Debre Berhan University

College of Computing

Department of Information Technology

Lempel-Ziv and Welch Compression with Cuckoo Search based Elliptic Curve Cryptography (LZWC-CS-ECC) to Enhance Image Encryption Security and Speed Performance

Woldeiyesus Ayele Agiz

A Thesis Submitted to the Department of Information Technology in Partial Fulfillment for the Degree of Master of Science in Computer Network and Security

Debre Berhan, Ethiopia

June 2021

Debre Berhan University

College of Computing

Department of Information Technology

Woldeiyesus Ayele Agiz

Advisor: Gnanaprakasam Thangavel (PHD)

This is to certify that the thesis prepared by Woldeiyesus Ayele, entitled: *Lempel-Ziv and Welch Compression with Cuckoo Search based Elliptic Curve Cryptography (LZWC-CS-ECC) to Enhance Image Encryption Security and Speed Performance* and submitted to College of Computing in partial fulfillment of the requirements for the Degree of Master of Science in Computer and Network Security complies with the regulations of the university and meets the accepted standards with respect to originality and quality.

Signed by the Examining Committee:

	Name	Signature	Date
Advisor:			
Internal Examiner:			
External Examiner:			
Chairman:			

Abstract

The usage of images transferring through the internet is dramatically increasing in today's digital era. These images are vulnerable to hacking and tamper by attackers. Securing publicly available images over untrusted internet is necessary to protect against network intruders. Nowadays, various encryption and decryption algorithms (RSA and DH) are available to offer a good level of security. However, in these algorithms, the size of the encryption key is large and much slower. Thereby Elliptic Curve Cryptography (ECC) is a better alternative, and it provides equal security with smaller key sizes. But, the private key is an issue unless chosen optimally. We used the Cuckoo Search (CS) algorithm to optimize the best private key to fill this gap, Lempel Ziv and Welch (LZW) algorithm for compression, compressed byte values converted into large numbers, and Net Bean IDE 8.2 for implementation and performance analysis. Based on the simulation result, the proposed algorithm improves the original ECC image (keys generation time by 23.6%, encryption time by 95.7%, decryption time by 102.1%, the overall processing time base on bit length by 45.9%, and the overall processing time base on image size by 111.9%). As a result of the performance investigation, the proposed algorithm is very fast and secure.

Key Words: - Cryptography, CS, ECC, Image Security, LZW.

Acknowledgments

First and foremost, I would like to thank God, his Holy mother (Kindest Mariam), for helping me throughout my life in general and complete this work.

I am grateful to my advisor Dr. Gnanaprakasam Thangavel, for his supervision and advice.

I would also like to thank my parents, brothers, sisters, and my friends (Especially thanks for Getaneh A., Amdework A, ...) for their support and patience. Without their encouragement, motivation, and understanding, it would not have been possible for me to complete this work.

Finally, my thanks go to all the people who have directly or indirectly supported me to complete the research work.

Big Thanks for all!!!

Table of Contents

List of Tables

List of Figures

List of Algorithm

List of Abbreviations

3DES	Triple Data Encryption Standard
AES	Advance Encryption Standard
bis	Big Integer Sequence
CS	Cuckoo Search
DES	Data Encryption Standard
DH	Diffie-Hellman
DLP	Discrete Logarithm Problem
DSA	Digital signature Algorithm
ECC	Elliptic Curve Cryptography
En/Decryption	Encryption and Decryption
LZWC	Lempel-Ziv and Welch Compression
PRkr	Private Key of Receiver
PRks	Private Key of Sender
PUkr	Public Key of Receiver
PUks	Public Key of Sender
SHA1	Secure Hash Algorithm 1

Chapter 1. Introduction

1.1 Background

In recent years, with the development of digital communication technology, computer network technology and information communication technology, information dissemination and access have become increasingly convenient and fast. Communication and transmission of data over networks have increased exponentially in the last few years. Every user can easily download digital multimedia such as images, audio, and video files from the Internet. Therefore, cybercriminal and data thieves have also become an increasingly serious problem [1].

Security is the means of protecting assets from adversaries. Data security refers to protecting data from destructive forces and the unwanted actions of unauthorized users. Data security hierarchy involves challenges and issues which encryption algorithms intend to reduce. To achieve the goal of data security, the ability of a specific type of cryptographic algorithm (symmetric and asymmetric) needs to be considered [2].

The Image plays a key role in part of providing information, significantly in remote sensing, medical specialty and video conferencing applications. The employment and dependence on information carrying image and its application is still growing. An image is made up of pixels which are displayed as a rectangular array. Each pixel can be a single value or a vector of components. The single value can be the grayness level of a monochrome image or can be an index into a color that maps a single value into a set of color components [3].

Digital images occupy a large part of our daily communications. Digital cameras, phones, and computers have made the process of capturing, processing and sharing images between interpersonal via instant and unrestricted communication extremely sensitive. Images are stored and then used throughout various applications such as Twitter, Facebook, Viber, Imo, Telegram, and WhatsApp [4].

The fact that many areas such as the medical field and the military field carry sensitive digital images must be secured and protected against attacks in effective ways. Encryption of sensitive data is necessary, and encryption algorithms are designed to protect data and ensure confidentiality and the authorized recipients can access the decryption data [5]. However, Image size is larger than text and this is one of the struggles of processing time and storage. To reduce the size of the image for storage and transmission bandwidth we apply compression the technique.

Compression is the process of reducing data file size and it gives benefits in terms of less storage, faster transmission and faster read or writes files [6][7]. Two types of compression are available namely, lossy and lossless. The Lossy compression strips some of a file's redundant data. Because of this data loss, only certain applications are fit for lossy compressions, like graphics, audio, and video. Lossy compression essentially reduces the quality of the file to arrive at the resulting highly compressed size [8].

Lossless compression, on the other hand, can recover the exact original data after compression. It is used mainly for compressing texts, executable programs, spreadsheets, etc., where exact replication of the original is essential & changing even a single bit cannot be tolerated. Examples: Run Length Encoding, Huffman Coding and LZW [9].

We select lossless image compression over other techniques because most images in most discipline areas such as health and military require high security and quality. Furthermore, LZW compression is advisable than other compression techniques because it is fast, simple and efficient [10][11].

Encryption is a process to convert data into an unreadable format to preserve data from unauthorized users using cryptographic algorithms. It is used to keep sensitive data so that it can be difficult for unauthorized users to see it. The good algorithms must have been tested to meet the requirements of the security which protect the encryption components[5]. Image encryption refers to applying a symmetric or asymmetric encryption algorithm on an input image to be converted into a cipher image [12].

Cryptography is used nowadays to facilitate secure communications. It is used in military and healthcare applications to protect multimedia files (image, audio, and video). The symmetric key system uses the same key for both Encryption and Decryption

(En/Decryption) processes and offers high security while the asymmetric key system uses a private key and a public key which is available for everyone and can be used by authorized users to derive the secret-key [13].

Asymmetric cryptography is the foundation of internet security, allowing for two parties to communicate securely without the need to exchange confidential key material in advance. All public-key cryptosystems in widespread use today are based on either the problem of factoring large integers (e.g., RSA) or the problem of computing discrete logarithms in some groups such as elliptic curves [14].

ECC was discovered in 1985 by Neal Koblitz and Victor Miller. The main advantage of ECC is that it provides better security with a smaller key size. For example, a 160-bit elliptic curve provides the same security as that of a 1024 bit RSA [15]. It is also difficult to solve discrete logarithmic problems on elliptic curves. The elliptic curve over the prime field comprises coefficients of the elliptic curve and the base point, which is a point on the curve. The curve selected to show be known to both the sender and the receiver. The major operation in ECC is scalar multiplication, which comprises Point addition and Points doubling. ECC provides an alternative to the popular public-key cryptosystems of the time, such as a multiplicative group over the finite field - RSA. In recent years, ECC has been the primary cryptographic protocol for secure web pages, online banking, encrypted email, and many other types of data [10][14].

Generally, in today's world image plays an important role in everyone's life. The security of images is required while transferring them across the network. Various En/Decryption algorithms are available to protect the image from unauthorized users. RSA and Diffie-Hellman(DH) key exchange provide a good level of security, but the size of the encryption key in these two is a big problem. ECC is a better alternative for public-key encryption. It provides equal security with a smaller key size. But choosing the private key is always an issue in all public key cryptographic algorithms such as RSA, and ECC. If tiny values are chosen in random the security of the complete algorithm becomes an issue. Since the Public key is computed based on the Private Key, if they are not chosen optimally they generate infinity values [16][17]. To overcome this issue we use the Cuckoo Search algorithm.

The CS algorithm is inspired by obligate brood parasitism of some cuckoo species by laying their eggs into the nest of host birds [18][19]. Those female parasitic cuckoos can imitate the colors and the pattern of the eggs of the host species. The algorithm works based on the following three assumptions:

1. A cuckoo chooses a nest randomly to lay the egg and at a time only one egg is laid by the cuckoo.
2. The best nests with the highest quality egg (solution) will carry over to the next generations.
3. The total number of available host nests is fixed and the host bird can discover a cuckoo's egg with a probability.

1.2 Research Motivation

Due to the growing popularity of networks and increasing rate of the use of technology including smartphones in every aspect of life tacking images, and sharing them is almost common in the fields of communication, transportation, military, medicine, education, social media, etc. In these fields, the digital image which contains critical information (e.g. Bank swift, military information, financial statements, and patient body part) can be transmitted. Due to this, the transmitted image has to be secured in the transmission medium because several attackers target these images. Thus, it is imperative to find a highly secured and fast method to protect these images data and their contents from any type of attack. Therefore, these issues motivated us to propose "LZW compression and CS-based ECC" to protect the image and its contents from any type of attack.

1.3 Statement of the Problem

Nowadays, digital images have become an inevitable source of information and are mostly used over a network. Every day the world comes across various images from various sources. Most of these images often include data on a high degree of confidentiality. Attackers always try to steal, damage, or use these private images to extort the owner of these images in different ways. In addition [20], the security of digital images has attracted much attention recently. As a result, the need aroused to suggest a strong way to protect these images against different types of attackers taking into consideration that the rapid development of the internet and the wide applications of multimedia technology enable people to exchange digital multimedia with others conveniently over the Internet. Although, different researches were conducted using RSA, Data Encryption Standard (DES), Advanced Encryption Standard (AES) and others to increase the security of the data [21]. But the size of En/Decryption keys, memory requirement and loss of random key search [17], processing time were not considered due to their large bit key [22]. ECC has gained attractiveness because it offers a similar security level comparing to traditional systems, such as RSA, but with significantly smaller key sizes. Thus, this feature makes it highly suited for implementation in very resource-constrained devices. However, the problem in ECC is that tiny values are chosen at random for the security of the complete algorithm and this becomes an issue. In ECC public key is computed based on the private Key, if they are not chosen optimally they generate infinity values and this leads to an incorrect En/Decryption process [17][16]. Another problem in image encryption is time to generate keys, encryption and decryption and transition issues in sharing cipher images. As the larger the size of the image, the higher the bandwidth it requires.

Therefore, instead of using the large key size and slow processing methods, we have proposed a new method called CS based ECC to Secure Image that overcomes the above shortcomings of existing algorithms. In addition to this, applying image compression by the LZW algorithm, before the encryption dramatically minimizes the time of encryption, decryption and transmission bandwidth.

Hence, this research work aimed to answer the following research questions:

- Which algorithm is better for image encryption and decryption for image data security?
- How can we enhance the performance and security of ECC for image encryption?
- Which algorithm is better for image compression and decompression to improve the perfocemace and security?
- How can we improve the key generation of ECC to improve the private key selection?

1.4 Objective of the Study

General Objective

The general objective of this work is to enhance image security using Cuckoo Search-based Elliptic Curve Cryptography with LZW compression.

Specific Objectives

To achieve the general objective of the thesis, the following specific objectives are identified.

- Exploring literature review related to encryption, decryption and Compression
- Identifying vulnerabilities and issues of ECC Image encryption
- Selecting a better algorithm for image compression and decompression
- Selecting a better optimal key generation algorithm
- Designing a better image security algorithm to overcome weaknesses of ECC and a new recent variant of ECC
- Implementing proposed cryptosystem
- Testing the proposed cryptosystem performance concerning existing works

1.5 Scope of Study

This research exhaustively discusses the ECC for image encryption performance, security, and its shortcomings. Then, it attempted to propose an algorithm to overcome these shortcomings. The scope of research is limited to improving image security and speed of image encryption, decryption, and key generation of ECC using CS algorithm and LZW compression. The proposed algorithm performance is compared to that of the original ECC because the new variations on ECC improved by adding new techniques and methods rather than modifying ECC special for image data security.

1.6 Research Methodology

Draw.io, an online diagramming application, was used to show our algorithms architectural design for key generation, encryption, decryption, and the whole process of the proposed system [23]. LZW compression and CS algorithm are used to speed up and optimize key search, respectively. Net Beans IDE 8.2 Java programming environment was used to implement algorithms because it has suitable for testing, bit manipulation, and other Big Integer operations. Microsoft Excel 2010 shows evaluation metrics such as speed performance (key generation, En/Decryption) and security performance (keyspace and key sensitivity) are used. Generally, the research processes include literature review, development of test cases, test and identify weaknesses, design solutions, and simulation.

1.7 Research Significance

Nowadays, image security is very important because it is used by almost all organizations, especially in healthcare, military, telemedicine, business institution, transportation, etc. Therefore, our work can be applied in these areas. This research work helps and satisfies numerous organizations by improving efficient real-time image encryption and decryption in various fields. Some of the significances of this thesis are as follows: it: -

- ➢ Optimize the ECC key generation based on the CS algorithm.
- ➢ Minimize the time for encryption, decryption.
- ➢ Fast and secure communication.
- ➢ Provides greater security performance in image encryption without difficulty.

1.8 Organization of the Thesis

The rest of this research work is organized as follows: Chapter two presents a detailed literature review research article related to image security and research article related to ECC, RSA, CS algorithm and image compression. It also provides a comprehensive summary of the literature review carried out for this research. It also introduces related works which are conducted for Image encryption using ECC and CS algorithm and discuss security issues and challenges of image. Chapter three presents our proposed algorithm. Chapter four provides an extensive simulation study and evaluation of the proposed algorithms and compares the proposed LZWC-CS-ECC with ECC. Finally, Chapter five concludes the research and summarizes the contributions of the thesis by comparison of all the proposed approaches for enhancing difference. Suggestions for future work are also included.

Chapter 2. Literature Review

2.1. Introduction

This chapter presents literature, the basic concept of security, cryptography and the type of cryptography, image encryption and compression algorithms and elliptic curve cryptography and Cuckoo search algorithm as follows:

2.2. Survey of Security

Security is one of the main challenges in the computing environment that results in the limitation of confidentiality, integrity, and availability of the data [24]. Many researchers and scholars have been providing different security schemes to achieve high-security encryption to protect the image from malicious attacks [25]. Nowadays, the security in digital images have become very important in many applications-confidential video conferencing, defense database, banking, finance, mobile computing, personal communication, etc. image share cryptography may play an important role particularly to those applications that require authentication based on shared keys maintained by multiple parties exclusively [26].

Since security in image transmission is a challenging issue during the transformation of an image for communication purposes, we need a strong mechanism to protect our data not be exposed to an attacker. To avoid this, the image should be protected before transmitting it from the sender to the recipient. For secure image transmission, researchers proposed that many techniques and methods to ensure security, out of the most commonly used techniques are encryption and data hiding [27][28]. Encryption ensures security but the problem with encryption algorithm is that the resultant noise image attracts the attention of hackers so it may be possible after many trials they can be decrypted by a hacker. Another method is data hiding in which pieces of secret information are hidden behind a carrier that may be anything a text file, video, audio or an image [27].

2.2.1. Introduction to Cryptography

Cryptography is the process used to convert data from readable type to unreadable type to realize the protection needs. It also provides an authentication mechanism for a user to protect their data from authorized access. In cryptography, the original data is called plaintext, and encrypted data is called a ciphertext [28][25]. The process of converting plaintext to ciphertext is called encryption whereas the inverse is called decryption [29]. Generally, we categorized cryptography into two main types. These are symmetric and asymmetric cryptography. In symmetric (private key) cryptography, the same key is used for both En/Decryption whereas, in asymmetric (public key) cryptography, the sender uses a private key for encryption, and the receiver uses the public key to decipher the ciphertext [30, 31, 32]. The keys used in cryptosystems represent the strength of the encryption algorithm. These keys should be complex and large enough to achieve high security to the secret data [33].

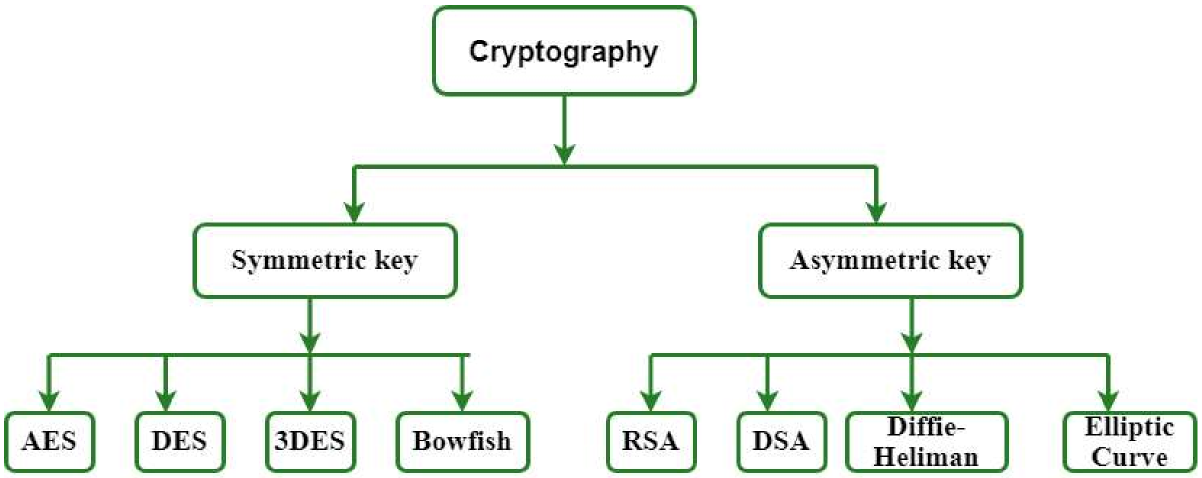


Figure 2. 1 Classification of Encryption algorithms

The use of cryptography also provides the authentication service for data during transmission. To secure our data from illegal users, there is a need to convert the information into a distorted format. In this case, cryptography is the best-known mechanism that encrypts an image using a different algorithm such as hashing algorithm, Blowfish, RSA, AES, and ECC[33][34].

2.2.2. Symmetric Key Cryptography

Symmetric key (privet key) cryptography is also called secret-key or shared key cryptography. In this type of mechanism, the sender and receiver share a common key for both En/Decryption. The method follows a self-certification method i.e. the key is selfcertified. The key needs to be shared through secret communication. If it is compromised, then the encrypted message can be easily decrypted by the attacker. This type of Cryptographic technique is required because it provides faster service without using many resources. Various algorithms have been developed so far as to describe symmetric key cryptography such as AES, DES, 3DES, Blowfish [33].

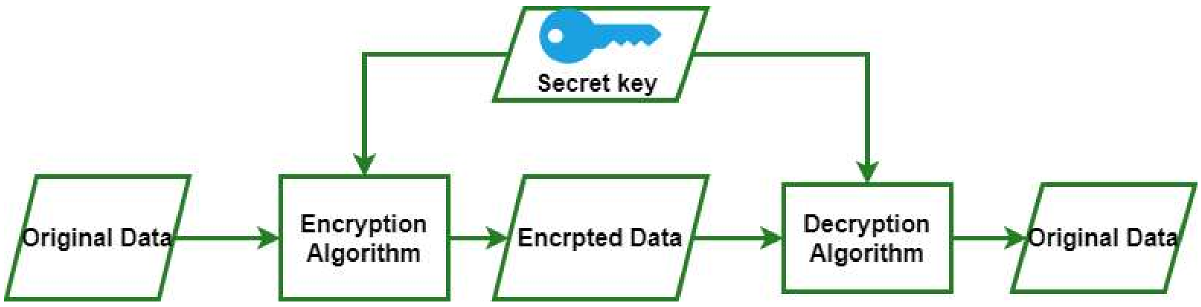


Figure 2. 2 Symmetric Key Cryptography

Recently symmetric key cryptography is well known and used in many applications because of its simplicity. Since a single key is used on both sides in symmetric techniques, the sharing of a key becomes sometimes insecure. The speed of symmetric key cryptography is better than public key cryptography, but the adversary knows the secret key anyway, then the entire private key cryptography will be unlocked. Therefore, the key must be secret between communicating parties. However, sharing or exchanging the same key is not a trivial task. These parties may share the secrete keys physically. However, it is not a good solution to exchange key physical for parties with a geographical barrier. The major attacks on symmetric key cryptography include the chosen-plaintext attacks, know-plain attacks, brute force attacks, linear cryptanalysis, etc. [28][35]. Different researchers perform image encryption using DES, AES, and Blowfish but because of the above limitations, we select asymmetric key cryptography [36].

2.2.3. Asymmetric Key Cryptography

Asymmetric key cryptography is also called public key cryptography. In this system, the sender uses a public key of the receiver to encrypt the plaintext and the receiver uses his private key to decrypt the ciphertext [32].

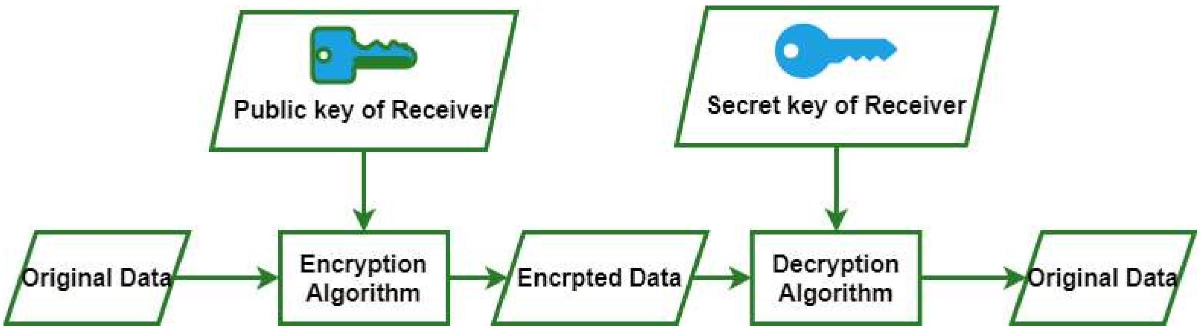


Figure 2. 3 Asymmetric key Cryptography

Asymmetric key cryptography is very important in data security to transmit secure data even when both communicating parties have no opportunity to agree on a given private algorithm [37]. It uses a longer key to improve the security of data during transmission. However, the limitation of public key cryptography is that slower than private key cryptography. As the study shows it needs more computing time and memory consumption for encrypting and decrypting the content of a message [22].

Table 2. 1 Comparison of Symmetric Encryption and Asymmetric Encryption

Basis For Comparison	**Symmetric Encryption**	**Asymmetric Encryption**
Basic	Symmetric encryption uses a single key for both En/Decryption.	Asymmetric encryption uses a different key for En/Decryption.
Performance	Symmetric encryption is fast in execution.	Asymmetric Encryption is slow in execution due to the high computational burden.
Algorithms	DES, 3DES, AES, and RC4.	DH, RSA, ECC.
Purpose	Symmetric encryption is used for bulk data transmission.	Asymmetric encryption is often used for securely exchanging secret keys.

There are various algorithms to implement this encryption mechanism. These are RSA, DH, ECC and Digital Signature Algorithm (DSA) [32]. In the incoming section, we continue our discussion on Public key crypitogrpy such as RSA, DH Key Exchange, DSA, and ECC.

A. Rivest, Shamir and Adleman

RSA is one of the public key encryption algorithm developed in 1977 by Adi Shamir, Ron Rivest and Len Adleman [38]. It is the most well-known public key cryptography that provides both digital signature and secrecy [39]. This cryptography generates a public and private key using mathematical functions depending on a large number of prime numbers. RSA algorithm uses a block cipher in a message both plaintext and ciphertext represent a number between 0 and n-1 for large number n. In this case the size of n 1024 bits or 309 decimal and make use of an exponential function. This algorithm provides a security service by using the following steps:

Step 1: Choose two large prime numbers p & q

Step 2: Compute n=pq and z=(p-1)(q-1)

1. Choose number e, less than n, which has no common factor (other than 1) with z
2. Find number d, such that ed – 1 is exactly divisible by z
3. Keys are generated using n, d, e
4. A public key is (n,e)
5. A private key is (n, d)
6. Encryption: c = me mod n

   m is plaintext

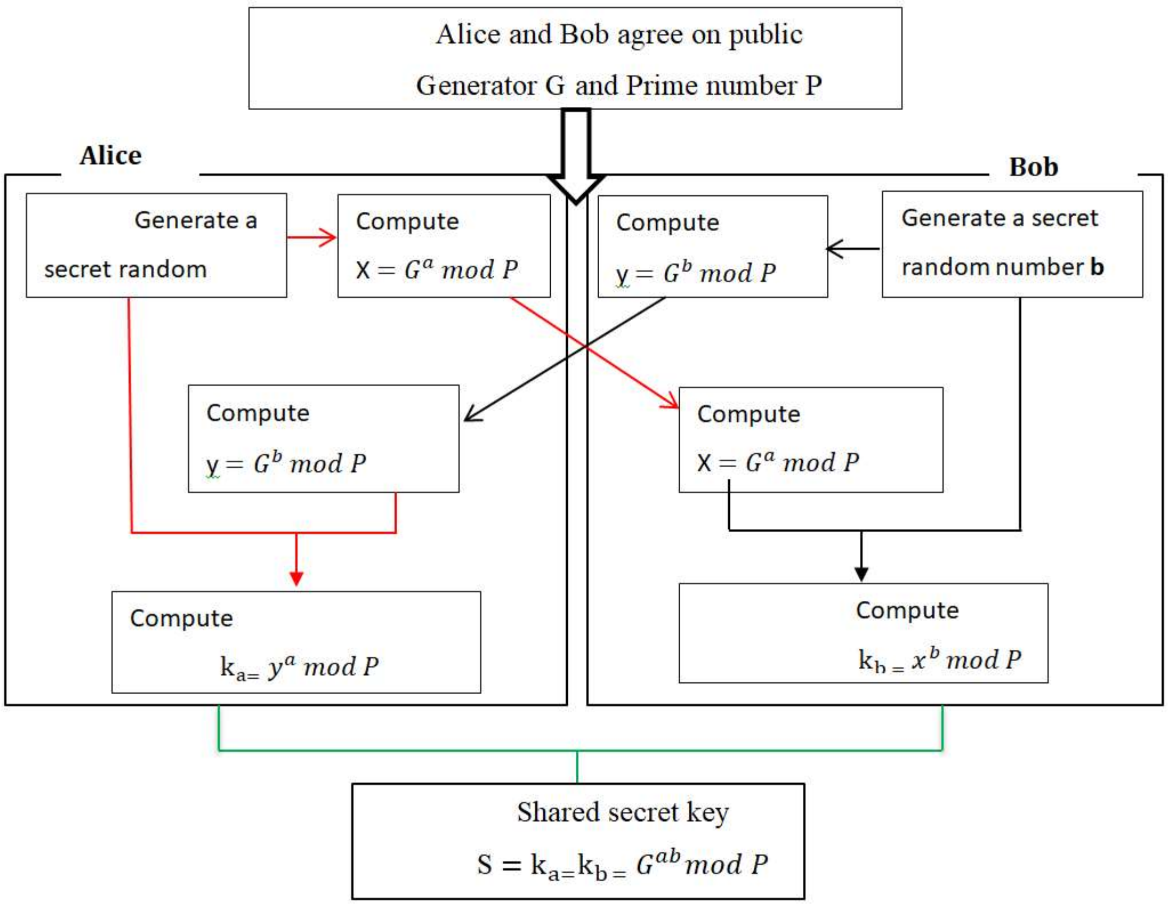
   c is ciphertext

   Decryption: $m = c^d \bmod n$
7. A public key is shared and the private key is hidden

However, the basic RSA is not secure enough for encrypting the same message more than once always gives the same ciphertext. Another primary drawback of RSA is the speed performance. This makes RSA to be exposed to different attacks such as indirect attacks like timing attacks, common modulus, known-plaintext, chosen-plaintext, and frequency of blocks attacks [39]. In addition to this, RSA is known to be much slower asymmetric key encryption and not advised for encrypting large data like image, video and sound.

B. Diffie-hellman Key Exchange

The other public key cryptography is called DH Key Exchange. This algorithm was first introduced by Witfield Diffie and Martin Hellman in 1976. It is used for key exchange. This type of key exchange mechanism consists of two keys: a private key and a secret key. The sender encrypts the message with his private key and public key. Then the receiver decrypts the message using his own private and sender's public key [30]. Figure 2.4 shows the basic the key exchange of DH.



Alice and Bob agree on public Generator G and Prime number P
Alice
Bob
Generate a secret random
Compute $X = G^a \bmod P$
Compute $y = G^b \bmod P$
Generate a secret random number **b**
Compute $y = G^b \bmod P$
Compute $X = G^a \bmod P$
Compute $k_a = y^a \bmod P$
Compute $k_b = x^b \bmod P$
Shared secret key $S = k_a = k_b = G^{ab} \bmod P$

Figure 2. 4 Diffie-Hellman Key exchange mechanism

However, the DH key exchange cannot be used for signing a digital signature. The nature of key exchange does make it susceptible to man-in-middle attacks since it doesn't authenticate either party involved in the key exchange.

C. Digital Signature Algorithm

On the other hand, the DSA is the other public key cryptography used for authentication and verification of the integrity of data. DSA was performed to be able to generate and verify signatures using a Secure Hash Algorithm (SHA). If the sender (source of message) wants to send a message to the receiver the signature generation in the sender uses its private key to generate a digital signature, once the receiver gets the message the signature verification uses the sender's public key. DSA is compatible with signing and verifying functions [30]. But DSA algorithm requires a lot of time for computing a signature and verification process. It is not encrypted data only used for authentication purposes. The other drawback of DSA is that it computes Secure Hash Algorithm 1(SHA1) and signs it due to any drawback in the cryptography security of SHA1 in the DSA [22][37].

D. Elliptical curve cryptography

The elliptic curve cryptosystem, which was proposed in 1985 by Neal Koblitz and Victor Miller [3][41], is one of the cryptosystems now in use for public key cryptography. The ECC uses the curve equation of the form as shown in equation (1), with a line only passes through three points along the curve (P, Q, and R), and that by knowing two of the points (P and Q), the other (R) can be calculated easily, but with just R, the other two, P and Q, cannot be derived. The basic diagram of ECC is shown in figure 2.5. In this equation (1), y, x, a, and b are all integers with modulo p. The coefficients a and b are called characteristic coefficients of the curve, which help us to determine what points will be on the curve [12, 17, 41].

$$y^2 = x^3 + ax + b \tag{1}$$

Where a and b are integers that satisfy equation (2) and p is a large prime number.

$$4a^2 + 27b^2 = 0 \tag{2}$$

The certain formula is defined for operation with the elliptic curve points are point addition, point doubling and point multiplication.

Point addition: Adding two points is not as easy as simply adding their x and y components and taking them modulo p. Instead, it is more like connecting the two points via a line and then intersecting that line with the curve. The two-point P(x_1, y_1) and Q(x_2, y_2) are distinct. P + Q = R(x_3, y_3) is calculate as equation (3, 4, 5).

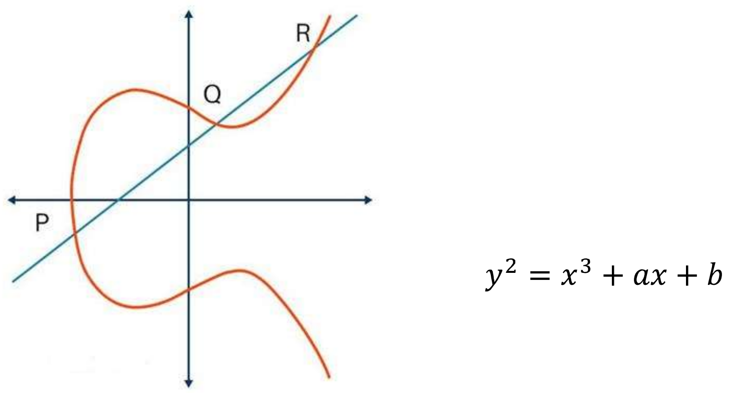


Figure 2. 5 Basic Elliptical Curve

$$x_3 = \{s^2 - x_1 - x_2\}\, mod\, p \quad (3)$$

$$y_3 = \{s(x_1 - x_3) - y_1\}\, mod\, p \quad (4)$$

Where

$$s = \frac{y_2 - y_1}{x_2 - x_1}\, mod\, p \quad (5)$$

Point doubling: Point doubling comes into play if two points are in the same (x, y) coordinate [12][40]. The two points P(x_1, y_1) and Q(x_1, y_1) overlap. P + Q = R(x_3, y_3) is calculate as equation (6, 7, 8).

$$x_3 = \{s^2 - 2x_1\}\, mod\, p \quad (6)$$

$$y_3 = \{s(x_1 - x_3) - y_1\}\, mod\, p \quad (7)$$

Where

$$s = \frac{3x_1{}^\wedge 2 + a}{2y_1}\, mod\, p \quad (8)$$

The other additional important calculation here is scalar point multiplication and checking if a point is on a curve. When point multiplication and point doubling are implemented, one can derive from those two basic building blocks scalar point multiplication, i.e. multiplying a scalar value (an integer) with a point. Let P be any point on the elliptic curve. Multiplication operation over P is defined by the repeated addition. nP = P + P + P +⋯+ n times [12, 41, 42].

In recent years, ECC has been the primary cryptographic protocol for secure text, images, web pages, online banking, encrypted email, and many other types of data [12, 14, 37]. The technology can be used in combination with a large amount of public key encryption techniques, like RSA, and DH. Some researchers project that ECC can computationally equal level of security measures with a 164-bit key that other systems need a 1,024-bit (RSA, DSA) key to attain. Since ECC helps to set up comparable security with lesser computing power and battery usage, it is becoming extensively used for mobile applications [15][30].

Elliptic curves are appropriate for encoding, digital marks, and other tasks. They are also used in several integer factorization techniques that have relevance in cryptography. The most important advantage promised by ECC is a lesser key range, sinking storage and transmission requirements, i.e. that an elliptic curve cluster could offer the same level of safety given by an RSA-based scheme with a huge modulus and equally superior key.

2.2.4. Image Compression

Compression is the technique of minimizing the size of a data file. Compression has advantages such as less storage, faster transmission, and faster reading and writing of files. The huge data file size is compressed to the smaller for the compressed data file which is limited network bandwidth [42][43].

There are two types of the compression algorithm. These are lossy and Lossless. The lossy compression algorithm loses some original data which is not necessary after decompression. It throws away perceptually insignificant information and cannot recover all bits. Some of the methods of lossy data compression methods are transform coding, discrete cosine transform, discrete wavelet transform, and fractal compression.

Lossless compression is a method used to reduce the size of a file while maintaining the same quality as before it was compressed**.** It is used in a text files, database tables, and medical images because of the law of regulations. Lossless image compression becomes significant because it can reduce the size of an image without quality loss. Some of the main techniques are Run Length Encoding, Arithmetic Encoding, Shannon fano, LZW, Huffman coding [43].

LZW COMPRESSION coding is invented by Abraham Lempel, Jacob Ziv, and Terry Welch. It is the process of encoding an image file in such a way that it consumes less space than the original file. It is a type of lossless compression technique that reduces the size of an image file without affecting or degrading its quality to a greater extent. It is a dictionary-based coding algorithm and the most preferred method for lossless file compression. It is the foremost technique for general purpose data compression due to its simplicity and versatility than other lossless compression techniques. It addresses spatial redundancies in an image. It is also an error-free compression approach that doesn't require prior knowledge of the probability of occurrence of symbols to be encoded [44, 45, 46].

Compression and encryption are interconnected with each other. Their objective is to reduce image file size, retain quality in reconstruction image from compression, manageable in available transmission bandwidth and secure during transferring [42, 47, 48].

- Compression followed by encryption: In this sequence, an intruder has less cleave to access the image but encryption may again increase the size.
- Encryption followed by Compression: In this sequence, size is not again increased but an intruder may have more clues to access the image.
- Joint compression and encryption: This approach is recently used which may be fast as compared to the previous two but the procedure is complicated.

2.2.5. Cuckoo Search

CS was inspired by the obligate brood parasitism of some cuckoo species by laying their eggs in the nests of other host birds (of other species). Some host birds can engage in direct contests with the infringing cuckoos. For example, if a host bird discovers the eggs are not its own, it will either throw these alien eggs away or simply abandon its nest and build a new nest elsewhere. In addition, the timing of the egg-laying of some species is also amazing. Parasitic cuckoos often choose a nest where the host bird just laid its eggs. In general, the cuckoo eggs hatch slightly earlier than their host eggs [18].

Once the first cuckoo chick is hatched, the first instinct action it will take is to evict the host eggs by blindly propelling the eggs out of the nest, which increases the cuckoo chick's share of food provided by its host bird. Studies also show that a cuckoo chick can also mimic the call of host chicks to gain access to more feeding opportunities [48].

CS is based on three idealized rules [49]: Each cuckoo lays one egg at a time and dumps its egg in a randomly chosen nest. The best nests with high-quality eggs will carry over to the next generation. The number of available host nests is fixed, and the egg laid by a cuckoo is discovered by the host bird with a probability $P \epsilon (0, 1)$ discovering operate on some set of worst nests, and discovered solutions dumped from further calculations.

S. Amtade and T. Miyamoto (2015) considered the problem of computational jobs capably considering system resource constraint and a CS algorithm is introduced. Experimental results show that CS outperforms the genetic algorithm in terms of fitness value [50].

B. M. Ismail et al., in 2016 proposed a cuckoo inspired fast search technique used for fractal image compression. This technique cuckoo inspired fast search uses an ordered vector of range blocks which was considered through their coordinate distance and also the ordered vector of range blocks by coordinating their similarity. The experimental results evinced that the suggested model is the most robust, scalable technique and there is a significant reduction in mean square error calculation, as there are only four transformations of the dihedral group and is sufficient for the similarity comparison in this proposed work [49].

P. Sekhar and S. Mohanty (2016) proposed a meta-heuristic algorithm which is called an enhanced CS algorithm for Contingency Constrained Economic Load Dispatch to alleviate transmission line overloading. The assessment of power system security deals with discovering the secure and the insecure operating states, whereas the improvement of security deals with the essential control action against overloads under a possible scenario. By generation of rescheduling, the overloaded lines are relieved from the rigorousness. In this proposed algorithm, to develop the solution vectors, dynamically variable parameters that are the step size and the possibility are incorporated instead of the fixed values of the parameters [51].

W. Yang et al., in 2017 designed a new local-enhanced CS algorithm, it differs from the standard version, and the local search for each cuckoo is attracted by the global best position found by the entire swarm. Simulation results show it achieves the best performance when compared with the other four algorithms [52].

2.2.6. Image Security

An image is defined as a two-dimension function, f(x, y) where x and y are spatial plane coordinates and the amplitude of f at any pair of coordinates (x, y) is called the intensity or gray level of the image at that point. When x, y and the intensity values of f are all finite, discrete quantities, we call the image a digital image [53]. The digital image is a kind of popular data widely used on the Internet [46].

Currently, images from various applications or sources are transmitted through the internet for various applications such as military images, medical imaging systems, confidential documents, online personal photo albums, cable TV and so on. These images contain some secret and confidential information that is required to be protected from leakage so that only the intended users should be able to read that image [26][54].

ECC is one approach to public-key cryptography with the algebraic structure of elliptic curves on finite fields. It requires smaller keys when we compared to non-ECC cryptography (based on finite field GF(p), where p is a prime number) to provide equivalent security. Before it becomes popular, all public-key algorithms which are based on the RSA cryptosystem, DSA and DH key exchange, and alternative cryptosystems which are based on modular arithmetic. However, the foundations and implementation skills of ECC are still a mystery and not understandable to most but, have equal security with non-ECC public key algorithms with smaller key sizes this makes ECC the suitable encryption technique since it needs small memory and high encryption performance[12].

2.3. Related Work

To securely transfer images across the network, various techniques have been developed in recent years using ECC.

2.3.1. ECC Image Encryption Mechanism

Ali Soleymani et al. [5], was proposed an encryption technique using Elliptic Curve over Prime Group field. They produced a mapping table with values ranging from 0 to 255 along each row, with the elliptic curve coordinates in the corresponding row. The pixel value of the image is mapped onto the elliptic curve coordinate using the table and encrypted with the public key of the receiver. The map table is simple to grasp and implement in this paper. However, it requires extra running time due to the map table.

P. Astya et al., in 2014 was proposed an image encryption algorithm using ECC. The image in the shape of a grid is first transformed on an elliptic curve in this article [55]. However, it is only for the Bitmap image En/Decryption.

As discussed by [14], elaborately proposed image encryption by using ECC during and before image compression, and they offered two ECC-based encryption algorithms, i.e. selective encryption of the quantized DCT coefficients and perceptual encryption based on selective bit-plane encryption. They apply encryption before compression, so it does not affect the structure of the ECC encryption, but the compression values for encryption time aren't more decreased.

Laiphrakpam D. and Khumanthem M. came with ECC to encrypt, decrypt and inclusion of digitally sign to the cipher image to provide authenticity and integrity [56]. In this paper, they apply pairing of the grouped pixel instead of mapping it to minimize the En/Decryption time, ECC, on the other hand, is employed to generate the random sequence that is utilized to disperse the simple image. However, the cryptographic operations are also very large

X. Zhang and X. Wang in 2018 [20], was proposed a digital image based on an ECC. With the Diffie–Hellman public key sharing approach, the sender and receiver agree on an elliptic curve point. Encrypted big integers are used to create the encrypted image. Their algorithm analysis and testing data show that they have strong security and efficiency. However, the En/Decryption times are not highly efficient, and the private keys selection is insufficient.

2.3.2. Cuckoo Search Algorithm Optimization

We have found two optimization techniques so far, namely Particle Swarm Optimisation (PSO) and Cuckoo Search algorithm. The PSO is an efficient algorithm for solving static function optimization for dynamic and noisy environments [c]. However, in many cases, the CS algorithm is better than PSO. Here are some of them: The CS algorithm is simple, can be used to adopt solutions to dynamic circumstances, It can be easily combined with others. The CS outperforms the PSO algorithm concerning the quality of the output produced and its reliability [d]. Other works related to CS algorithms are discussed as follows:

M. Dash and R. Mohanty (2014) applied a CS algorithm in the field of Speaker Recognition systems and voice. This algorithm can aim at discovering and shortlisting the features from the voice which can exclusively recognize them. Following feature extraction with a threshold to minimize discarded signal or disturbance and just considering the voice sample, a Fitness Function based on the mean of the individual sample is used to extract a small number of unique and best features while removing the rest. As a result, there is no require matching the voice of the speaker through every feature. In the matching phase, to enhance the security a threshold value is added to the correlation. As only optimized features are extracted, therefore this cannot only optimize this method but can also save resources [48].

In 2016, D. Manjumdar and S. Mallick proposed a new approach of the population-based CS algorithm that has been applied to a constraint satisfaction problem. Constraint Satisfaction and Optimization is a powerful pattern that includes planning, scheduling or configuration kind of complex combinational optimization problems. A set of variables and constraints are defining these problems. To satisfy several constraints, course timetabling requires scheduling rooms, teachers and also the number of subjects to the pre-specified time slots. In this, few constraints are hard, and they should be satisfied by maximum soft constraints. They presented an optimal solution namely the CS algorithm, which can solve the above-mentioned problems. It is evident with their experimental results that CS can be used effectively to solve the course timetable problem and it produces satisfactory results [18].

As proposed by [57], A novel oriented CS algorithm to improve Currently, distance vector-hop method performance for cyber-physical systems designed an oriented CS, in which, the local search capability was dominated by the combination of two different random distributions, and the global search pattern is directed by the historical best position, and the result shows modification achieves better precision performance when compared with three other Currently, distance vector-hop method algorithms. It is inspired by the behavior of cuckoos' brood parasitism and levy flight. Due to the simple concepts, it has been widely applied to many applications.

The paper [18, 49, 57], shows the CS performs well for almost all these test problems to optimize and fitness value in a different area. So we apply the cuckoo search algorithm for selecting the private keys.

2.3.3. LZW Compression algorithm

Run Length Encoding, Entropy Encoding, Huffman Encoding, Arithmetic Coding, Lempel–Ziv–Welch, Deflation and Chain Codes algorithms are lossless kinds of compression [58]. From these compression algorithms, the LZW algorithm is better in terms of compression ratio, the types of data it can compress, and faster computation time [59][60]. Moreover, it is simple and has good compression, a dynamic codeword table built for each file, and decompression creates the code word table, so it does not need to be passed [44][61].

2.3.4. CS with ECC

Gnanaprakasam Thangavel and Rajivkannan Athiyappan [19] proposed novel text encryption using the Optimal ECC method using Cloud Simulator. Experimental results and security analysis between the existing ECC and the new Optimal ECC are presented. They used the CS algorithm to find the best key as the input of the Optimal ECC text En/Decryption process. They also suggest applying CS-based Optimal ECC for Image encryption, Audio/Video encryption methods [19].

To get fast and secure transmission is by using compression before the encryption of multimedia data like images. It also increases security because the original image pixel is converted into a small coded byte, and it also extremely decreases En/Decryption process [47, 48, 59].

2.4. Summary of the Chapter

Many studies show that the standard ECC is effective for image encryption, and CS is also best for optimal key generation in En/Decryption. However, unless it uses optimization techniques such as CS for private keys selection, the ECC sometimes encounters failure during public keys generation. Moreover, the CS with ECC technique is used in encrypting and decrypting texts. But we don't found the image so far. In addition, we have seen that the LZW algorithm is better for image compression by preserving the necessary contents as it is a lossless compression technique. Hence, using CS and LZW with ECC will give a better image of En/Decryption. Its summarize on the following table (Table 2.2):

Table 2. 2 Summary of related works

Title	**Author(s), Year**	**Publisher**	**Contribution**	**Drawbacks**
A Novel Public Key Image Encryption Based on Elliptic Curves over Prime Group Field [5]	A. Soleymani et al., 2013	Journal of Image and Graphics	A new mapping method to convert each image pixel to a dot on an EC. This mapping technique is very fast with low complexity and computation, easy to implement.	Mapping and shared look-up table need additional cost. Not use any optimization. Image compression is not considered. Computationally very slow. Pixels into large integers are not considered.
JPEG mage encryption with ECC [14]	S. Bakhtiari et al., 2014	IEEE	They combined image compression with encryption. Enhance the decryption and the overall processing time.	Pixels to EC mapping process is more expensive. Not use any optimization. Only for JPEG. Computationally slow. Pixels into large integers are not considered.
Image Encryption using ECC [56]	D. Singh and M. Singh, 2015	Elsevier	They group pixels into large integers. They use digital signatures to provide authenticity and integrity.	Not use any optimization. Image compression is not considered. The cryptographic operations are also very large. Computationally slow.
Digital Image Encryption Algorithm Based on EC Public Cryptosystem [20]	X. Zhang and X. Wang, 2018	IEEE	They groups pixel values together and convert them into big integers. They used the DH algorithm for keys and necessary parameters sharing.	Not use any optimization. Image compression is not considered. The En/Decryption time is not efficient.

Chapter 3. The Proposed Enhanced Image Security Algorithm

3.1. Introduction

This chapter depicts the proposed enhanced image security using LZW compression and CS-based ECC image encryption and decryption. It encompasses the following components: LZW image compression, key generation, compressed byte values converted into large numbers, image encryption, cipher image decryption, and image decompression. It also deals with which encryption algorithm is selected and how compression and encryption processes are performed.

3.2. Architecture of LZWC-CS-ECC

The basic idea for the proposed LZWC-CS-ECC is to use LZW coding/decoding and CS algorithm to improve the security and speed performance of conventional ECC. Figure 3.1 shows the LZWC-CS-ECC algorithm procedures. The overall processes to be computed by the sender and receiver are as follows: On the sender side: compress the image, receive the receiver's public key, encrypt the coded image, and then send the encrypted image to the receiver. Again on the receiver side, receive the cipher image, then decrypt it, finally, uncompressed on the decrypted image. The detailed description of each step is presented from subsection 3.2.1 up to 3.2.4.

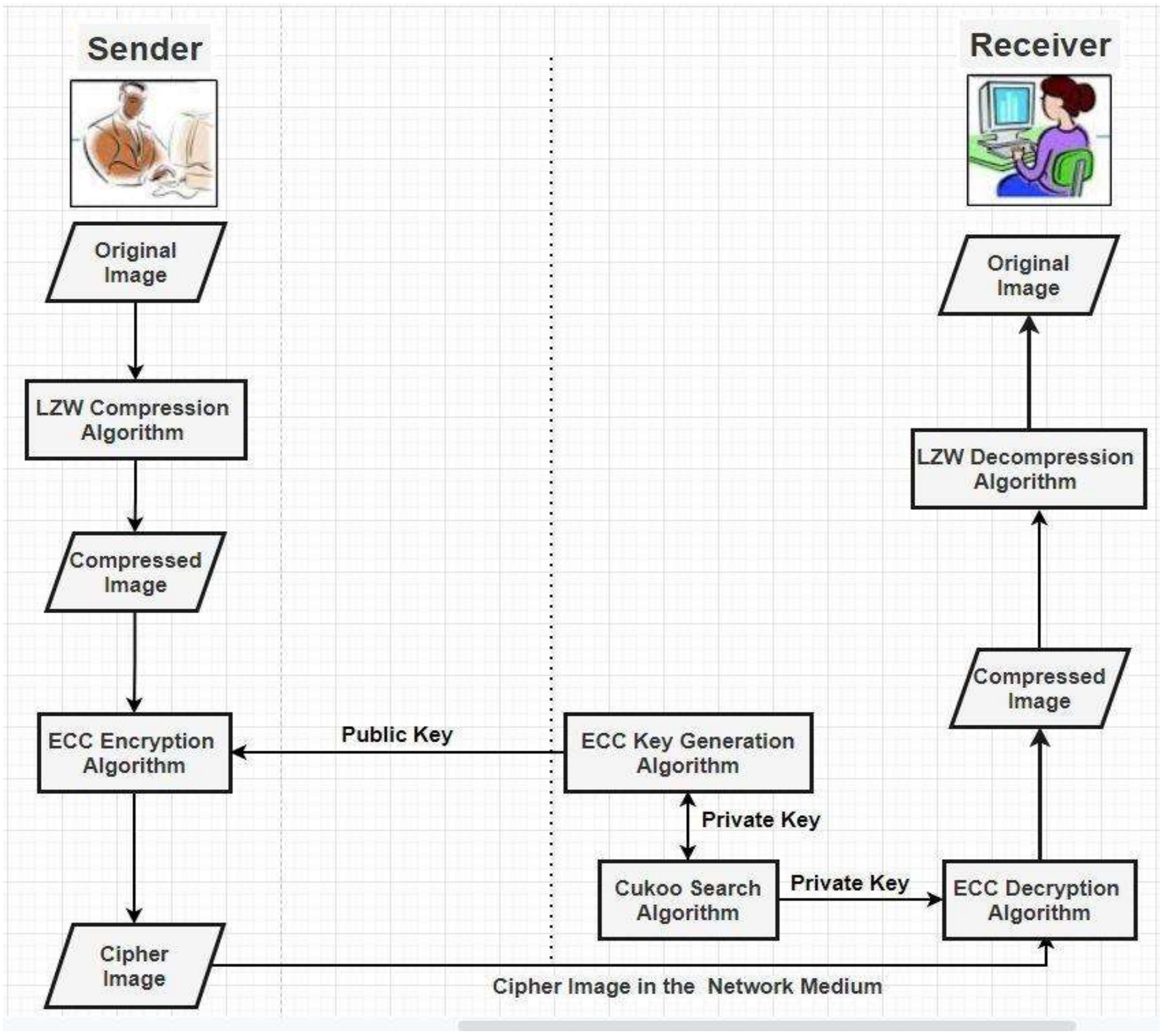


Figure 3. 1 Architecture of the LZWC-CS-ECC

3.2.1. LZW image Compression

To improve speed and security of image in ECC, reduce file size and coded in other formats using LZW compression. We use the dictionary size of 4096. The LZW compression process basic steps are as follows: First, read the image, then enter all pixels to the string table, then initialize the first pixel to string (The coding starts with an initial dictionary, which is enlarged with the arrival of new symbol sequences from the pixel, initialize table to contain single byte and then read first input byte, assign string (symbol)), then read the next input byte and check whether it is the similar byte with the previous one or not, if it exists, assigned the previous symbol, if not, assign new symbol, apply this steps until the end of the pixel. Finally, the LZW COMPRESSION gives the compressed coded String (byte) file.

Algorithm 3.1 LZW COMPRESSION Algorithm

```
LZW Compression ()
Input: Image
Read the image
Enter all image pixel to the table
Initialize pixel p to the first byte from the input pixel
While any input left
       Read Symbol s
       If p + s is in the table
              p = p + s
       else output codedword (p)
              enter p + s to the table
              p = s
       End if
End while
Output: codedword(p) - sequence of bytes - .lzw file
```

3.2.2. Key generation

The keys generation process basic steps to be computed by the receiver are as follows: To define a mathematical curve of a finite prime field equation (1), first select the prime number p with generator G, the coefficient value of a and b, then select the privet key (Private Key of Receiver (PRkr)) using CS algorithm, then compute the public key (Public Key of Receiver (PUkr)) its equal to G*PRkr, finally, sent defined curve and PUkr to sender. It computes the shared secret key by multiplying PUks and PRkr after getting the sender's public key (PUks). The generator of the curve G is the point whose point multiplication with different scalars produces every point on the curve. Further, we define the order of elliptic curve n as the smallest integer whose scalar point multiplication with generator G gives us the point at infinity O for the curve, i.e., nG = O. Under key generation the CS algorithm plays a great role to select the optimized privet key.

At the sender side, the simple keys generations are the following: First, get and hold the defined curve and PUkr of Receiver, then generate privet key (private Key of Sender (PRks)) using CS algorithm, then compute the public key (PUks) its equal to G* PRks and register inside the shared directory, finally compute the shared secret keys PRks multiplying through PUkr (Shard secret key = PRks * PUkr)

3.2.3. Cuckoo Search Algorithm

We use the CS algorithm to select the best and optimized random number for the private key of sender and receiver. The basic steps used in CS algorithm phases are initialization, generating a new cuckoo, fitness evaluation, updating, reject worst-case and stopping criteria. Algorithm 3.2 shows the detailed CS algorithm procedures.

Algorithm 3. 2 Cuckoo Search Algorithm

```
CS Algorithm ()
Input: Possible random numbers
Generate initial random number location
   eval = 0
While termination condition not meet do
    For i = 1 to Np do
        $x_i$ = generate new solution($x_i$)
        $f_i$ = evaluate the new solution($x_i$)
           eval = eval + 1
         j = [ rand(0, 1) * Np + 1]
        If $f_i < f_j$ then
           $x_j = x_i$; $f_j = f_i$; // replace j-th
           solution
        End if
        if rand(0, 1) < $p_a$ then
            init nest($x_{worst}$)
        End if
        if $f_i < f_{min}$ then
          $x_{best} = x_i$; $f_{min} = f_i$;
        End if
End while
Output: Private Key value of ($f_{min} = \min(f(x))$)
```

In this subsection, the main advantage of the CS algorithm is generating the best optimized random number for private keys and feeding it to the ECC public key generator. Both sender and receiver generate their private key using the CS algorithm.

3.2.4. Byte to Big Integer Conversion

The compressed LZW file is a sequence of bytes. We apply other good options to speed up our encryption and decryption process by grouping the compressed bytes. The basic steps to group bytes to big integer are as follow: First, create two BigInteger objects, then create two-byte arrays, then create and assign value to byte array b3, then assign byte array representation of bi1, bi2 to b1, b2, then print byte array b1 using for loop and finally print byte array b2 using for loop. Input to encryption is extremely minimized image using compression and grouping to Big Integer. The reverse process is done in the decryption process.

3.2.5. Encryption and Decryption Process

Encryption and decryption LZWC-CS-ECC are as follows: when the sender receives the receiver public key (PUkr), the generator point G and the order n, then compute the cipher image. During encryption steps: First, read the big integer sequence (bis) and encrypt the Pbis using a public key. The cipher image (Ci) by splitting two messages (C1 and C2), C1 = k*G and C2 = (Pbis + k*PUkr). k is the randomly chosen integer. Then it sends to the receiver.

During decryption step, the receiver received cipher image and decrypt to the big integer using the private key (PRkr). It received two points C1 and C2. First, multiply the first point by the private key (PRkr) to get the shared secret key (SSk), then to get the original Pbis to subtract the shared secret key(SSk) from the second point of cipher image (C2). So Pbis is equal to k*PUkr - SSk. Finally, to recover the original image apply the reverse order of byte grouping and decompress it.

Algorithm 3. 3 LZWC-CS-ECC En/Decryption algorithm

```
LZWC-CS-ECC_ Encryption ()
Input: a,b, n, p, G, PUkr and Coded big integer sequence(bis)
Procedure:
    Read big integer sequence on elliptic curve - Pbis
    Encrypt Pbis
    Obtain the ciphertext $C_i$
    Compute ciphertext by splitting two message (C1 and C2),
         C1 = k*G
         C2 = Pbis + k*PUkr
         C represent the encrypted message
Output: Ciphertext $C_1$ and $C_2$

LZWC-CS-ECC_ Decryption ()
Input: Ciphertext $C_1$ and $C_2$
```

```
Procedure:
   Get the ciphertext C1 and C2
   Decrypt plainImage,
       Pbis = k*PUkr - SSk
Output: big integer sequence
```

3.2.6. LZW Decompression

On the receiver side the after decryption, the inverse procedure LZW COMPRESSION process is to be computed by the receiver. During decompression step: First, received decrypted coded byte file. Then enter it into the table, then read the first byte and generate the corresponding pixel, then read the next byte if it is not in the dictionary, we just add the previous byte in the entry, if it is present, we form a word with the previous word and the first pixel of the current pixel and add it in a new entry, then repeat it until the end of the table. Finally, it gives the original image. Algorithm 3.4

Algorithm 3. 4 LZW Decompression Algorithm

```
LZW Decompression ()
Input: sequence of bytes (compressed .lzw file)
Read the codedword bytes
Enter all codedword bytes to the table
Read priorcodedword and output one symbol(byte) corresponding
to it
While codedwords are still left
       Read codedword
       If codedword is not in the table
// special case: s+p+s+p+s, also if p is null
      Enter in table string(priorcodedword)
      + firstchar(symbol(priorcodedword))
        Output symbol(priorcodedword)
        + firstchar(symbol(priorcodedword))
       else enter in table symbol(priorcodedword)
        + firstchar(symbol(codedword))
        output symbol(codedword)
       End if
End while
Output: priorcodedword - Original image
```

Chapter 4. Implementation and Performance Evaluation

This chapter presents the implementation of LZWC-CS-ECC, performance analysis and result discussions are explained.

4.1. Implementation

To conduct our experiment, we have used Net Beans IDE 8.2 Java programming environment installed on windows 10 that was running on Intel-R, Core-TM i5, CPU 2.3-GHz, 64-bit processor with 4 GB of RAM. We have chosen six different bit sizes: 28-bit, 128-bit, 256-bit, 512-bit, 1024-bit and 2048-bit for processing time, key generation, encryption and decryption. The image is first compressed through the LZW algorithm and then encrypted using ECC with CS private key selection. The decryption algorithm after decompression correctly recovers the original image.

4.2. Performance Analysis and Discussion

The performance improvement of the LZWC-CS-ECC algorithm is compared to standard ECC Image by considering speed performance parameters: key generation time, encryption time, decryption time, and overall processing time of the algorithms.

To simulate the speed performance of the algorithms, we have used five different size images; 193KB, 258KB, 1,729KB, 3,073KB and 6,913KB, six different bit-sizes; 28, 128, 256, 512, 1024 and 2048, and different combinations of randomly chosen key pairs. To make our result more reliable for the analysis, we have executed the algorithms five times for each input and the average execution time is considered as shown from Table 4.8 up to Table 4.10 in Annex B.1 up to Annex B.5. Then based on these tables, we have plotted graphs: Figure 4.1, Figure 4.4 and Figure 4.5 shown in this section.

4.1.1 Key Generation Time of Algorithms

For evaluating the key generation performance of algorithms, we have used the six different bit sizes. By running the simulation 5 times, we have summarized the key generation time of algorithms as shown in Table 4.1. As shown in Table 4.1, when compared to key generation time costs, LZWC-CS-ECC is better than standard ECC.

Table 4.1 Key Generation Time (in Seconds)

Bit size / algorithm	28bit	128bit	256bit	512bit	1024bit	2048bit
ECC	1.0953504	1.154155	1.15244	1.389163	1.991185	2.5065
LZWC-CS-ECC	1.1939976	1.117847	0.958987	1.179261	1.262496	1.66351

Based on key generation time Table 4.1, we have organized a comparison table: Table 4.2. Table 4.2 shows that the key generation performance of LZWC-CS-ECC improves the key generation performance of standard ECC by 23.6%.

Table 4.2 Key Generation Time Comparison of LZWC-CS-ECC with ECC (%)

Bit size / Algorithm	28bit	128bit	256bit	512bit	1024bit	2048bit	Average
ECC	-8%	3%	20%	18%	58%	51%	23.6%

We have generated Figure 4.1 from key generation time Table 4.1. From this figure, it is observed that the key generation time performance of LZWC-CS-ECC is far better than standard ECC. Therefore, LZWC-CS-ECC key generation complexity is less than ECC because of using CS algorithm.

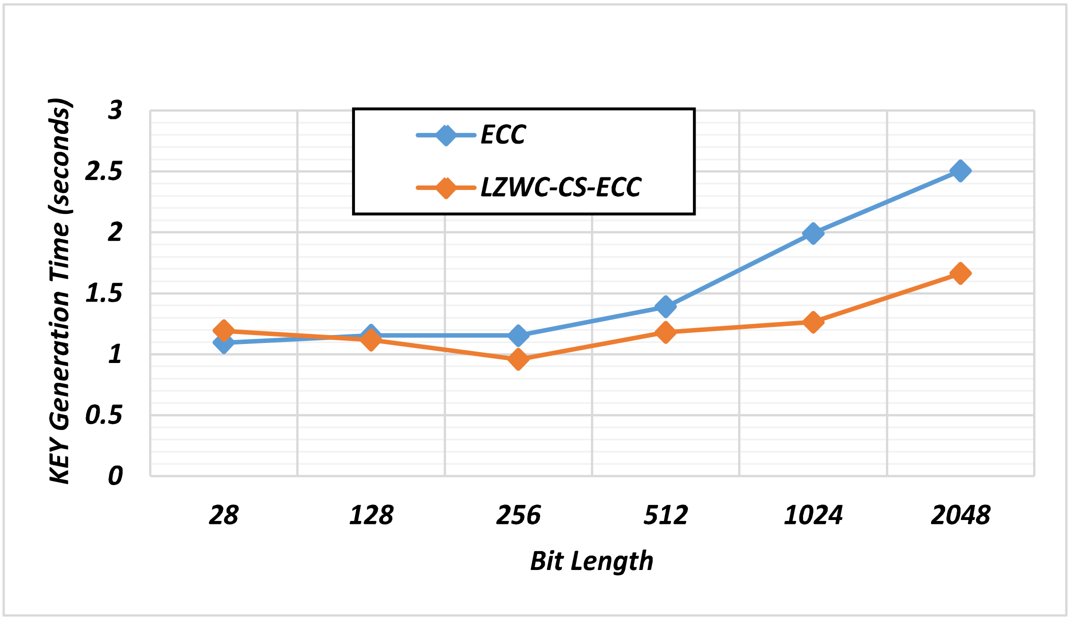


Figure 4. 1 Analysis of Key Generation Performance (seconds)

4.1.2 Encryption Time of Algorithms

For evaluating the encryption performance of algorithms, we have used the six different bit sizes and 258KB images. By running the simulation 5 times, we have summarized the encryption time of algorithms as shown in Table 4.3. As shown in Table 4.3, when compared to encryption time cost, LZWC-CS-ECC outperforms standard ECC.

Table 4.3 Encryption Time (in Seconds)

Bit size/ algorithm	**28bit**	**128bit**	**256bit**	**512bit**	**1024bit**	**2048bit**
ECC	6.1462471	6.1235497	6.312986	6.899473	8.030719	7.9976935
LZWC-CS-ECC	2.9516911	3.1507045	3.36389	3.713022	4.017537	4.0345631

Based on encryption time Table 4.3, we have analyzed Table 4.4. From Table 4.4, it is found that LZWC-CS-ECC has improved the encryption performance of standard ECC by 95.7%.

Table 4.4 Encryption Time Comparison of LZWC-CS-ECC with ECC (%)

Bit size algorithm	**28bit**	**128bit**	**256bit**	**512bit**	**1024bit**	**2048bit**	**Average**
ECC	108%	94%	88%	86%	100%	98%	95.7%

Based on Table 4.3, we have analyzed Figure 4.2. This figure shows that LZWC-CS-ECC encryption performance is far better than the standard ECC algorithm.

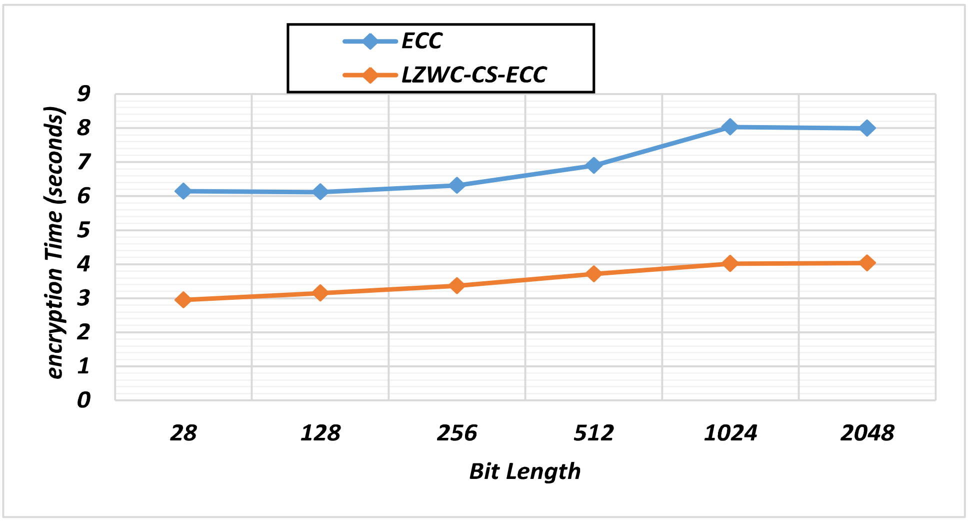


Figure 4.2 Analysis of Encryption Performance (in seconds)

4.1.3 Decryption Time of Algorithms

For evaluating the decryption performance of algorithms, we have used six different bit sizes and 258KB images. By running the simulation 5 times, we have summarized the decryption time of algorithms as shown in Table 4.5. As shown in Table 4.5, when we compared the decryption time cost, LZWC-CS-ECC is far better than standard ECC.

Table 4.5 Decryption Time (in Seconds)

Bit size / algorithm	**28bit**	**128bit**	**256bit**	**512bit**	**1024bit**	**2048bit**
ECC	11.7222402	12.2638626	13.72885	13.7354	13.7354	15.95815
LZWC-CS-ECC	6.6275945	5.751773	5.716506	7.210619	7.57155	7.581441

Based on the decryption time in Table 4.5, we have analyzed Table 4.6. Table 4.6 shows that the decryption speed of LZWC-CS-ECC is higher 102.1% than the standard ECC which makes LZWC-CS-ECC more cost-effective than the standard ECC algorithm.

Table 4.6 Decryption Time Comparison of LZWC-CS-ECC with ECC (%)

Bit size / algorithm	**28bit**	**128bit**	**256bit**	**512bit**	**1024bit**	**2048bit**	**Average**
ECC	77%	113%	140%	90%	81%	110%	102.1%

Based on decryption time Table 4.5, we have analyzed Figure 4.3. The Figure shows that the decryption speed of LZWC-CS-ECC is better than the standard ECC image decryption algorithm compared with. Especially, as the bit length of keys used and image size increase our algorithm performance when compared with standard ECC is more significant.

The Figure shows decryption speed of LZWC-CS-ECC is highly better than standard ECC, due to the LZW COMPRESSION process to improve the encryption and decryption performance of our algorithm.

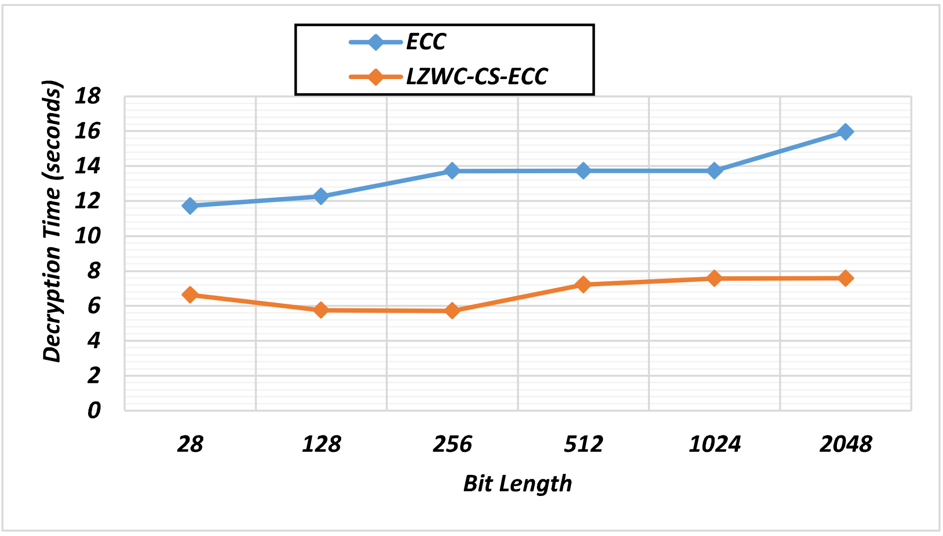


Figure 4.3 Analysis of Decryption Performance (in seconds)

4.1.4 Overall Processing Time based on Bit Length

For evaluating the overall processing time performance of algorithms, we have used the six different bit sizes and 258KB images. By running the simulation 5 times, we have summarized the overall processing time of algorithms as shown in Table 4.7. As shown in Table 4.7, when we compared the processing time cost, LZWC-CS-ECC outperforms standard ECC.

Table 4.7 Overall Processing Time based on Bit Length (in Seconds)

Bit size / algorithm	**28bit**	**128bit**	**256bit**	**512bit**	**1024bit**	**2048bit**
ECC	12.5	13.6	13.8	13.8	13.9	15.2
LZWC-CS-ECC	9.4	9.2	9	9	9.6	10.6

Based on the overall processing time in Table 4.7, we have analyzed Table 4.8. Table 4.8 shows that the overall processing speed of LZWC-CS-ECC is 45.9% higher than the standard ECC which makes LZWC-CS-ECC more cost-effective than the standard ECC algorithm.

Table 4.8 Overall Processing Time Comparison of LZWC-CS-ECC with ECC based on Bit Length (%)

Bit size / algorithm	**28bit**	**128bit**	**256bit**	**512bit**	**1024bit**	**2048bit**	**Average**
ECC	33%	48%	53%	53%	45%	43%	45.9%

Based on the overall processing time in Table 4.7, we have analyzed Figure 4.4. The Figure shows that the overall processing speed of LZWC-CS-ECC is better than the standard ECC image processing algorithm compared with different bit lengths.

The Figure shows the overall processing speed of LZWC-CS-ECC is highly better than standard ECC, due to the CS-based key generation process and LZW image compression process to improve the overall speed performance of our algorithm for different bit lengths.

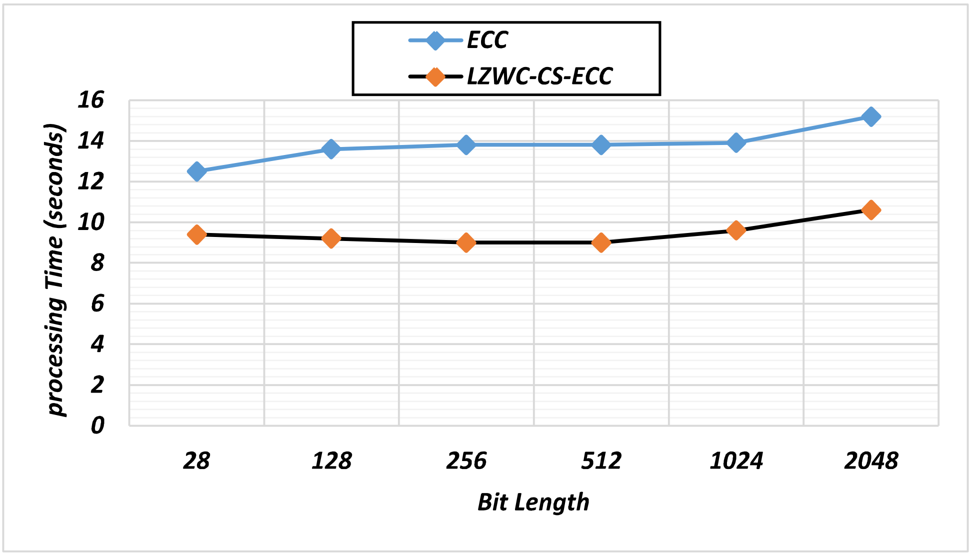


Figure 4.4 Analysis of Overall Speed Performance based on Bit Length (in seconds)

4.1.5 Overall Processing Time Based on Image Size

For evaluating the overall processing time performance of algorithms, we have used the five different size images and 512-bit length. By running the simulation 5 times, we have summarized the overall processing time of algorithms as shown in Table 4.9. As shown in Table 4.9, when we compared the overall processing time cost based on image size, LZWC-CS-ECC outperforms standard ECC.

Table 4.9 Overall Processing Time based on Image Size (in Seconds)

Bit size / algorithm	**193KB**	**258KB**	**1,729KB**	**3,073KB**	**6,913KB**
ECC	9.333	14.333	179.254	391.322	1766.312
LZWC-CS-ECC	4.25	9.667	79.789	179.546	709.787

Based on the overall processing time in Table 4.9, we have analyzed Table 4.10. Table 4.10 shows that the overall processing speed of LZWC-CS-ECC is 111.9% better than standard ECC which makes LZWC-CS-ECC more cost-effective than the standard ECC algorithm.

Table 4.10 Overall Processing Time Comparison of LZWC-CS-ECC with ECC based on Image Size (%)

Bit size / algorithm	193KB	258KB	1,729KB	3,073KB	6,913KB	Average
ECC	120%	48%	125%	118%	149%	111.9%

Based on the overall processing time in Table 4.9, we have analyzed Figure 4.5. The Figure shows that the overall processing speed of LZWC-CS-ECC is better than the standard ECC image processing algorithm compared with.

The Figure shows overall processing speed of LZWC-CS-ECC for different image sizes is highly better than standard ECC, due to the LZW image compression process to improve the overall speed performance of our algorithm.

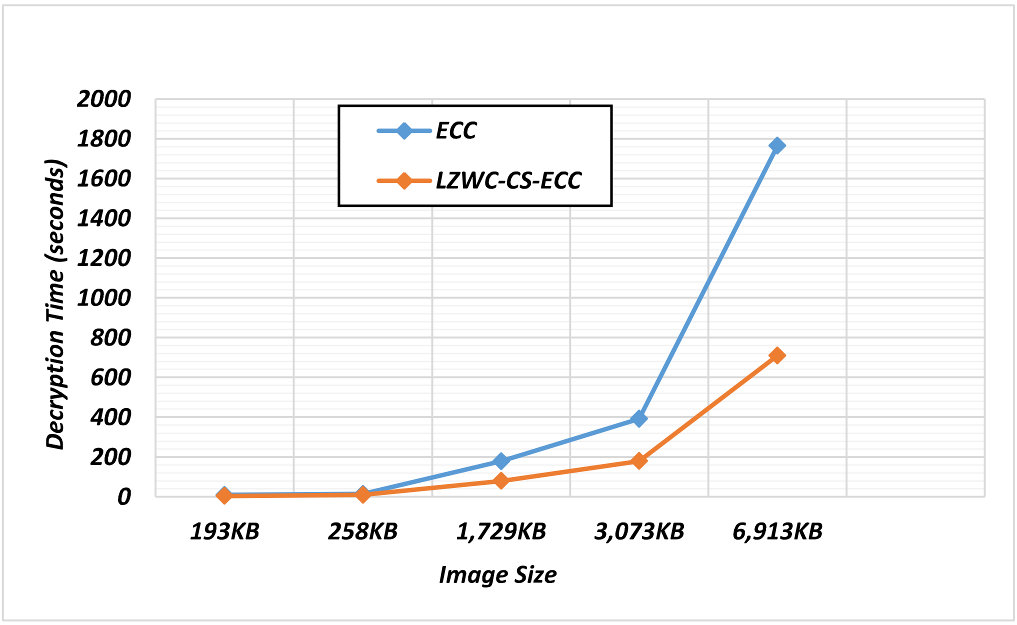


Figure 4.5 Analysis of Overall Speed Performance based on Image Size (in seconds)

4.1.6 Key Space

The size of the key used determines how secure an En/Decryption algorithm is. The larger the key size, the more difficult it is to carry out a Brute Force attack. Concerning the key size, ECC gives an exponentially complex Elliptic Curve Discrete Logarithmic Problem. It is one of the most difficult problems in mathematics, and there is currently no approach that has been successfully deciphered. We utilized a 512-bit standard Elliptic curve in our implementation, which has a large enough key size to ensure the required security. In LZWC-CS-ECC key paces is 2^{512}, so it is sufficiently vast to withstand a brute-force attack [20][56].

4.1.7 Key Sensitivity

Key sensitivity shows the dependence of the encryption scheme on cipher keys. There are two ways of determining the key's sensitivity. First, a 1-bit change in cipher key must produce an entirely different cipher image, and the second one is a 1-bit change in cipher key must decrypt an entirely random image. The original image was recovered from the cipher image using the correct private key of the receiver only. The decrypted image with the wrong key is just one digit different from the original key is completely different. As a result, it's been showed that the proposed LZWC-CS-ECC En/Decryption is highly sensitive to the key, such that even a near-perfect estimate of the keys yields no information about the plain image.

4.1.8 Security Strength Analysis

Table 4.11 indicates, the security strength of LZWC-CS-ECC is significantly higher than the standard ECC and and a new recent variant of ECC because it uses the cuckoo search algorithm to improve private keys selection and compression, as well as byte to Big integer conversion.

Table 4.11 Security Strength Comparison of Algorithms

Algorithm	Key Length, Key Space	Speed	Comparative security level	Reason
ECC	192, 2^{192}	Slow	Moderate	DLP only
A Novel Public Key Image Encryption Based on Elliptic Curves over Prime Group Field [5]	128, 2^{128}	Moderate	Moderate	DLP + fast mapping technique
JPEG mage encryption with ECC [14]	64, 2^{64}	Fast	Moderate	DLP + Compression
Image Encryption using ECC [56]	512, 2^{512}	Fast	High	DLP + Compression + Byte to BigInterger
Digital Image Encryption Algorithm Based on EC Public Cryptosystem [20]	512, 2^{512}	Fast	High	DLP + Compression + Byte to BigInterger
LZWC-CS-ECC	512, 2^{512}	Very Fast	Very High	DLP + Optimal key + Compression + Byte to BigInterger

Chapter 5. Conclusions and Future Work

5.1. Conclusions

In this paper, we have proposed the LZWC-CS-ECC algorithm. We discovered an issue with the private key selection process and a processing time challenge on hug data encryption processes, such as image data, as we analyzed existing systems. To address our research problems, the proposed LZWC-CS-ECC algorithm uses the LZWC and the CS algorithm to reduce the size of the image for encryption and to select an optimal private key by minimizing the selection loop for advancing security and speeding up the decryption process.

From our result analysis, we have found that our LZWC-CS-ECC algorithm has better performance than the existing ECC image encryption.

Generally, improved algorithm strength, key generation and decryption speed of our LZWC-CS-ECC algorithm because of CS algorithm, and improved encryption speed of our LZWC-CS-ECC algorithm because of LZW algorithm makes it more secured and fast. As a result, our proposed LZWC-CS-ECC algorithm can be implemented in image security demanding environments like Medical centers and defenses.

5.2. Future Work

As future work, one can extend our algorithm with other multimedia like voice and video and implementing our algorithm in working environments. Applying both lossy and lossless compression algorithms to separate regions of a single image can also be done to extend this work.

References

[1] M. Tayel, G. Dawood, and H. Shawky, “A Proposed Serpent-Elliptic Hybrid Cryptosystem for Multimedia Protection,” *2018 Int. Conf. Adv. Comput. Commun. Informatics, ICACCI 2018*, pp. 387–391, 2018, doi: 10.1109/ICACCI.2018.8554950.

[2] B. Koziel, R. Azarderakhsh, M. Mozaffari Kermani, and D. Jao, “Post-Quantum Cryptography on FPGA Based on Isogenies on Elliptic Curves,” *IEEE Trans. Circuits Syst. I Regul. Pap.*, vol. 64, no. 1, pp. 86–99, 2017, doi: 10.1109/TCSI.2016.2611561.

[3] K. Keerthi and B. Surendiran, “Elliptic curve cryptography for secured text encryption,” *Proc. IEEE Int. Conf. Circuit, Power Comput. Technol. ICCPCT 2017*, 2017, doi: 10.1109/ICCPCT.2017.8074210.

[4] J. Stephy and V. Subramaniyaswamy, “Analysis of digital image data hiding techniques,” *Proc. Int. Conf. I-SMAC (IoT Soc. Mobile, Anal. Cloud), I-SMAC 2018*, pp. 140–144, 2019, doi: 10.1109/I-SMAC.2018.8653794.

[5] A. Soleymani, M. J. Nordin, and Z. M. Ali, “A Novel Public Key Image Encryption Based on Elliptic Curves over Prime Group Field,” *J. Image Graph.*, vol. 1, no. 1, pp. 43–49, 2013, doi: 10.12720/joig.1.1.43-49.

[6] H. Hu, Y. Cao, J. Xu, C. Ma, and H. Yan, “An Image Compression and Encryption Algorithm Based on the Fractional-Order Simplest Chaotic Circuit,” *IEEE Access*, vol. 9, pp. 22141–22155, 2021, doi: 10.1109/ACCESS.2021.3054842.

[7] Z. Liu, L. Meng, Y. Tan, J. Zhang, and H. Zhang, “Image compression based on octave convolution and semantic segmentation,” *Knowledge-Based Syst.*, vol. 228, p. 107254, 2021, doi: 10.1016/j.knosys.2021.107254.

[8] W. Khalaf, A. S. Mohammad, and D. Zaghar, “Chimera: A new efficient transform for high quality lossy image compression,” *Symmetry (Basel).*, vol. 12, no. 3, 2020, doi: 10.3390/sym12030378.

[9] K. Rajasekaran, P. D. Sathya, and V. P. Sakthivel, “Quasi-lossless based fractal image compression using krill herd algorithm,” *Int. J. Adv. Trends Comput. Sci. Eng.*, vol. 9, no. 5, pp. 7763–7769, 2020, doi: 10.30534/ijatcse/2020/121952020.

[10] Y. Manjula and K. B. Shivakumar, “Enhanced secure image steganography using double encryption algorithms,” *Proc. 10th INDIACom; 2016 3rd Int. Conf. Comput. Sustain. Glob. Dev. INDIACom 2016*, vol. 7, pp. 705–708, 2016.

[11] C. H. Lin, Y. Xie, and W. Wolf, “LZW-based code compression for VLIW embedded

systems," *Proc. -Design, Autom. Test Eur. DATE*, vol. 3, pp. 76–81, 2004, doi: 10.1109/DATE.2004.1269210.

[12] X. Fang and Y. Wu, "Investigation into the elliptic curve cryptography," *2017 3rd Int. Conf. Inf. Manag. ICIM 2017*, no. i, pp. 412–415, 2017, doi: 10.1109/INFOMAN.2017.7950418.

[13] S. Behnia, A. Akhavan, A. Akhshani, and A. Samsudin, "Image encryption based on the Jacobian elliptic maps," *J. Syst. Softw.*, vol. 86, no. 9, pp. 2429–2438, 2013, doi: 10.1016/j.jss.2013.04.088.

[14] S. Bakhtiari, S. Ibrahim, M. Salleh, and M. Bakhtiari, "JPEG mage encryption with Elliptic Curve Cryptography," *Proc. - 2014 Int. Symp. Biometrics Secur. Technol. ISBAST 2014*, pp. 144–149, 2015, doi: 10.1109/ISBAST.2014.7013111.

[15] M. Bafandehkar, S. M. Yasin, R. Mahmod, and Z. M. Hanapi, "Comparison of ECC and RSA algorithm in resource constrained devices," *2013 Int. Conf. IT Converg. Secur. ICITCS 2013*, pp. 0–2, 2013, doi: 10.1109/ICITCS.2013.6717816.

[16] Shankar K. and Eswaran P., "Australian Journal of Basic and Applied Sciences A Secure Visual Secret Share (VSS) Creation Scheme in Visual Cryptography using Elliptic Curve Cryptography with Optimization Technique," vol. 9, no. December, pp. 150–163, 2015.

[17] S. Kota, V. N. R. Padmanabhuni, K. Budda, and K. Sruthi, "Authentication and Encryption Using Modified Elliptic Curve Cryptography with Particle Swarm Optimization and Cuckoo Search Algorithm," *J. Inst. Eng. Ser. B*, vol. 99, no. 4, pp. 343–351, 2018, doi: 10.1007/s40031-018-0324-x.

[18] D. Majumdar and S. Mallick, "Cuckoo search algorithm for constraint satisfaction and optimization," *Proc. - 2016 2nd IEEE Int. Conf. Res. Comput. Intell. Commun. Networks, ICRCICN 2016*, pp. 235–240, 2017, doi: 10.1109/ICRCICN.2016.7813662.

[19] G. Thangavel and R. Athiyappan, "Cuckoo Search based Optimal Elliptic Curve Cryptography (OECC) for Text Encryption," *IOSR J. Comput. Eng.*, vol. 18, no. 04, pp. 77–81, 2016, doi: 10.9790/0661-1804027781.

[20] X. Zhang and X. Wang, "Digital Image Encryption Algorithm Based on Elliptic Curve Public Cryptosystem," *IEEE Access*, vol. 6, pp. 70025–70034, 2018, doi: 10.1109/ACCESS.2018.2879844.

[21] N. A. Wahid, A. Ali, B. Esparham, and M. Marwan, "A Comparison of Cryptographic

Algorithms: DES, 3DES, AES, RSA and Blowfish for Guessing Attacks Prevention," *J. Comput. Sci. Appl. Inf. Technol.*, vol. 3, no. 2, pp. 1–7, 2018, [Online]. Available: www.symbiosisonline.orgwww.symbiosisonlinepublishing.com.

[22] T. Kumar, Yogesh Bala, "Asymmetric Algorithms and Symmetric Algorithms: A Review Tannu Bala," *Int. J. Comput. Appl.*, no. Icaet, pp. 975–8887, 2015.

[23] M. B., G. Holi, and S. Murthy, "An Overview of Image Security Techiques," *Int. J. Comput. Appl.*, vol. 154, no. 6, pp. 37–46, 2016, doi: 10.5120/ijca2016911834.

[24] N. S. Abraham and P. Nair, "Survey on Image Encryption , Data Hiding and Secret Fragment Visible Mosaic Image Creation Techniques," pp. 115–119, 2015.

[25] A. Hasnat, D. Barman, and S. Sarkar, "Color image share cryptography: A novel approach," *J. Intell. Fuzzy Syst.*, vol. 36, no. 5, pp. 4491–4506, 2019, doi: 10.3233/JIFS-179002.

[26] D. G. Singhavi and P. N. Chatur, "A new method for creation of secret-fragment-visible-mosaic image for secure communication," *ICIIECS 2015 - 2015 IEEE Int. Conf. Innov. Information, Embed. Commun. Syst.*, pp. 0–4, 2015, doi: 10.1109/ICIIECS.2015.7192929.

[27] M. Ubaidullah and Q. Makki, "A Review on Symmetric Key Encryption Techniques in Cryptography," *Int. J. Comput. Appl.*, vol. 147, no. 10, pp. 43–48, 2016, doi: 10.5120/ijca2016911203.

[28] J. Singh, K. Lata, and J. Ashraf, "Image Encryption & Decryption with Symmetric Key Cryptography using MATLAB," *Int. J. Curr. Eng. Technol.*, vol. 5, no. 1, pp. 448–451, 2015, [Online]. Available: http://inpressco.com/category/ijcet.

[29] M. Islam, M. Shah, Z. Khan, T. Mahmood, and M. J. Khan, "A New Symmetric Key Encryption Algorithm Using Images as Secret Keys," *Proc. - 2015 13th Int. Conf. Front. Inf. Technol. FIT 2015*, pp. 1–5, 2016, doi: 10.1109/FIT.2015.12.

[30] M. B. Yassein, S. Aljawarneh, E. Qawasmeh, W. Mardini, and Y. Khamayseh, "Comprehensive study of symmetric key and asymmetric key encryption algorithms," *Proc. 2017 Int. Conf. Eng. Technol. ICET 2017*, vol. 2018-Janua, pp. 1–7, 2018, doi: 10.1109/ICEngTechnol.2017.8308215.

[31] Z. E. Dawahdeh, S. N. Yaakob, and R. Razif bin Othman, "A new image encryption technique combining Elliptic Curve Cryptosystem with Hill Cipher," *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 30, no. 3, pp. 349–355, 2018, doi: 10.1016/j.jksuci.2017.06.004.

[32] S. Chandra, S. Paira, S. S. Alam, and G. Sanyal, "A comparative survey of symmetric and asymmetric key cryptography," *2014 Int. Conf. Electron. Commun. Comput. Eng. ICECCE 2014*, pp. 83–93, 2014, doi: 10.1109/ICECCE.2014.7086640.

[33] M. H. Abood, "Steganography with RC4 and Pixel Shuffling Encryption Algorithms," no. March, pp. 7–9, 2017.

[34] A. Kaur and G. Singh, "A Random Selective Block Encryption Technique for Secure Image Cryptography Using Blowfish Algorithm," *Proc. Int. Conf. Inven. Commun. Comput. Technol. ICICCT 2018*, no. Icicct, pp. 1290–1293, 2018, doi: 10.1109/ICICCT.2018.8473273.

[35] A. Devi, A. Sharma, and A. Rangra, "A Review on DES, AES and Blowfish for Image Encryption & Decryption," *Int. J. Eng. Comput. Sci.*, vol. 4, no. 6, pp. 12646–12651, 2015, [Online]. Available: https://www.ijecs.in/index.php/ijecs/article/download/3887/3623/.

[36] Mohammad Ali Bani Younes, "Literature Survey on Different Technique of Image Encryption," *Int. J. Sci. Eng. Res.*, vol. 7, no. 1, pp. 93–98, 2016, [Online]. Available: https://www.ijser.org/researchpaper/Literature-Survey-on-Different-Techniques-of-Image-Encryption.pdf.

[37] R. Tripathi and S. Agrawal, "Comparative Study of Symmetric and Asymmetric Cryptography Techniques," *Int. J. Adv. Found. Res. Comput.*, vol. 1, no. 6, pp. 68–76, 2014.

[38] R. S. Jamgekar and G. S. Joshi, "File Encryption and Decryption Using Secure RSA," *Int. J. Emerg. Sci. Eng.*, vol. 1, no. 4, pp. 11–14, 2013.

[39] A. Karakra and A. Alsadeh, "A-RSA: Augmented RSA," *Proc. 2016 SAI Comput. Conf. SAI 2016*, pp. 1016–1023, 2016, doi: 10.1109/SAI.2016.7556103.

[40] A. Dua and A. Dutta, "A study of applications based on elliptic curve cryptography," *Proc. Int. Conf. Trends Electron. Informatics, ICOEI 2019*, vol. 0, no. Icoei, pp. 249–254, 2019, doi: 10.1109/ICOEI.2019.8862708.

[41] J. Lin *et al.*, "An Image Compression-Encryption Algorithm Based on Cellular Neural Network and Compressive Sensing," *2018 3rd IEEE Int. Conf. Image, Vis. Comput. ICIVC 2018*, pp. 673–677, 2018, doi: 10.1109/ICIVC.2018.8492882.

[42] M. Yang and N. Bourbakis, "An Overview of Lossless Digital Image Compression Techniques," pp. 1099–1102.

[43] C. Paper, “A Survey of Data Compression Algorithms and their Applications A Survey of Data Compression Algorithms and their Applications,” no. JANUARY 2012, 2015, doi: 10.13140/2.1.4360.9924.

[44] S. Funasaka, K. Nakano, and Y. Ito, “Fast LZW compression using a GPU,” 2015, doi: 10.1109/CANDAR.2015.20.

[45] M. E. Scholar, “A Biometric Iris Image Compression using LZW and Hybrid LZW Coding Algorithm,” 2017.

[46] Z. Tang, S. Xu, H. Yao, C. Qin, and X. Zhang, “Reversible data hiding with differential compression in encrypted image,” *Multimed. Tools Appl.*, vol. 78, no. 8, pp. 9691–9715, 2019, doi: 10.1007/s11042-018-6567-3.

[47] A. Razzaque, “Image Compression and Encryption : An Overview,” *Int. J. Eng. Res. Technol.*, vol. 1, no. 5, pp. 1–7, 2012.

[48] R. Ghose, T. Das, A. Saha, T. Das, and S. P. Chattopadhyay, “Cuckoo search algorithm for speech recognition,” *2015 Int. Conf. Work. Comput. Commun. IEMCON 2015*, vol. 3, no. 10, pp. 3540–3545, 2015, doi: 10.1109/IEMCON.2015.7344522.

[49] B. M. Ismail, B. Eswara Reddy, and T. Bhaskara Reddy, “Cuckoo inspired fast search algorithm for fractal image encoding,” *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 30, no. 4, pp. 462–469, 2018, doi: 10.1016/j.jksuci.2016.11.003.

[50] S. Amtade and T. Miyamoto, “Cuckoo search algorithm for job scheduling in cloud systems,” *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, vol. e98, no. 2, pp. 645–649, 2015, doi: 10.1587/transfun.E98.A.645.

[51] P. Sekhar and S. Mohanty, “An enhanced cuckoo search algorithm based contingency constrained economic load dispatch for security enhancement,” *Int. J. Electr. Power Energy Syst.*, vol. 75, pp. 303–310, 2016, doi: 10.1016/j.ijepes.2015.09.018.

[52] W. H. Yang, J. R. Liu, and Y. Zhang, “A new local-enhanced cuckoo search algorithm,” *Int. J. Comput. Sci. Math.*, vol. 8, no. 2, pp. 175–182, 2017, doi: 10.1504/IJCSM.2017.083756.

[53] K. Gupta and R. Singh, “Enhanced Secured Image Transfer over Internet,” no. April, pp. 112–115, 2015.

[54] S. Anandakumar, “Image Cryptography Using RSA Algorithm in Network Security,” vol. 5, no. 9, pp. 326–330, 2015.

[55] G. Noida, U. Pradesh, G. Noida, U. Pradesh, and U. Pradesh, “IMAGE ENCRYPTION

AND DECRYPTION USING ELLIPTIC CURVE CRYPTOGRAPHY," vol. 8354, no. 3, pp. 198–205, 2014.

[56] L. D. Singh and K. M. Singh, "Image Encryption using Elliptic Curve Cryptography," *Procedia Comput. Sci.*, vol. 54, pp. 472–481, 2015, doi: 10.1016/j.procs.2015.06.054.

[57] Z. Cui, B. Sun, G. Wang, Y. Xue, and J. Chen, "A novel oriented cuckoo search algorithm to improve DV-Hop performance for cyber–physical systems," *J. Parallel Distrib. Comput.*, vol. 103, no. October 2017, pp. 42–52, 2017, doi: 10.1016/j.jpdc.2016.10.011.

[58] F. I. Khandwani and P. E. Ajmire, "A Survey of Lossless Image Compression Techniques," *Int. J. Electr. Electron. Comput. Sci. Eng.*, vol. 5, no. 1, pp. 2348–2273, 2018, [Online]. Available: http://www.ijeecse.com/V5N1-013.pdf.

[59] M. A. Alam *et al.*, "Faster image compression technique based on LZW algorithm using GPU parallel processing," *2018 Jt. 7th Int. Conf. Informatics, Electron. Vis. 2nd Int. Conf. Imaging, Vis. Pattern Recognition, ICIEV-IVPR 2018*, no. 1, pp. 272–275, 2019, doi: 10.1109/ICIEV.2018.8640956.

[60] M. R. Hasan, M. I. Ibrahimy, S. M. A. Motakabber, M. M. Ferdaus, and M. N. H. Khan, "Comparative data compression techniques and multi-compression results," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 53, no. 1, 2013, doi: 10.1088/1757-899X/53/1/012081.

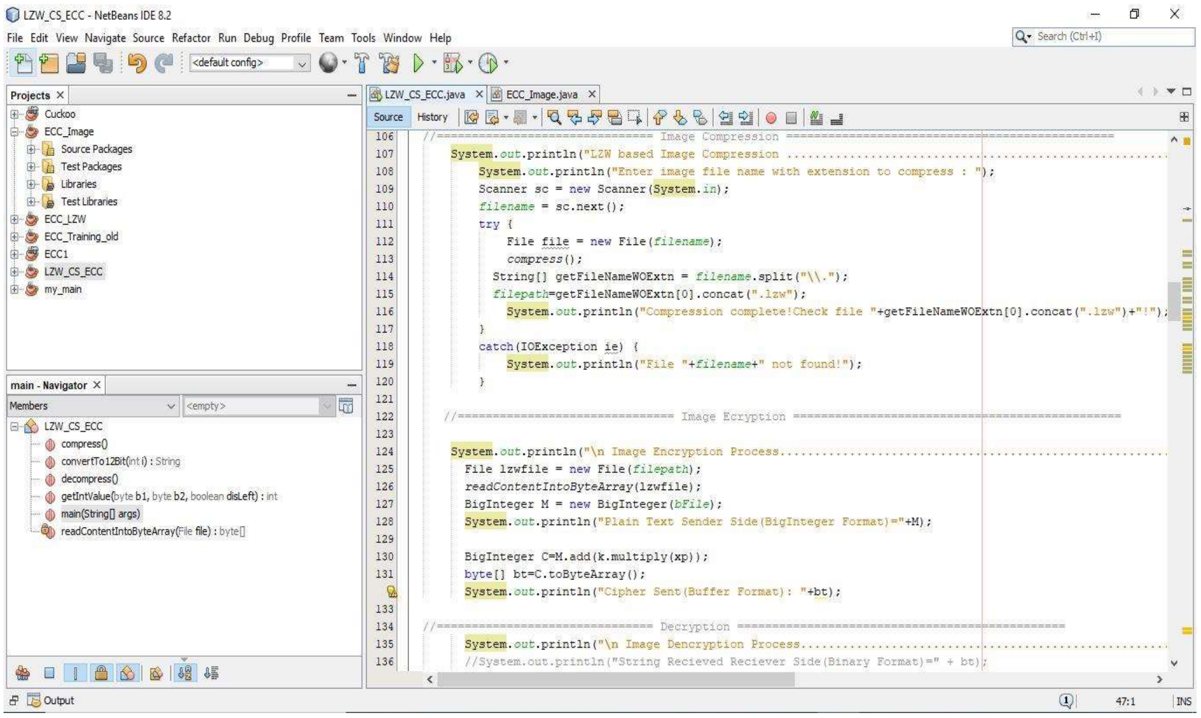
[61] K. A. Ramya and M. Pushpa, "A Survey on Lossless and Lossy Data Compression Methods," *Int. J. Comput. Sci. Eng. Commun.*, vol. 4, no. 1, pp. 1277–1280, 2016.

[62] T. Thulasi and P. Basu, "A lossless joint image compression using wavelet decomposition and hierarchical tree encoding," *2019 Int. Conf. Intell. Comput. Control Syst. ICCS 2019*, no. Iciccs, pp. 533–538, 2019, doi: 10.1109/ICCS45141.2019.9065487.

Annexes

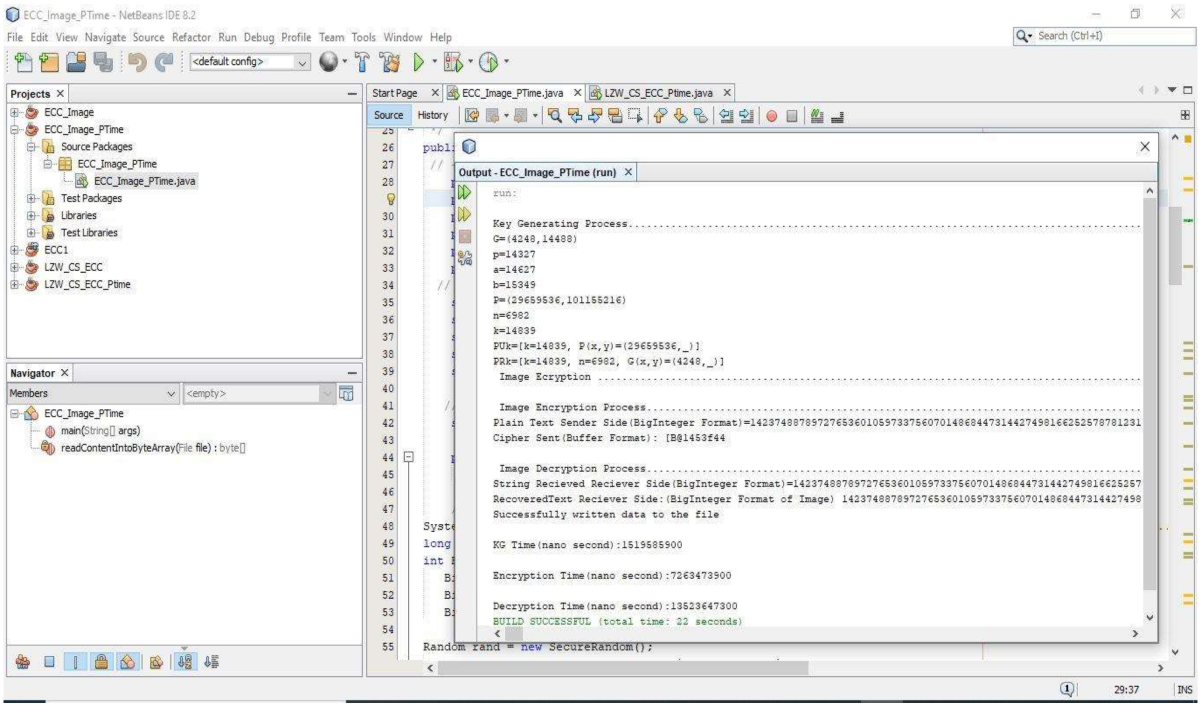
Annex A. Implementation Examples

A. LZWC-CS-ECC Implementation Screenshot



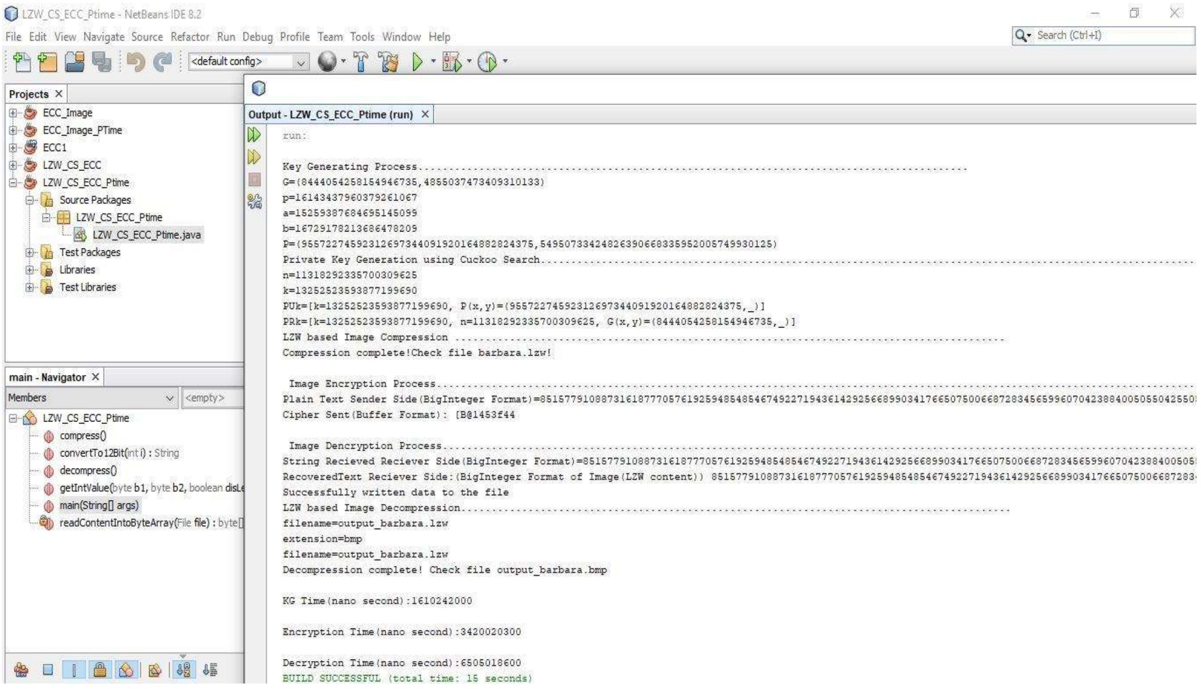
```
//============================= Image Compression ==============================================
    System.out.println("LZW based Image Compression ...................................
        System.out.println("Enter image file name with extension to compress : ");
        Scanner sc = new Scanner(System.in);
        filename = sc.next();
        try {
            File file = new File(filename);
            compress();
          String[] getFileNameWOExtn = filename.split("\\.");
          filepath=getFileNameWOExtn[0].concat(".lzw");
            System.out.println("Compression complete!Check file "+getFileNameWOExtn[0].concat(".lzw")+"!");
        }
        catch(IOException ie) {
            System.out.println("File "+filename+" not found!");
        }

   //============================== Image Ecryption ==============================================

    System.out.println("\n Image Encryption Process.........................................
      File lzwfile = new File(filepath);
      readContentIntoByteArray(lzwfile);
      BigInteger M = new BigInteger(bFile);
      System.out.println("Plain Text Sender Side(BigInteger Format)="+M);

      BigInteger C=M.add(k.multiply(xp));
      byte[] bt=C.toByteArray();
      System.out.println("Cipher Sent(Buffer Format): "+bt);

//============================== Decryption ==============================================
      System.out.println("\n Image Dencryption Process...........................................
      //System.out.println("String Recieved Reciever Side(Binary Format)=" + bt);
```

B. ECC Overall Speed Performance evaluation Screenshot



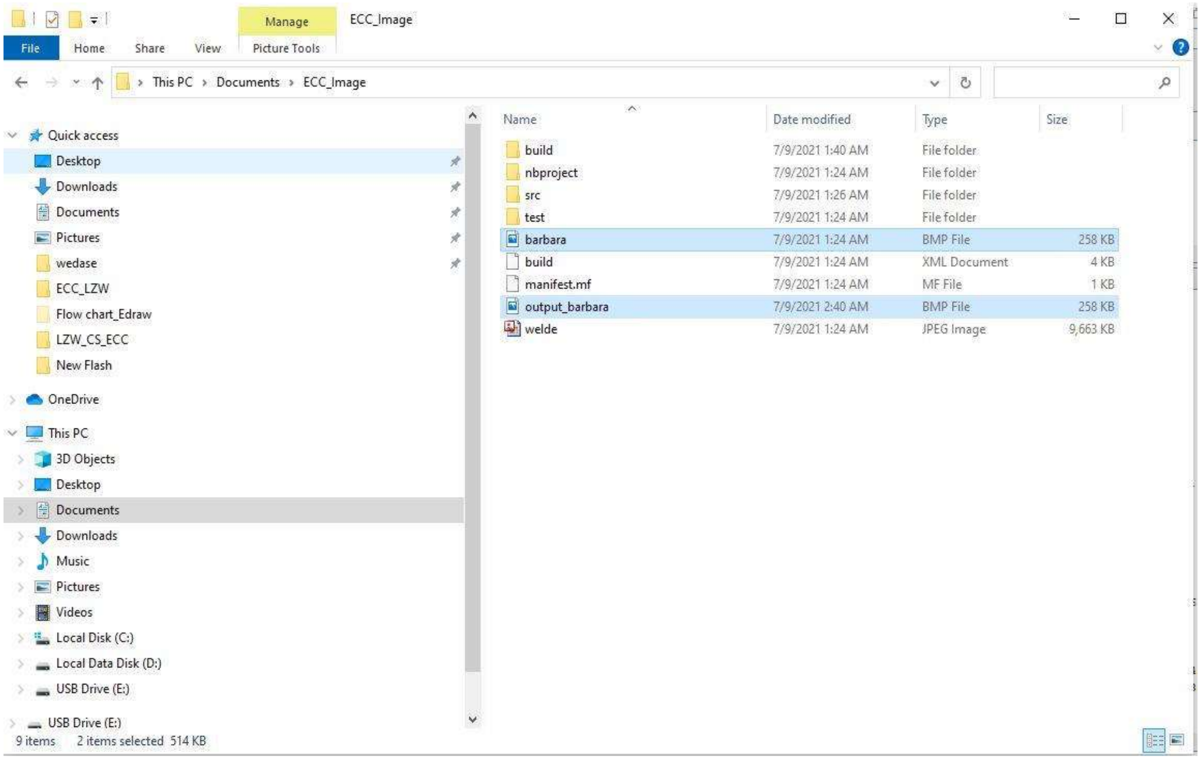
```
run:

Key Generating Process..........................................................................
G=(4248,14488)
p=14327
a=14627
b=15349
P=(29659536,101155216)
n=6982
k=14839
PUk=[k=14839, P(x,y)=(29659536,_)]
PRk=[k=14839, n=6982, G(x,y)=(4248,_)]
 Image Ecryption ...............................................................................

 Image Encryption Process.......................................................................
Plain Text Sender Side(BigInteger Format)=1423748878972765360105973375607014868447314427498166252578781231
Cipher Sent(Buffer Format): [B@1453f44

 Image Decryption Process.......................................................................
String Recieved Reciever Side(BigInteger Format)=142374887897276536010597337560701486844731442749816625257
RecoveredText Reciever Side:(BigInteger Format of Image) 1423748878972765360105973375607014868447314427498
Successfully written data to the file

KG Time(nano second):1519585900

Encryption Time(nano second):7263473900

Decryption Time(nano second):13523647300
BUILD SUCCESSFUL (total time: 22 seconds)
```

C. LZWC-CS-ECC Overall Speed Performance evaluation Screenshot



```
run:

Key Generating Process.............................................................................
G=(8444054258154946735,4855037473409310133)
p=16143437960379261067
a=15259387684695145099
b=16729178213686478209
P=(95572274592312697344091920164882824375,54950733424826390668335952005749930125)
Private Key Generation using Cuckoo Search..................................................................................
n=11318292335700309625
k=13252523593877199690
PUk=[k=13252523593877199690, P(x,y)=(95572274592312697344091920164882824375,_)]
PRk=[k=13252523593877199690, n=11318292335700309625, G(x,y)=(8444054258154946735,_)]
LZW based Image Compression .......................................................................
Compression complete!Check file barbara.lzw!

 Image Encryption Process.............................................................................................
Plain Text Sender Side(BigInteger Format)=85157791088731618777057619259485485467492271943614292566899034176650750066872834565996070423884005055042550
Cipher Sent(Buffer Format): [B@1453f44

 Image Dencryption Process............................................................................................
String Recieved Reciever Side(BigInteger Format)=851577910887316187770576192594854854674922719436142925668990341766507500668728345659960704238840050
RecoveredText Reciever Side:(BigInteger Format of Image(LZW content)) 8515779108873161877705761925948548546749227194361429256689903417665075006687283
Successfully written data to the file
LZW based Image Decompression.....................................................................
filename=output_barbara.lzw
extension=bmp
filename=output_barbara.lzw
Decompression complete! Check file output_barbara.bmp

KG Time(nano second):1610242000

Encryption Time(nano second):3420020300

Decryption Time(nano second):6505018600
BUILD SUCCESSFUL (total time: 15 seconds)
```

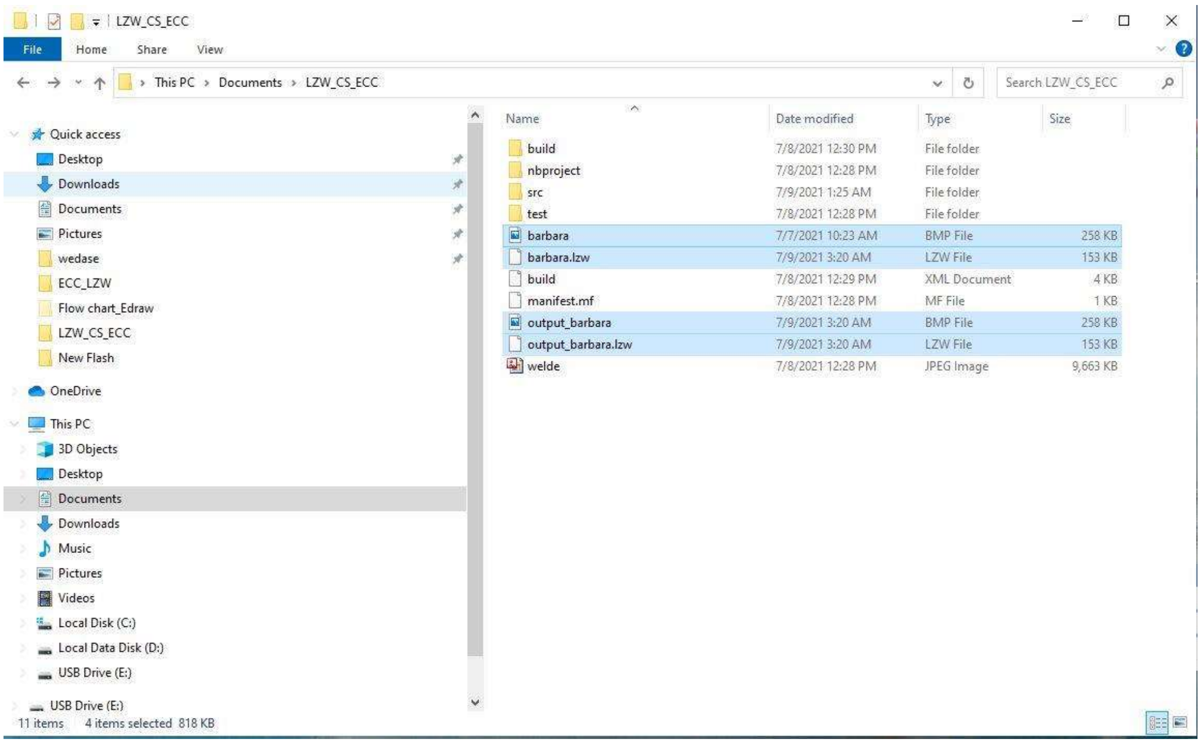
D. ECC Image En/Decryption Input/output Files Screenshot



Name	Date modified	Type	Size
build	7/9/2021 1:40 AM	File folder	
nbproject	7/9/2021 1:24 AM	File folder	
src	7/9/2021 1:26 AM	File folder	
test	7/9/2021 1:24 AM	File folder	
barbara	7/9/2021 1:24 AM	BMP File	258 KB
build	7/9/2021 1:24 AM	XML Document	4 KB
manifest.mf	7/9/2021 1:24 AM	MF File	1 KB
output_barbara	7/9/2021 2:40 AM	BMP File	258 KB
welde	7/9/2021 1:24 AM	JPEG Image	9,663 KB

9 items   2 items selected 514 KB

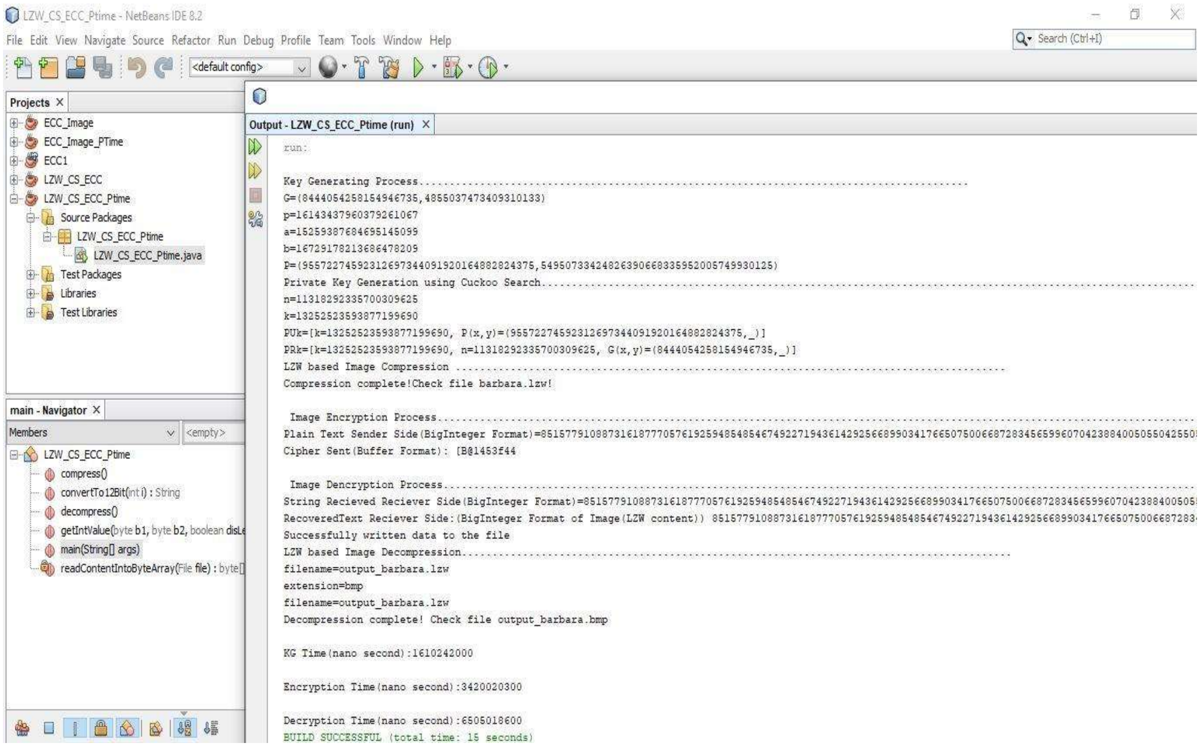
E. LZWC-CS-ECC Image En/Decryption Input/output Files Screenshot



Name	Date modified	Type	Size
build	7/8/2021 12:30 PM	File folder	
nbproject	7/8/2021 12:28 PM	File folder	
src	7/9/2021 1:25 AM	File folder	
test	7/8/2021 12:28 PM	File folder	
barbara	7/7/2021 10:23 AM	BMP File	258 KB
barbara.lzw	7/9/2021 3:20 AM	LZW File	153 KB
build	7/8/2021 12:29 PM	XML Document	4 KB
manifest.mf	7/8/2021 12:28 PM	MF File	1 KB
output_barbara	7/9/2021 3:20 AM	BMP File	258 KB
output_barbara.lzw	7/9/2021 3:20 AM	LZW File	153 KB
welde	7/8/2021 12:28 PM	JPEG Image	9,663 KB

11 items   4 items selected 818 KB

Annex B. Key Generation and En/Decryption Time screenshot



```
run:

Key Generating Process.......................................................................
G=(8444054258154946735,4855037473409310133)
p=16143437960379261067
a=15259387684695145099
b=16729178213686478209
P=(95572274592312697344091920164882824375,54950733424826390668335952005749930125)
Private Key Generation using Cuckoo Search......................................................................................
n=11318292335700309625
k=13252523593877199690
PUk=[k=13252523593877199690, P(x,y)=(95572274592312697344091920164882824375,_)]
PRk=[k=13252523593877199690, n=11318292335700309625, G(x,y)=(8444054258154946735,_)]
LZW based Image Compression ..............................................................................
Compression complete!Check file barbara.lzw!

 Image Encryption Process.........................................................................................
Plain Text Sender Side(BigInteger Format)=85157791088731618777057619259485485467492271943614292566899034176650750066872834565996070423884005055042550
Cipher Sent(Buffer Format): [B@1453f44

 Image Dencryption Process........................................................................................
String Recieved Reciever Side(BigInteger Format)=851577910887316187770576192594854854674922719436142925668990341766507500668728345659960704238840050
RecoveredText Reciever Side:(BigInteger Format of Image(LZW content)) 8515779108873161877705761925948548546749227194361429256689903417665075006687283
Successfully written data to the file
LZW based Image Decompression..............................................................................
filename=output_barbara.lzw
extension=bmp
filename=output_barbara.lzw
Decompression complete! Check file output_barbara.bmp

KG Time(nano second):1610242000

Encryption Time(nano second):3420020300

Decryption Time(nano second):6505018600
BUILD SUCCESSFUL (total time: 15 seconds)
```

Annex C. LZWC-CS-ECC Sample Java Code

```
import java.io.*;
import java.io.File;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.io.FileInputStream;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.*;
/** @author Woldeiyesus Ayele
... - Indicates the Remaining jumped codes here
*/

public class LZW_CS_ECC {
 // -------------- Compression Instace Variables-------------
--------------------------
  public static HashMap<String, Integer> dictionary = new
HashMap<>();
// variable declarations here
    public static void main(String[] args) {
  //============= Key Generation ============
   int BIT_LENGTH = 128;
    // Generate random primes
System.out.println("\nKey                          Generating
Process.................. ");
Random rand = new SecureRandom();
 BigInteger p= BigInteger.probablePrime(BIT_LENGTH/2, rand);
 BigInteger xg,yg;
 xg= new BigInteger(BIT_LENGTH/2,rand);
 yg= new BigInteger(BIT_LENGTH/2,rand);
   BigInteger a,b;
```

```
do{
 a= BigInteger.probablePrime(BIT_LENGTH/2, rand);
 b = BigInteger.probablePrime(BIT_LENGTH/2, rand);
}while(a.compareTo(one)<1     ||     b.compareTo(one)<1     ||
a.compareTo(b)==0                                            ||
((yg.pow(2).mod(p)).compareTo((xg.pow(3).add(xg.multiply(a)).
add(b.mod(p)))))==1);
//System.out.println("p= "+p);
 // Generate random primes
System.out.println("G=("+xg+","+yg+")");
//System.out.println("K=("+xk+","+yk+")");
//System.out.println("L=("+xl+","+yl+")");
System.out.println("p="+p);
//System.out.println("s="+s);
System.out.println("a="+a);
System.out.println("b="+b);

BigInteger n= new BigInteger(BIT_LENGTH/2,rand);
BigInteger xp= xg.multiply(n);
BigInteger yp= yg.multiply(n);

System.out.println("P=("+xp+","+yp+")");

//=======   Cuckoo   Search   Based   Private   Key   Generation
==============
BigInteger k;
do
{
k= new BigInteger(BIT_LENGTH/2,rand);
…
}while(n.compareTo(k)==1);
```

```
System.out.println("n="+n);
System.out.println("k="+k);
 ...
System.out.println("PUk=[k=" +k+ ", P(x,y)=(" +xp+ ",_)]");
System.out.println("PRk=[k=" +k+ ", n=" +n+ ", G(x,y)=(" +xg+
",_)]");
//======= Image Compression ==================
...
        Scanner sc = new Scanner(System.in);
        filename = sc.next();
        try {
            File file = new File(filename);
            compress();
          String[] getFileNameWOExtn = filename.split("\\.");
          filepath=getFileNameWOExtn[0].concat(".lzw");
System.out.println("Compression      complete!Check      file
"+getFileNameWOExtn[0].concat(".lzw")+"!");
        }
        catch(IOException ie) {
            System.out.println("File     "+filename+"     not
found!");
        }
 //========== Image Ecryption ==================
      ...
     BigInteger C=M.add(k.multiply(xp));
      byte[] bt=C.toByteArray();
      System.out.println("Cipher Sent(Buffer Format): "+bt);

//=========== Decryption ============
System.out.println("\n Image Dencryption Process............
");
      BigInteger Crecieved=new BigInteger(bt),Mlocal;
      System.out.println("String       Recieved       Reciever
Side(BigInteger Format)=" + Crecieved);
      Mlocal=Crecieved.subtract(n.multiply(k.multiply(xg)));
      System.out.println("RecoveredText               Reciever
Side:(BigInteger Format of Image(LZW content)) " + Mlocal);
       bt=Mlocal.toByteArray();
       byte[]          decryptedMessage          =          new
BigInteger(bt).toByteArray();
       String dmsg= new String(decryptedMessage);
       Path path = Paths.get("output_".concat(filepath));
        byte[] bytes = decryptedMessage;
          ...
//============== Image Decompression ========================
   String[] getFileNameWOExtn = filename.split("\\.");
   dfilename = "output_".concat(filepath);
   System.out.println("filename=" + dfilename);
          ...
  System.out.println("filename=" + dfilename);
            decompress();
        }
        catch(IOException ie) {
     System.out.println("File "+dfilename+" not found!");
        } }
// ======= Compression Functions ===========
public static void compress() throws IOException {
        int i,byteToInt;
        char C;
        // Character dictionary
        for(i=0;i<256;i++) {
            dictionary.put(Character.toString((char)i),i);
        }
// Read input file and output file
RandomAccessFile           inputFile           =           new
RandomAccessFile(filename,"r");
        String[] getFileNameWOExtn = filename.split("\\.");
        RandomAccessFile        outputFile         =         new
RandomAccessFile(getFileNameWOExtn[0].concat(".lzw"),"rw");
         try {
         ...
            if(byteToInt < 0) byteToInt += 256;
            C = (char) byteToInt;
            P = ""+C;
            while(true) {
                inputByte = inputFile.readByte();
                byteToInt = new Byte(inputByte).intValue();
                if(byteToInt < 0) byteToInt += 256;
                C = (char) byteToInt;

                // if P+C is present in dictionary
                if(dictionary.containsKey(P+C)) {
                    P = P+C;
                }
      else {
      BP = convertTo12Bit(dictionary.get(P));
          if(isLeft) {
buffer[0] = (byte) Integer.parseInt(BP.substring(0,8),2);
                        buffer[1]          =          (byte)
Integer.parseInt(BP.substring(8,12)+"0000",2);
                    }
```

h

```
                    else {
                        buffer[1]          +=          (byte)
Integer.parseInt(BP.substring(0,4),2);
                        buffer[2]           =          (byte)
Integer.parseInt(BP.substring(4,12),2);
                        for(i=0;i<buffer.length;i++) {
                            outputFile.writeByte(buffer[i]);
                            buffer[i]=0;
                        }  }
                 isLeft = !isLeft;
if(dictSize < 4096) dictionary.put(P+C,dictSize++);
                    P=""+C;
           ...
          }

          catch(IOException ie) {
              BP = convertTo12Bit(dictionary.get(P));
              if(isLeft) {
                  buffer[0]              =              (byte)
Integer.parseInt(BP.substring(0,8),2);
                  buffer[1]              =              (byte)
Integer.parseInt(BP.substring(8,12)+"0000",2);
                  …
                  }
              }
              inputFile.close();
              outputFile.close();
          }
      }

      public static String convertTo12Bit(int i) {
        String to12Bit = Integer.toBinaryString(i);
        while  (to12Bit.length()  <  12)  to12Bit  =  "0"  +
to12Bit;
        return to12Bit;
    }

    //        ==========        Decompression        Functions
====================
public static void decompress() throws IOException {
        arrayOfChar = new String[4096];
        int i;

 for         (i=0;i<256;i++)         arrayOfChar[i]         =
Character.toString((char)i);

        // Read input file and output file

        RandomAccessFile        inputFile        =        new
RandomAccessFile(dfilename,"r");
        RandomAccessFile        outputFile        =        new
RandomAccessFile("output_".concat(filename),"rw");
        try {
            dbuffer[0] = inputFile.readByte();
                       ...
                    }
                    ddictSize++;

outputFile.writeBytes(arrayOfChar[dpreviousword]           +
arrayOfChar[dpreviousword].charAt(0));
                }
                /*
```

```
                 If word is present, we form a word with the
previousword and the first character of the
                 currentword and add it in a new entry
                */

                else {
                    if (ddictSize < 4096) {
                        arrayOfChar[ddictSize]               =
arrayOfChar[dpreviousword]                                   +
arrayOfChar[dcurrentword].charAt(0);
                    }
                    ddictSize++;

outputFile.writeBytes(arrayOfChar[dcurrentword]);
                }
                dpreviousword = dcurrentword;
            }
        }
        catch (EOFException e) {
            inputFile.close();
            outputFile.close();
        }
    }

    /*
    public static int getIntValue(byte b1, byte b2, boolean
disLeft) {
        String t1 = Integer.toBinaryString(b1);
        String t2 = Integer.toBinaryString(b2);

        …
        if    (disLeft)    return    Integer.parseInt(t1    +
t2.substring(0, 4), 2);
        else return Integer.parseInt(t1.substring(4, 8) + t2,
2);

    }
    //=========  read  compressed  lzw  file  for  Encryption
==========
    private static byte[] readContentIntoByteArray(File file)
   {
      FileInputStream fileInputStream = null;
      bFile = new byte[(int) file.length()];
      try
      {
          ...
      }
      catch (Exception e)
      {
         e.printStackTrace();
      }
      return bFile;
   }
}
```