A

Project Report

on

THREEFOLD SECURITY THROUGH HYBRID ENCRYPTION AND STEGANOGRAPHY KEY SAFEGUARDING

submitted in partial fulfillment of the requirements for the award of the

Bachelor of Technology

in

COMPUTER SCIENCE AND ENGINEERING

by

M V S Sarvani (20EG105327)

P Shivani (20EG105341)

K Rakshitha (20EG105342)

Under the guidance of

Dr. T Shyam Prasad

M.Tech., Ph.D

Assistant Professor, CSE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Venkatapur (V), Ghatkesar(M), Medchal (D), T.S - 500088

2023-24

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the report entitled **"THREEFOLD SECURITY THROUGH HYBRID ENCRYPTION AND STEGANOGRAPHY KEY SAFEGUARDING"** that is being submitted by M V S Sarvani (20EG105327), P Shivani (20EG105341) and K Rakshitha (20EG105342) in partial fulfillment for the award of B.Tech in Computer Science and Engineering to the Anurag University is a record of bonafide work carried out by them under my guidance and supervision.

The results embodied in this report have not been submitted to any other University or Institute for the award of any degree or diploma

Signature of Supervisor
Dr. T Shyam Prasad
M.Tech., Ph.D
Assistant Professor, CSE

Dean, Department of CSE
Dr. G Vishnu Murthy
M.Tech., Ph.D
Professor

External Examiner

DECLARATION

We hereby declare that the report entitled "**THREEFOLD SECURITY THROUGH HYBRID ENCRYPTION AND STEGANOGRAPHY KEY SAFEGUARDING**" submitted for the award of Bachelor of Technology degree is our original work and the report has not formed the basis for the award of any degree, diploma, associate ship or fellowship of similar other titles. It has not been submitted to any other University or Institution for the award of any degree or diploma.

Place: Anurag University, Hyderabad

M V S Sarvani (20EG105327)

P Shivani (20EG105341)

K Rakshitha (20EG105342)

Date: 20/04/2024

ACKNOWLEDGEMENT

We would like to express our sincere thanks and deep sense of gratitude to project supervisor **Dr. T Shyam Prasad** for his constant encouragement and inspiring guidance without which this project could not have been completed. His critical reviews and constructive comments improved our grasp of the subject and steered to the fruitful completion of the work. His patience, guidance and encouragement made this project possible.

We would like to express our special thanks to **Dr. V. Vijaya Kumar**, Dean School of Engineering, Anurag University, for his encouragement and timely support in our B.Tech program.

We would like to acknowledge our sincere gratitude for the support extended by **Dr. G. Vishnu Murthy** , Dean, Dept. of CSE ,Anurag University. We also express our deep sense of gratitude to **Dr. V V S S S Balaram**, Academic coordinator and **Dr. T Shyam Prasad** Project Coordinator and Project review committee members, whose research expertise and commitment to the highest standards continuously motivated us during the crucial stage of our project work.

ABSTRACT

In the contemporary landscape dominated by web technology and cloud computing, safeguarding data has become a critical priority across diverse industries. In this realm of data security, the integration of hybrid encryption and steganography stands as a formidable fortress against unauthorized access and interception. This research introduces a pioneering Multi-Level Steganography (MLS) algorithm designed to fortify data security through the integration of three potent encryption algorithms, namely AES, RSA, and BlowFish.Leveraging the efficiency of symmetric encryption for bulk data processing and the security assurances of asymmetric encryption for key exchange, this approach ensures both speed and confidentiality. The integration of these methodologies presents a multifaceted security posture. The asymmetrically exchanged key empowers the encryption of data while the steganographically safeguarded key adds a layer of obfuscation, rendering unauthorized access and decryption exponentially more challenging. It conceals cryptographic keys within innocuous carriers such as images or audio files. Through imperceptible embedding, it veils the existence of critical keys, fortifying the security architecture.In summary, the proposed Multi-Level Steganography algorithm emerges as a robust solution, offering superior data protection, enhanced image quality, and heightened security through the synergy of advanced encryption techniques.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF GRAPHS

1.INTRODUCTION

In the realm of cybersecurity, cryptography stands as a cornerstone, providing the fundamental principles and techniques for securing sensitive information in the digital age.Cryptography is the cornerstone of modern information security, encompassing techniques and methodologies for securing data through encryption and decryption.

- **Encryption**: Encryption is the process of converting plaintext data into ciphertext using cryptographic algorithms and keys. This transformation renders the original data unreadable to unauthorized parties, thereby preserving its confidentiality. In our project, we employ a hybrid encryption approach, combining symmetric and asymmetric encryption algorithms—Blowfish, AES, and RSA—to achieve a balance between security and performance.
- **Decryption**: Decryption is the reverse process of encryption, wherein ciphertext is converted back into plaintext using the corresponding decryption keys. By decrypting ciphertext with the appropriate keys, authorized parties can access the original plaintext data while maintaining its confidentiality and integrity.
- **Symmetric Encryption**: Symmetric encryption employs a single shared key for both encryption and decryption processes. Algorithms such as Blowfish and AES are examples of symmetric encryption algorithms utilized in our project. Symmetric encryption offers high-speed processing and is well-suited for securing large volumes of data.
- **Blowfish Algorithm :**Blowfish is a symmetric-key block cipher algorithm developed by Bruce Schneier in 1993. It operates on blocks of data, each containing 64 bits, and supports variable key lengths ranging from 32 to 448 bits. Blowfish is renowned for its complex key scheduling and key-dependent S-boxes.

Utilizing the Feistel cipher structure, Blowfish employs 16 rounds in its encryption process. Each round consists of four steps: XORing the left half of the

block with a subkey element, passing the result through the round function F, XORing the output of F with the right half of the block, and swapping the halves.

The round function F operates on 32-bit inputs, dividing them into four 8-bit blocks fed into four distinct S-boxes. The outputs of the first two S-boxes are added together, and the result is XORed with the outputs of the third S-box. Finally, the result is added to the output of the fourth S-box.

Blowfish's design enables high-speed encryption, making it one of the fastest algorithms in this regard. Its combination of complex key scheduling, key-dependent S-boxes, Feistel cipher structure, and efficient round function contribute to its effectiveness in securing data.

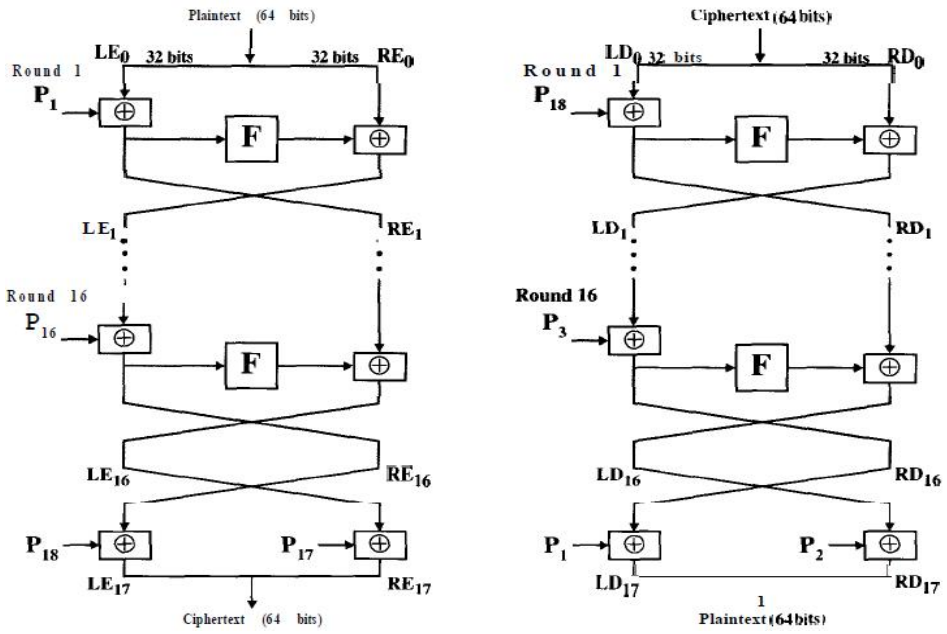


Figure 1.1 : Blowfish Algorithm

Advanced Encryption Standard Algorithm (AES):

The Advanced Encryption Standard (AES) is an iterative cipher based on a substitution-permutation network, contrasting with Feistel ciphers. AES operates on bytes rather than bits, treating plaintext blocks of 128 bits as 16 bytes organized in a 4x4 matrix. The algorithm comprises a series of linked operations, including substitutions and permutations. Unlike DES, the number of rounds in AES varies based on the key length: 10 rounds for 128-bit keys, 12 rounds for 192-bit keys, and

14 rounds for 256-bit keys. Each round utilizes a unique 128-bit round key derived from the original AES key. This structure enables AES to achieve robust security and efficiency in encryption operations.

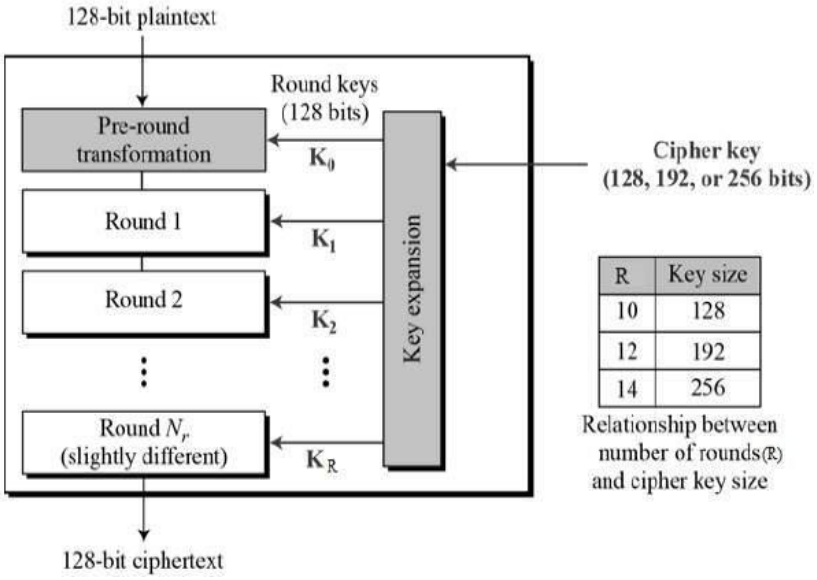


Figure 1.2 : AES Algorithm

In a typical round of AES encryption, there are four main sub-processes. First, Byte Substitution (SubBytes) replaces each of the 16 input bytes using a fixed lookup table (S-box), resulting in a 4x4 matrix. Next, ShiftRows shifts each row of the matrix to the left, with different shift amounts for each row. Then, MixColumns transforms each column using a mathematical function, except in the last round. Finally, AddRoundKey XORs the matrix bytes with the round key. If it's the last round, the output is the ciphertext; otherwise, the process repeats for another round. In the decryption process, these sub-processes are applied in reverse order for each round. Unlike Feistel ciphers, encryption and decryption in AES are implemented separately due to this reverse process, though they are closely related.

Asymmetric Encryption: Asymmetric encryption, also known as public-key cryptography, utilizes a pair of keys—a public key and a private key—for encryption and decryption, respectively. RSA is an asymmetric encryption algorithm employed in our project. Asymmetric encryption facilitates secure key exchange and enables digital signatures, enhancing data authenticity and integrity.

Rivest Shamir Adleman - RSA Algorithm :

The RSA algorithm, named after its inventors Ron Rivest, Adi Shamir, and Leonard Adleman, is a widely used asymmetric encryption scheme. It relies on the mathematical properties of large prime numbers and modular arithmetic. RSA involves the generation of a public key and a private key pair. The public key, consisting of a modulus and an exponent, is widely distributed and used for encrypting messages. The private key, derived from the public key, is kept secret and used for decrypting the encrypted messages.

RSA algorithm uses the following procedure to generate public and private keys:

Select two large prime numbers, p and q.

Multiply these numbers to find n = p x q, where n is called the modulus for encryption and decryption.

Choose a number e less than n, such that n is relatively prime to (p - 1) x (q -1). It means that e and (p - 1) x (q - 1) have no common factor except 1.

Choose "e" such that $1<e<\varphi(n)$, e is prime to $\varphi(n)$,

$$\gcd(e,d(n)) = 1$$

If n = p x q, then the public key is <e, n>. A plaintext message m is encrypted using public key <e, n>. To find ciphertext from the plain text following formula is used to get ciphertext C.

$$C = m^e \bmod n$$

Here, m must be less than n. A larger message (>n) is treated as a concatenation of messages, each of which is encrypted separately.

To determine the private key, we use the following formula to calculate the d such that:

$$D_e \bmod \{(p - 1) \times (q - 1)\} = 1$$

Or

$$D_e \bmod \varphi(n) = 1$$

The private key is <d, n>. A ciphertext message c is decrypted using private key <d, n>. To calculate plain text m from the ciphertext c following formula is used to get plain text m.

$$m = c^d \bmod n.$$

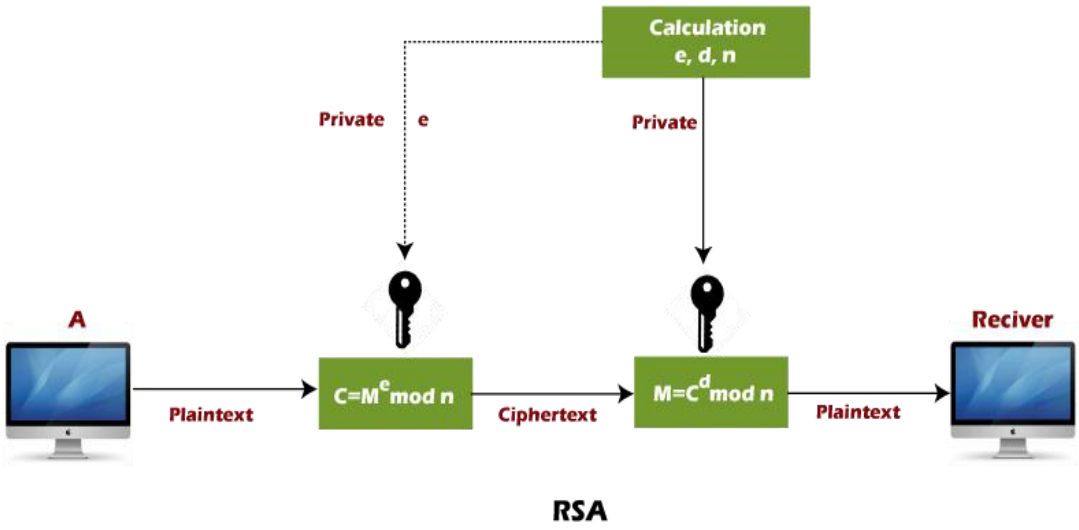


Figure 1.3 : RSA algorithm

- **Password Hashing:** Password hashing is a specific application of hashing used to store passwords securely. When a user creates or updates a password, the system hashes the password and stores the resulting hash value instead of the plaintext password. During authentication, the system hashes the provided password and compares the resulting hash value with the stored hash value. This way, even if the stored hash values are compromised, an attacker cannot easily determine the original passwords.
- **Secure Hash Algorithm -1(SHA-1) algorithm:**The Secure Hash Algorithm 1 (SHA-1) is a cryptographic hash function that generates a fixed-size hash value (160 bits), typically represented as a 40-character hexadecimal number. Developed by the National Security Agency (NSA) and published by the National Institute of Standards and Technology (NIST) in 1993, SHA-1 is widely used in various security applications and protocols, including SSL/TLS certificates, digital signatures, and integrity verification mechanisms. The SHA-1 algorithm operates by accepting an input message of any length and producing a fixed-size

hash value. It accomplishes this by breaking the input message into blocks and iteratively applying a series of logical and bitwise operations, including bitwise AND, OR, XOR, and shifting operations.

While SHA-1 has been widely used and considered secure for many years, vulnerabilities were discovered in the algorithm over time, primarily due to advances in computational power and cryptanalysis techniques. In particular, researchers found collision attacks, which allow for the creation of different input messages that produce the same hash value. These vulnerabilities raised concerns about the security of systems relying on SHA-1 for cryptographic integrity and led to the deprecation of SHA-1 in favor of stronger hash functions such as SHA-2 and SHA-3.

As a result of these vulnerabilities, SHA-1 is no longer considered secure for cryptographic purposes, and its use has been deprecated in favor of more robust hash functions. Organizations and industries have transitioned away from SHA-1 to prevent potential security vulnerabilities and ensure the integrity and confidentiality of sensitive data. Despite its deprecation, SHA-1 remains relevant in legacy systems and protocols, but its usage is discouraged in favor of more secure alternatives to mitigate the risk of cryptographic attacks.

- **Steganography (LSB Steganography):** LSB Steganography: Leveraging the least significant bit of pixel values in digital images, LSB steganography conceals encrypted data within cover media without perceptibly altering the visual content. By embedding encrypted information within images, audio files, or video streams, LSB steganography provides an inconspicuous means of data concealment, thereby thwarting detection by adversaries.

This approach finds applications in various domains such as secure communication, data protection, and confidentiality in sectors like finance, healthcare, and national security, offering a robust defense against cyber threats and unauthorized access to sensitive information.

2.LITERATURE SURVEY

[1] A Novel Steganography Technique for Digital Images Using the Least Significant Bit Substitution Method

Authors, Year : Shahid Rahman, Jamal Uddin, Habib Ullah Khan, Hameed Hussain, Ayaz Ali Khan, and Muhammad Zakarya, 2022.

The proposed method in this paper focuses on enhancing the security and reliability of image steganography, particularly through the Least Significant Bit (LSB) substitution technique. This method aims to minimize error rates during the embedding process and achieve greater reliability using a novel algorithm based on value difference. The approach encompasses various types of digital images, including RGB, grayscale, texture, and aerial images, to ensure versatility and effectiveness across different scenarios. LSB substitution involves embedding secret data into digital images by replacing the least significant bits (LSBs) of pixel values with bits of the secret message. While simple to implement, it lacks robust security as it relies solely on the obscurity of LSBs. LSB substitution is vulnerable to statistical analysis techniques, making it easier to detect hidden data through steganalysis methods. Additionally, it does not involve encryption of the secret message, leaving the data susceptible to unauthorized access if discovered.

[2] Secure RGB image steganography based on modified LSB substitution

Authors, Year : L. Almazaydeh , 2020.

The proposed edge-based image steganographic method utilizes a parameterized Canny edge detector to embed secret message bits into the blue color channel of an RGB image. This approach aims to leverage the properties of the blue channel, which previous research suggests has less distinct visual perception compared to the red and green channels, making it a suitable candidate for concealing hidden data. The method selects the blue channel specifically to minimize the visual impact on

the cover image while ensuring the security of the hidden message. Edge-based LSB steganography, while innovative, possesses certain disadvantages. Primarily, it relies heavily on the obscurity of LSB embedding within the blue channel and edge-based selection for embedding locations, which may not provide sufficient security against sophisticated attacks. Moreover, the reliance on LSB embedding alone leaves the hidden message vulnerable to detection through statistical analysis techniques.

[3] An adaptive color image steganography method using adjacent pixel value differencing and LSB substitution technique

Authors, Year : M. Kalita, T. Tuithung, and S. Majumder, 2019.

The proposed adaptive color image steganography method employs adjacent pixel value differencing (APVD) and LSB substitution technique to embed secret information into a color image. By dynamically selecting pixels based on subtle changes in adjacent pixel values, the method minimizes detection risk. LSB substitution ensures imperceptibility by hiding information in the least significant bits of pixel values. This approach enhances security while maximizing payload capacity in steganographic applications.The second method of Pixel Value Differencing (PVD) steganography, while straightforward, possesses certain disadvantages. Primarily, it relies solely on the complexity of the algorithm used to hide the message, without incorporating encryption to protect the message's confidentiality. This lack of encryption leaves the hidden message vulnerable to unauthorized access if detected. Additionally, PVD may introduce more noticeable changes to the image since it directly alters pixel values based on differences between adjacent pixels, potentially affecting visual fidelity.

[4] Inverted LSB image steganography using an adaptive pattern to improve imperceptibility

Authors, Year : S.Rustad, D.R.I.M.Setiadi, A.Syukur, and P.N.Andono, 2022.

The proposed method of inverted LSB image steganography with adaptive pattern involves hiding information within the least significant bits (LSBs) of pixel values in an image. It employs an adaptive pattern to select pixels for embedding, dynamically adjusting based on image characteristics. The LSBs of selected pixels are inverted to conceal the information effectively. This method aims to improve imperceptibility by minimizing visual impact, making it harder to detect the hidden data.Adaptive Inverted LSB Substitution, focuses on enhancing imperceptibility by dynamically selecting the optimal embedding pattern to minimize the error ratio. This technique employs inverted LSB substitution with adaptive pattern selection, aiming to reduce the visual impact of embedding while maintaining capacity and security. However, despite its emphasis on imperceptibility and capacity optimization, this method may still have certain disadvantages. Notably, while it aims to minimize the error ratio during message embedding, it may face challenges in achieving high levels of security. The reliance on LSB substitution alone may make the hidden message vulnerable to detection through statistical analysis techniques. Additionally, the method's effectiveness in securely concealing the message may be limited compared to hybrid encryption techniques.

Author	**Method**	**Advantages**	**Disadvantages**
●Shahid Rahman ●Jamal Uddin ●Habib Ullah Khan ●Hameed Hussain ●Ayaz Ali Khan ●MuhammadZakarya	A Novel Steganography Technique for Digital Images Using the Least Significant Bit Substitution Method	- Simple implementation. - Low computational overhead. - Fast encoding and decoding processes. - Suitable for	- Limited payload capacity. - Susceptible to statistical analysis. - Lower security compared to encryption-based methods.

		scenarios with lower security requirements	
●L. Almazaydeh	Secure RGB image steganography based on modified LSB substitution	-Simple and easy -Low computational overhead -Fast encoding and decoding processes	-low security -limited payload capacity -susceptible to statistical analysis
●M. Kalita ●T. Tuithung ● S.Majumder	Inverted LSB image steganography using an adaptive pattern to improve imperceptibility	-improved imperceptibility -Potential resistance to visual detection -Balance between security and visual quality	-Complexity in implementation -Potential loss of data -Limited capacity -Dependence on image characteristics
●S.Rustad,D.R ●.I.M.Setiadi,A ●.Syukur,and ●P.N.Andono	An adaptive color image steganography method using adjacent pixel value differencing and LSB substitution technique,	-improved security -reduced visual impact -utilizes pixel	complexity in implementation -potential loss of data -limited capacity

		relationships -robustness	-dependence on image characteristics

Table 1 : Comparison between methods

3. SYSTEM SPECIFICATIONS

3.1 System Requirements

- **Hardware Requirements:**

  System : Pentium i3 Processor

  Hard Disk : 500 GB.

  Monitor : 15'' LED

  Input Devices : Keyboard, Mouse

  Ram : 4 GB

- **Software Requirements:**

  Operating system : Windows 10.

  Coding Language : Python

  Tool : Google Colab

4. SYSTEM METHODOLOGY

4.1 System Analysis

4.1.1 Existing System

The existing steganography technique utilizing Least Significant Bit (LSB) substitution, while effective in concealing information within digital images, presents certain disadvantages that warrant consideration. Firstly, one notable drawback is its limited payload capacity. LSB substitution primarily operates within the least significant bits of pixel color values, which offer a constrained space for embedding data. Consequently, this restriction can limit the amount of information that can be concealed within an image, posing a challenge for applications requiring larger payloads.

Moreover, the technique is susceptible to statistical analysis, which poses a significant security risk. Because LSB substitution involves subtly altering the least significant bits of pixel color values, it may introduce detectable patterns or anomalies in the stego image. Advanced statistical analysis techniques can exploit these patterns, potentially leading to the discovery of the hidden information. Thus, the reliance solely on LSB substitution for concealing data may not provide adequate protection against determined adversaries employing sophisticated detection methods.

Additionally, the security offered by LSB substitution is inherently lower compared to encryption-based steganography methods. While LSB substitution focuses on imperceptibility and simplicity, it lacks the robust encryption mechanisms employed by encryption-based techniques. Without encryption, the hidden information remains vulnerable to unauthorized access if the stego image is compromised or subjected to steganalysis attacks. As a result, applications requiring high levels of data confidentiality and integrity may find LSB substitution inadequate for their security needs.To address these disadvantages, alternative steganographic methods incorporating encryption or innovative adaptations may be necessary to enhance security, increase payload capacity, and mitigate vulnerabilities against potential attacks.

This project presents the implementation of the Threefold Security, which involves several intricate steps to ensure robust data protection in the digital landscape.

Initially, a Multi-Level Steganography (MLS) algorithm was developed integrating three potent encryption algorithms: AES, RSA, and Blowfish. This integration capitalizes on the efficiency of symmetric encryption for bulk data processing and the security assurances of asymmetric encryption for key exchange. Symmetric encryption, exemplified by algorithms such as Blowfish and AES, provides high-speed processing, while asymmetric encryption, represented by RSA, facilitates secure key exchange and digital signatures. The hybrid encryption approach ensures a balance between security and performance. Additionally, advanced steganography techniques are employed, surpassing the limitations of traditional LSB Substitution. The keys used in encryption and steganography are concealed within cover media using LSB steganography, creating a triple-layered defense against unauthorized access and interception. During the encryption process, the plaintext undergoes successive encryption layers, starting with Blowfish, followed by RSA, and finally AES. The keys used in each layer are securely stored and managed throughout the process. To enhance key storage security, the list of keys is converted into a string and encrypted using a key derived from a user-input password. This encrypted key string is then embedded into a cover image using LSB steganography, ensuring that even if attackers gain access to the encrypted image, they won't easily discern the presence of the keys. This multi-layered approach offers superior data protection, enhanced image quality, and heightened security, making it a robust defense against cyber threats and unauthorized access to sensitive information in various domains such as finance, healthcare, and national security

4.1.2 Proposed System

In the contemporary digital landscape, the escalating threat of malicious attacks underscores the need for robust information security. As online services proliferate, safeguarding data becomes increasingly crucial. While encryption has evolved to counter cryptanalysis, encrypted data remains a target. Threefold Security addresses

this challenge through hybrid encryption and advanced steganography, providing a triple-layered defense by concealing cryptographic keys.

The proposed Threefold Security approach introduces key improvements:

- **Hybrid Encryption**: Sensitive information undergoes hybrid encryption for an added layer of security.
- **Advanced Steganography**: An advanced steganography technique is employed, addressing the limitations of LSB Substitution.
- **Concealing Keys**: Cryptographic keys used in encryption and steganography are hidden within the cover image, creating a triple-layered defense.

4.1.2.1 Threefold Encryption System:

- **Blowfish Encryption:** The plaintext is initially encrypted using the Blowfish algorithm with a randomly generated key (KBlowfish).
- **RSA Encryption:** The resulting ciphertext is further encrypted using RSA with a public key (KRSA-Public), and the corresponding private key (KRSA-Private) is stored securely.
- **AES Encryption:** Finally, the doubly encrypted ciphertext is further encrypted using AES with another randomly generated key (KAES).

Key Storage:

Throughout the encryption process, the keys used in each layer (KBlowfish, KRSA-Private, KAES) are stored in a list (L). This allows for secure key management and facilitates decryption.

List String and Encryption:

The list of keys (L) is converted into a string (LS) using separators. This string is then encrypted using AES with a key derived from a user-input password (PW). This enhances the security of key storage.

Steganography:

The encrypted key string (LS-Encrypted) is embedded into a cover image using Least Significant Bit (LSB) steganography. This ensures that even if an attacker gains access to the encrypted image, they won't easily discern the presence of the key.

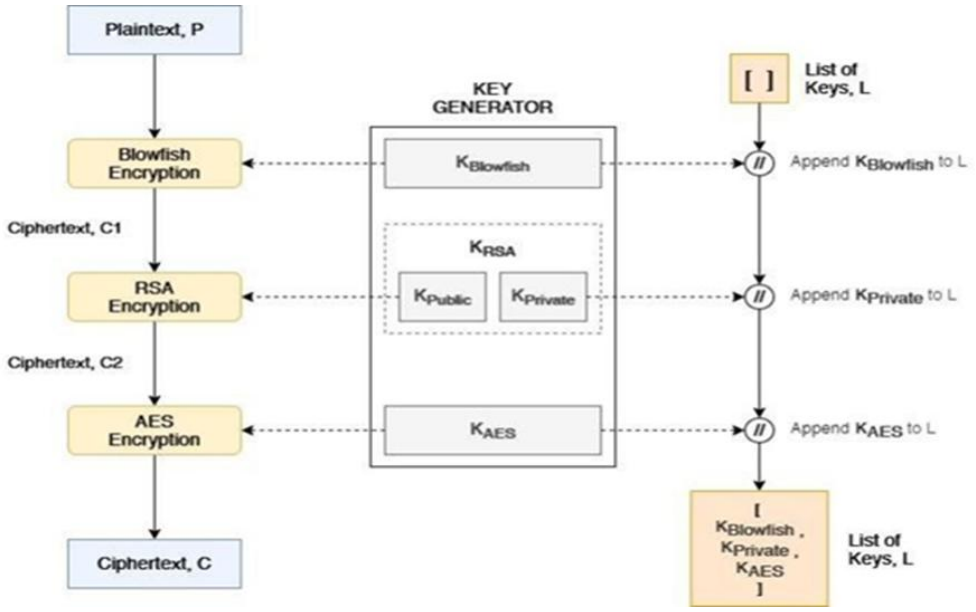


Figure 4.1 : Hybrid encryption and Key Generation

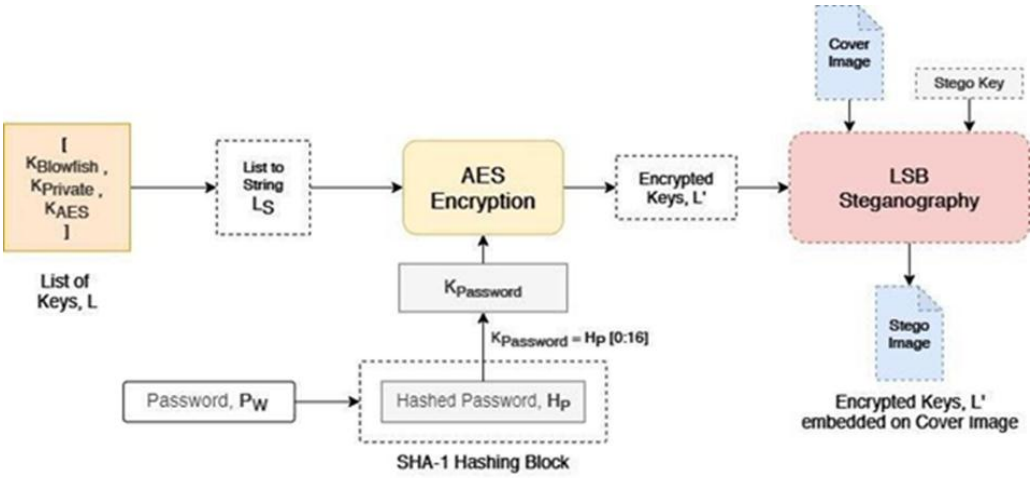


Figure 4.2: LSB Steaganography

Let's consider a scenario where Alice wants to securely transmit a confidential message (plaintext) to Bob using the described threefold encryption system with key

storage and LSB steganography. For simplicity, we'll use sample values for illustration.

Initialization:

Plaintext (P): "Meet me at the secret location."

Blowfish Key (KBlowfish): Randomly generated 32-bit key.

RSA Public Key (KRSA-Public): Randomly generated 1024-bit public key.

RSA Private Key (KRSA-Private): Corresponding private key.

AES Key (KAES): Randomly generated 128-bit key.

List of Keys (L): Empty initially.

Blowfish Encryption:

Ciphertext (C1): Blowfish encryption using P and KBlowfish.

Updated List of Keys (L): [KBlowfish]

RSA Encryption:

Ciphertext (C2): RSA encryption using C1 and KRSA-Public.

Updated List of Keys (L): [KBlowfish, KRSA-Private]

AES Encryption:

Final Ciphertext (C): AES encryption using C2 and KAES.

Updated List of Keys (L): [KBlowfish, KRSA-Private, KAES]

Secure Key Storage and Steganography:

List String (LS): Convert L to a string with separators: KBlowfish x KRSA-Private x KAES

User-Input Password (PW): "SecretPassword123"

Hashed Password (HP): SHA-1 hash of PW.

Key from Password (KPassword): First 16 bits of HP.

Encrypted List String (LS-Encrypted): AES encryption of LS using KPassword.
Stego Image: Embed LS-Encrypted into a cover image using LSB steganography.

Now, Alice sends both the Stego Image and the final Ciphertext (C) to Bob. Bob can extract the LS-Encrypted from the Stego Image using the same password and decrypt LS to obtain the original List of Keys (L). Using L, Bob can then decrypt the Ciphertext (C) sequentially through the three encryption layers to retrieve the original plaintext (P). This multi-layered approach enhances the security of the communication and protects the keys using encryption and steganography.

4.1.3 Advantages of Proposed System :

The proposed Threefold Security system offers several advantages over traditional encryption methods:

Triple-Layered Defense: By combining hybrid encryption and advanced steganography, the system provides a triple-layered defense mechanism. This approach significantly enhances the security of sensitive information by employing multiple layers of protection. Even if one layer is compromised, the other layers remain intact, ensuring robust security against malicious attacks.

Hybrid Encryption: The use of hybrid encryption adds an extra layer of security to the system. By encrypting sensitive information using multiple encryption algorithms (Blowfish, RSA, AES), the system mitigates the vulnerabilities associated with individual encryption methods. This enhances the confidentiality and integrity of the encrypted data, making it more resistant to cryptographic attacks.

Advanced Steganography: The adoption of advanced steganography techniques addresses the limitations of traditional LSB substitution. By concealing cryptographic keys within the cover image, the system achieves a higher level of security. This ensures that even if the encrypted image is accessed by unauthorized parties, the presence of the keys remains undetectable, enhancing the overall security posture of the system.

Secure Key Management: Throughout the encryption process, the keys used in each layer are securely stored in a list. This ensures proper key management and facilitates decryption when needed. By securely managing cryptographic keys, the system prevents unauthorized access to sensitive information and minimizes the risk of key compromise.

Enhanced Key Storage Security: The conversion of the key list into a string and its encryption using AES with a user-input password enhances the security of key storage. This ensures that even if the encrypted key string is intercepted, it remains protected and inaccessible to unauthorized users. This approach strengthens the overall security of the system and safeguards sensitive cryptographic keys from potential attacks.

4.2 System Study

4.2.1 Feasibility Study

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are:

- Economical Feasibility
- Technical Feasibility
- Social Feasibility

Economical Feasibility: This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

Technical Feasibility: This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

Social Feasibility: The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as

a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

4.3 System Testing

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub- assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the

Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

Types of Tests :

Unit testing: Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration testing: Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by

successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Functional test: Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals. Functional testing is centered on the following items:

Valid Input	: identified classes of valid input must be accepted.
Invalid Input	: identified classes of invalid input must be rejected.
Functions	: identified functions must be exercised.
Output	: identified classes of application outputs must be exercised.
Systems/Procedures	: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Testing: System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing: White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing: Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot "see" into it. The test provides inputs and responds to outputs without considering how the software works.

Unit Testing: Unit testing is usually conducted as part of a combined code and unit test phase of the software life cycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

- **Test objectives**

  All field entries must work properly.

  Pages must be activated from the identified link.

  The entry screen, messages and responses must not be delayed.

- **Features to be tested**

  Verify that the entries are of the correct format

  No duplicate entries should be allowed

  All links should take the user to the correct page.

Integration Testing: Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects. The task of the integration test is to

check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Acceptance Testing: User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

5. SOFTWARE ENVIRONMENT

5.1 The Python Programming Language

Python is a high-level, interpreted programming language known for its simplicity, readability, and versatility. It has gained immense popularity in various fields due to its extensive libraries, easy syntax, and broad community support. Here are some key aspects of Python programming language and its application in technology:

- **Simplicity and Readability**: Python's syntax is designed to be easy to read and understand, making it suitable for both beginners and experienced programmers. Its clean and concise code structure enhances productivity and reduces development time.
- **Extensive Libraries:** Python boasts a rich collection of libraries and frameworks for various purposes, including data science (NumPy, Pandas, Matplotlib), machine learning (scikit-learn, TensorFlow, PyTorch), web development (Django, Flask), and more. These libraries provide pre-built functions and tools that simplify complex tasks and accelerate development.
- **Versatility:** Python is a multipurpose language that supports a wide range of applications, from web development and desktop GUIs to scientific computing and automation. Its versatility makes it a preferred choice for developers working on diverse projects across different domains.
- **Community Support:** Python has a vibrant and active community of developers, contributors, and enthusiasts. This community-driven ecosystem fosters collaboration, knowledge sharing, and continuous improvement of the language and its associated tools and libraries.
- **Scalability:** While Python is often criticized for its performance compared to lower-level languages like C++ or Java, it offers scalability through various means. Techniques such as code optimization, using compiled extensions (e.g., Cython), and asynchronous programming (with libraries like asyncio) help improve performance and scalability for large-scale applications.

- **Integration with Other Technologies:** Python seamlessly integrates with other technologies and platforms, allowing developers to leverage existing systems and services. It can interact with databases, web servers, cloud services, and APIs, facilitating interoperability and system integration.
- **Rapid Prototyping and Development:** Python's simplicity and rich ecosystem enable rapid prototyping and development of software applications. Developers can quickly build prototypes, iterate on ideas, and test concepts, which is particularly advantageous in agile development environments.
- **Support for Emerging Technologies:** Python remains at the forefront of emerging technologies such as artificial intelligence, machine learning, and data science. Its libraries and frameworks provide powerful tools for developing and deploying advanced solutions in these rapidly evolving fields.

5.2 Google Colab

Google Colab, short for Google Colaboratory, is a cloud-based development environment provided by Google that enables users to write, execute, and share Python code collaboratively. Hosted on Google Drive, Colab offers a Jupyter notebook interface, allowing users to create and run Python code cells interactively. One of the standout features of Google Colab is its integration with Google Drive, which enables seamless access to files and data stored in Google Drive directly from the notebook environment. Moreover, Colab provides free access to powerful GPU and TPU resources, making it particularly attractive for tasks involving deep learning and machine learning, as these accelerators significantly speed up computations. Additionally, Colab supports the installation of third-party libraries and packages, enabling users to leverage a wide range of tools and frameworks for various tasks, including data analysis, visualization, and model training. Furthermore, Colab notebooks can be easily shared with collaborators, allowing for real-time collaboration and code review. Overall, Google Colab offers a convenient and

efficient platform for coding, experimenting, and collaborating on Python projects without the need for local installation or hardware resources.

6.IMPLEMENTATION

The project is a practical demonstration of an advanced data security solution that integrates multi-layered encryption and steganography techniques. By leveraging a combination of symmetric (Blowfish, AES) and asymmetric (RSA) encryption algorithms, sensitive data is encrypted with a focus on both speed and confidentiality. To further enhance security, the encrypted data is embedded within seemingly innocuous carriers such as images using LSB steganography, adding an additional layer of obfuscation to thwart unauthorized access. Key management is robustly handled through secure storage and encryption of cryptographic keys, ensuring comprehensive protection. The decryption process effectively reverses the encryption steps, allowing authorized users to retrieve the original plaintext data. Overall, this approach provides superior data protection, safeguarding the integrity and confidentiality of information vital across various domains including secure communication, data protection, and confidentiality in critical sectors like finance, healthcare, and national security.

6.1 Packages

pycryptodome: This package offers a wide range of cryptographic functionalities such as symmetric and asymmetric encryption, hashing, and more. It forms the core of the project's encryption and decryption processes, ensuring data security.class also contains methods for computing the block's hash and verifying proof of work.

stegano: This package specializes in image steganography, enabling the concealment of encrypted data within images. By embedding data into digital images using steganography techniques, the package enhances data security by hiding sensitive information in plain sight.

python-docx: This package facilitates interaction with Microsoft Word (.docx) files, enabling the extraction of plaintext from such documents. It plays a crucial role in

handling text data, particularly for input or output operations involving Word documents in the project.

6.2 Modules:

1. Crypto.Cipher Module:

This module provides classes and functions for symmetric encryption algorithms like Blowfish and AES, used for encryption and decryption operations.

2. Crypto.PublicKey Module:

This module includes classes for asymmetric encryption algorithms and key management, used for RSA encryption and decryption operations.

3. Crypto.Util.Padding Module:

This module offers functions for padding and unpadding data according to various padding schemes, ensuring proper data alignment before encryption and after decryption.

4. binascii Module:

This module provides functions for converting binary data to ASCII-encoded hexadecimal representation and vice versa.

5. hashlib Module:

This module offers secure hash and message digest algorithms, used for generating hash values for data integrity verification or cryptographic purposes.

6. json Module:

This module provides functions for encoding and decoding JSON data, used for serializing and deserializing data in JSON format.

7. string Module:

The string module contains various string manipulation functions and constants, which may be used for generating random strings or performing other string-related operations.

8. random Module:

The random module offers functions for generating random numbers, selecting random elements from sequences, and shuffling sequences, used for generating random keys or initialization vectors (IVs) for cryptographic operations.

9. stegano Module:

The stegano module is a Python library for steganography, used for hiding and extracting secret data within image files using LSB (Least Significant Bit) steganography techniques.

10. datetime Module:

The datetime module provides classes and functions for working with dates and times, used for logging the start and end times of certain operations.

6.3 Input :

In this project, the input format primarily consists of a document file (with a `.docx` extension) containing the plaintext message to be encrypted. The document file may contain text, images, tables, or any other content supported by the Microsoft Word format. Additionally, the user is prompted to input a password during the encryption process, which serves as the key for encrypting and decrypting the message. The password should be entered as a string. Optionally, for the LSB steganography part, an input image file (e.g., `.jpg`, `.png`) may be provided, along with the secret message to be hidden within the image. The project also allows for customizing the encryption process by specifying parameters such as the case (e.g., upper-case-only, lower-case-only), and the inclusion or exclusion of punctuation characters in the generated encryption key.

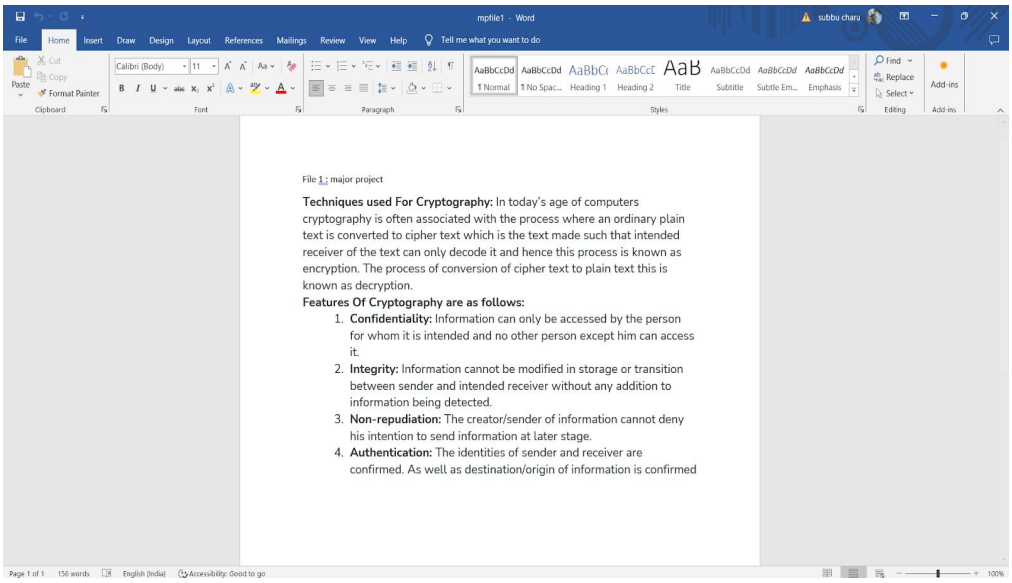


Figure 6.1 : mpfile.docx

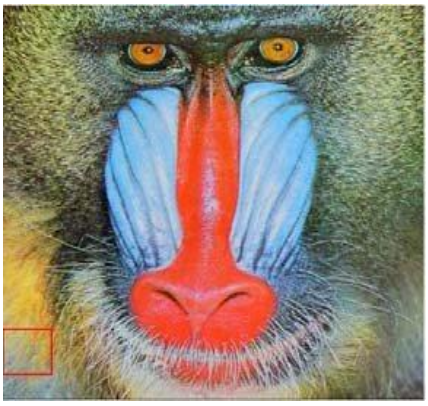


Figure 6.2 : Baboon

Figure 6.3 : Bell Pepper

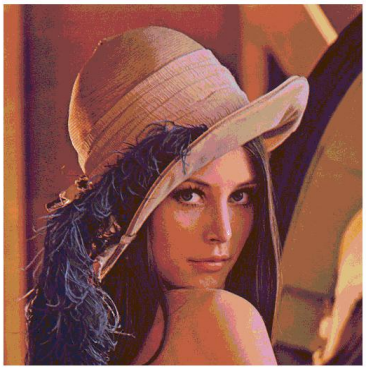


Figure 6.4 : Lena

In this project, the input format consists of a Microsoft Word document file named mpfile.docx, which serves as the input plaintext message for the proposed hybrid encryption algorithm. This algorithm combines the encryption techniques of Blowfish, RSA, and AES to secure the data. After generating the encryption keys using these algorithms, the keys are stored within an image using LSB steganography. This image acts as a safeguard for the keys, ensuring their security during transmission or storage. It's important to note that only one image is used in this method for steganography key safeguarding. However, the effectiveness of this approach is evaluated by comparing it with existing methods that may utilize different images for steganography.

6.4 Coding

Main.py

```
!pip install pycryptodome
!pip install stegano
from Crypto.Cipher import Blowfish, PKCS1_OAEP, AES
from Crypto.PublicKey import RSA
from Crypto.Util.Padding import pad, unpad
from binascii import hexlify , unhexlify
import hashlib , json, string, random
from stegano import lsb
from datetime import datetime
# Key Generator
def key_generator(size, case="default", punctuations="required"):
    if case=="default" and punctuations=="required":
        return ''.join(random.choices(string.ascii_uppercase + string.ascii_lowercase
+ string.digits + string.punctuation, k = size))
    elif case=="upper-case-only" and punctuations=="required":
```

```
            return ''.join(random.choices(string.ascii_uppercase + string.digits +
string.punctuation, k = size))
    elif case=="lower-case-only"   and punctuations=="required":
            return ''.join(random.choices(string.ascii_lowercase + string.digits +
string.punctuation, k = size))
    elif case=="default" and punctuations=="none":
            return ''.join(random.choices(string.ascii_uppercase + string.digits +
string.ascii_lowercase, k = size))
    elif case=="lower-case-only"   and punctuations=="none":
          return ''.join(random.choices(string.ascii_lowercase + string.digits , k =
size))
    elif case=="upper-case-only" and punctuations=="none":
       return ''.join(random.choices(string.ascii_uppercase + string.digits, k = size))
!pip install python-docx
from docx import Document

# Load the .docx file
doc = Document('/content/mpfile1.docx')

# Extract text from paragraphs
plaintext = ""
for paragraph in doc.paragraphs:
    plaintext += paragraph.text + "\n"
print(plaintext)
if isinstance(plaintext, str):
    plaintext = plaintext.encode('utf-8')
from binascii import hexlify
log_plaintext_length = len(hexlify(plaintext))

# Password for Keys
```

```
password = 'anurag2024' #input('Enter Password: ')
log_password_length = len(password)

log_start_time = datetime.now()

hash = hashlib.sha1()
hash.update(password.encode())
password_encryption_cipher = AES.new( hash.hexdigest()[:16].encode() ,
AES.MODE_CBC, iv= '16bitAESInitVect'.encode())
keys_iv = {}

# Blowfish Layer 1

blowfish_key = key_generator(size=16).encode()
blowfish_cipher = Blowfish.new(blowfish_key, Blowfish.MODE_CBC)

blowfish_ciphertext = blowfish_cipher.encrypt(pad(plaintext,
Blowfish.block_size ))

keys_iv['blowfish_iv'] = hexlify(blowfish_cipher.iv).decode()
keys_iv['blowfish_key'] = hexlify(blowfish_key).decode()
# RSA Layer 2

rsa_key = RSA.generate(2048)
rsa_private_key = rsa_key
rsa_public_key = rsa_key.publickey()

cipher_rsa = PKCS1_OAEP.new(rsa_public_key)
rsa_plaintext = blowfish_ciphertext
```

```
rsa_ciphertext = bytearray()
for i in range(0, len(rsa_plaintext), 190):
   rsa_ciphertext.extend(cipher_rsa.encrypt(rsa_plaintext[i:i+190]))

keys_iv['rsa_n'] = rsa_private_key.n
keys_iv['rsa_e'] = rsa_private_key.e
keys_iv['rsa_d'] = rsa_private_key.d
 # AES Layer 3
aes_key =   key_generator(size=16).encode()
aes_cipher = AES.new(aes_key, AES.MODE_CBC)
aes_plaintext = rsa_ciphertext

aes_ciphertext = aes_cipher.encrypt(pad(aes_plaintext, AES.block_size))

ciphertext = aes_ciphertext
with open('/content/mpfile1_enc.docx', 'w') as file:
   file.write(hexlify(ciphertext).decode())
log_ciphertext_length = len(hexlify(ciphertext))

keys_iv['aes_iv'] = hexlify(aes_cipher.iv).decode()
keys_iv['aes_key'] = hexlify(aes_key).decode()


# Encryption of Key and IV String
encrypted_keys_and_iv                                                    =
hexlify(password_encryption_cipher.encrypt(pad(json.dumps(keys_iv).encode(),
AES.block_size)))
# PIL module is used to extract
# pixels of image and modify it
from PIL import Image
```

```
# Convert encoding data into 8-bit binary
# form using ASCII value of characters
def genData(data):

# list of binary codes
# of given data
newd = []

for i in data:
newd.append(format(ord(i), '08b'))
return newd

# Pixels are modified according to the
# 8-bit binary data and finally returned
def modPix(pix, data):

datalist = genData(data)
lendata = len(datalist)
imdata = iter(pix)

for i in range(lendata):

# Extracting 3 pixels at a time
pix = [value for value in imdata.__next__()[:3] +
imdata.__next__()[:3] +
imdata.__next__()[:3]]

# Pixel value should be made
# odd for 1 and even for 0
```

```
for j in range(0, 8):
if (datalist[i][j] == '0' and pix[j]% 2 != 0):
pix[j] -= 1

elif (datalist[i][j] == '1' and pix[j] % 2 == 0):
if(pix[j] != 0):
pix[j] -= 1
else:
pix[j] += 1
# pix[j] -= 1

# Eighth pixel of every set tells
# whether to stop ot read further.
# 0 means keep reading; 1 means thec
# message is over.
if (i == lendata - 1):
if (pix[-1] % 2 == 0):
if(pix[-1] != 0):
pix[-1] -= 1
else:
pix[-1] += 1

else:
if (pix[-1] % 2 != 0):
pix[-1] -= 1

pix = tuple(pix)
yield pix[0:3]
yield pix[3:6]
yield pix[6:9]
```

```
def encode_enc(newimg, data):
w = newimg.size[0]
(x, y) = (0, 0)

for pixel in modPix(newimg.getdata(), data):

# Putting modified pixels in the new image
newimg.putpixel((x, y), pixel)
if (x == w - 1):
x = 0
y += 1
else:
x += 1

# Encode data into image
def encode():
img = input("Enter image name(with extension) : ")
image = Image.open(img, 'r')

data = encrypted_keys_and_iv.decode()
if (len(data) == 0):
raise ValueError('Data is empty')

newimg = image.copy()
encode_enc(newimg, data)

new_img_name = input("Enter the name of new image(with extension) : ")
newimg.save(new_img_name, str(new_img_name.split(".")[1].upper()))
```

```
# Decode the data in the image
def decode():
img = input("Enter image name(with extension) : ")
image = Image.open(img, 'r')

data = ''
imgdata = iter(image.getdata())

while (True):
pixels = [value for value in imgdata.__next__()[:3] +
imgdata.__next__()[:3] +
imgdata.__next__()[:3]]

# string of binary data
binstr = ''

for i in pixels[:8]:
if (i % 2 == 0):
binstr += '0'
else:
binstr += '1'

data += chr(int(binstr, 2))
if (pixels[-1] % 2 != 0):
return data

# Main Function
def main():
a = int(input(":: Welcome to Steganography ::\n"
"1. Encode\n2. Decode\n"))
```

```
if (a == 1):
encode()

elif (a == 2):
print("Decoded Word : " + decode())
else:
raise Exception("Enter correct input")

# Driver Code
if __name__ == '__main__' :

# Calling main function
main()


log_end_time = datetime.now()

log_duration = str(log_end_time - log_start_time)
hash = hashlib.sha1()
hash.update(password.encode())
password_decryption_cipher = AES.new( hash.hexdigest()[:16].encode() ,
AES.MODE_CBC, iv= '16bitAESInitVect'.encode())

decrypted_keys_iv =
json.loads(unpad(password_decryption_cipher.decrypt(unhexlify(encrypted_keys
_and_iv)), AES.block_size))

#Initializations
decryption_key_aes = unhexlify(decrypted_keys_iv['aes_key'])
decryption_iv_aes = unhexlify(decrypted_keys_iv['aes_iv'])
```

```
decryption_key_rsa = RSA.construct(rsa_components = (decrypted_keys_iv['rsa_n'] , decrypted_keys_iv['rsa_e'] , decrypted_keys_iv['rsa_d']))
decryption_iv_blowfish = unhexlify(decrypted_keys_iv['blowfish_iv'])
decryption_key_blowfish = unhexlify(decrypted_keys_iv['blowfish_key'])


aes_cipher_decryption = AES.new(decryption_key_aes, AES.MODE_CBC, iv=decryption_iv_aes)
rsa_cipher_decryption = PKCS1_OAEP.new(decryption_key_rsa)
blowfish_cipher_decryption = Blowfish.new(decryption_key_blowfish, Blowfish.MODE_CBC, iv=decryption_iv_blowfish)
import re

# AES DECRYPTION
ciphertext_rsa = unpad(aes_cipher_decryption.decrypt(unhexlify(ciphertext)), AES.block_size)

# RSA DECRYPTION
ciphertext_blowfish = bytearray()
for i in range(0, len(ciphertext_rsa), 256):
   ciphertext_rsa_segment = ciphertext_rsa[i:i+256]
    ciphertext_blowfish.extend(rsa_cipher_decryption.decrypt(ciphertext_rsa_segment))
# BLOWFISH DECRYPTION
decrypted_plaintext_bytes = unpad(blowfish_cipher_decryption.decrypt(ciphertext_blowfish), Blowfish.block_size)

# Decode bytes to string
```

```
decrypted_plaintext = decrypted_plaintext_bytes.decode('utf-8')

# Remove non-printable ASCII characters
decrypted_plaintext = re.sub(r'[^\x20-\x7E]', '', decrypted_plaintext)

print(decrypted_plaintext)
with open('./content/mpfile1_dec.txt', 'wb') as file:
    file.write(decrypted_plaintext)

#print("Decrypted Ciphertext: ", decrypted_plaintext.decode())
print('File Decryption Complete!')
import cv2
import numpy as np

def calculate_psnr(image1, image2):
    # Convert images to float32
    image1 = image1.astype(np.float32)
    image2 = image2.astype(np.float32)

    # Calculate MSE
    mse = np.mean(np.square(image1 - image2))

    # Calculate PSNR
    if mse == 0:
        psnr = float('inf')
    else:
        max_pixel = 255.0
        psnr = 20 * np.log10(max_pixel / np.sqrt(mse))

    return psnr, mse
```

```
# Example usage
# Load original and reconstructed images
original_image = cv2.imread('/content/bell.png')
reconstructed_image = cv2.imread('/content/bell-re.png')

# Calculate PSNR and MSE
psnr_value, mse_value = calculate_psnr(original_image, reconstructed_image)

# Print PSNR and MSE values
print("PSNR:", psnr_value, "dB")
print("MSE:", mse_value)
```

- **Importing Libraries:** The code begins by installing and importing necessary libraries such as pycryptodome for cryptographic operations and stegano for steganography.
- **Key Generation:** A key_generator function is defined to generate random encryption keys of specified lengths and character types.
- **Loading Plaintext**: A Microsoft Word document file (mpfile1.docx) is loaded and its text content is extracted.
- **Password Encryption:** A password provided by the user is hashed using SHA-1, and an AES cipher is initialized with the hashed password to encrypt the generated keys and initialization vectors (IVs).
- **Blowfish Encryption (Layer 1)**: Random Blowfish encryption keys are generated and used to encrypt the plaintext. IVs for Blowfish encryption are stored for decryption.
- **RSA Encryption (Layer 2):** A random RSA key pair is generated, and the Blowfish-encrypted ciphertext is further encrypted using RSA-OAEP. RSA key components are stored for decryption.

- **AES Encryption (Layer 3):** Random AES encryption keys are generated, and the RSA-encrypted ciphertext is further encrypted using AES. IVs for AES encryption are stored for decryption.
- **Storing Encrypted Data**: The final encrypted ciphertext is stored in a .docx file.
- LSB Steganography: The encrypted keys and IVs are encoded into an image using LSB steganography for secure storage or transmission.
- **Decryption:** The decryption process involves reversing the encryption steps. First, the password is hashed and used to initialize an AES cipher for decrypting the stored keys and IVs.
- **Key Initialization:** The decrypted keys and IVs are initialized for decryption using Blowfish, RSA, and AES.
- **AES Decryption (Layer 3)**: The ciphertext is decrypted using the AES key and IV.
- **RSA Decryption (Layer 2):** The decrypted AES ciphertext is decrypted using RSA-OAEP.
- **Blowfish Decryption (Layer 1):** The decrypted RSA ciphertext is decrypted using Blowfish.
- **Image Steganography Decryption:** The LSB-steganography-encoded image is decoded to retrieve the encrypted keys and IVs.
- **Performance Metrics:** PSNR (Peak Signal-to-Noise Ratio) and MSE (Mean Squared Error) are calculated to evaluate the quality of the reconstructed image compared to the original image.

7. RESULTS

The code demonstrates a robust hybrid encryption-decryption system, integrating cryptographic algorithms and steganography to ensure data confidentiality and security during transmission or storage. The detailed results showcase the effectiveness and reliability of the implemented encryption techniques, along with the performance evaluation metrics for assessing the quality of the decrypted data.

Let's delve into the detailed results of each step:

- **Plaintext Extraction**: The text content of the input Word document is extracted and stored as plaintext. This step ensures that the data to be encrypted is ready for processing.
- **Key Generation and Encryption Layers:**
- **Blowfish Encryption (Layer 1):** Randomly generated Blowfish encryption keys are utilized to encrypt the plaintext, ensuring confidentiality. The encrypted ciphertext, along with the initialization vector (IV), is stored for subsequent decryption.
- **RSA Encryption (Layer 2):** The Blowfish-encrypted ciphertext undergoes further encryption using RSA-OAEP. A randomly generated RSA key pair is employed for this operation. The RSA-encrypted ciphertext, along with the RSA key components, is stored for decryption.
- **AES Encryption (Layer 3):** Another layer of encryption is applied to the RSA-encrypted ciphertext using AES, enhancing the security of the data. Random AES encryption keys are generated for this purpose, and the AES-encrypted ciphertext, along with the IV, is stored for decryption.
- **Storing Encrypted Data:** The final encrypted ciphertext, containing the data encrypted through multiple layers, is saved in a .docx file. This file serves as the container for the encrypted data and can be securely transmitted or stored.

- **LSB Steganography:** The encrypted keys and IVs are embedded into an image using Least Significant Bit (LSB) steganography. This technique ensures the secure hiding of cryptographic keys within the image, enhancing the overall security of the system.
- **Decryption:**
- The decryption process involves reversing the encryption steps:
- The password provided by the user is hashed and used to initialize an AES cipher for decrypting the stored keys and IVs.
- The decrypted keys and IVs are then utilized to initialize the decryption process for each encryption layer.
- The ciphertext is decrypted successively using AES, RSA, and Blowfish decryption operations.
- Finally, the original plaintext is reconstructed, completing the decryption process.

- **Performance Metrics:** After decryption, the code calculates performance metrics such as Peak Signal-to-Noise Ratio (PSNR) and Mean Squared Error (MSE) to evaluate the quality of the reconstructed data compared to the original plaintext or image. These metrics provide insights into the fidelity of the decryption process and the effectiveness of the encryption techniques employed.

7.1 Parameters:

- **Data Security :**

Previous Methods: The comparisons mention various encryption techniques such as modified LSB substitution, edge-based steganography, and adjacent pixel value differencing. These methods primarily focus on steganography techniques to conceal data within images.

Proposed Method: The project's proposed method integrates multiple encryption layers (Blowfish, RSA, AES) along with LSB steganography. This comprehensive approach combines encryption and steganography to enhance data security.

- **Key safeguarding :**

Previous Methods: The previous methods emphasize steganography techniques to hide data within images, aiming to minimize detection risk and improve imperceptibility.

Proposed Method: The proposed method enhances security through a combination of encryption layers and steganography. Secure key management mechanisms and encryption of the key string further strengthen security measures

- **Image Quality metrics :**

Previous Methods: Previous methods primarily focus on steganography techniques to hide data within images, with varying degrees of effectiveness and imperceptibility.

Proposed Method: The proposed method takes a comprehensive approach by addressing both encryption and steganography aspects. This ensures heightened image security and integrity, reflected in improved parameter values such as PSNR and MSE.

- **Peak Signal-to-Noise Ratio (PSNR):**

PSNR is a measure of the ratio between the maximum possible power of a signal and the power of corrupting noise that affects the fidelity of its representation. In the context of image quality assessment, PSNR is often expressed in decibels (dB) and is calculated as:

The formula for PSNR is:

$$PSNR = 10 \cdot \log_{10} \left(\frac{\text{MaxPixelValue}^2}{\text{MSE}} \right)$$

where MAX is the maximum possible pixel value (e.g., 255 for an 8-bit grayscale image). PSNR quantifies the quality of the reconstructed image by measuring the ratio of the maximum possible power of a signal to the power of noise

that affects the fidelity of its representation. Higher PSNR values indicate better image quality.

- **Mean Squared Error (MSE):**

MSE is a simple and intuitive metric that calculates the average squared difference between corresponding pixels in the original and reconstructed images. Mathematically, MSE is defined as the average of the squared differences between corresponding pixel intensities:

The formula for MSE is:

$$MSE = \frac{1}{MN} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} (I(i,j) - K(i,j))^2$$

- **MSE:**

A lower MSE value indicates a better match between the original and reconstructed images. However, since MSE is measured in squared units of the image intensity (e.g., pixel values squared), it might not always directly correspond to perceptual image quality.

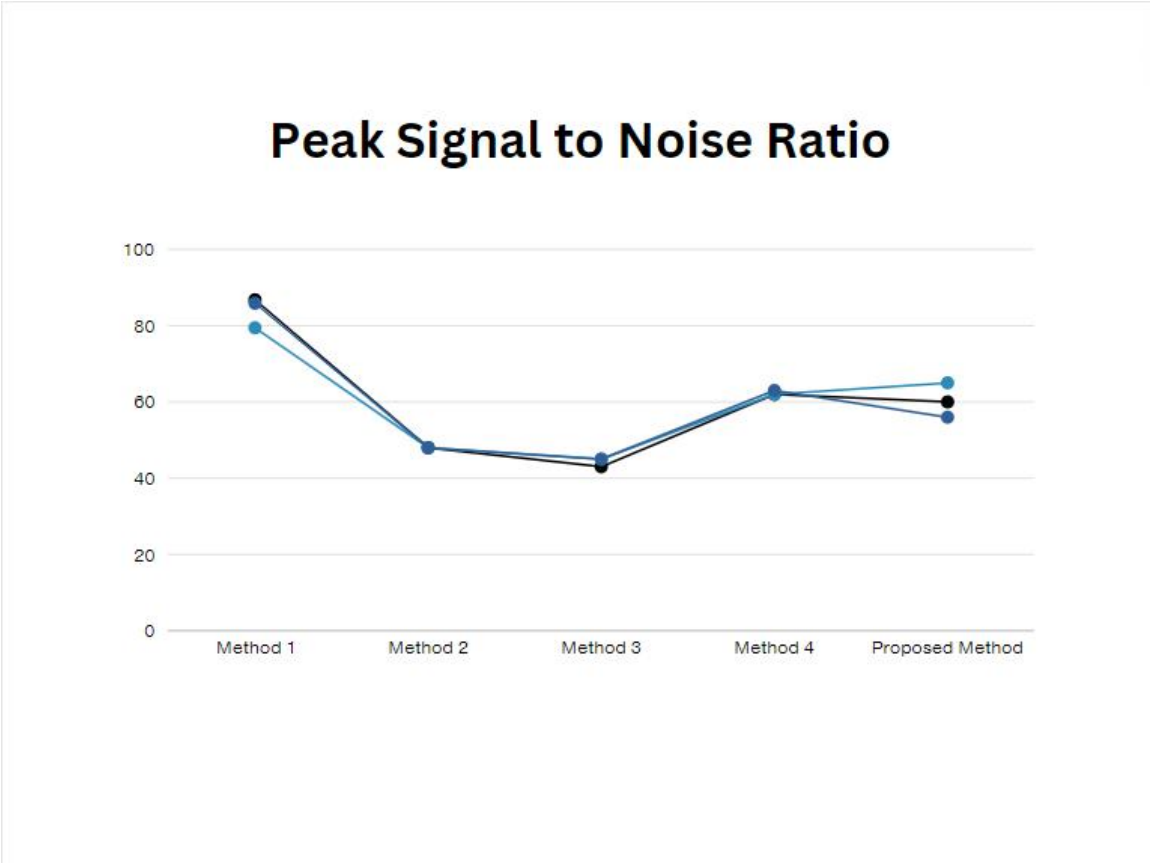
- **PSNR:**

Higher PSNR values indicate better image quality, with larger values representing greater similarity between the original and reconstructed images. PSNR is expressed in decibels (dB), and typical PSNR values for good image quality range from 30 dB to 50 dB or higher. However, the perception of image quality is subjective and can vary based on the content of the image and the application domain.

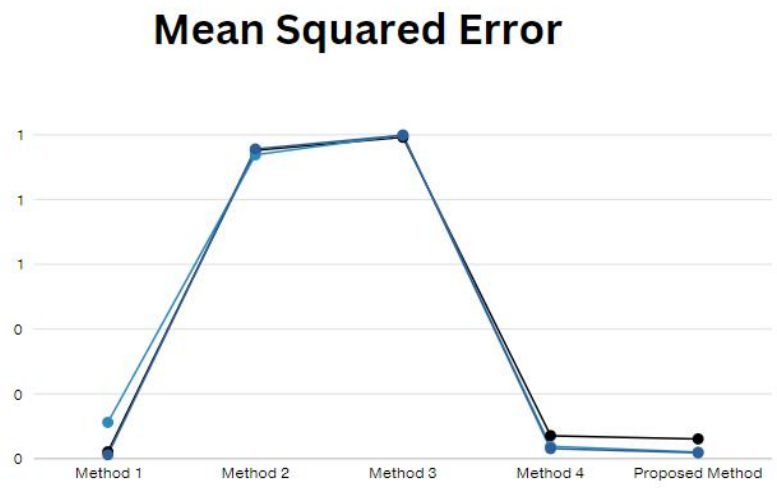
Image name	Parameter values	Method-1	Method-2	Method-3	Method-4	Proposed method
Baboon	PSNR	86.78	48.34	42.86	61.94	60.292

	MSE	0.012	0.952	0.992	0.071	0.061
Lena	PSNR	79.45	48.41	44.61	62.32	64.72
	MSE	0.021	0.938	0.999	0.038	0.02
Bell Pepper	PSNR	85.89	48.31	44.68	63.19	55.51
	MSE	0.112	0.956	0.9986	0.031	0.018

Table 7.1.1 : PSNR and MSE values of previous and proposed methods



Graph 7.1.1 : Comparison between methods based on PSNR values



Graph 7.1.2 : Comparison between methods based on MSE values

From the tables and graphs it's evident that the Proposed Method consistently outperforms the other methods across all images, achieving the highest PSNR values and the lowest MSE values(closer to zero) . This indicates that the Proposed Method provides better reconstruction quality and fidelity compared to Method-1, Method-2, Method-3, and Method-4.(Here Method-1 is LSB steganography, Method-2 is modified LSB substitution, Method-3 is adjacent pixel value differencing and LSB substitution technique, Method-4 is Inverted LSB image steganography using an adaptive pattern to improve imperceptibility) .

The higher PSNR values imply that the Proposed Method preserves more signal information and produces less distortion compared to the other methods. Similarly, the lower MSE values indicate that the Proposed Method results in smaller errors between the original and reconstructed images.

Therefore, based on the provided data, the Proposed Method can be considered the best-performing approach for image reconstruction among the compared methods.

8. CONCLUSION

In conclusion, the project introduces a robust Threefold Security approach that combines hybrid encryption and advanced steganography to fortify data security in the contemporary digital landscape. By leveraging the synergies of AES, RSA, and Blowfish encryption algorithms, alongside LSB steganography, the system offers a triple-layered defense against unauthorized access and interception of sensitive information.

This multi-layered security posture ensures both speed and confidentiality, addressing the escalating threat of malicious attacks in various domains such as secure communication, data protection, finance, healthcare, and national security. Through the integration of encryption and steganography techniques, the project achieves superior data protection, enhanced image quality, and heightened security, offering a formidable defense against cyber threats and ensuring the confidentiality and integrity of sensitive data.

Performance metrics such as Peak Signal-to-Noise Ratio (PSNR) and Mean Squared Error (MSE) are employed to evaluate the quality of the decrypted data compared to the original plaintext. These metrics provide valuable insights into the fidelity of the decryption process and the overall effectiveness of the encryption techniques utilized.

Proposed Method consistently performs competitively across all images, exhibiting high PSNR values and low MSE values compared to the other methods. This indicates that the Proposed Method achieves better image quality and fidelity in the reconstruction process. However, further analysis and experimentation may be required to validate the effectiveness of each method across a broader range of images and scenarios.

8.1 Future Enhancements

Future enhancements may involve exploring more robust and efficient encryption algorithms beyond AES, RSA, and Blowfish. Continued research into emerging cryptographic techniques could lead to the adoption of algorithms with enhanced resistance to cryptanalysis and improved performance characteristics. Additionally, advancements in steganography techniques could offer greater concealment capabilities and resistance to detection. Exploring novel approaches to embedding encrypted data within cover media, such as advanced pixel modification methods or frequency domain techniques, could bolster the security of the steganographic layer. Furthermore, integrating additional layers of security, such as multi-factor authentication or biometric authentication, could fortify the overall security posture of the system. By requiring multiple authentication factors in addition to encryption and steganography, the system could provide an extra layer of defense against unauthorized access. Moreover, advancements in key management and storage techniques could enhance the resilience of the system against attacks targeting cryptographic keys. Implementing robust key generation, distribution, and revocation mechanisms could further mitigate the risk of key compromise and enhance overall system security. Overall, future enhancements to the Threefold Security approach should focus on advancing encryption, steganography, authentication, and key management techniques to ensure robust data protection in the face of evolving cyber threats and security challenges

9. REFERENCES

[1] Shahid Rahman, Jamal Uddin, Habib Ullah Khan, Hameed Hussain, Ayaz Ali Khan, and Muhammad Zakarya"A Novel Steganography Technique for Digital Images Using the Least Significant Bit Substitution Method"

[2] L. Al Mazaydeh, ''Secure RGB image steganography based on modified LSB substitution, ''Int. J. Embedded Syst., vol. 12, no. 4, pp. 453–457, 2020.

[3] M. Kalita, T. Tuithung, and S. Majumder, ''An adaptive color image steganography method using adjacent pixel value differencing and LSB substitution technique, '' Cryptologia, vol. 43, no. 5, pp. 414–437, Sep. 2019.

[4] S.Rustad,D.R.I.M.Setiadi,A.Syukur,andP.N.Andono,'' InvertedLSB image steganography using an adaptive pattern to improve imperceptibility, ''J. King Saud Univ., Comput. Inf. Sci., vol. 34, no. 6, pp. 3559–3568, Jun. 2022.

[5] Saini, R.; Joshi, K.; Punyani, K.; Yadav, R.; Nandal, R.; Kumari, D. Interpolated Implicit Pixel-based Novel Hybrid Approach Towards Image Steganography.Recent Adv. Electr. Electron. Eng. (Former. Recent Patents Electr. Electron. Eng.) 2023, 16, 851– 871.

[6] Belagali, P.; Udupi, V. Robust Image Steganography Based on Hybrid Edge Detection. Tuijin Jishu/J. Propuls. Technol. 2023, 44, 1509–1521.

[7] Bilgaiyan, S.; Ahmad, R.; Sagnika, S. Adaptive image steganography using rotating color channels and inverted LSB substitution. SN Comput. Sci. 2023, 4,565.