

**National College of Ireland  
Project Submission Sheet**

**Student Name:** Rahul Sureshbabu Pillai

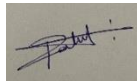
**Student ID:** X23103892

**Programme:** MSc Cybersecurity **Year:** 2024  
**Module:** Msc Research Project  
**Lecturer:** Mark Monagan  
**Submission Due Date:** 12/08/2024  
**Project Title:** Thesis Report  
**Word Count:** 7854

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the references section. Students are encouraged to use the Harvard Referencing Standard supplied by the Library. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action. Students may be required to undergo a viva (oral examination) if there is suspicion about the validity of their submitted work.

**Signature:**



**Date:** 12/08/2024

**PLEASE READ THE FOLLOWING INSTRUCTIONS:**

1. Please attach a completed copy of this sheet to each project (including multiple copies).
2. Projects should be submitted to your Programme Coordinator.
3. **You must ensure that you retain a HARD COPY of ALL projects**, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer. Please do not bind projects or place in covers unless specifically requested.
4. You must ensure that all projects are submitted to your Programme Coordinator on or before the required submission date. **Late submissions will incur penalties.**
5. All projects must be submitted and passed in order to successfully complete the year. **Any project/assignment not submitted will be marked as a fail.**

<b>Office Use Only</b>	
Signature:	
Date:	
Penalty Applied (if applicable):	

# AI Acknowledgement Supplement

[Msc Research Project]

Your Number	Name/StudentCourse	Date
Rahul Pillai	MSc Cybersecurity	12/08/2024

This section is a supplement to the main assignment, to be used if AI was used in any capacity in the creation of your assignment; if you have queries about how to do this, please contact your lecturer. For an example of how to fill these sections out, please click [here](#).

## AI Acknowledgment

This section acknowledges the AI tools that were utilized in the process of completing this assignment.

Tool Name	Brief Description	Link to tool
NAH		

## Description of AI Usage

This section provides a more detailed description of how the AI tools were used in the assignment. It includes information about the prompts given to the AI tool, the responses received, and how these responses were utilized or modified in the assignment. **One table should be used for each tool used.**

[Insert Tool Name]	
[N.A]	
[Insert Sample prompt]	[Insert Sample response]

## Evidence of AI Usage

This section includes evidence of significant prompts and responses used or generated through the AI tool. It should provide a clear understanding of the extent to which the AI tool was used in the assignment. Evidence may be attached via screenshots or text.

### Additional Evidence:

[N.A]

### Additional Evidence:

[N.A]

## Contents

Abstract .....	4
1 Introduction .....	4
1.1 Background .....	4
1.2 Problem Definition .....	6
2 Literature review .....	7
2.1 Data Encryption .....	7
2.2 Data Compression .....	8
2.3 Steganography .....	9
2.4 Data Storage System .....	10
2.5 Summary .....	10
3 Research Methodology .....	14
4 Design Specification .....	17
5 Implementation .....	18
6 Results And Evaluation .....	19
7 Conclusion and future enhancements .....	22
References .....	23

# **Abstract**

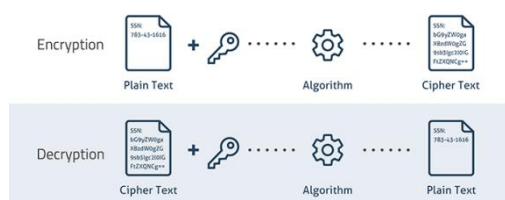
Data security becomes more and more important in the digital era. Every day, our personal, sensitive data is at risk of being stolen or changed during transmission or storage. In this study, these issues are addressed, and a four-layer security system is proposed to significantly increase data security. In other words, four algorithms and techniques are used to make the data as safe as possible: Elliptical Curve Cryptography (ECC), Huffman compression, steganography, and InterPlanetary File System(IPFS) storage. ECC is used for the encryption of the text; it provides one of the most secure and efficient types of data encryption. ECC makes the text coded, and, thus, very hard for unauthorized persons to read and understand. After the encryption of the text, Huffman compression is applied to reduce the size of the coded text and make it easier and faster to manage and send. To reduce the size of the data even more, steganography, another security technique, hidden the file inside an image. The resulting data is then stored using, to gain the most secure decentralized storage possible. These interrelated security techniques make the data almost impossible to be stolen or changed, and very well hidden that will be invisible and delicate to access for unauthorized users. ECC is also compared with RSA, another widely used encryption type, in the study. The results show that ECC is more secure and faster than RSA. Thus, ECC is the best option to be used in the proposed system. Overall, this study makes a significant contribution to both science and practice because it offers a clear solution to the highly important issue of data security.

## **1 Introduction**

### **1.1 Background**

The era of digital technologies has changed everything, starting from the way we live and connect to each other to the ways we work and do business. It has provided us with an outstanding approach to communication, sources of any information, and types of technological progress. However, with all these advantages, there is a serious question our data is not always safe. Everything we do from sharing personal information with each other to run business is based on the idea of data. It is the central aspect of keeping everything in order and to make the operational process smooth. However, this is the fact that also makes us be extremely concerned about its safety: when we transfer data or data is stored, we need to ensure it remains uncorrupted and unreadable to any unauthorized party(Malik & Abbas, 2023). The following paper is devoted to explaining several important methods helping to protect our data from theft and changes. Several essential methods can be employed to protect data from theft or tampering. The first one on the list is encryption that presupposes changing the way the data look and, therefore, making it inappropriate for use. Compression stands for data reduction; in this case, the information is delivered in a more general format. The next approach to discuss is steganography, a kind of secret storage. Finally, there is a range of secure storage systems like TrueCrypt that often applies a combination of the approaches mentioned above.

Encryption is a base point conversion algorithm to convert plain-text data into ciphertext (Figure(1)). This renders the data useless to anyone who does not possess that key. This guarantees that the data is unusable to interceptors unless they have programming knowledge and the key. Encryption is in widespread use to protect the classified information, which includes financial transactions and corporate data as well that has improved significantly over decade (Omotunde & Ahmed,2023).



**Figure (1): Encryption and decryption process (Lithmee, 2018)**

Compression is a mathematical function that attempts to reduce data file sizes by cutting off those parts of the image there which do not contain any useful information. It will not only save storage space, but also affect the efficiency of transmission. This is done using compression algorithms by, again, encoding data more efficiently with less amount of bits in a computer. Compression, however, reduces the size of files NOT security by design It can indirectly contribute to the security of data transmission speed and height (Fitriya et al. 2017). 2017).

The practice of concealing data into other seemingly non-secret data – such as images, audio files, or text messages – other, is called Steganography. Steganography differs from encryption because, whereas encryption changes the data in question to make it secure, steganography changes the data in order to hide the existence of the data in the first place. By putting messages into what seems to be relatively benign cover data, steganography ensures that knowledge of the existence of a hidden message is kept secret. In this respect, as Srinivasan et al. note, steganography adds another layer of security as well, as it makes it more difficult for others to conclude that there could be hidden data(Srinivasan et al. 2024).

Systems related to secure storage include a broad range of techniques meant to ensure a safe environment for sensitive data storage. They include encryption and signatures and ensure all-around confidentiality, integrity, and availability of data through a multitude of similar means. With only a little overlap, all these approaches have the same goal of preventing unauthorized data access, information leakage, and loss of placed data. More specifically, the secure storage approaches were meant to ensure safe data storage at rest, that is to say – to protect the data stored by the data owner on their databases, file servers, or cloud storage solutions. That is how confidential and reliable information at storage is assured, preventing unauthorized and malicious use of the data and avoiding possible information loss and data breaches(Sunday & Olufunmini, 2023).

## 1.2 Problem Definition

Encryption is a considerable part of data security as it works as the basis for protecting information. However, many existing encryption methods have vulnerabilities that decrease their capacity of safeguarding data. These include the use of key lengths that are too short and subject the data to brute force attacks, issues related to the selected encryption algorithms and problems that occur due to key generation, key management, or implementation (Ahmed & Naeem, 2022). In order to provide the required level of security information, several security layers should be used (Jan et al., 2021). In their absence, the vulnerability of the data to tampering and breach would increase. Previous research highlights that, accordingly, to additional measures to make the attackers' work more complicated have been developed. An approach to maximize data security that presumes the use of four protection layers, one of which is a data compression method. The remaining layers are also being suggested in this paper: the first of them is encryption with ECC that is used to encode the information; the second one is the concealment of the encrypted information in the image with the help of steganography. The third measure that is taken to ensure the safety of the data sent with the help of InterPlanetary File System (IPFS) is a compression approach which, as mentioned previously, is Huffman coding. The fact that the selected system combines four relatively well-known techniques does not make it less unique and integrated. The system unites all four components in one integrated system and supports security at every stage, from the encryption of the text with ECC to the arranging of the data inside the image together with the other information.

The study has the following research question:

How effective is the combination of ECC encryption, Huffman compression, LSB steganography, and IPFS in enhancing data security and privacy within a desktop application?

The aim of the study is to enhance data security and privacy by concealing data through a combination of ECC cryptography, compression, steganography techniques, and IPFS.

The objectives of the study are:

1. Text encryption using the ECC algorithm to securely transform plain text into encrypted form.
2. Implement Huffman compression to effectively reduce the size of encrypted text.
3. Apply LSB steganography to embed compressed ciphertext into an image.
4. Store the stego-image in IPFS to ensure decentralized and secure storage of the concealed data.
5. Integrate all encryption, compression, steganography, and IPFS storage methods into a user-friendly desktop application.

The novelty of the system introduced here is that it is made by using advanced ECC cryptography that strongly encrypts text and while it is efficient Huffman compression that reduces data size and thus guarantees data integrity. The study's contribution is that it uses the techniques of F5 algorithm, which is based on the LBS, and while ECC is used as a complex cryptanalysis mechanism. Data that has been processed by the system and stored using IPFS is guaranteed to be secure because it is decentralized. The desktop application is user-friendly and easy to use which ensures enhanced data security and privacy.

Section 1 of the report briefly introduces the project and define the concepts of encryption, compression, steganography, data security storage, and the system that the paper will develop. Section 2 reviews existing work done in the fields in consideration here. Section 3 describes how the study is done, presenting the objectives, and questions that the study is based on. Section 4 details presents the specifications of the different resources that the present study is based on. Section 5 describes how the system was finally implemented. Section 6 discusses the results of the study and how they were evaluated. Section 7 offers conclusions and suggests ways to improve the system in the future.

## **2 Literature review**

This section reviews previous studies and their challenges. It explores current techniques for encryption, steganography, compression, and IPFS for data protection. The main aim of this research is to enhance data security by combining all these methods for stronger protection and quicker processing.

### **2.1 Data Encryption**

The strengths and weaknesses of RSA were examined, and novel solutions to overcome its weaknesses were proposed in the study by (Nisha and Farik, 2017). The study employs the RSA public key cryptography algorithm to secure communication over networks, focusing on dual keys for encryption and decryption. It reviews the algorithm's usage, strengths, and weaknesses, proposing solutions to overcome identified drawbacks. The study further examines RSA's current applications, such as in Pretty Good Privacy and cloud services like Google's G Suite, which utilize RSA for key transportation and authentication. Results indicate RSA's effectiveness is due to the computational difficulty of factoring large integers into primes, with a 768-bit RSA modulus containing a 232 decimal digit number being factored as a significant achievement. The study has a limitations, particularly the vulnerability of RSA to attacks if weak key generation occurs, emphasizing the necessity of using large random prime numbers to calculate the modulus for the public and private keys.

A systematic and critical review of RSA-based public key cryptographic schemes, focusing on various domains such as Hybrid, Parallel, Cloud, Wireless, and Core-Modifications, was conducted in the study by (Imam et al., 2021). The study employs

a two-stage search strategy to ensure a wide coverage of RSA-related research, starting from the year 1978 up to July 2021. Other methods used in the study include a quality assessment checklist to evaluate the selected papers and data extraction to analyze the primary research subject, objectives, domain categories, limitations, and key conclusions. The results of the study indicate that the most recent applications of standard RSA incorporate key exchanges, digital signatures and other types of communication where data transmission between two parties is required. The study admits that even though they reviewed many papers, there might be some limitations. This is because they left out certain papers based on set rules, which means they might have missed some important improvements to the RSA algorithm.

A review of cryptographic techniques, focusing on RSA's role in ensuring secure communications across various digital platforms, was discussed in the study by (Upadhyay and Gupta, 2022). The study compared symmetric and asymmetric cryptography, they explained what makes RSA special. The study emphasized that RSA is strong, especially in preventing brute force attacks due to its use of logarithmic functions. However, it also pointed out some drawbacks, such as RSA's slow key generation speed, which can be a problem in situations where quick key generation is needed.

A comparative analysis of two cryptographic algorithms, RSA and Elliptic Curve Cryptography, focusing on key generation, encryption, and decryption times, was examined by (Mohammad Rafeek Khan et al., 2023). The study recorded key generation, encryption, and decryption times for both algorithms across different security bit levels. Results unveiled a notable disparity in key generation times between ECC and RSA, with ECC exhibiting significantly faster generation times across various security bit levels. Notably, ECC demonstrated a linear relationship between time and key size, contrasting with RSA's exponential increase. ECC outperformed RSA in decryption, particularly at higher security levels. Further highlighting the advantages of ECC, the study used its efficiency and speed compared to RSA. ECC's inherent characteristics, including its reliance on elliptic curves, allow for smaller key sizes without compromising security. This makes ECC especially good for environments with limited resources, like IoT devices, where it's important to have efficient computing and use as little memory as possible.

## **2.2 Data Compression**

A thorough overview of data compression techniques, particularly focusing on methods that preserve data integrity such as Huffman coding, Shannon Fano coding, Tunstall coding, Lempel Ziv Welch algorithm, and Run-Length Encoding was discussed in the study by (Fitriya, Purboyo and Prasasti, 2017). These methods are discussed in their applications to compress various data types like text, images, and video. Audio compression is explored in detail, covering formats like Vorbis, MP3, and lossless FLAC, highlighting considerations for audio professionals. Text compression's importance is emphasized through its impact on compression ratios, which measure how effectively data is reduced in size. Video compression is also examined, stressing the role of codecs in efficient data transmission. Image compression methods such as



GIF, PNG, and JPEG, particularly focusing on the LZW algorithm in GIFs, are reviewed. The study demonstrates how these techniques effectively reduce file sizes, optimizing storage use and speeding up data transfer. However, it also recognizes challenges in compression, including potential data loss in lossy compression and the difficulty in achieving optimal compression ratios while preserving data integrity.

Comparing several methods for compressing data without losing information was discussed in the study by (Anup, Ashok and Raundale, 2019). The study focused on Run-Length Encoding, Shannon-Fano Algorithm, Huffman Coding, and Lempel-Ziv-Welch algorithm. Each method was examined for how it reduces data size: RLE shortens patterns to smaller strings, Shannon-Fano uses a tree based on frequency, Huffman Coding assigns shorter codes to common data, and LZW groups symbols based on a dictionary. Results showed these methods effectively shrink data, measured by metrics like compression ratio, compression factor, and savings percentage. For example, RLE compressed data to 2.7 times its original size, Shannon-Fano saved 77.6% of data, and Huffman Coding balanced ratio and savings. LZW was noted for its use in GIFs and UNIX file compression, showing practical applications. However, the study noted that current compression techniques have limits. It suggested using advanced methods like pattern deduction or reducing redundancy, possibly with machine learning, to improve compression ratios. Ultimately, the study concluded that the best compression method depends on the specific needs of the data being compressed.

## **2.3 Steganography**

The study presents a comprehensive survey of image steganography techniques, focusing on methods such as spatial domain methods, which include least significant bit techniques, pixel value differencing, and transform domain techniques like discrete cosine transformation, as discussed in the study by (Hussain, 2013). It utilizes various digital mediums like images, audio, and video as data carriers, employing algorithms that modify pixel values or transform coefficients to conceal information. Results indicate that spatial domain methods, particularly LSB techniques, offer high capacity and less image degradation, while transform domain techniques provide stronger resistance to image manipulations. The study does not specify numerical values for results, as it is a survey rather than an experimental research. However, it highlights the trade-offs between capacity and perceptual transparency, noting that higher data embedding can lead to more noticeable image alterations. The study's limitation lies in its focus on the theoretical aspects of steganography without providing empirical data or performance metrics for the surveyed techniques, which could have offered a more practical evaluation of their effectiveness in real-world applications.

Using AES and Blowfish encryption to protect cover images and hide encryption keys within stego images, which enhances data security was discussed in the study by (Alanzy et al., 2023). The algorithm employs pixel randomization and hybrid encryption to make the encryption process more complex, aiming to preserve stego image quality while ensuring encryption and decryption of confidential messages. The research also explores LSB steganography, where the least significant bit of the cover

image is replaced with bits from a secret message. Performance is evaluated using metrics like Peak Signal-to-Noise Ratio (PSNR) and Mean Square Error (MSE). Results show that the proposed algorithm effectively secures data with high PSNR and low MSE values, indicating good image quality and strong encryption. The study notes the need for further enhancements in encryption and compression techniques, as well as the potential use of machine learning and AI to improve resistance against attacks on cryptography and steganography, acknowledging current limitations in the method's complexity during encoding. A limitation of the study concerning AES or Blowfish encryption is their requirement for substantial processing power and memory resources. This demand can pose challenges for resource-constrained devices or applications that need quick encryption and decryption cycles.

## 2.4 Data Storage System

Exploring IPFS and its decentralized storage architecture, which relies on peer-to-peer networking and content addressing principles, was discussed in the study by (Trinh Viet Doan et al., 2022). By combining concepts from P2P networking, Linked Data, and other domains, IPFS enables file exchanges reminds of Bittorrent, employing a unique naming and addressing scheme using multihashes. In terms of deployment figures, the study cites data from 2022, revealing that the IPFS network boasted 3.7 million unique users and facilitated the transfer of over 125TBs of data through more than 805 million requests per week. The periodic measurements indicate a substantial growth to approximately 17k reachable nodes per crawl as of 2022. The results of the study underscore IPFS's significant traction among web-related projects seeking to future-proof their products. Notably, IPFS has gained adoption by major players such as Cloudflare and has been seamlessly integrated into browsers like Mozilla Firefox, Brave, and Opera. One of the key advantages of IPFS over traditional databases lies in its decentralized nature, which eliminates single points of failure and enhances resilience against censorship and data loss. IPFS employs cryptographic hashing to ensure data integrity and authenticity. Each file is assigned a unique hash based on its content, making it virtually impossible for malicious actors to tamper with or forge data without detection.

## 2.5 Summary

The summary of all the studies reviewed in this research is provided below:

Study	Methods	Results	Limitations
Nisha and Farik, (2017)	RSA	The research highlighted RSA's effectiveness in resisting attacks due to the computational complexity involved in factoring large integers into primes. It provided an example of a 768-bit RSA modulus and underscored the importance	The study shows potential vulnerabilities in RSA if weak key generation practices are employed. It emphasized the need for robust key generation protocols to

		of using large, random prime numbers to enhance security.	mitigate security risks effectively.
Imam et al., (2021)	RSA	The research categorized modifications and applications of RSA into 11 distinct domains, showcasing its broad applicability and evolution. It highlighted recent advancements and practical implementations of RSA in domains such as key exchanges, digital signatures, and secure communications like web browsers and chat applications.	The study recognized limitations related to the exclusion of certain research papers, which may have omitted significant advancements in RSA algorithms and applications. It suggested that future studies should ensure a comprehensive review to capture all relevant developments in RSA-based cryptography.
Upadhyay and Gupta, (2022)	RSA	The study shows RSA's robustness in cryptographic security, particularly in preventing brute force attacks. It highlighted RSA's computational complexity in factoring large integers and its applications in secure communication systems.	A limitation identified was RSA's slower key generation speed, which could pose challenges in scenarios requiring rapid key generation. The study suggested that future research could explore improvements in key generation processes to enhance RSA's practical applications in real-time secure communications.
Mohammad Rafeek Khan et al., (2023)	RSA and Elliptic Curve Cryptography (ECC).	The research revealed significant disparities in key generation times between ECC and RSA, with ECC showing faster generation times across various security bit levels. ECC demonstrated a linear relationship between time and key size, whereas RSA exhibited exponential increases. ECC also	A potential limitation could include the scope of the analysis, such as specific scenarios or implementations not covered in detail. Additionally, variations in hardware or software configurations could affect performance

		outperformed RSA in decryption speed, especially at higher security levels. The study highlighted ECC's efficiency and speed advantages over RSA, particularly in environments with limited resources like IoT devices.	comparisons.
Fitriya, Purboyo, and Prasasti, (2017)	Huffman coding, Shannon Fano coding, Tunstall coding, Lempel Ziv Welch algorithm, and Run-Length Encoding.	The research demonstrated the effectiveness of these compression techniques in reducing file sizes while preserving data integrity. Compression ratios were highlighted as a measure of effectiveness, with techniques like Huffman coding and LZW algorithm showing significant reductions in data size. The study also underscored the role of codecs in efficient video compression and the impact of compression methods on data transfer speeds and storage optimization.	One limitation of the study may be the complexity involved in comparing various compression techniques comprehensively across all data types and applications. Additionally, while the study acknowledged challenges such as data loss in lossy compression and difficulties in achieving optimal compression ratios, it did not delve deeply into specific algorithms' limitations in real-world scenarios.
Anup, Ashok, and Raundale, (2019)	Run-Length Encoding (RLE), Shannon-Fano Algorithm, Huffman Coding, and Lempel-Ziv-Welch algorithm.	The study found that these compression methods effectively reduced data size, as measured by metrics such as compression ratio, compression factor, and savings percentage. For instance, RLE achieved a compression ratio of 2.7, Shannon-Fano saved 77.6% of data, and Huffman Coding balanced compression ratio with	A limitation identified in the study was the inherent complexity in achieving optimal compression ratios while preserving data integrity across different data types and applications. The study suggested future research directions could explore advanced techniques like pattern deduction

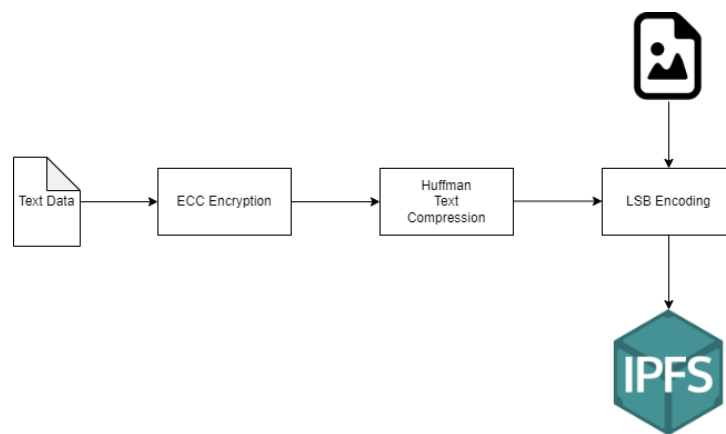
		savings. LZW's practical applications in GIFs and UNIX file compression were highlighted, demonstrating its versatility and effectiveness in real-world scenarios.	and redundancy reduction, potentially leveraging machine learning to enhance compression effectiveness.
Hussain, (2013)	LSB steganography	Spatial domain methods, especially LSB techniques, were found effective for high capacity with minimal image degradation. Transform domain techniques showed resilience against image manipulations. However, specific numerical results or performance metrics were not provided, focusing more on theoretical evaluations.	The study mentions a limitation that is when the image space is small or low-resolution, embedding a lot of text can reduce the hiding efficiency.
Alanzy et al., (2023)	AES, Blowfish and LSB steganography	The algorithm achieved strong data security with high PSNR and low MSE values, indicating effective encryption and good image quality.	A challenge noted was the significant processing power and memory resources required for AES and Blowfish encryption, potentially limiting use in resource-constrained devices.
Trinh Viet Doan et al., (2022)	IPFS	As of 2022, IPFS supported 3.7 million users and facilitated over 125TB of data transfer weekly with around 17,000 reachable nodes per crawl. It gained traction in web projects and browser integrations due to its decentralized nature, resilience against censorship, and data loss prevention.	Ensuring consistent performance and reliability across a decentralized network architecture remains a challenge, especially under heavy network traffic and increased user demands.

**Table (1):Summary table of reviewed literature studies**

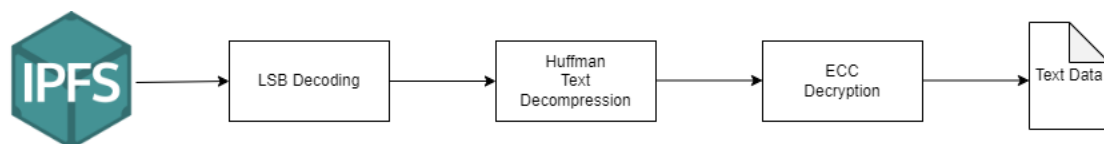
From Table (1), it's clear that previous studies didn't integrate encryption, compression, steganography, and secure storage to enhance data security. Instead, these studies often used these methods individually or in pairs, relying on established algorithms such as RSA. In contrast, this study innovatively combines ECC encryption, Huffman compression, LSB steganography, and IPFS to significantly enhance data security and privacy.

### 3 Research Methodology

This research project introduces a complete approach to enhance data protection. It starts by encrypting data using ECC encryption. Next, it uses Huffman compression to make files smaller without losing any data. Then, it hides this compressed data inside images using steganography. Finally, these "hidden" images are stored securely in IPFS to prevent unauthorized access. Figure (2) below shows the steps of the encryption process. It shows how data is encrypted, compressed to save space, hidden inside images using steganography, and finally stored securely in IPFS. Figure (3) below, on the other hand, explains the decryption process. It demonstrates how encrypted data is retrieved from IPFS, and then the hidden compressed data is extracted using reverse steganography. After extraction, the data is decompressed using Huffman coding and decrypted using ECC encryption.



**Figure (2):Data hiding**



**Figure (3):Data retrieving**

### **3.1 Data Encryption**

Data encryption is a process that converts plain text into a coded form to prevent unauthorized access. This ensures that only authorized individuals can access and read the original information. The main goal of data encryption is to protect sensitive information from being intercepted or accessed by unauthorized parties, thus maintaining the confidentiality and integrity of the data. In this study, Elliptic Curve Cryptography (ECC) is utilized for encrypting text. ECC is a form of public key cryptography based on the algebraic structure of elliptic curves over finite fields. This encryption method generates a pair of keys: a public key for encryption and a private key for decryption. ECC encryption is preferred because it provides strong security with smaller key sizes compared to other common cryptographic methods.(Shantha, Renita and Edna, 2019). This results in faster computations and reduced storage requirements, which are significant advantages, especially in environments with limited resources. The ECC encryption process begins with the generation of an elliptic curve and a base point. These elements are used to create a public-private key pair. The public key is then used to transform the plain text into an encrypted format, making it unreadable to anyone who does not possess the corresponding private key. One of the challenges encountered during this process is ensuring the robustness of the key generation step, as weak or improperly generated keys can compromise the security of the entire system. Additionally, there is limited library support for ECC, which means extensive research is required to find suitable implementations. This lack of readily available tools can slow down development and complicate the process. Although these difficulties, ECC provides several advantages. Its smaller key sizes mean less computational overhead, which enhances performance without sacrificing security. This efficiency makes ECC suitable for a wide range of applications, from securing communications in mobile devices to protecting sensitive data in large-scale enterprise systems.

### **3.2 Data Compression**

Data compression is a technique used to reduce the size of data, making it easier and faster to store and transmit (man and Utama Siahaan, 2016). It works by eliminating redundancy and encoding information more efficiently. This process is essential for improving storage utilization and reducing bandwidth consumption, which is particularly important in environments with limited resources or where large volumes of data need to be transferred quickly. In this study, Huffman Coding Compression is employed to compress the encrypted text. Huffman coding is a lossless data compression algorithm that assigns variable-length codes to input characters, with shorter codes assigned to more frequent characters. This method ensures that no data is lost during compression, which is crucial for maintaining the integrity of the encrypted information. The Huffman coding process begins by creating a frequency table of all characters in the input data. Characters that appear more frequently are given shorter codes, while less frequent characters are assigned longer codes. This is done by building a binary tree where each leaf node represents a character and its frequency. Characters with lower frequencies are placed deeper in the tree, resulting in longer codes, while higher frequency characters are placed closer to the root, yielding shorter codes. The tree is traversed to generate the codes for each character, and the original data is then

encoded using these variable-length codes. One of the difficulties faced during this process is ensuring the accuracy of the frequency table. Any errors in calculating character frequencies can lead to incorrect code assignment, which would affect the compression efficiency and possibly compromise data integrity. Huffman coding offers significant advantages. It effectively reduces the size of the data without losing any information, making it an ideal choice for compressing encrypted text. This efficiency in reducing data size helps in faster data transmission and saves storage space, which is particularly beneficial in data-sensitive applications.

### **3.3 Image Steganography**

The Steganography word originates from Greek roots "steganos" and "graphein," meaning "covered" or "concealed." Its fundamental goal is to hide data within a cover material in a way that remains imperceptible to others (Sabah et al., 2023). Steganography is often used for secure communication, protecting intellectual property, and maintaining privacy, as it allows information to be transmitted or stored without drawing attention. In this study, the Least Significant Bit embedding method is used for image steganography. LSB embedding is a straightforward approach where data bits are inserted into the least significant bit of image pixels, altering their values slightly. This method is advantageous because it minimally affects the visual quality of the image, making it difficult for unauthorized users to detect the hidden data without specific tools. The LSB embedding process begins by converting the encrypted and compressed data into a binary format. Each bit of this binary data is then sequentially embedded into the least significant bit of selected pixels in the cover image. One significant challenge in LSB embedding for image steganography is the constraint imposed by image size versus the length of the hidden text. When the available space within the image is limited, such as with small or low-resolution images, embedding a larger amount of text can lead to inadequate hiding efficiency.

### **3.4 Data Storing with IPFS**

Data storing is a fundamental aspect of digital information management, involving the saving and maintenance of data in a secure and accessible manner. Traditional centralized storage systems, like those used in cloud services, store data on servers managed by a single entity. While effective, these systems can suffer from issues such as single points of failure, high costs, and limited control over data distribution and access. To address these challenges, decentralized storage solutions like the InterPlanetary File System have emerged, offering a more robust and efficient approach to data storage and retrieval. In this study, IPFS is used to store and manage the stego-image data. IPFS is a peer-to-peer distributed file system that enables the storage and sharing of hypermedia in a decentralized manner. Unlike traditional systems that rely on a central server, IPFS uses a distributed network of nodes, each storing pieces of the overall data. This method improves redundancy, reduces the risk of data loss, and enhances accessibility (Trautwein et al., 2022). The process of storing data on IPFS begins with breaking down the data into smaller chunks. Each chunk is cryptographically hashed, resulting in a unique identifier known as a content identifier. These CIDs are used to retrieve the data chunks from the IPFS network. When data is added to IPFS, it is shared across many nodes, so the data stays available even if some



nodes go offline. This decentralized approach significantly enhances the robustness and resilience of the data storage system. A difficulty faced was that sometimes the IPFS connection would be lost. Many users have reported this issue in the official repository, making it time-consuming to establish connections between nodes and ensure proper functionality, sometimes necessitating a reinstallation of the system. One of the main advantages of using IPFS is its ability to ensure data integrity and authenticity. Since each chunk of data is hashed, any alteration to the data would change its CID, making tampering easily detectable. Also, IPFS reduces bandwidth usage and improves load times by enabling content to be retrieved from the nearest node, thus optimizing the data access process.

### **3.5 Data Decryption**

The data decryption process begins by retrieving the stego-image from the InterPlanetary File System . Once the stego-image is successfully accessed, the next step is to perform LSB decoding. This method involves examining the least significant bits of the image pixels to extract the hidden compressed cipher text. Since these bits were modified during the embedding process, they now contain the encrypted and compressed data. After extracting the compressed cipher text, Huffman decoding is applied. This technique reverses the Huffman coding used during compression, transforming the compressed data back into its original encrypted form. Huffman decoding works by using a tree structure to decode the variable-length codes into the corresponding characters or data units, effectively decompressing the cipher text. Finally, the decompressed cipher text undergoes decryption using Elliptic Curve Cryptography (ECC). ECC decryption uses a private key to transform the encrypted data back into its original, readable form. This step is crucial for retrieving the original plain text, ensuring that the data is accurately and securely restored to its initial state before encryption and compression.

## **4 Design Specification**

### **4.1 ECC Encryption**

Elliptic Curve Cryptography is known for its security and efficiency due to its use of smaller keys compared to traditional cryptographic methods. This lightweight algorithm requires less computational resources, making it ideal for ensuring data security without compromising performance (Shantha, Renita and Edna, 2019).

### **4.2 Huffman Text Compression**

Huffman compression is a well-known method for reducing the size of text. It's widely recognized for its efficiency, akin to how Morse code works. Each letter gets a short code if it's common, and a longer one if it's rare, making it effective for compressing data efficiently (Joshi et al., 2020).

### 4.3 LSB Embedding

The least significant bit technique is commonly used in image steganography to embed information subtly. It takes advantage of the least noticeable parts of digital images to hide data securely and is favored for its simplicity and reliability in concealing information (Aslam et al., 2022).

### 4.4 IPFS

IPFS, short for Inter-Planetary File System, is a decentralized file storage and sharing system. It operates peer-to-peer, meaning files are stored and accessed directly between users without relying on a central server. IPFS is closely related to blockchain technology. While IPFS manages the storage and distribution of files, blockchain provides a secure and transparent way to record transactions and manage access rights (Nishara Nizamuddin, Hasan and Salah, 2018).

## 5 Implementation

The integration of encryption, compression, steganography, and IPFS methods into a desktop application was realized using Python's Tkinter library, which offers a straightforward approach to building graphical user interfaces (GUIs). The GUI application comprises several pages, with the encryption and decryption pages serving as its core components. The encryption page is designed to facilitate the secure transformation of plaintext data into encrypted and steganographically embedded form. Users begin by entering their text data into a designated text area. They are then presented with two essential buttons: one for uploading an image and another to initiate the encryption process. Upon clicking the encryption button, the user's input undergoes a series of steps. First, the data is encrypted using Elliptic Curve Cryptography (ECC), ensuring robust security with minimal computational overhead. Following encryption, the data is compressed using Huffman coding, effectively reducing its size without compromising integrity. The compressed data is then embedded into the selected image using the Least Significant Bit (LSB) steganography technique. Once the data has been encrypted, compressed, and embedded, the final step involves saving the stego-image to the InterPlanetary File System. IPFS provides a decentralized and resilient platform for storing data, ensuring that the encrypted information remains accessible and tamper-proof. In contrast, the decryption page focuses on retrieving and processing encrypted data stored within a stego-image retrieved from IPFS. Users begin by selecting the stego-image containing the encrypted data through a file selection button. Upon selection, the GUI displays the image and provides a text area where the decrypted output will be shown. To initiate the decryption process, users click the decryption button. The application performs the following operations sequentially: reverse steganography, decompression using Huffman coding, and decryption using the ECC algorithm. Reverse steganography extracts the hidden compressed data from the stego-image without altering the image's visible content. Huffman decompression then restores the original size and format of the encrypted data, preparing it for decryption.

Finally, the ECC decryption algorithm decrypts the data, revealing the plaintext output in the designated text area. Throughout both encryption and decryption processes, the GUI's design emphasizes usability and clarity. Users are guided through each step with intuitive buttons and clear instructions, minimizing the complexity typically associated with cryptographic operations. The application's integration of Tkinter ensures a responsive and user-friendly experience, allowing even non-technical users to securely encrypt and decrypt their data with confidence. The GUI's usability extends beyond its core functionality. Additional features include error handling mechanisms to alert users of input mistakes or unexpected file formats, ensuring a smooth and reliable operation. Furthermore, the application incorporates informative prompts and tooltips to guide users through the encryption and decryption procedures, promoting understanding and confidence in handling sensitive data securely. By using Python's Tkinter library, the development team achieved a balance between functionality and accessibility in the desktop application. The choice of Tkinter facilitated rapid prototyping and iterative development, enabling quick adjustments based on user feedback and evolving security requirements. This iterative approach ensured that the final application not only met but exceeded user expectations for usability, security, and performance.

## 6 Results And Evaluation

### 6.1 Results

```
#####
#####
#####
#####
#####
Text encrypted successfully.
Ciphertext compressed and saved successfully.
Steganography process completed
Retrieved compressed cipher text successfully.
Decompressed Cipher Text: bytearray(b'*\x13~\x08 \x98k2A\xee\x08\xfd\xd6s\x90\x8b\x86u0\xe2z5xX\xc9')
decrypted msg: b'there is a mole in the wall'
```

**Figure (4): ECC decryption**

In Figure (4), the results demonstrate successful decryption of the ciphertext using the implemented cryptographic techniques. The ECC decryption process effectively reconstructed the original plaintext message from the encrypted ciphertext, showcasing the efficacy of Elliptic Curve Cryptography in secure data decryption. This successful decryption reaffirms the reliability and functionality of ECC in cryptographic applications, particularly in scenarios where efficient and secure data transmission is crucial.

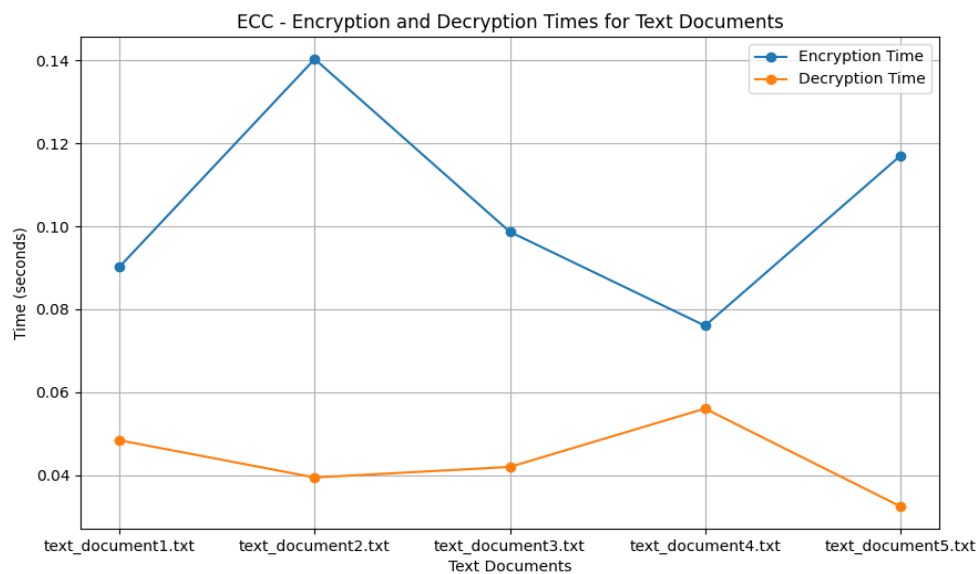
```
ECC encryption_times: [0.08196043968200684, 0.09287214279174805, 0.07688188552856445, 0.12340879440307617, 0.06815767288
208008]
ECC decryption_times: [0.031240463256835938, 0.032759904861450195, 0.04748225212097168, 0.03892946243286133, 0.046216726
303100586]
```

**Figure (5):ECC encryption and decryption time**

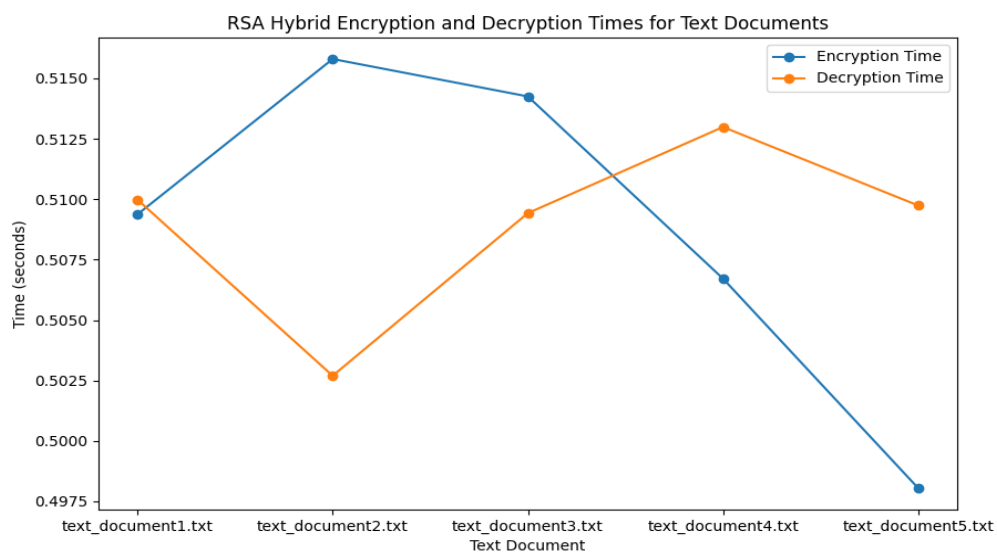
```
RSA encryption_times: [0.5093751, 0.515803, 0.5142548000000002, 0.5067146999999999, 0.4980451000000006]
RSA decryption_times: [0.5099797000000001, 0.5026950000000001, 0.5094243999999999, 0.5129896, 0.5097414999999996]
```

**Figure (6): RSA encryption and decryption time**

Figure (5) and (6) display the encryption and decryption times for ECC and RSA, respectively. The results clearly indicate that ECC performs faster than RSA. ECC encrypts and decrypts data more swiftly than RSA, showcasing its efficiency in cryptographic tasks. This comparison emphasizes ECC's superiority in scenarios where speed and performance are crucial, such as in secure communications and data transactions. Implementing ECC can significantly enhance the responsiveness and efficiency of cryptographic operations, ensuring quicker data protection and retrieval in various applications.

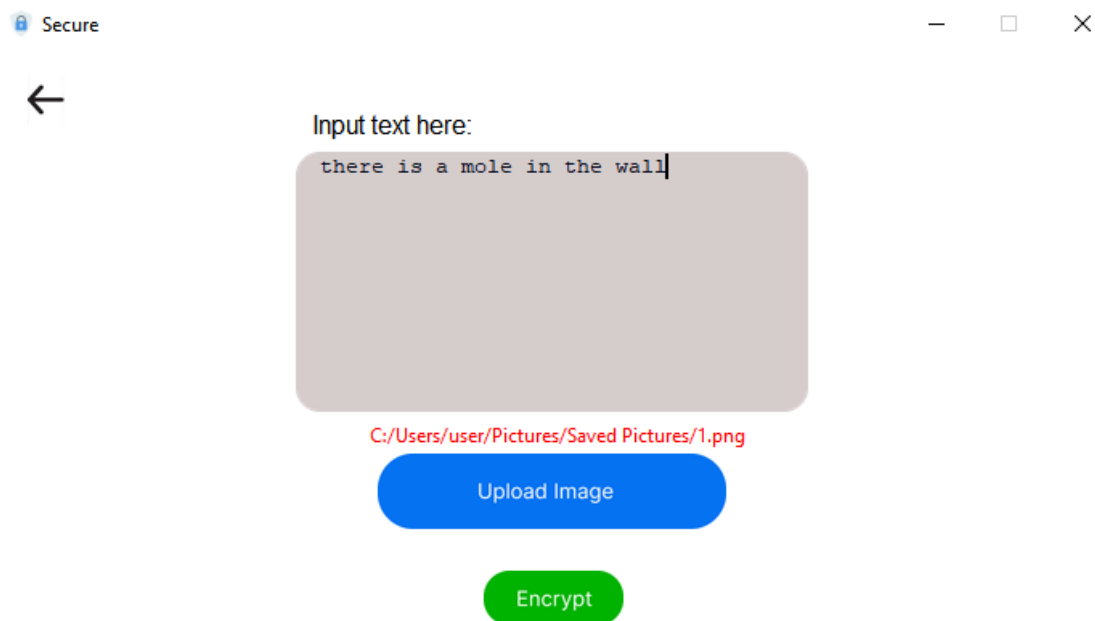


**Figure (7):ECC Encryption and Decryption Time Plot for Text Documents**

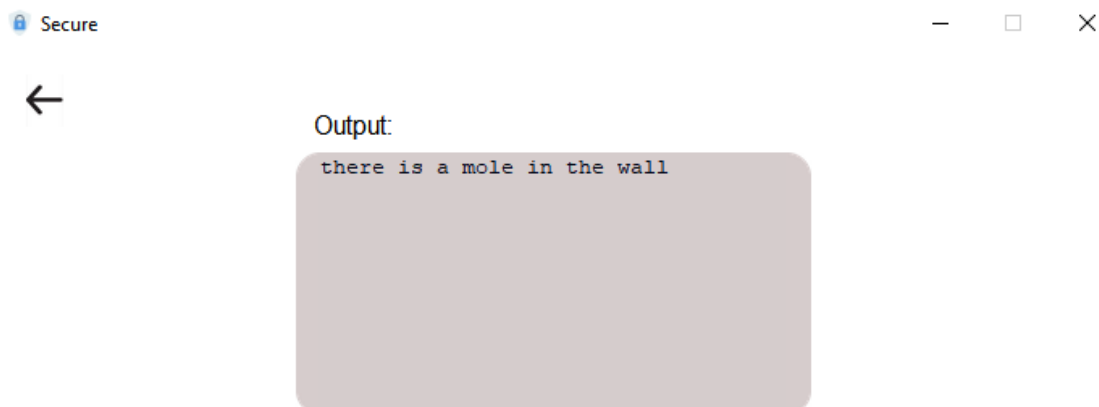


**Figure (8):RSA Encryption and Decryption Time Plot for Text Documents**

Figure (7) shows the encryption and decryption times for ECC across five text documents of increasing size, with the x-axis representing each file and the y-axis indicating the time taken. As file sizes increase incrementally, ECC consistently demonstrates shorter encryption and decryption times compared to RSA, as depicted by the downward trend on the plot. Similarly, Figure (8) shows the corresponding times for RSA encryption and decryption across the same set of text documents. These figures provide a clear visual comparison showing ECC's efficiency in handling cryptographic tasks swiftly, making it a favorable choice for applications where fast processing of secure data is essential.



**Figure (9):Encryption page in the GUI**



**Figure (10):Decryption page in the GUI**

Figure (9) illustrates the encryption page within the graphical user interface (GUI), providing a simple platform for users to secure their data. In this interface, users can input their plain text and select an image through a straightforward process. Figure (10) complements this by showing the decryption page in the same user-friendly GUI environment. Here, users can decrypt encrypted data by uploading the stego-image. The GUI makes the encryption and decryption processes intuitive and accessible. It streamlines the steps involved, offering a friendly interface that guides users through each operation effectively. This approach not only enhances usability but also ensures that users can perform cryptographic tasks with ease and confidence.

## 6.2 Evaluation

The evaluation of the proposed system shows that it effectively combines encryption, compression, steganography, and IPFS to enhance data security and privacy. The results confirm that all the project's objectives were met successfully. Objective 1, the system achieved text encryption using the ECC algorithm, ensuring that plain text was securely encrypted. Objective 2, it effectively compressed the encrypted text using Huffman compression, reducing the size without losing any data. Objective 3, it successfully applied LSB steganography to embed the compressed ciphertext into an image, creating a stego-image. Objective 4, the system demonstrated the ability to store the stego-image in IPFS, ensuring decentralized and secure storage. Final objective was integrate all these methods into a user-friendly desktop application built with Python's Tkinter module. This approach not only improves the security of the data but also makes the process accessible and efficient for users, proving the system's overall effectiveness in protecting sensitive information.

The research question, "How effective is the combination of ECC encryption, Huffman compression, LSB steganography, and IPFS in enhancing data security and privacy within a desktop application?" has been answered. As shown in Figures (9) and (10), the desktop application successfully encrypts, compresses, and embeds data into an image using these techniques. The system also retrieves the stego-image from IPFS, performs LSB decoding, decompresses the data, and decrypts it. One area for improvement in this study is its focus only on image-based steganography, without extending to audio or video formats. Also, handling large data sizes could be challenging due to the requirement for higher-resolution images.

## 7 Conclusion and future enhancements

The study developed an effective system that combines ECC encryption, Huffman compression, LSB steganography, and IPFS to enhance data security and privacy. Through the desktop application created using Python's Tkinter module, the system demonstrated its ability to securely encrypt, compress, and hide data within images, then retrieve and decrypt it successfully. The encryption process begins with the user inputting plain text, which is then encrypted using the Elliptic Curve Cryptography (ECC) algorithm. ECC generates a unique key, ensuring the text is securely encrypted into an unreadable format. This encrypted text is then compressed using Huffman encoding, resulting in a smaller, more efficient version. Next, the Least Significant Bit

(LSB) technique embeds the compressed encrypted text into an image, creating a stego-image that appears normal but contains hidden data. The stego-image is then stored on IPFS, ensuring decentralized and secure storage. During decryption, the stego-image is retrieved from IPFS, and the system extracts the compressed encrypted message using LSB decoding. This message is decompressed with Huffman coding to retrieve the original encrypted text. The encrypted text is then decrypted using the ECC key, converting it back into the original plain text. A comparison was also performed between RSA and ECC showed that ECC is not only faster but also more secure, making it a better choice for this application. The process of encrypting text, compressing it, embedding it into an image, and storing it on IPFS was seamless and efficient. This system can be applied in various fields requiring high security, such as confidential communication, secure data storage, and privacy protection in digital transactions. By combining these advanced techniques, the system provides robust data protection suitable for both personal and professional use.

Future enhancements could include extending the steganography capabilities beyond images to include audio and video files and also implementing blockchain technology to securely store metadata or cryptographic hashes generated from IPFS. By recording these hashes on a blockchain, the system can provide an immutable and transparent ledger of data transactions and accesses.

## References

- Agrawal, E., & Pal, P. R. (2017). A secure and fast approach for encryption and decryption of message communication. ResearchGate.  
[https://www.researchgate.net/publication/320149845\\_A\\_Secure\\_and\\_Fast\\_Approach\\_for\\_Encryption\\_and\\_Decryption\\_of\\_Message\\_Communication](https://www.researchgate.net/publication/320149845_A_Secure_and_Fast_Approach_for_Encryption_and_Decryption_of_Message_Communication)
- Ahmed, A., & Naeem, M. (2022). Analysis of Most Common Encryption Algorithms. 04(02). <https://doi.org/10.24032/ijeacs/0402/003>
- Alanzy, M., Alomrani, R., Alqarni, B. and Almutairi, S. (2023). Image Steganography Using LSB and Hybrid Encryption Algorithms. Applied Sciences, [online] 13(21), p.11771. doi:<https://doi.org/10.3390/app132111771>.
- Anup, A., Ashok, R. and Raundale, P. (2019). Comparative Study of Data Compression Techniques. International Journal of Computer Applications, 178(28), pp.15–19. doi:<https://doi.org/10.5120/ijca2019919104>.
- Aslam, M.A., Rashid, M., Azam, F., Abbas, M., Rasheed, Y., Alotaibi, S.S. and Anwar, M.W. (2022). Image Steganography using Least Significant Bit (LSB) - A Systematic Literature Review. [online] IEEE Xplore.  
doi:<https://doi.org/10.1109/ICCIT52419.2022.9711628>.
- Fitriya, L.A. & Purboyo, Tito & Prasasti, Anggunmeka. (2017). A review of data compression techniques. International Journal of Applied Engineering Research. 12. 8956-8963.

- Fitriya, Purboyo and Prasasti (2017). A review of data compression techniques. [https://www.researchgate.net/publication/322557949\\_A\\_review\\_of\\_data\\_compression\\_techniques](https://www.researchgate.net/publication/322557949_A_review_of_data_compression_techniques)
- Hussain, (2013). A Survey of Image Steganography Techniques. [https://www.researchgate.net/publication/271863505\\_A\\_Survey\\_of\\_Image\\_Steganography\\_Techniques](https://www.researchgate.net/publication/271863505_A_Survey_of_Image_Steganography_Techniques)
- Imam, R., Areeb, Q.M., Alturki, A. and Anwer, F. (2021). Systematic and Critical Review of RSA Based Public Key Cryptographic Schemes: Past and Present Status. *IEEE Access*, 9, pp.155949–155976. doi:<https://doi.org/10.1109/access.2021.3129224>.
- Jan, A., Parah, S. A., Hussan, M., & Malik, B. A. (2021). Double layer security using crypto-stego techniques: a comprehensive review. *Health and Technology*, 12(1), 9–31. <https://doi.org/10.1007/s12553-021-00602-1>
- Joshi, R., Githika G, Ch, P., Fairouz SK and Shaik Mohammed Rafi (2020). Efficient Data compression using variable length Huffman coding. *SSRG international journal of VLSI & signal processing*, 7(2), pp.6–10. doi:<https://doi.org/10.14445/23942584/ijvsp-v7i2p102>.
- Lithmee. (2018, August 19). Difference between encryption and decryption. *Pediaa.Com*. <https://pediaa.com/difference-between-encryption-and-decryption/>
- Malik, N., & Abbas, A. (2023). Innovative Strategies: Ensuring data security in an evolving digital landscape. *ResearchGate*. [https://www.researchgate.net/publication/375609649\\_Innovative\\_Strategies\\_Ensuring\\_Data\\_Security\\_in\\_an\\_Evolving\\_Digital\\_Landscape](https://www.researchgate.net/publication/375609649_Innovative_Strategies_Ensuring_Data_Security_in_an_Evolving_Digital_Landscape)
- man, S. and Utama Siahaan, A.P. (2016). Huffman Text Compression Technique. *International Journal of Computer Science and Engineering*, 3(8), pp.103–108. doi:<https://doi.org/10.14445/23488387/ijcse-v3i8p124>.
- Mohammad Rafeek Khan, Kamal Upreti, Alam, M., Khan, H., Shams Tabrez Siddiqui, Haque, M. and Parashar, J. (2023). Analysis of Elliptic Curve Cryptography & RSA. *Journal of ICT standardisation*. doi:<https://doi.org/10.13052/jicts2245-800x.1142>.
- Nisha and Farik (2017). RSA Public Key Cryptography Algorithm – A Review. [https://www.researchgate.net/publication/318729097\\_RSA\\_Public\\_Key\\_Cryptography\\_Algorithm\\_-\\_A\\_Review](https://www.researchgate.net/publication/318729097_RSA_Public_Key_Cryptography_Algorithm_-_A_Review)
- Nishara Nizamuddin, Hasan, H.R. and Salah, K. (2018). IPFS-Blockchain-Based Authenticity of Online Publications. pp.199–212. doi:[https://doi.org/10.1007/978-3-319-94478-4\\_14](https://doi.org/10.1007/978-3-319-94478-4_14).
- Omotunde, H., & Ahmed, M. (2023). A Comprehensive Review of Security Measures in Database Systems: Assessing Authentication, Access Control, and Beyond. 115–133. <https://doi.org/10.58496/mjcs/2023/016>



Sabah, N., Abbas, N., None Lubna Emad Kadhim and Majeed, M. (2023). Hiding Information in Digital Images Using LSB Steganography Technique. *International journal of interactive mobile technologies*, 17(07), pp.167–178. doi:<https://doi.org/10.3991/ijim.v17i07.38737>.

Shantha, A., Renita, J. and Edna, E.N. (2019). Analysis and Implementation of ECC Algorithm in Lightweight Device. 2019 International Conference on Communication and Signal Processing (ICCSP). doi:<https://doi.org/10.1109/iccsp.2019.8697990>.

Srinivasan, Divyasree & Manojkumar, Khushi & Syed, Afreen & Nutakki, Harichandana. (2024). A Comprehensive Review on Advancements and Applications of Steganography. 10.13140/RG.2.2.13568.44807.

Sunday, A. E., & Olufunminiyi, O. E. (2023). An efficient data protection for cloud storage through encryption. *International Journal of Advanced Networking and Applications*, 14(05), 5609–5618. <https://doi.org/10.35444/ijana.2023.14505>

Trautwein, D., Raman, A., Tyson, G., Castro, I., Scott, W., Schubotz, M., Gipp, B. and Psaras, Y. (2022). Design and evaluation of IPFS. *Proceedings of the ACM SIGCOMM 2022 Conference*. doi:<https://doi.org/10.1145/3544216.3544232>.

Trinh Viet Doan, Psaras, Y., Ott, J. and Bajpai, V. (2022). Towards Decentralised Cloud Storage with IPFS: Opportunities, Challenges, and Future Directions. *arXiv (Cornell University)*. doi:<https://doi.org/10.48550/arxiv.2202.06315>.

Upadhyay, Gupta (2022). A LITERATURE REVIEW ON THE CONCEPT OF CRYPTOGRAPHY AND RSA ALGORITHM. [https://www.researchgate.net/publication/360175324\\_A\\_LITERATURE\\_REVIEW\\_ON\\_THE\\_CONCEPT\\_OF\\_CRYPTOGRAPHY\\_AND\\_RSA\\_ALGORITHM](https://www.researchgate.net/publication/360175324_A_LITERATURE_REVIEW_ON_THE_CONCEPT_OF_CRYPTOGRAPHY_AND_RSA_ALGORITHM)