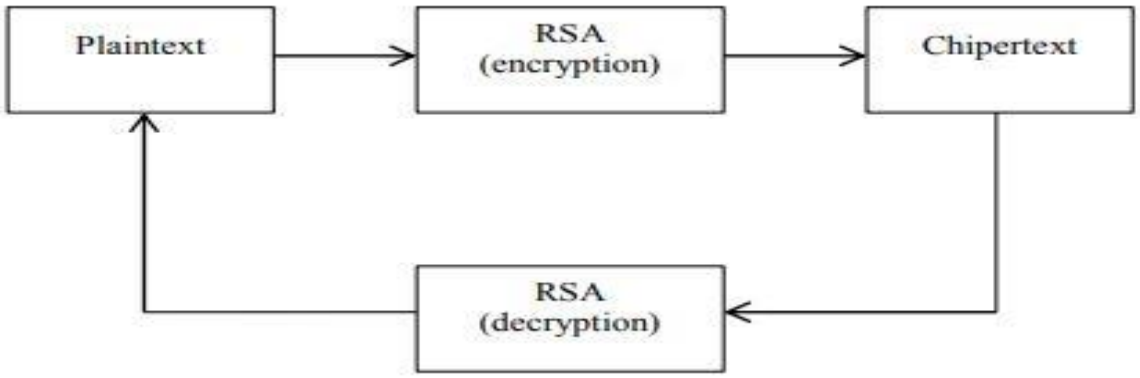


Figure 2.6: Process of RSA algorithm method [2].

Encryption converts the original data (plaintext) to the encoded data (cipher text), while the decryption is returning the cipher text to the plaintext. RSA algorithm is divided into 3 steps:

1. Key Generators

   a. Select 2 large primes for p and q
   b. Calculate the modulus value n = px q (1)
   c. Calculate using the Euler function φ (n) = (p 1) x (q-1) (2)
   d. Select the random integer e as a public key, provided that it meets the Greater Common Divisor (e, φ (n)) = 1, 1<e< φ(n)
   e. Calculate private key d so dxe = 1 (mod φ (n))

2. Encryption C = M e mod n (5)

3. Decryption M = C d mod n (6)

2.4 Trivium

Trivium is a synchronous stream cipher designed to generate up to 2^{64} bits of key stream from an 80-bit secret key and an 80-bit initial value (IV). As for most stream ciphers, this process consists of two phase, first the internal state of the cipher is initialized using the key and the IV, then the state is repeatedly updated and used to generate key stream bits [28].

2.4.1 Key stream generation

The proposed design contains a 288-bit internal state denoted by (s1 ……...s288). The key stream generation consists of an iterative process which extracts the values of 15 specific state bits and uses them both to update 3 bits of the state and to compute 1 bit of key stream z_i. The state bits are then rotated and the process repeats itself until the requested $N \leq 2^{64}$ bits of key stream have been generated. A complete description is given by the following simple pseudo-code:

for i = 1 to N do

t1←s66 + s93

t2 ← s162 + s177

t3← s243 + s288

zi← t1 + t2 + t3

t1← t1 + s91 * s92 + s171

t2← t2 + s175 * s176 + s264

t3 ← t3 + s286 * s287 + s69

(s1, s2,,,,,,,s93) ← (t3, s1,,,,,,, s92)

(s94, s95,,,,,,, s177) ← (t1 ,s4,,,,,,, s176)

(s178, s279,,,,, s288) ← (t2, s178,,,,,,s287)

end for

Note that here, and in the rest of this document, the `+' and `*' operations stand for addition and multiplication over GF(2) (i.e., XOR and AND), respectively [23].

2.4.2 Key and IV setup

The [26] algorithm is initialized by loading an 80-bit key and an 80-bit IV into the 288-bit initial state, and setting all remaining bits to 0, except for s286, s287, and s288 . Then, the state is rotated over 4 full cycles, in the same way as explained above, but without generating key stream bits. This is summarized in the pseudo-code below

(s1, s2,,,,,,,s93) ← (K1,,,,, K80, 0,,,,,,, 0)

(s94, s95,,,,,,,, s177) ← (IV1,,,,,,IV80, 0,,,,,,, 0)

(s178, s279,,,,, s288) ← (0,,,,,,, 0,1,1,1)

for i = 1 to 4 * 288 do

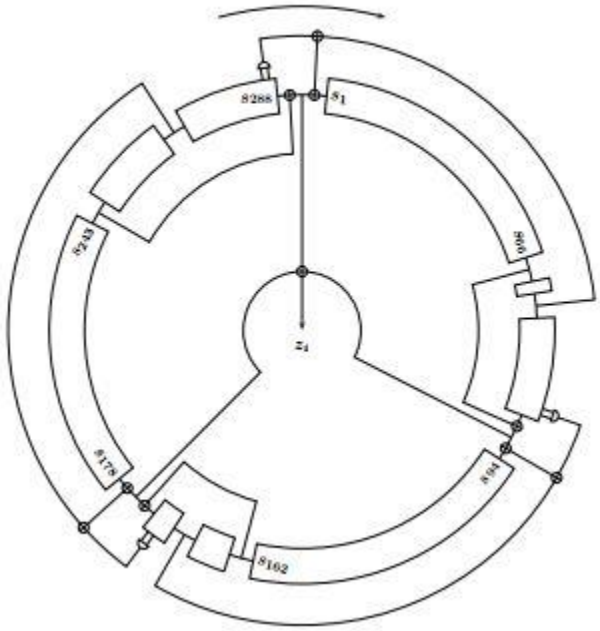


Figure 2.7: Trivium [31].

2.5 Elliptic curve

Elliptic curve cryptography (ECC) is a public key encryption technique based on elliptic curve theory that can be used to create faster, smaller, and more efficient cryptographic keys. ECC generates keys through the properties of the elliptic curve equation instead of the traditional method of generation as the product of very large prime numbers. The technology can be used in conjunction with most public key encryption methods, such as RSA, and Diffie-Hellman. According to some researchers, ECC can yield a level of security with a 164-bit key that other systems require a 1,024-bit key to achieve. Because ECC helps to establish equivalent security with lower computing power and battery resource usage. ECC was developed by Certicom, a mobile e-business security provider, and was recently licensed by Hifn, a manufacturer of integrated circuitry (IC) and network security products [2,29,33].

2.5.1 Elliptic Curve Encryption/Decryption

Several approaches to encryption/decryption using elliptic curves have been analyzed in the literature. In this subsection we look at perhaps the simplest. The first task in this system is to encode the plaintext message m to be sent as an x-y point Pm. It is the point Pm that will be encrypted as a cipher text and subsequently decrypted.

Note that we cannot simply encode the message as the x or y coordinate of a point, because not all such coordinates are in Eq(a, b); Again, there are several approaches to this encoding, which we will not address here, but suffice it to say that there are relatively straightforward techniques that can be used.As with the key exchange system, an encryption/decryption system requires a point G and an elliptic group Eq(a, b) as parameters [2].

Each user A selects a private key n A and generates a public key PA = nA xG. To encrypt and send a message Pm to B, A chooses a random positive integer k and produces the cipher text Cm consisting of the pair of points:Cm= {kG, Pm+ kPB} Note that A has used B's public key PB. To decrypt the ciphertext, B multiplies the first point in the pair by B's secret key and subtracts the result from the second point: Pm+ kP B nB(kG) = Pm + k(nBG) nB(kG) = Pm A has masked the message Pm by adding kP B to it. Nobody but A knows the value of k, so even though PB is a public key, nobody can remove the mask kP B.

However, A also includes a "clue," which is enough to remove the mask if one knows the private key n B. For an attacker to recover the message, the attacker would have to compute k given G and kG, which is assumed hard [24].

Chapter three

System model

In this thesis design for symmetric and asymmetric algorithms which they have the different implementation. The symmetric side use single secrete key and asymmetric uses the public and private key.

3.1 In Transmitter Side

The first phase is input Audio/Video data which is a user can enter to system for encryption. audio/video file is analog therefore the compression and converting techniques include all phase of integer transformation, quantization and Entropy coding finally digital forms which is covert to hexadecimal values and then covert to binary finally the binary data use for the encryption by using one common key which they exchange in secure channel and transmit the data in insecure channel.

For asymmetric the transmitter uses the public key of receiver to encrypt the data.

3.2 In Receiver Side

the encrypted data receive from insecure channel then decryption done by using the common key that send from the transmitter side then inverse compression done finally converts digital data to analog the finally the receiver uses the secure Audio/video data. For asymmetric the receiver uses the private key to decrypt the data. The Encryption and decryption done in binary data's.

The following figure in the next page describes the system model.

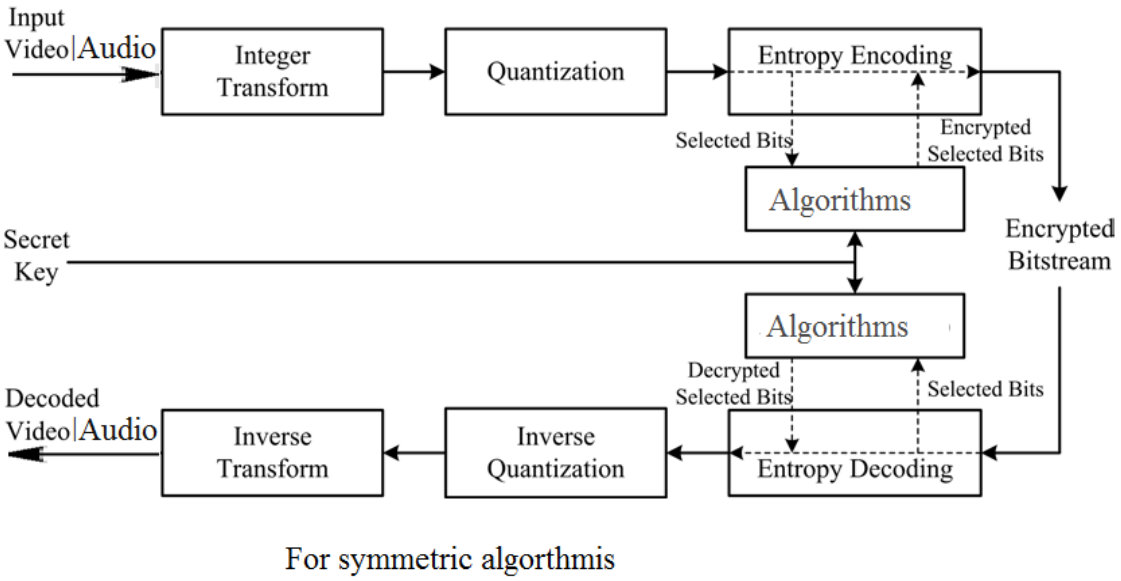


Figure 3.1: System model for symmetric

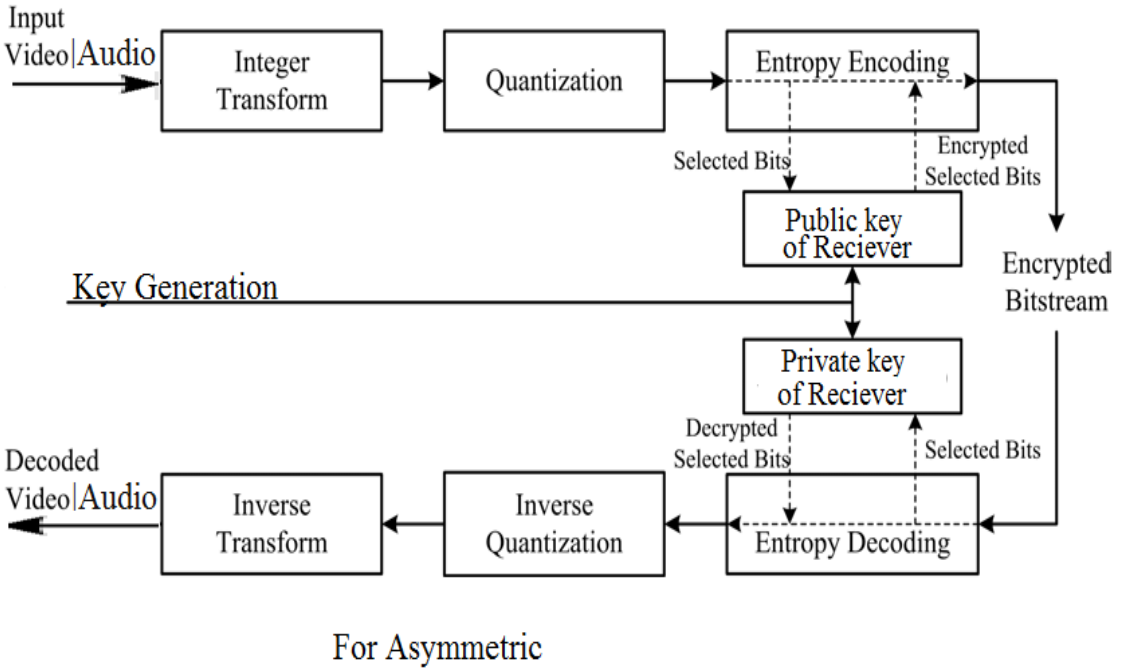


Figure 3.2: System model for asymmetric

Chapter four

Results and Discussions

4.1. Introduction

Performance measured in encryption time, decryption time, throughput encryption rate, throughput decryption rate, CPU time, power efficiency and memory usage algorithm is calculated in view of the following machine performance.

Software Environment: Experimental evaluation on Java Development Kit 8.2, Windows 8.1 Pro64 bit Operating System.

Hardware Specification: All the algorithms are tested on Intel(R) Core™ i3-4005U @1.7GHz processor with 4GB of RAM with 500GB.

IoT devices are low-power devices, according to this the parameters we used to evaluate the performance of the algorithms [9]. In this thesis the key length used for analysis the algorithm's is AES is 128 bit, Blowfish is 128 bit, Trivium key is 80 bit, RSA is 1024 bit and ECC is 224 bit. The analysis done by using the following metrics under which the cryptosystems are compared.

- ✓ **Encryption time**- The time required to convert plaintext to cipher text is encryption time. Measured in time of starting and ending difference. EncTime=EndTime-StartTime.
- ✓ **Decryption time**- The time to recover plaintext from cipher text is called decryption time. Measured in time of starting and ending difference. DecTime=EndTime-StartTime.
- ✓ **Throughput of Encryption and Decryption-** The throughput of the encryption scheme defines the speed of encryption.

$$\text{throughput encryption} = \frac{\text{total plain text}}{Encryption\ time} = \frac{Tp}{Et}$$

- ✓ **Power efficiency-** is measured in power usage of algorithms during execution.
- ✓ **CPU time-** is measured in clock ticks or seconds. time reflects the load of the CPU and this load depends on the CPU time used in the encryption/decryption process.
- ✓ **Memory usage-** is measured in memory used during execution.

  find the space that each algorithm takes to encrypt/decrypt is calculated as taking the measurement of memory before encryption/decryption processes and after the encryption/decryption processes. The result will be subtracted from before to after usage of memory in kilo bytes.

4.2 Analysis of Results

This section shows the results which obtained by running the simulation program using different audio/video data loads. The results show the impact of changing data load on each algorithm and the impact used for analysis encryption time, decryption time, throughput, memory usage, power efficiency and CPU usage to measure the performance of the algorithms finally suggesting best algorithms for IoT devices.

- ✓ To calculate the encryption time, decryption time and CPU usage is measured in millisecond
- ✓ throughput is measured in byte per millisecond
- ✓ memory usage measured in kilobyte
- ✓ power efficiency is measured in Joule/Ms

The testing is done by classifying data on different size and various formats

i. The same file format but different size to evaluate the encryption time selection random select 6 video which the various size [37].

Table 4.1: Encryption time for different video file.

The size in MB in MP4 format	AES Encryption time (ms)	Blowfish Encryption time. (ms)	RSA Encryption time. (ms)	ECC Encryption time. (ms)	Trivium Encryption time. (ms)
1	5	60	1807	179385	11
6.23	62	141	6918	1220133	67
15.2	125	313	21867	1478212	137
19.6	168	438	25206	1957535	140
42.7	219	1045	44137	4156687	328
127.2	426	3161	143462	6879221	860

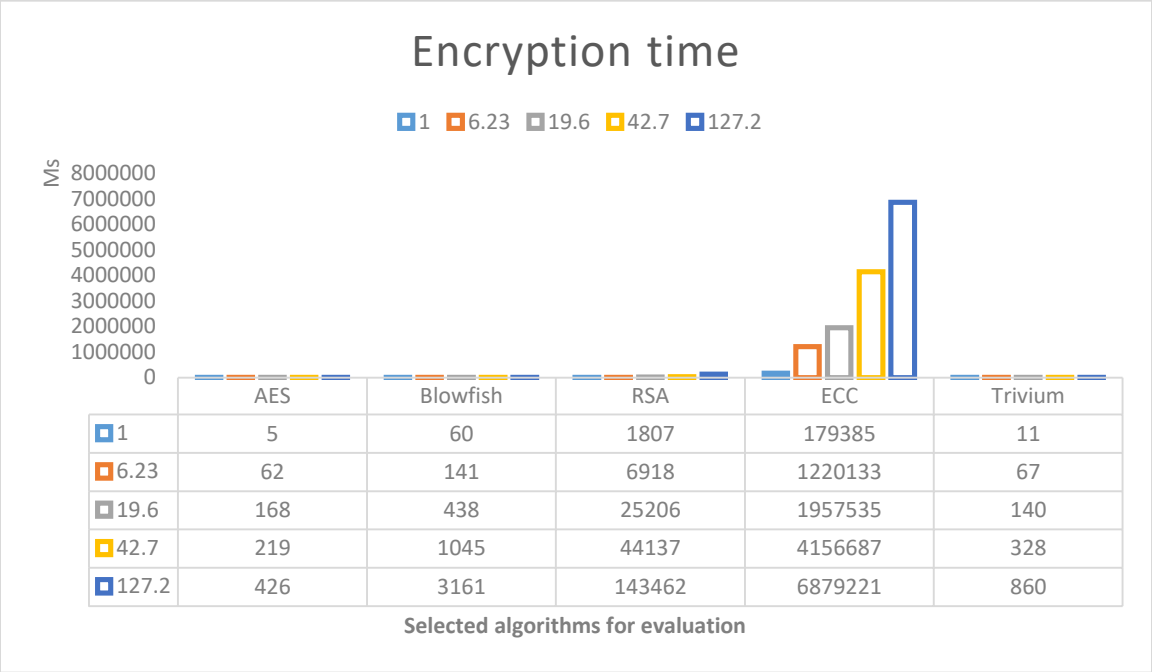


Figure 4.1: Encryption time same type and different video file

The simulation shows that in the encryption time for 1, 6.23, 19.6, 42.7, 127 megabyte MP4 video files, AES take least time relative to others algorithms and Trivium is the second algorithms which take least time relative to others. RSA and Elliptic Curve take enormous amount of encryption time compared with others because their key length and the phase of encryption process.

ii. The same format but different size to analyze the decryption time selection random select 6 video which the size have

Table 4.2: Decryption time for same type and different video file.

The video size in MB in MP4 format	AES Decryption time. (ms)	Blowfish Decryption time. (ms)	RSA Decryption time. (ms)	ECC Decryption time. (ms)	Trivium Decryption time.(ms)
1	5	60	1801	93238	9
6.23	31	141	14241	1001545	28
15.2	125	313	15742	1274233	98
19.6	47	454	21458	1842153	114
42.7	172	937	48721	4424585	148
127.2	298	789	2387323	12879221	698

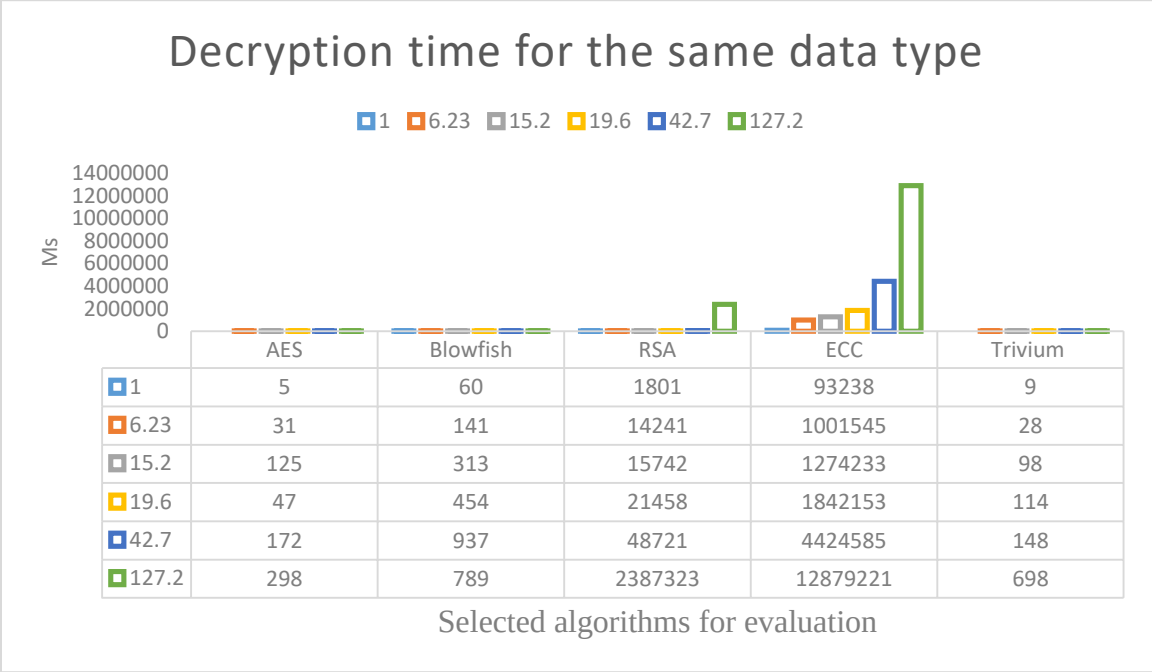


Figure 4.3: Decryption time for same type and different video file

Decryption time which Elliptic curve and RSA take much time compare with the others algorithms and AES use minimum decryption time and secondly Trivium takes less time compared with Blowfish. Those symmetric algorithms that uses small key and their decryption phase less complicated.

iii. Throughput for both encryption and decryption

Table 4.3: Total throughput

The size in MB in MP4 format	AES throughput (byte/ms)	Blowfish throughput (byte/ms)	RSA throughput (byte/ms)	ECC throughput (byte/ms)	Trivium throughput (byte/ms)
1	52786	32990	110	220	175956
6.23	712892	93664	156	150	291636
15.2	35136	33748	1696	196	270174
19.6	219540	88002	2090	138	294812
42.7	409214	85758	2402	144	273224
127.2	249584	77678	174	108	265294

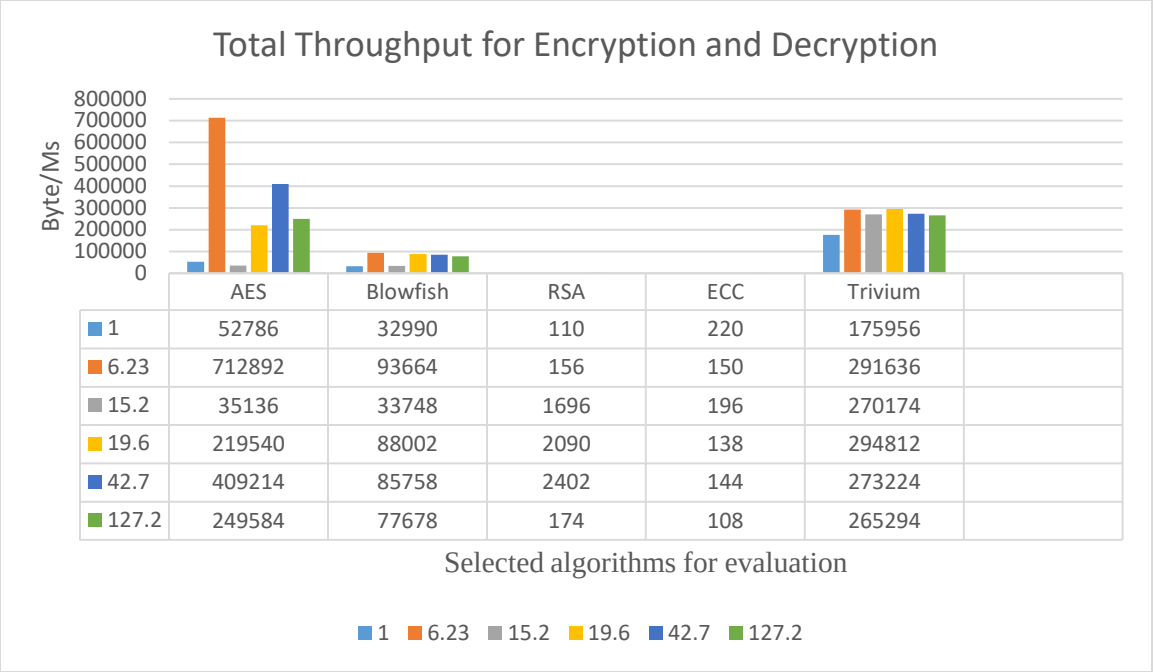


Figure 4.4: Throughput for encryption and decryption

AES and Trivium have better performance relative to the other algorithms. RSA and ECC have low throughput because the throughput is depending on the encryption time for this reason the time used to encrypt and decrypt the data is high.

iv. Memory usage for encryption and decryption process

For encryption

Table 4.4: Memory usage for encryption and decryption

The size in MB in MP4 format	AES Encryption memory usage in kb	Blowfish Encryption memory usage in kb	RSA Encryption memory usage in kb	ECC Encryption memory usage in kb	Trivium Encryption memory usage in kb
1	10034	96884	9710	24928	9665
6.23	20436	20506	20956	113818	20503
15.2	38363	39162	41177	148819	39113
19.6	47104	47526	152096	269731	47569
42.7	95894	95748	118477	618867	48459
127.2	263155	263155	283407	1387591	263279

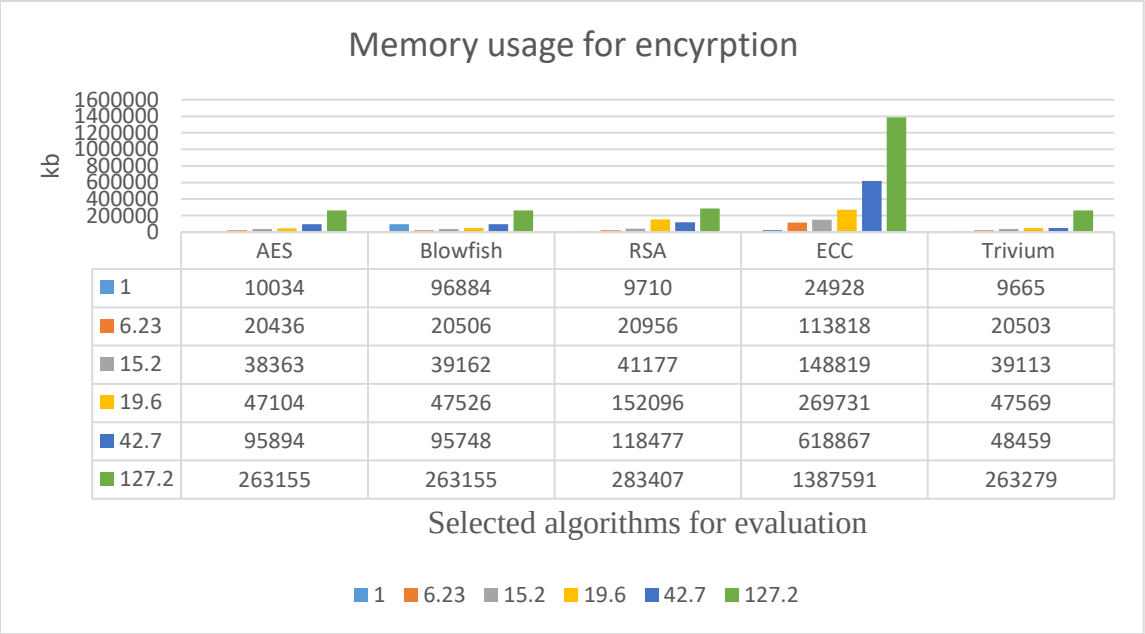


Figure 4.5: Memory usage for encryption

AES and Trivium use lesser amount memory during their process and blowfish also use small amount memory comparing with RSA and ECC. RSA also uses small amount of memory compared with ECC. The main reason is memory requirement depends on the number of operations to be done by the algorithms.

For Decryption

Table 4.5: Memory usage for decryption

The size in MB in MP4 format	AES Decryption memory usage (kb)	Blowfish Decryption memory usage (kb)	RSA Decryption memory usage (kb)	ECC Decryption memory usage (kb)	Trivium Decryption memory usage (kb)
1	11581	10784	10821	28986	11631
6.23	20436	20656	420949	313818	20844
15.2	38784	39164	45414	149943	39323
19.6	47473	47721	45721	269731	47729
42.7	95748	95599	118431	688867	95610
127.2	263155	233056	283494	2817231	263425

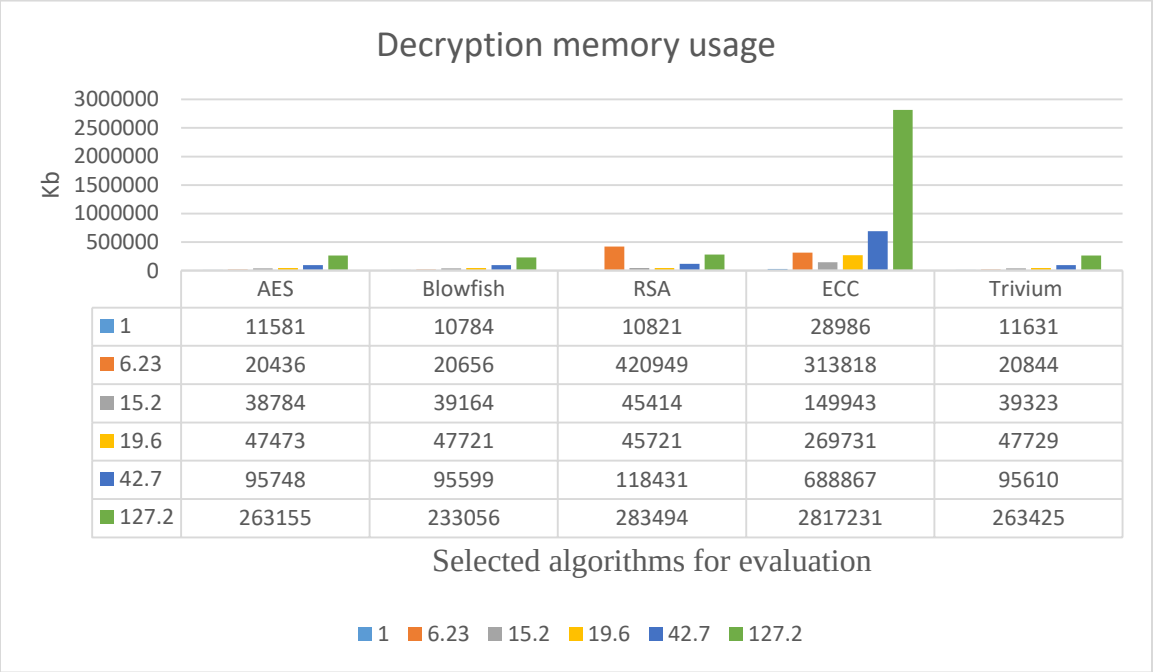


Figure 4.6: Decryption memory usage

The simulation result show that ECC use high memory throughout process and relatively AES and Trivium uses less memory relative to Blowfish, RSA and ECC. The number operation done in AES is less complicated relative with other.

v. CPU usage for encryption and decryption process

For encryption

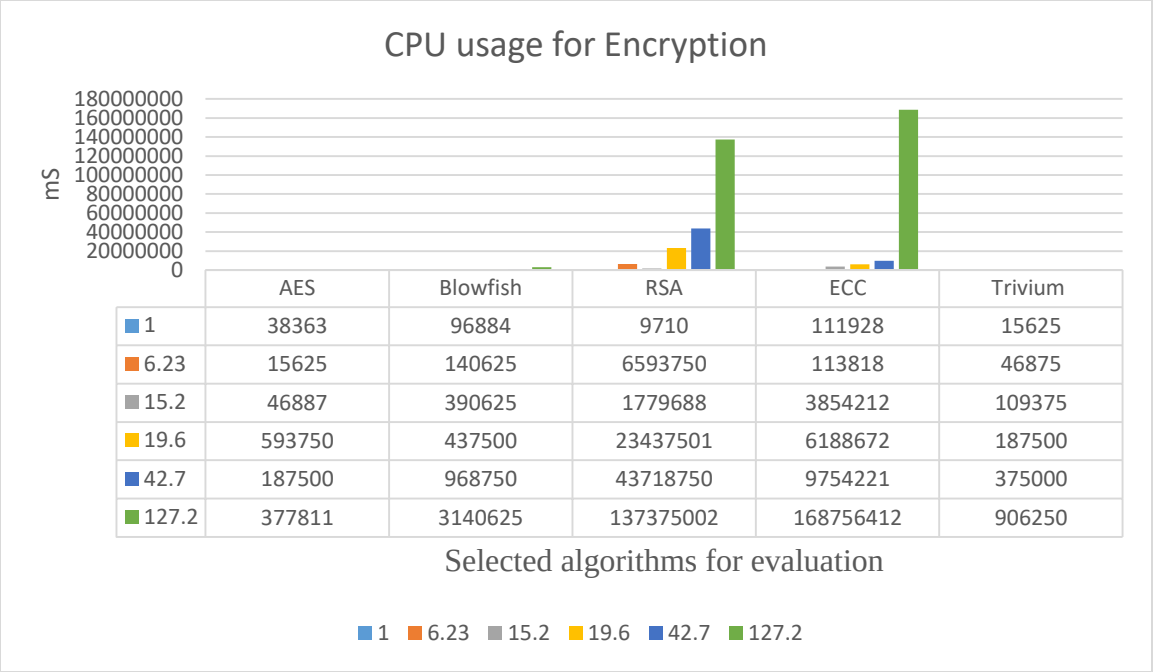
Table 4.6: CPU usage for encryption

The size in MB in MP4 format	AES Encryption CPU usage (ms)	Blowfish Encryption CPU usage (ms)	RSA Encryption CPU usage (ms)	ECC Encryption CPU usage (ms)	Trivium Encryption CPU usage (ms)
1	38363	96884	9710	111928	15625
6.23	15625	140625	6593750	113818	46875
15.2	46887	390625	1779688	3854212	109375
19.6	593750	437500	23437501	6188672	187500
42.7	187500	968750	43718750	9754221	375000
127.2	377811	3140625	137375002	168756412	906250

For Decryption

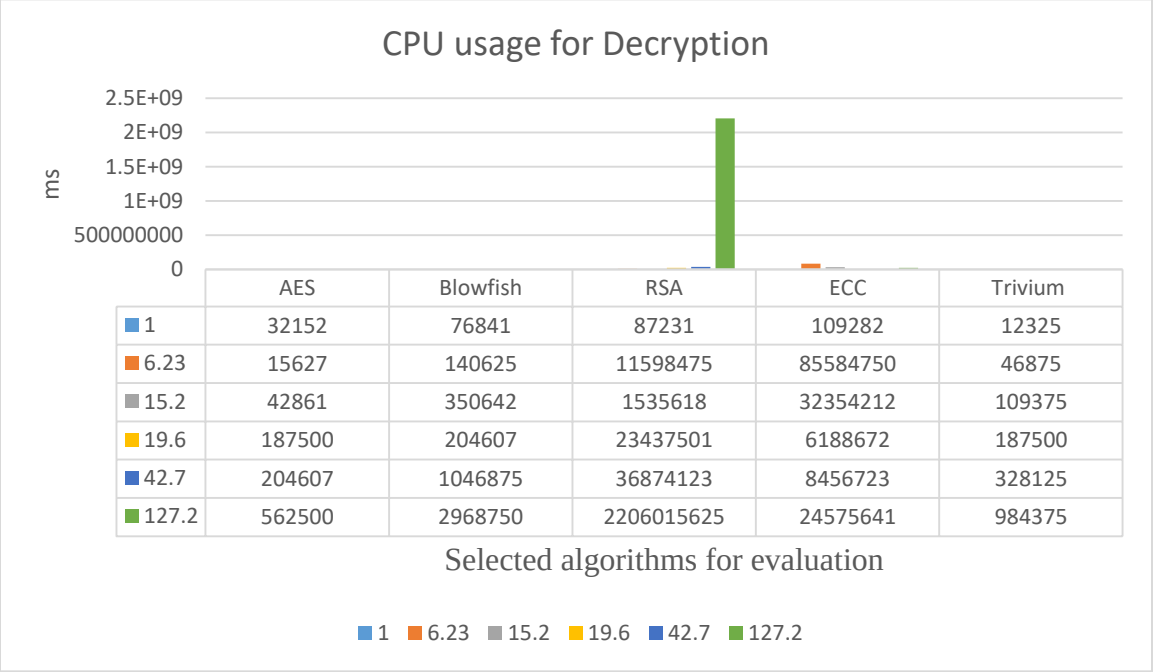
Table 4.7: CPU usage for decryption

The size in MB in MP4 format	AES Encryption CPU usage in (ms)	Blowfish Encryption CPU usage (ms)	RSA Encryption CPU usage (ms)	ECC Encryption CPU usage (ms)	Trivium Encryption CPU usage
1	32152	76841	87231	109282	12325
6.23	15627	140625	11598475	85584750	46875
15.2	42861	350642	1535618	32354212	109375
19.6	187500	204607	23437501	6188672	187500
42.7	204607	1046875	36874123	8456723	328125
127.2	562500	2968750	2206015625	24575641	984375



	AES	Blowfish	RSA	ECC	Trivium
1	38363	96884	9710	111928	15625
6.23	15625	140625	6593750	113818	46875
15.2	46887	390625	1779688	3854212	109375
19.6	593750	437500	23437501	6188672	187500
42.7	187500	968750	43718750	9754221	375000
127.2	377811	3140625	137375002	168756412	906250

Figure 4.7: CPU usage for encryption



	AES	Blowfish	RSA	ECC	Trivium
1	32152	76841	87231	109282	12325
6.23	15627	140625	11598475	85584750	46875
15.2	42861	350642	1535618	32354212	109375
19.6	187500	204607	23437501	6188672	187500
42.7	204607	1046875	36874123	8456723	328125
127.2	562500	2968750	2206015625	24575641	984375

Figure 4.8: CPU usage for decryption

The CPU usage for both encryption and decryption is AES and Trivium is low relative to RSA, Blowfish and ECC. Blowfish also have small amount time compared with RSA and ECC. This shows AES and Trivium algorithms is less complicate and low CPU load.

vi. Power efficiency

For both encryption and decryption measured in j/ms

Table 4.8: Power efficiency for encryption

The size in MB in MP4 format	AES Encryption Power efficiency (j/ms)	Blowfish Encryption Power efficiency (j/ms)	RSA Encryption Power efficiency (j/ms)	ECC Encryption Power efficiency (j/ms)	Trivium Encryption Power efficiency (j/ms)
1	$2.1*10^{-9}$	$2.3*10^{-8}$	$7.5*10^{-7}$	$5.4*10^{-4}$	$3.4*10^{-9}$
6.23	$7*10^{-9}$	$2.75*10^{-8}$	$8.4*10^{-7}$	$9.1*10^{-4}$	$15*10^{-9}$
15.2	$1.3*10^{-8}$	$2.9*10^{*-8}$	$15.5*10^{-7}$	$8.4*10^{-3}$	$1.9*10^{-8}$
19.6	$1.6*10^{-8}$	$3.7*10^{-8}$	$84*10^{-7}$	$9.9*10^{-3}$	$2.1*10^{-8}$
42.7	$8.4*10^{-8}$	$4.7*10^{-8}$	$92*10^{-7}$	0.018	$9.2*10^{-8}$
127.2	$9.1*10^{-8}$	$4.32*10^{-8}$	$12*10^{-6}$	0.026	$9.7*10^{-8}$

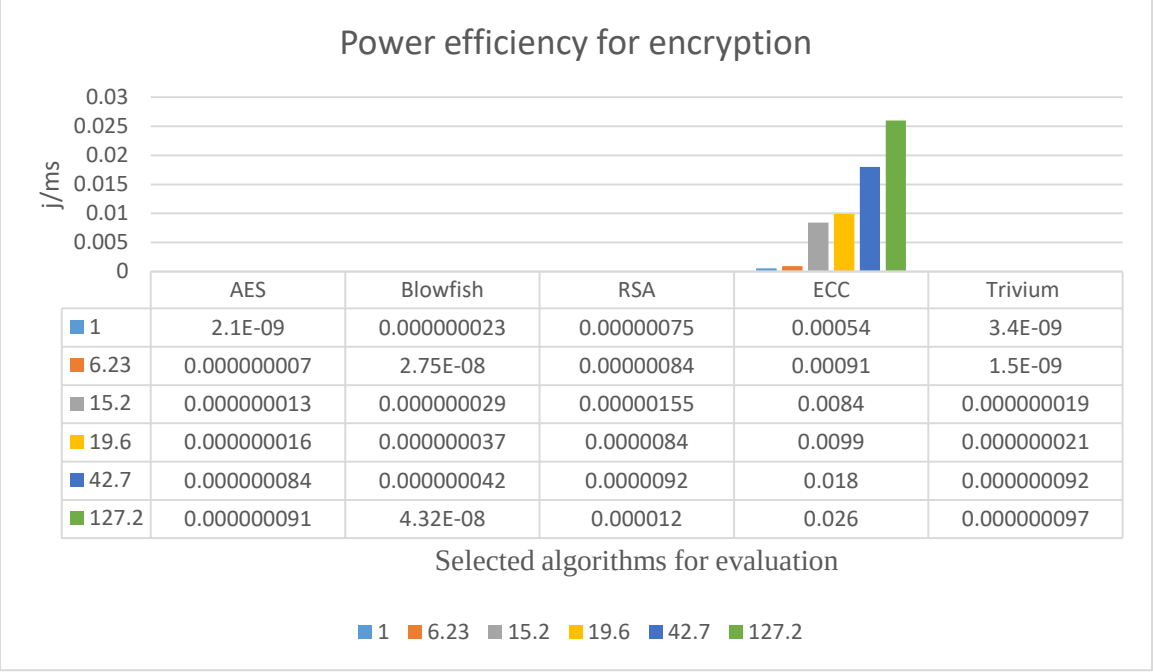


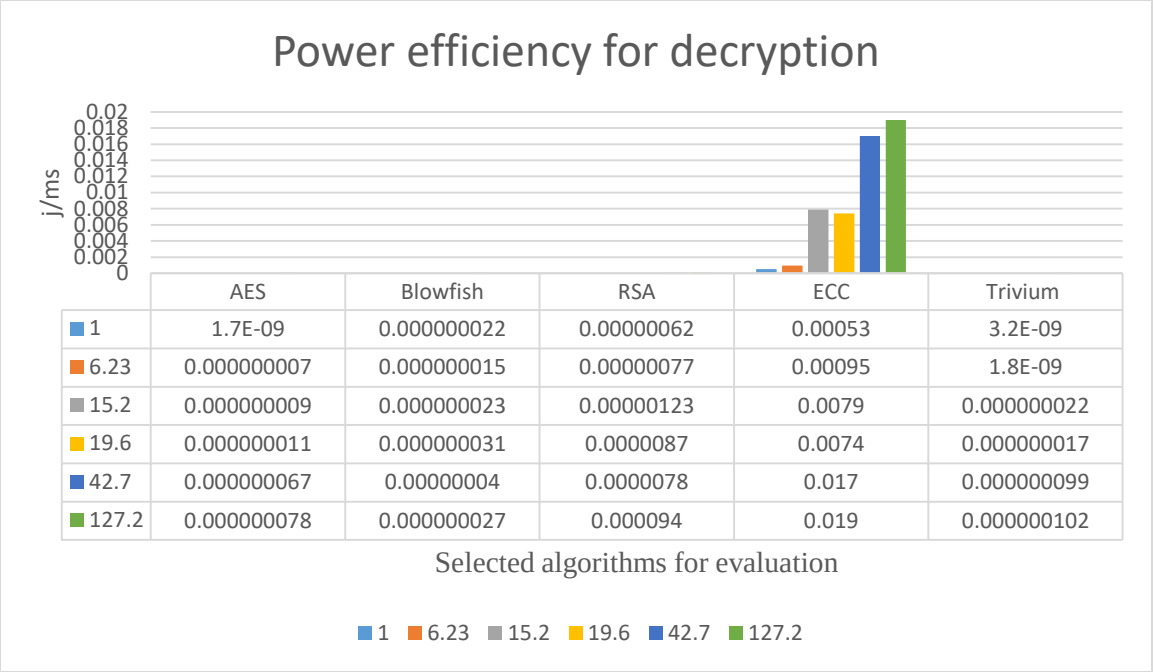
Figure 4.9: Power efficiency for encryption

For Decryption

Table 4.9: Power efficiency for decryption

The size in MB in MP4 format	AES Decryption Power efficiency (j/ms)	Blowfish Decryption Power efficiency (j/ms)	RSA Decryption Power efficiency (j/ms)	ECC Decryption Power efficiency (j/ms)	Trivium Decryption Power efficiency (j/ms)
1	$1.7*10^{-9}$	$2.2*10^{-8}$	$6.2*10^{-7}$	$5.3*10^{-4}$	$3.2*10^{-9}$
6.23	$7*10^{-9}$	$1.5*10^{-8}$	$7.7*10^{-7}$	$9.5*10^{-4}$	$18*10^{-9}$
15.2	$9*10^{-9}$	$2.3*10^{-8}$	$12.3*10^{-7}$	$7.9*10^{-4}$	$22*10^{-9}$
19.6	$1.1*10^{-8}$	$3.1*10^{-8}$	$8.7*10^{-7}$	$74*10^{-4}$	$1.7*10^{-8}$
42.7	$6.7*10^{-8}$	$4*10^{-8}$	$7.8*10^{-6}$	$17*10^{-3}$	$9.9*10^{-8}$
127.2	$7.8*10^{-8}$	$2.7*10^{-7}$	$9.4*10^{-5}$	$19*10^{-3}$	$10.2*10^{-8}$

The result show that the AES used low power relative to other and Trivium also the second low power used algorithms, blowfish uses also third low voltage uses algorithms compared with ECC and RSA. Because AES and Trivium the high throughput for this reason they are better to implement in low power devices.



	AES	Blowfish	RSA	ECC	Trivium
1	1.7E-09	0.000000022	0.00000062	0.00053	3.2E-09
6.23	0.000000007	0.000000015	0.00000077	0.00095	1.8E-09
15.2	0.000000009	0.000000023	0.00000123	0.0079	0.000000022
19.6	0.000000011	0.000000031	0.0000087	0.0074	0.000000017
42.7	0.000000067	0.00000004	0.0000078	0.017	0.000000099
127.2	0.000000078	0.000000027	0.000094	0.019	0.000000102

Figure 4.10: Power efficiency for decryption

The result show that the AES uses less power relative to other and Trivium also the second low power uses algorithms, blowfish also third low power used algorithms compare to ECC and RSA. AES and Trivium is relatively better to implement in low power devices.

vii. The different file format video encryption, decryption time, throughput for encryption and decryption, CPU usage and power efficiency.

Table 4.10: Encryption time for different file video

Video Type and Size	AES Encryption Time (ms)	Blowfish Encryption Time (ms)	Trivium Encryption Time (ms)	ECC Encryption Time (ms)	RSA Encryption Time (ms)
3GP-16.1 MB	109	359	141	828914	20971
WEBM-46.4 MB	125	985	343	112872	54032
MKV 40.4 MB	78	828	288	124589	40755

Table 4.11: Decryption time for different file format video

Video Type and Size	AES Decryption Time (ms)	Blowfish Decryption Time (ms)	RSA Decryption Time (ms)	ECC Decryption Time (ms)	Trivium Decryption Time (ms)
3GP-16.1 MB	102	391	30514	7456253	156
WEBM-46.4 MB	187	1094	632521	9852315	344
MKV 40.4	94	922	785323	1078521	297

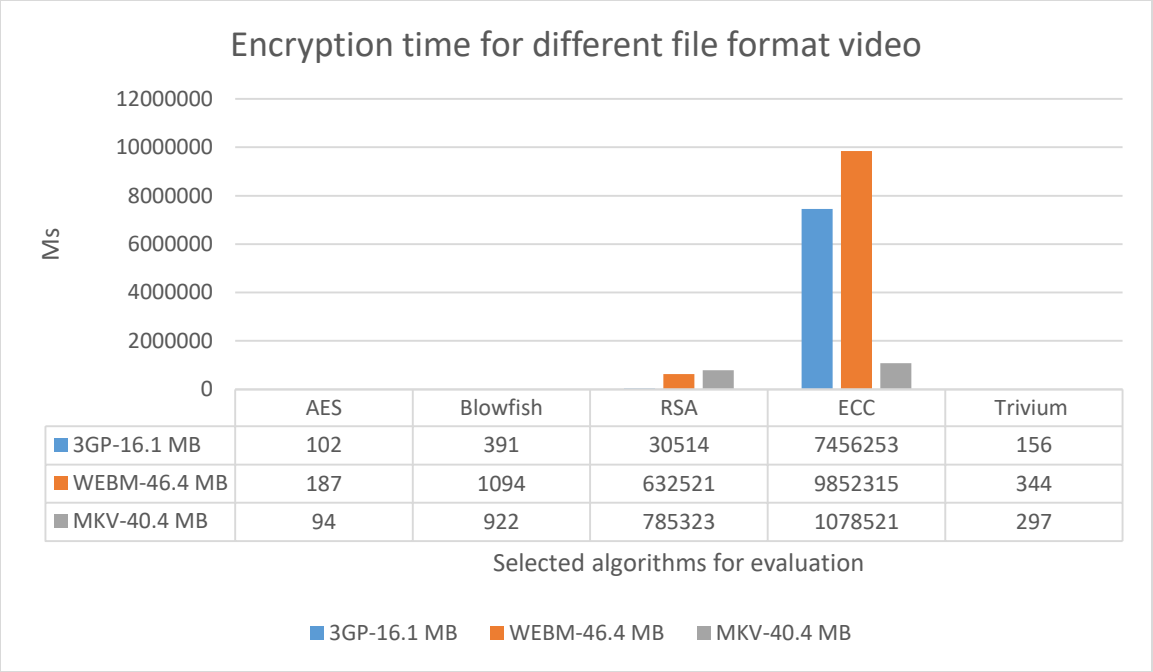


Figure 4.11: Encryption time for different file format video

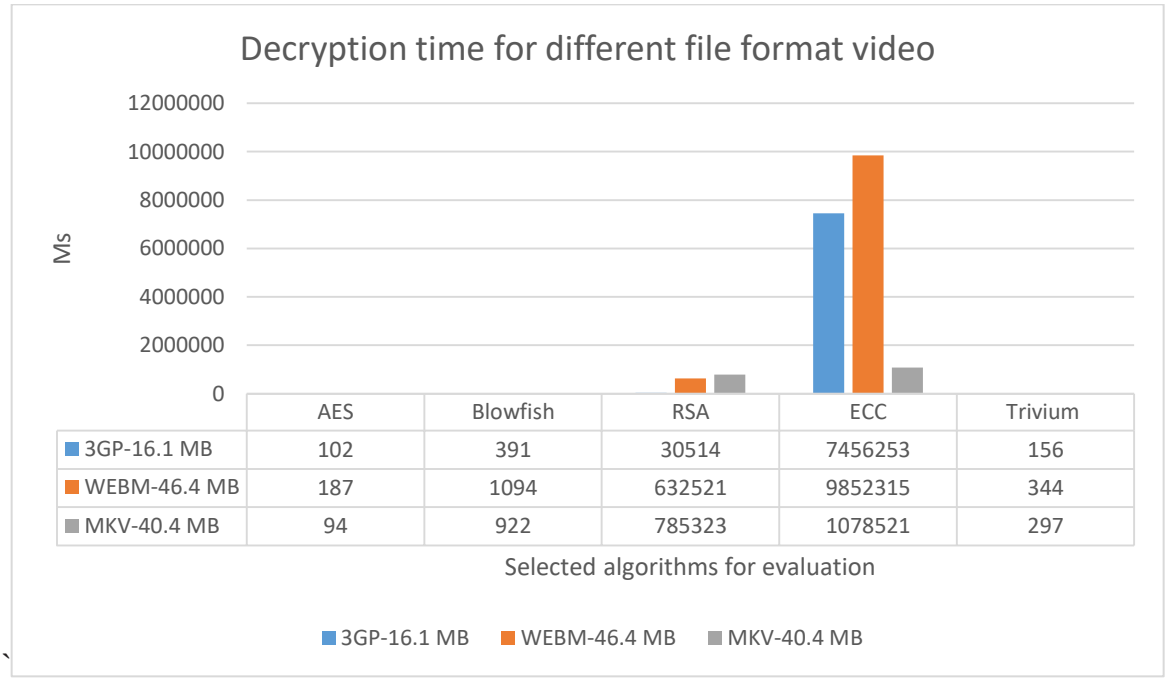


Figure 4.12: Decryption time for different file format video

AES use least time in different video file formats and RSA and ECC curve use huge time relative with other algorithms for encryption and decryption. RSA and ECC encryption is more complicated relatively with other algorithms.

For encryption

Table 4.12: Throughput for video file different file format

Video Type and Size	AES Total Throughput (byte/ms)	Blowfish Total Throughput (byte/ms)	RSA Total Throughput (byte/ms)	ECC Total Throughput (byte/ms)	Trivium Total Throughput (byte/ms)
3GP-16.1 MB	230220	90263	1487	418	228105
WEBM-46.4 MB	650383	50562	1629	116	206466
MKV 40.4	1086966	502172	2991	58	238044

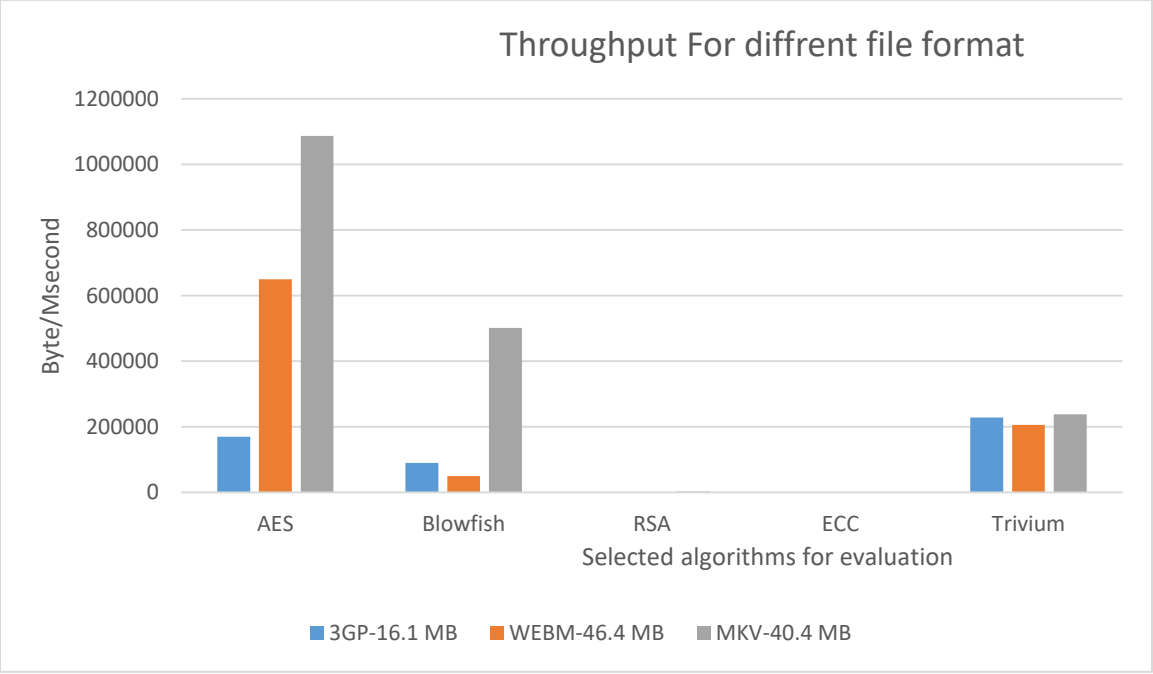


Figure 4.13: Throughput for different file type video

The result shows that AES high throughput relative to the other and RSA and ECC have low throughput compared with the other algorithms. Because RSA and ECC encryption/decryption consuming time is high relatively.

CPU usage

Table 4.13: CPU usage for encryption video data

Video Type and Size	AES Encryption CPU usage (ms)	Blowfish Encryption CPU usage (ms)	RSA Encryption CPU usage (ms)	ECC Encryption CPU usage (ms)	Trivium Encryption CPU usage (ms)
3GP-16.1 MB	562500	359375	19937500	26812523	140625
WEBM-46.4 MB	125000	953125	52187500	85752315	312500
MKV 40.4	78125	828125	4000000	1157869	281250

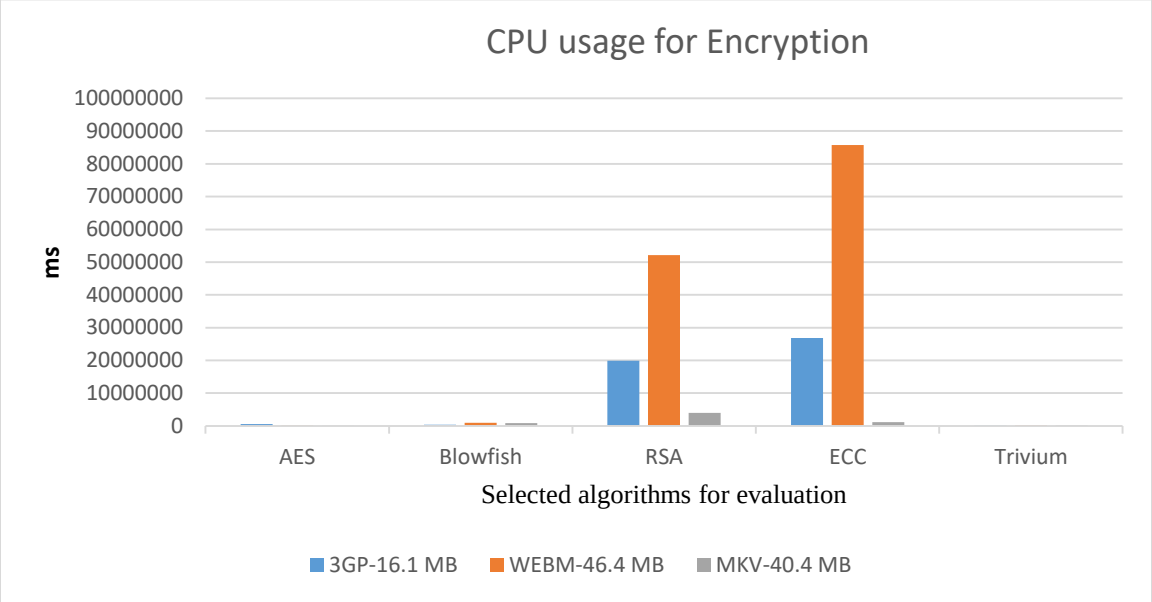


Figure 4.14: CPU usage for encryption

Table 4.13: CPU usage for decryption

Video Type and Size	AES Decryption CPU usage (ms)	Blowfish Decryption CPU usage (ms)	RSA Decryption CPU usage (ms)	ECC Decryption CPU usage (ms)	Trivium Decryption CPU usage (ms)
3GP-16.1 MB	109375	390625	2979218	32812523	140625
WEBM-46.4 MB	187500	1078125	80701562	92356522	328125
MKV 40.4	93750	921875	4235841	10237843	296875

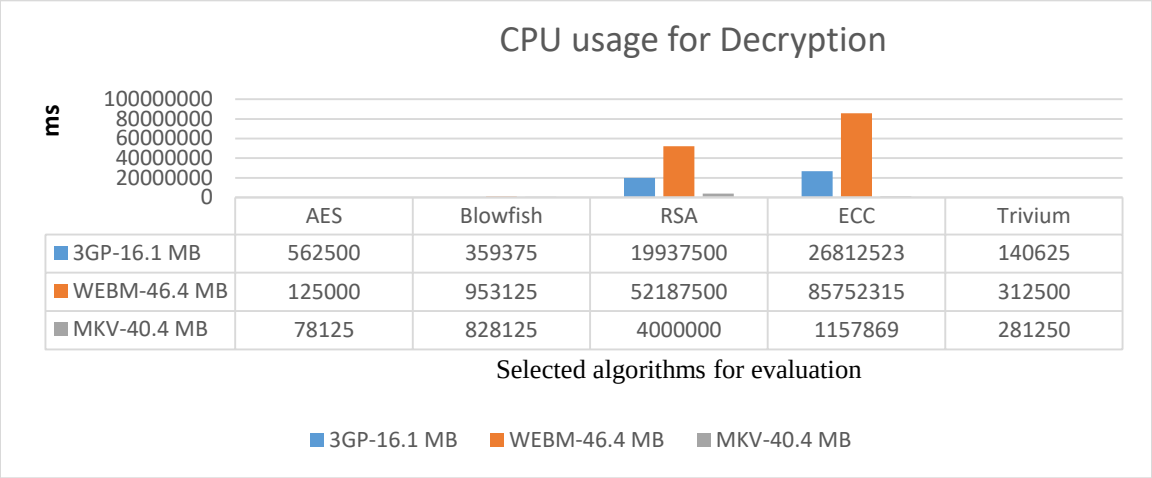


Figure 4.15: CPU usage for decryption

AES and Trivium also uses less CPU usage time compared with blowfish, RSA and ECC. This shows AES and Trivium algorithms is less complicate and low CPU load.

Memory usage

Table 4.14: Encryption memory usage video data

Video Type and Size	AES Encryption memory usage (kb)	Blowfish Encryption memory usage (kb)	RSA Encryption memory usage (kb)	ECC Encryption memory usage (kb)	Trivium Encryption memory usage (kb)
3GP-16.1 MB	41359	39154	1423568	2345655	43262
WEBM-46.4 MB	103009	103188	110252	258123	102935
MKV 40.4	90496	90565	3871352	4682254	90780

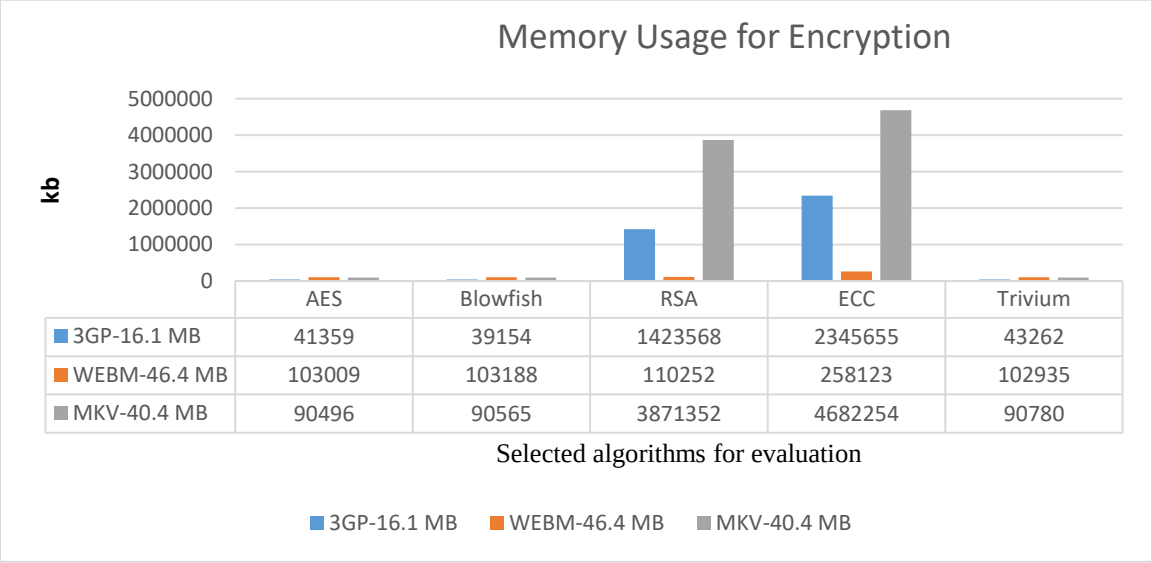


Figure 4.16: Memory usage for encryption for different format

Table 4.15: Memory for decryption for video data

Video Type and Size	AES Decryption memory usage (kb)	Blowfish Decryption memory usage (kb)	RSA Decryption memory usage (kb)	ECC Decryption memory usage (kb)	Trivium Decryption memory usage (kb)
3GP-16.1 MB	40039	39651	1658253	3127562	43921
WEBM-46.4 MB	103203	1078125	110544	3526251	102333
MKV 40.4	90872	90572	3803242	4256296	90851

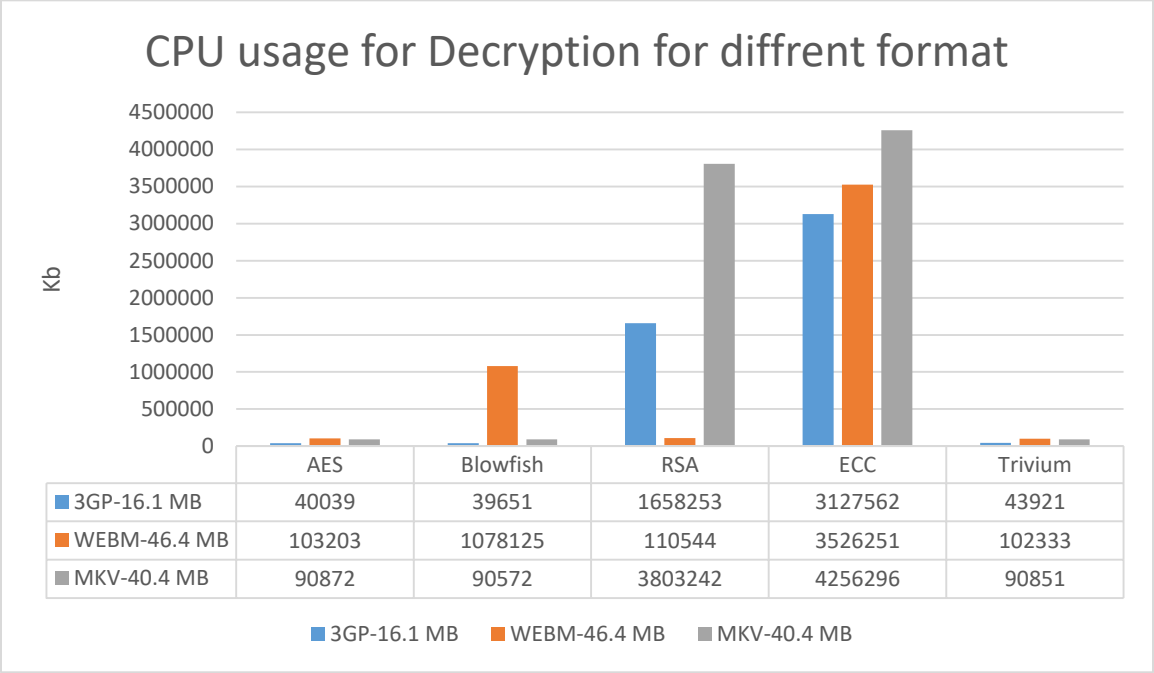


Figure 4.17: CPU usage for decryption for different format

In different data type of video file AES and Trivium use small amount of memory compared with other. RSA and ECC use high CPU usage for both encryption and decryption process. Because RSA and ECC both encryption and decryption process relatively highly complicated.

❖ For Audio file

Audio file Encryption time, Decryption time, throughput encryption and throughput decryption in mp3 format.

i. For Encryption time in Millisecond

Table 4.16: Encryption time for audio file.

Size MB	AES Encryption time (ms)	Blowfish Encryption time (ms)	Trivium Encryption time (ms)	RSA Encryption time (ms)	ECC Encryption time (ms)
5.65 mp3	15	156	110	7894	9857235
7.85 Mp3	31	172	63	8126	12822651
1.72 wma	5	32	16	2093	7835232

ii. For Decryption time in Millisecond

Table 4.17: Decryption time for audio file

Size MB	AES Decryption time (ms)	Blowfish Decryption time (ms)	Trivium Decryption time (ms)	RSA Decryption time (ms)	ECC Decryption time (ms)
5.65	62	172	47	114879	79374512
7.85	31	414	269	151598	92374621
1.72 wma	11	31	16	32198	65712225

The result shows that AES and Trivium uses less time for both encryption and decryption comparing with Blowfish, RSA and Elliptic Curve. This means encryption and decryption process of Blowfish, RSA and Elliptic Curve are complication is increased according to their sequence.

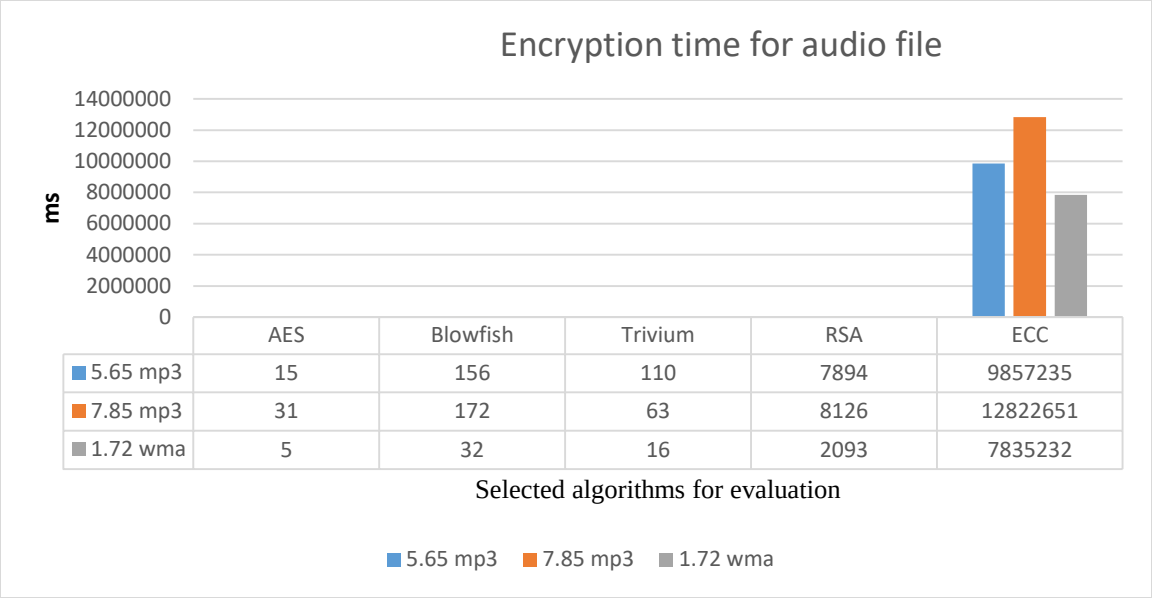


Figure 4.18: Encryption time for audio file

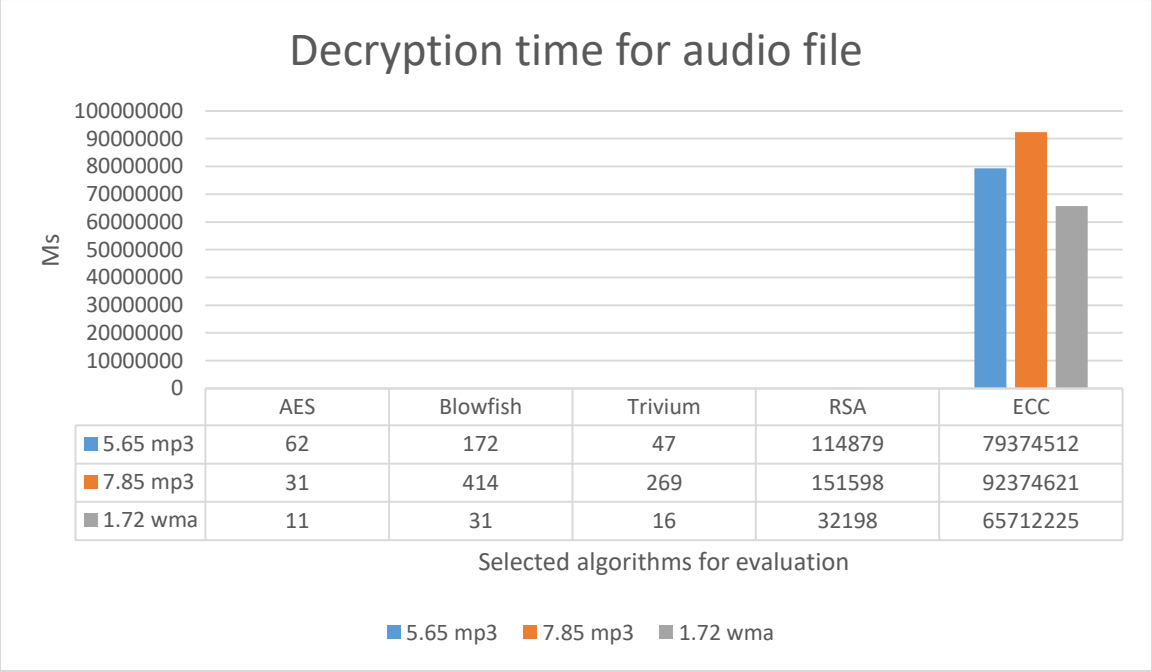


Figure 3.19: Decryption time for audio file

iii. Total Throughput for Encryption and Decryption in Byte per Millisecond

Table 4.18: Throughput for encryption

Size MB	AES Throughput (byte/ms)	Blowfish Throughput (byte/ms)	Trivium Throughput (byte/ms)	RSA Throughput (byte/ms)	ECC Throughput (byte/ms)
5.65 mp3	490881	68938	180043	1624	179
7.85 mp3	602762	85546	261630	1720	150
1.72 wma	526534	89635	226244	4190	846

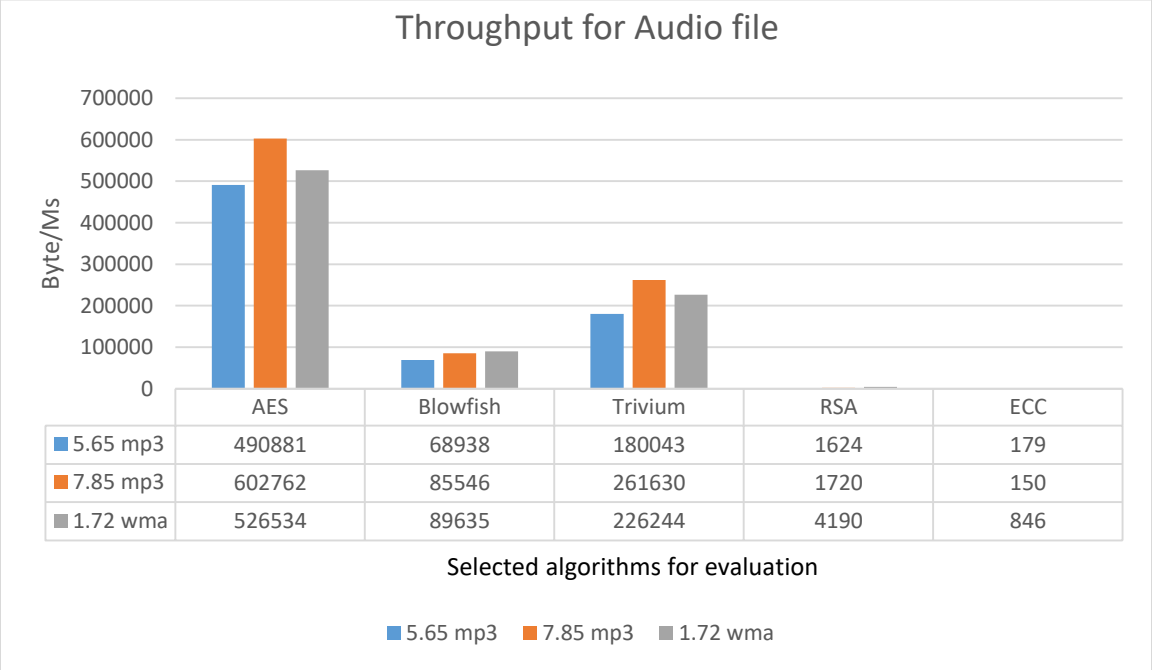


Figure 4.20: Throughput for audio

AES and Trivium algorithms have high throughput rate for both encryption and decryption comparing with other in audio file with various format. Because both algorithms used less amount of time to encrypt and decrypt.

iv. CPU usage for Audio file

For Encryption

Table 4.19: CPU usage for encryption

Audio Type and Size	AES Encryption CPU usage (ms)	Blowfish Encryption CPU usage (ms)	RSA Encryption CPU usage (ms)	ECC Encryption CPU usage (ms)	Trivium Encryption CPU usage (ms)
5.65 mp3	15625	156250	7593750	14568291	78125
7.85 mp3	16223	171875	8031250	95632341	62500
1.72 wma	1	32	2015625	32488931	16

For Decryption

Table 4.20: CPU usage for decryption

Audio Type and Size	AES Decryption CPU usage (ms)	Blowfish Decryption CPU usage (ms)	RSA Decryption CPU usage (ms)	ECC Decryption CPU usage (ms)	Trivium Decryption CPU usage (ms)
5.65 mp3	62500	171875	10996875	68712368	46875
7.85 mp3	46875	37631	1480000	87456313	62500
1.72 wma	0.5	31250	31843756	34156982	16

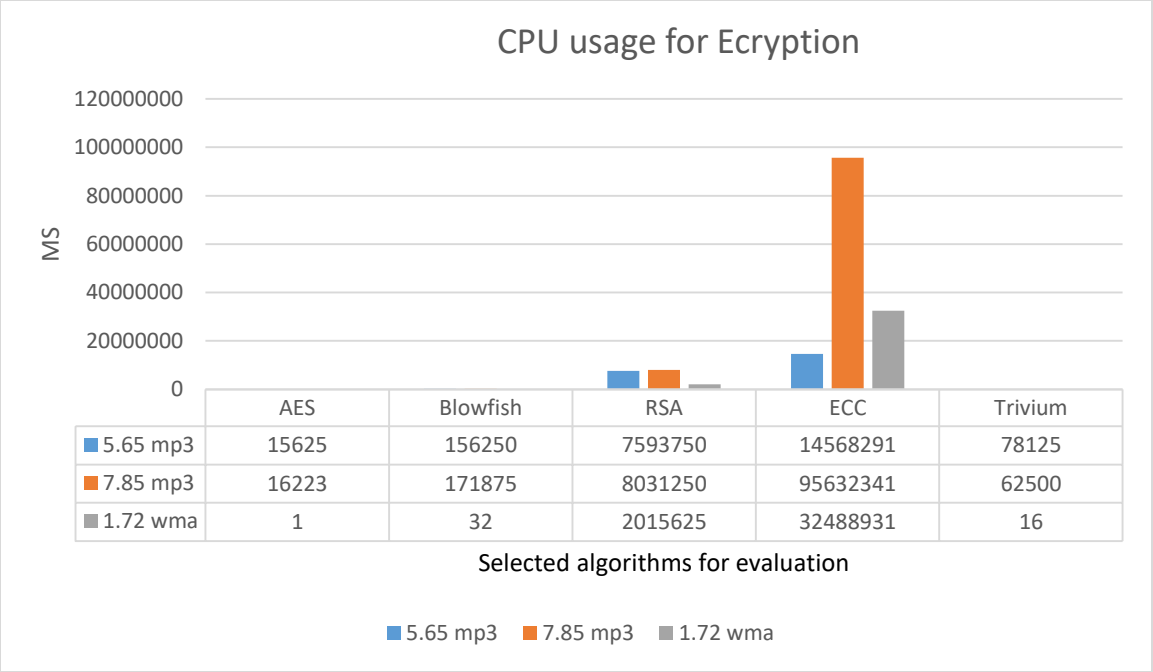


Figure 4.21: CPU usage for encryption

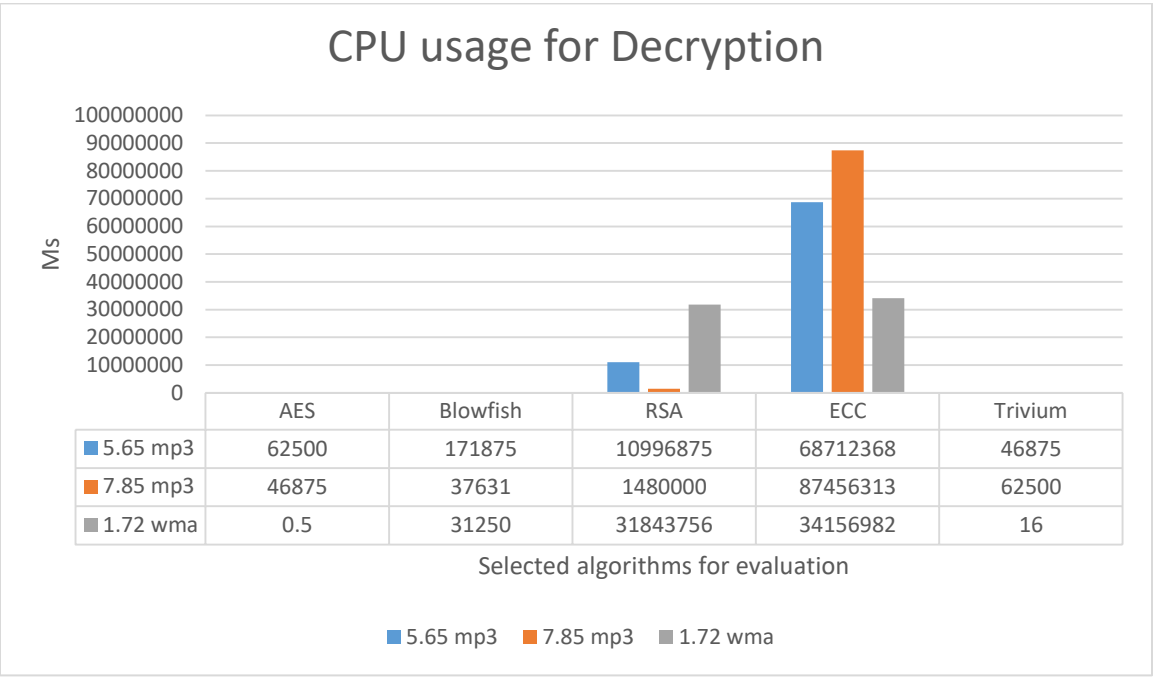


Figure 4.21: CPU usage for decryption

In both encryption and decryption of algorithms in CPU usage of audio file AES and Trivium uses petite time comparing with Blowfish, RSA and ECC. Relatively Blowfish uses short time comparing with RSA and ECC.

v. Power efficiency in j/ms

Table 4.21: Power efficiency for encryption

The size in MB in MP4 format	AES Encryption Power efficiency (j/ms)	Blowfish Encryption Power efficiency (j/ms)	RSA Encryption Power efficiency (j/ms)	ECC Encryption Power efficiency (j/ms)	Trivium Encryption Power efficiency (j/ms)
5.65 mp3	$6.4*10^{-9}$	$7.1*10^{-8}$	$5*10^{-7}$	$5.4*10^{-4}$	$6.54*10^{-9}$
7.85 mp3	$7.8*10^{-9}$	$8.5*10^{-8}$	$11*10^{-7}$	$9.1*10^{-4}$	$8.1*10^{-9}$
1.72 wma	$1.1*10^{-9}$	$1.9*10^{*-9}$	$3.7*10^{-8}$	$6.4*10^{-9}$	$1.3*10^{-8}$

The result show that the AES used low power relative to other and Trivium also the second low power used algorithms, blowfish uses also third low voltage uses algorithms compared with ECC and RSA. Because AES and Trivium the high throughput for this reason they are better to implement in low power devices.

vi. Memory usage for Audio file

Encryption

Table 4.22: Memory usage for encryption

Audio Type and Size	AES Encryption memory usage (kb)	Blowfish Encryption memory usage (kb)	RSA Encryption memory usage (kb)	ECC Encryption memory usage (kb)	Trivium Encryption memory usage (kb)
5.65 mp3	19089	18897	20736	45821265	18952
7.85 mp3	23743	23942	24803	58123472	24145
1.72 wma	12375	11745	11343	4682254	11760

For Decryption

Table 4.21: Memory usage for decryption

Audio Type and Size	AES Decryption memory usage (kb)	Blowfish Decryption memory usage (kb)	RSA Decryption memory usage (kb)	ECC Decryption memory usage (kb)	Trivium Decryption memory usage (kb)
5.65 mp3	18372	19092	20747	41038745	19101
7.85 mp3	23408	23457	25267	48952214	241445
1.72 wma	12423	11729	11506	2684123	11551

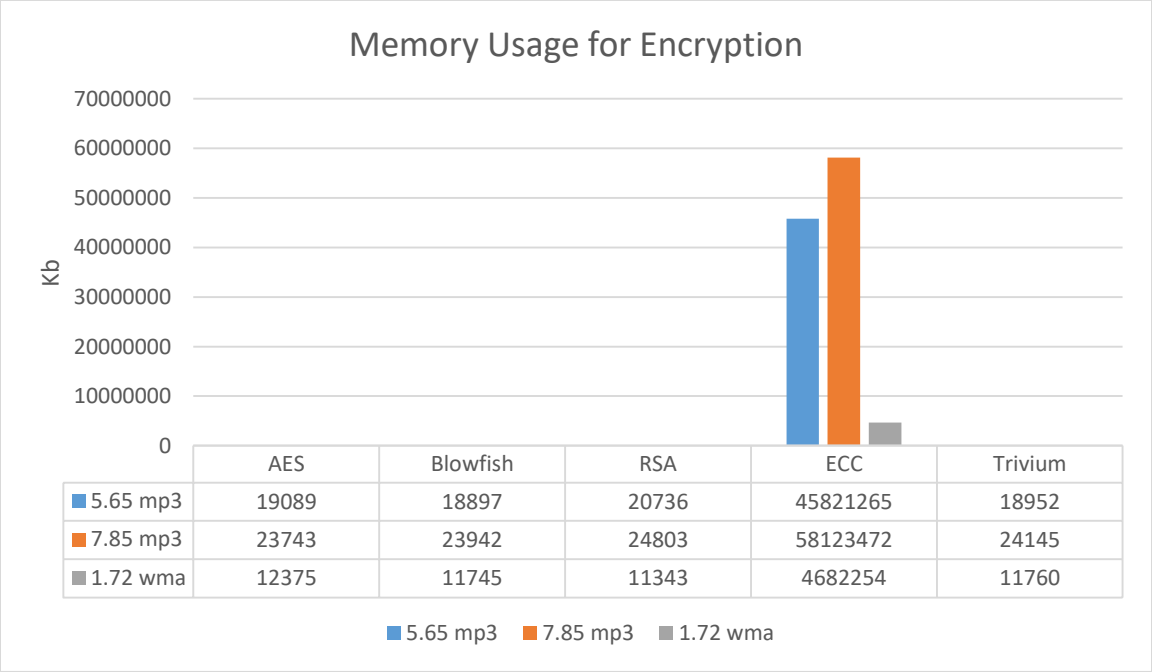


Figure 4.22: Memory usage for encryption

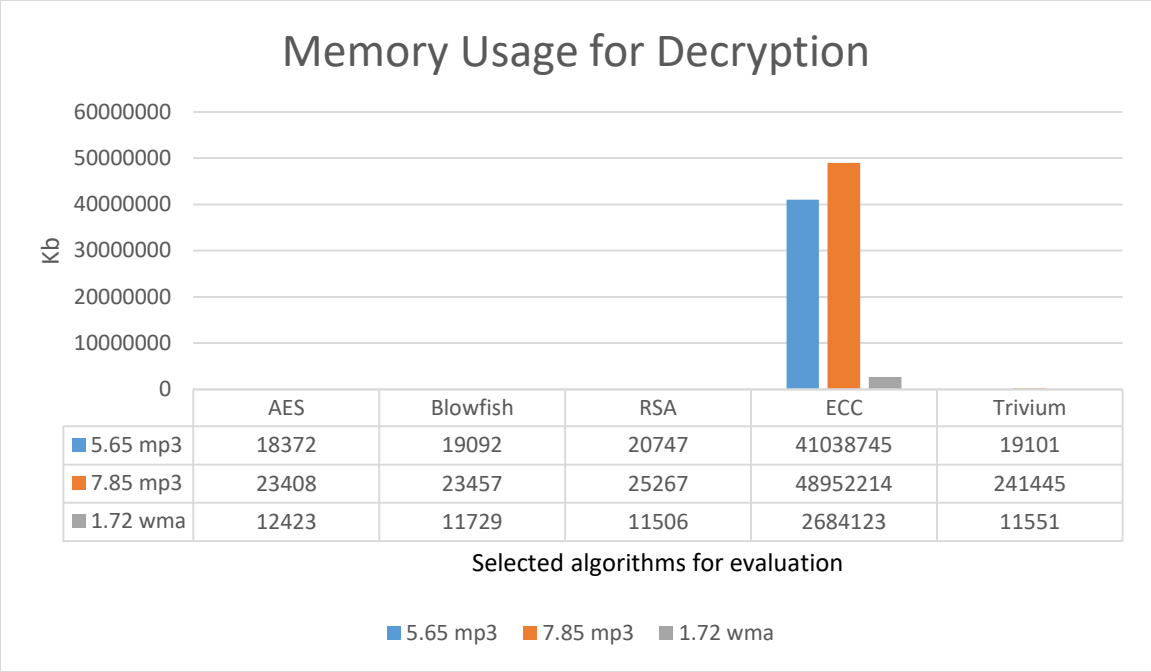


Figure 4.23: Memory usage for decryption

Memory usage in both encryption and decryption of algorithms in audio file AES and Trivium uses less memory comparing with Blowfish, RSA and ECC. Relatively Blowfish uses less memory comparing with RSA and ECC. Because number of operation to encrypt and decrypt audio file in relatively less for AES and Trivium.

4.3 comparative analysis of relative similar works

In previous works different cryptography techniques implemented

No	Done	Algorithm's	Parameters	Data	Results
1	[11]	AES,RSA, DES, Blowfish	Encryption and Decryption time	Text & Image	AES has better performance relatively with other algorithms.
2	[12]	AES, Blowfish and DES	Encryption and Decryption time	Image	Blowfish algorithm consumes least encryption time and DES consume maximum encryption time. They also observed that Decryption of Blowfish and AES algorithms is better than DES algorithm.
3	[37]	DES, TDES, AES, RC4	Encryption, Decryption time, memory usage and Throughput	Text	DES has least encryption time and also it takes less memory for decryption but low throughput. TDES has high decryption time and also it uses more space to encrypt/decrypt. But TDES throughput is better than DES and RC4. RC4 uses less memory, high encryption/decryption time but low throughput.

Their other papers are try to compare the algorithms but their focus area is encryption and decryption time for this reason the IoT devices require optimized resource constraints therefore this thesis that fill gap of resource of usage of cryptography algorithms in different parameter.

This thesis shows the encryption time, decryption time, throughput, memory usage for encryption and decryption, power efficiency for encryption and decryption CPU usage for encryption and decryption of algorithms that can help to decide the algorithms used to implement in low power IoT devices regarding with resource constraints.

Chapter five

Conclusion and recommendation

5.1 Conclusion

The aim of this thesis is evaluating cryptography algorithms that is AES, Blowfish, RSA, Trivium and Elliptic-Curve Cryptosystems in the parameters of encryption time, decryption time, throughput rate, CPU time, power efficiency and memory usage. algorithms are done by java.

The comparative analysis first done by finding the encryption time and Decryption time is done by comparing the algorithms using java simulation result, using sample video data with the same type of file in different sizes the result shows AES use less encryption time relative to the other algorithms, Trivium uses less time next to AES, Blowfish the third algorithms uses less, RSA less time comparing with Elliptic Curve uses high time for encryption and decryption comparing with other. In the parameter of memory usage in encryption and decryption process using sample video data with the same type of file in different sizes Elliptic Curve uses high memory comparing with other and AES uses less memory relative with Trivium, Blowfish and RSA. The Trivium is the second less memory used in the simulation, Blowfish third less memory and RSA uses relatively to Elliptic Curve. In CPU usage in the first sample small size data Trivium uses less time but overall sample data AES uses less time and Blowfish is the third less usage of CPU and RSA uses less comparing Elliptic Curve, using sample video data with the same type of file in different sizes.

Consumption of power AES and Trivium are low relative with Blowfish, RSA and Elliptic Curve using sample video data with the same type of file in different sizes.

In different file type with different size encryption time, decryption time, throughput rate, CPU time, power efficiency and memory usage using sample video data AES uses less resource in all parameter listed in the above, Trivium is the second better in all parameters listed in this thesis, Blowfish the third algorithms use less resource, RSA are the fourth algorithms and Elliptic Curve uses high computational resources of all parameter listed in the above.

For the audio file with the same file type and different size over all result from the all parameters AES and Trivium use less resource in comparing with Blowfish, RSA and Elliptic Curve. For

different audio file type AES are better performance with all parameter and Trivium also second less resource used during the simulation result, Blowfish are third less resource used in and RSA uses high relatively also less resource comparing with Elliptic Curve and Elliptic Curve uses high resource.

Overall result show that performance analysis of the algorithms AES, Blowfish, RSA, ECC and Trivium on different parameters, those parameters need in IoT devices for system resource requirements. Therefore, the simulation show that AES have better performance in all parameters that can used in our paper, Trivium have the second better algorithms in our test and Blowfish have third algorithm which have better performance from RSA and ECC.

5.2 Recommendations for Future work

In this thesis addressed the Evaluating cryptographic algorithms of audio and video data for system resource constrained of IoT devices. The main focus of the thesis was the system resource constraints. For the future the following ideas could be recommended:

1. Study the Evaluating cryptographic algorithms of audio and video data in terms of other parameters which are not covered with this study such as speedup ratio.
2. Study the evaluating other algorithm's which is not covered by this thesis in different parameters.

References

[1] Network Associates, Inc. and its Affiliated Companies, "*An Introduction to cryptography* ",PGP, Version 6.0.2, 1998, [Online]. Available: http://www.nai.com [Accessed: Nov. 22, 2018].

[2]W. Stallings, "*Cryptography and Network Security Principles and Practices*", 4th ed. Prentice Hall, 2005.

[3]Applications of Modern cryptography Technologies, applications and choices by Roland van Rijswijk (SURFnet) and Martijn Oostdijk (Novay)

[4]N. Smart, "*Cryptography: An Introduction*". 3rd ed. , University of Bristol.

[5] A. Safi. , "*Improving the Security of Internet of Things Using Encryption Algorithms*" ,World Academy of Science, Engineering and Technology, International Journal of Computer and Information Engineering, vol. 11, no. 5, 2017

[6] G. C. Pereira, R. C. Alves, F. L. da Silva, R. M. Azevedo, B. C. Albertini, and Cíntia B. Margi ,"*Performance Evaluation of Cryptographic Algorithms over IoT Platforms and Operating Systems*", Hindawi Security and Communication Networks ,no. 2046735, 2017

[7] U. Kumar, T. Borgohain and S. Sanyal ,"Comparative Analysis of Cryptography Library in IoT", arXiv:1504.04306v1 ,2015.

[8] M. A. Razzaq, S. H. Gill, M. A. Qureshi and S. Ullah , "*A Comprehensive Study on Security Issues in the Internet of Things (IoT)*" , International Journal of Advanced Computer Science and Applications (IJACSA), vol. 8, no. 6, 2017

[9] Z. Wang, H. Ding, J. Han, and J. Zhao, "*Secure and Efficient Control Transfer for IoT Devices*", International Journal of Distributed Sensor Networks, no. 503404, 2013.

[10] S. K. Kim, B. G. Kim, and B. J. Min, "*Reducing Security Overhead to Enhance Service Delivery in Jini IoT*" , International Journal of Distributed Sensor Networks, no. 205793, 2015.

[11] P. Madhumita, "*Performance analysis of encryption algorithms for security*", International Journal of Advanced Research Trends in Engineering and Technology (IJARTET), vol. 4, no. 11, November 2017.

[12] A. Devi1, A. Sharma and A. Rangra, "*Performance analysis of Symmetric Key Algorithms: DES, AES and Blowfish for Image encryption and decryption*", International Journal Of Engineering And Computer Science, vol. 4 ,no. 6, pp. 12646-12651,June 2015.

[13] .B. Bharathi, G. Manivasagam and M. A. Kumar , " *Metrics for performance evaluation of encryption algorithms*". IJARSE, no. 03,March 2017.

[14] F. Maqsood, M. Ahmed, M. M. Ali and M. A. Shah, "*A Comparative Analysis for Modern Cryptography Techniques*". International Journal of Advanced Computer Science and Applications (IJACSA), vol. 8, no. 6, 2017.

[15] P. Madhumita, "*Performance analysis of encryption algorithms for security*", In Proc. IEEE International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES), 2016.

[16] M. A. Khan , K. Salah,a "*IoT security: Review, blockchain solutions, and open challenges*", Science Direct, Future Generation Computer Systems,no.82,pp. 395-411,2018.

[17] M. Ammar, G. Russello, B. Crispo , *"Internet of Things: A survey on the security of IoT frameworks*", Science Direct, Journal of Information Security and Applications,no. 38,pp.8-27,2018.

[18] D. Manojkumar, K. SenthilKumar, R. V. Amudhan, G. V. Moorthi, and J. Deny, *"Improve Security in IOT Using Blowfish Algorithm*", International Journal of Pure and Applied Mathematics,vol. 119, no. 2, pp. 16261-16267, 2018.

[19] N. Mangla and P. Rathod, "*A Comprehensive Review: Internet of Things (IOT)*", IOSR Journal of Computer Engineering, vol. 19, no. 4, pp. 62-72, 2017.

[20] M. R. Asassfeh, M. Qatawneh and F. M. ALAzzeh, *"Performance evaluation of Blowfish algorithm on supercomputer IMAN*", International Journal of Computer Networks & Communications (IJCNC), vol.10, no.2, March 2018.

[21] R. K. Addluri, *"An Efficient Implementation of the Blowfish*", M. Sc. thesis, Osmania University, India, 2011.

[22] N. Koblitz, *"Good and bad uses of Elliptic curves*", Moscow mathematical journal,vol. 2, no. 4,pp.693-715,2002.

[23] S. Ravali, P. Neelima, P. Sruthi, S. Dileep , B. Manasa , *"Implementation of Blowfish Algorithm for Efficient Data Hiding in Audio*", International Journal of Computer Science and Information Technologies, vol. 5 , no. 1 ,pp. 748-750, 2014.

[24] A. Safi, *"Improving the Security of Internet of Things Using Encryption Algorithms*", World Academy of Science, Engineering and Technology, International Journal of Computer and Information Engineering, vol. 11, no.5, 2017.

[25] S. Kuma, "*Ways to Encrypt Data - [AES, DES, RSA, ECC Encryption]*", tips2secure.com , February 5, 2016. [Online]. Available: https://www.tips2secure.com/2016/02/ways-of-data-encryption.html. [Accessed Jun. 22, 2018].

[26] S. Kuma, *" AES (Advanced Encryption Standard) in Cryptography*", tips2secure.com , February 5, 2016. [Online]. https://www.tips2secure.com/2016/02/aes-encryption-cryptography.html. [Accessed Jun. 22, 2018].

[27] S. Kuma, "*Blowfish Encryption Algorithm: [Explanation with Examples]*", tips2secure.com.February5, 2016.[Online]. https://www.tips2secure.com/2016/02/blowfish-encryption.html [Accessed Jun. 22, 2018].

[28] V. Tilborg and C.A. Henk, "*Fundamentals of Cryptology A Professional Reference and Interactive Tutorial* ", The Springer International Series in Engineering and Computer Science,2000.

[29] B. Schneier, "*Applied Cryptography: Protocols, Algorithms and Source Code in C*", Wile press, March 30, 2015.

[30] D. R. Stinson ,"*Cryptography: Theory and Practice*", 3rd ed. , Chapman and Hall press,2005.

[31] C. De Canniere and B. Preneel, "*Trivium Specification*", Katholieke Universiteit Leuven, Dept. ESAT/SCD-COSIC, Kasteelpark Arenberg 10, B-3001 Heverlee, Belgium.

[32] D. Boneh and V. Shoup ,"*A Graduate Course in Applied Cryptography*" ,cryptobook,2017.

[33] V. Tilborg, C.A. Henk, and S. Jajodia, "*Encyclopedia of Cryptography and Security*" The Springer International Series in Data Structures and Information Theory,2011.

[34] Z. Wang, H. Ding, J. Han, and J. Zhao, "*Secure and Efficient Control Transfer for IoT Devices*", Hindawi Publishing Corporation, International Journal of Distributed Sensor Networks ,no.503404, 2013.

[35] S. K. Kim, B. G. Kim, and B. J. Min, "*Reducing Security Overhead to Enhance Service Delivery in Jini IoT*", Hindawi Publishing Corporation ,International Journal of Distributed Sensor Networks ,no. 205793, 2015.

[36] B. Shah and Z. Aalam "*Implementation and Performance Evaluation of the AES Algorithm for Data Transmission using Various Programming Languages*". Communications on Applied Electronics (CAE) – ISSN: 2394-4714 Volume 3– No.4, November 2015.\

[37] B. Nithya and P. Sripriya "*Comparative Analysis of Symmetric Cryptographic Algorithms on .Net Platform*" Indian Journal of Science and Technology, Vol 9(27), July 2016.

[38] K. Deshpande and P. Singh" *Performance Evaluation of Cryptographic Ciphers on IoT Devices*" International Conference on Recent Trends in Computational Engineering and Technologies (ICTRCET'18), May 17-18, 2018, Bangalore, India.