

SECURITY FRAME WORK FOR IOT HEALTH DATA

PROJECT REPORT

*Submitted in the fulfillment of the requirements for
the award of the degree of*

Bachelor of Technology

in

Electronics and Communication Engineering

BOLLU RAHUL

SURYADEVARA SAI LATHA

KAMBHAMPATI DHANUSH

[211LA05009]

[211LA05010]

[211LA05016]

Under the guidance of

Under the co-guidance of

Dr.M.S.S Rukmini

Ms.G.Ramya Sri

Professor

Assistant Professor

Department of ECE

Department of ECE



VIGNAN'S

Foundation for Science, Technology & Research

(Deemed to be University)

-Estd. u/s 3 of UGC Act 1956

(ACCREDITED BY NAAC WITH "A+" GRADE)

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

(ACCREDITED BY NBA)

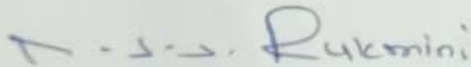
VIGNAN'S FOUNDATION FOR SCIENCE, TECHNOLOGY AND RESEARCH

(Deemed to be University)

Vadlamudi, Guntur, Andhra Pradesh, India -522213

CERTIFICATE


This is to certify that the project report entitled "SECURITY FRAME WORK FOR IOT HEALTH DATA" that is being submitted by BOLLU RAHUL[211LA05009], SURYADEVARA SAI LATHA[211LA05010], KAMBHAMPATI DHANUSH [211LA05016] in partial fulfillment for the award of B. Tech degree in Electronics and Communication Engineering to Vignan's Foundation for Science Technology and Research University, is a record of bonafide work carried out by them under the supervision of Dr.M.S.S.Rukmini and Ms.G.Ramya Sri of ECE Department.



Signature of the faculty guide

Dr.M.S.S.Rukmini

Professor



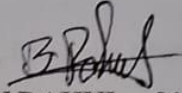
Dr. T. Pitchaiah, M.E, Ph.D, MIEEE, FIETE

Professor & HoD, ECE

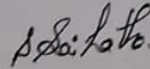
DECLARATION

I hereby declare that the project work entitled "SECURITY FRAME WORK FOR IOT HEALTH DATA" is being submitted to Vignan's Foundation for Science, Technology and Research (Deemed to be University) in partial fulfillment for the award of B. Tech degree in Electronics and Communication Engineering. The work was originally designed and executed by us under the guidance of our supervisor Dr.M.S.S.Rukmini and Ms.G.Ramya Sri as faculty guide at Department of Electronics and Communication Engineering, Vignan's Foundation for Science Technology and Research (Deemed to be University) and was not a duplication of work done by someone else. We hold the responsibility of the originality of the work incorporated into this thesis.

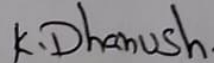
Signature of the candidates



BOLLU RAHUL - 211LA05009



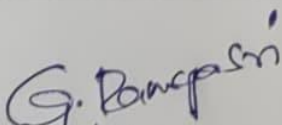
SURYADEVARA SAI LATHA - 211LA05010

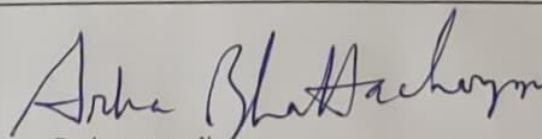


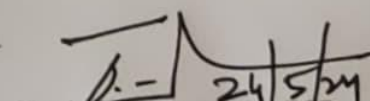
KAMBHAMPATI DHANUSH - 211LA05016

Major Design (Final Year Project Work) Experience Information

| | | | |
|---|---|--|--|
| Student group | BOLLURAHUL (211LA05009) | SURYADEVARA SAI LATHA (211LA05010) | KAMBHAMPATI DHANUSH (211LA05016) |
| Project Title | Security Framework for IoT Health Data | | |
| Program | Development of framework for IoT Health Data | | |
| Concentration Area | | | |
| Constraints - Examples | | | |
| Economic | Yes | | |
| Environmental | Yes | | |
| Sustainability | Yes | | |
| Implementable | Yes | | |
| Ethical | Followed the standard professional ethics | | |
| Health and Safety | Yes | | |
| Social | Applicable for defense also | | |
| Political | NA | | |
| Other | Can be used in Data verification and authorization of data | | |
| Standards | | | |
| 1. IEEE 802.15.6 | -Wireless Body Area Network | | |
| 2. IEEE 2413 | Standard for an Architecture Framework for Internet of Things | | |
| Prerequisite courses for the Major Design Experience | 1. Internet Of Things | | |
| | 2. Cryptography and Network Security | | |


Supervisor


Project co-ordinator


Head of the Department ECE

V ACKNOWLEDGEMENT

The satisfaction that comes from successfully completing any task would be incomplete without acknowledging the people who made it possible, whose ongoing guidance and encouragement have been essential to the achievement.

We are greatly indebted to **Dr.M.S.S.Rukmini** Professor and **Ms.G.Ramya Sri**, Asst.professor my revered guide and Associate Professor in the Department of Electronics and Communication Engineering, VFSTR (Deemed to be University), Vadlamudi, Guntur, for her valuable guidance in the preparation of this project report. He has been a source of great inspiration and encouragement to us. She has been kind enough to devote considerable amount of her valuable time in guiding us at every stage. This is our debut, but we are sure that we are able to do many more such studies, under the lasting inspiration and guidance given by respectable guide.

We would also like to thank to **Dr. T. Pitchaiah**, Head of the Department, ECE for his valuable suggestion.

We would like to specially thank, **Dr. N. Usha Rani**, Dean, School of Electrical, Electronics and Communication Engineering for her help and support during the project work.

We thank our project coordinators **Dr. Satyajee Sahoo**, **Dr. Arka Bhattacharyya**, **Mr. Abhishek Kumar** and **Mr. M. Vamsi Krishna** for continuous support and suggestions in scheduling project reviews and verification of the report. Also, thank to supporting staff of ECE Department for their technical support for timely completion of project.

We would like to express our gratitude to **Dr. P. Nagabhusan**, Vice-Chancellor, VFSTR (Deemed to be University) for providing us the greatest opportunity to have a great exposure and to carry out the project.

Finally, we would like to thank our parents and friends for the moral support throughout the project work.

We wish to express our heart full thanks to our parents for their support and encouragement throughout our life.

Name of the candidates

BOLLU RAHUL – 211LA05009

SURYADEVARA SAI LATHA – 211LA05010

KAMBHAMPATI DHANUSH – 211LA05016

TABLE OF CONTENT

| CHAPTER NO. | NAME OF THE CONTENT | PAGE NO. |
|--------------------|--|-----------------|
| | ABSTRACT | |
| | LIST OF FIGURES | |
| | LIST OF ABBREVIATIONS | |
| 1 | INTRODUCTION | 1 |
| 2 | LITERATURE SURVEY 2.1 LITERATURE STUDY 2.2 EXISTING SYSTEM 2.3 LIMITATIONS OF EXISTING METHODS | 3-10 |
| 3 | PROPOSED SYSTEM 3.1 STEGANOGRAPHY 3.2 LSB IMAGE STEGANOGRAPHY 3.3 ADVANCED ENCRYPTION STANDARD 3.4 ELLIPTICAL CURVE CRYPTOGRAPHY 3.5 ELLIPTICAL CURVE DIFFIE- HELLMAN | 11-15 |
| 4 | SYSTEM DESING 4.1 BLOCK DIAGRAM 4.2 ENCODING ALGORITHM 4.3 DECODING ALGORITHM 4.4 ELLIPTICAL CURVE DIFFIE- HELLMAN 4.5 SYSTEM IMPLEMENTATION 4.6 ELLIPTICAL CURVE CRYPTOGRAPHY | 16-30 |

| | | |
|----|---|-------|
| 5 | QUALITY EVALUATION METRICS 5.1 MEAN SQUARE ERROR 5.2 PEAK SIGNAL TO NOISE RATIO | 31-33 |
| 6 | RESULTS & DISCUSSION | 34-36 |
| 7 | CONCLUSION | 37-38 |
| 8 | FUTURE SCOPE | 39-40 |
| 9 | REFERENCES | 41-42 |
| 10 | APPENDIX | 43-51 |

TABLE OF FIGURES

| Fig No | Figure Name | Page No |
|---------------|---------------------|----------------|
| 4.1 | Block Diagram | 16 |
| 4.4.1 | SECP256K1 | 20 |
| 6.1 | ECDH KEY EXCHANGE | 35 |
| 6.2 | IMAGE STEGANOGRAPHY | 35 |
| 6.3 | Input image | 36 |
| 6.4 | Output image | 36 |

TABLE OF ABBREVIATIONS

| SL NO: | ABBREVIATIONS | DESCRIPTION |
|--------|---------------|----------------------------------|
| 1 | AES | Advanced Encryption Standard |
| 2 | ECC | Elliptical Curve Cryptography |
| 3 | EDCH | Elliptical Curve Diffie- Hellman |
| 4 | JPG | Joint Photographic Experts Group |
| 5 | LSB | Least Significant Bit |
| 6 | MSE | Mean Square Error |
| 7 | PDF | Portable Document Format |
| 8 | PNG | Portable Network Graphic |
| 9 | PSNR | Peak Signal To Noise Ratio |
| 10 | RGB | Red Green Blue |

ABSTRACT

Internet of Things (IoT) connects everyday objects to the internet, allowing them to collect and share data. This data can be used to automate tasks, improve efficiency and gain new insights across various fields. In short, IoT makes our world smarter and more interconnected. The Steganography is the technique of hiding secret messages within seemingly harmless covers to elude examination by censors, particularly in IoT environments where data security is paramount. Although proposed several decades ago, provably secure steganography has not gained popularity among researchers due to its rigorous data requirements. Recent advancements in generative models have enabled researchers to provide explicit data distributions, which has contributed to the development of provably secure steganography methods. However, these methods depend on the assumption of a preshared key, facing challenges such as key agreement, key updating, and user expansion in practical settings. Although public-key steganography offers a viable solution, existing approaches are burdened with inefficiency and complex implementation. In this paper, we propose a practical public-key steganography method that integrates Elliptic Curve Cryptography (ECC) with a generative model, incorporating Elliptic Curve Diffie-Hellman (ECDH) for secure key exchange. This method is the first comprehensive and practical approach to public-key steganography and steganographic key exchange, particularly suitable for IoT applications. Additionally, we provide a specific instance to illustrate the proposed method. The security of the proposed construction is also proven based on computational complexity theory. Further experiments have demonstrated the security and efficiency of the proposed method.

CHAPTER-1

INTRODUCTION

INTRODUCTION

The Internet of Things (IoT)[1] connects everyday objects through the internet, enabling data exchange and automation to enhance efficiency and decision-making. With applications in homes, healthcare, industry, and smart cities, IoT drives significant innovation and economic growth. However, addressing data security, privacy, and interoperability challenges is crucial to fully realize its benefits. Ultimately, IoT represents a paradigm shift in how we interact with our environment, marking a cornerstone of the future digital ecosystem.

Steganography is the practice of hiding information within non-secret data, making the message's existence undetectable. It can be used for secure communication, digital watermarking, and protecting intellectual property. However, it also poses risks, such as hiding malware, necessitating advanced detection techniques. Overall, steganography is a crucial tool in data security, combining creativity and technical precision Least Significant Bit (LSB) image steganography[6] is a method of hiding messages within digital images by altering the least significant bits of pixel values. This technique is simple, efficient, and visually imperceptible, making it useful for secure communication and watermarking. However, it is vulnerable to detection and image processing alterations, necessitating additional security measures to ensure message integrity AES-128 (Advanced Encryption Standard with a 128-bit key length) is a widely used symmetric encryption algorithm known for its strong security, efficiency, and standardization. It encrypts data in 128-bit blocks using a 128-bit key, making it highly secure against brute-force attacks and suitable for various applications, including secure communications and data protection. Its balance of security and performance has made it a cornerstone of modern cryptographic practices.

Elliptic Curve Cryptography (ECC) [2] is an advanced public key cryptography method that offers high security with significantly smaller key sizes compared to traditional systems like RSA. Its efficiency and reduced resource consumption make it ideal for secure communications, digital signatures, and encryption in resource-constrained environments like mobile devices and IoT. ECC's robust security and scalability make it a preferred choice for modern cryptographic applications Elliptic Curve Diffie-Hellman (ECDH) is a key agreement protocol leveraging elliptic curve cryptography for secure key exchange over insecure channels[5]. Its use of smaller key sizes and mathematical properties of elliptic curves ensures high security and efficiency. ECDH is widely adopted in protocols like SSL/TLS, offering flexibility and robust security for various applications, particularly in resource-constrained environments.

CHAPTER-2
LITERATURE SURVEY

LITERATURE SURVEY

2.1 LITERATURE STUDY:

INTERNET OF THINGS: APPLICATIONS, CHALLENGES AND RESEARCH ISSUES. [1]

Description: The paper discusses the Internet of Things (IoT) and its growing impact. IoT connects everyday objects to the internet, allowing them to collect and share data. This technology has the potential to revolutionize many aspects of life, from healthcare and agriculture to transportation and city management. It explores various communication protocols used in IoT devices, including Sigfox, Cellular, 6LoWPAN, Zigbee, and RFID. Each protocol has its advantages and disadvantages in terms of range, data rate, and security. It also highlights some of the key applications of IoT technology, including smart cities, smart agriculture, retail and logistics, healthcare, and security. IoT devices can be used to monitor and improve a wide range of processes and services. It acknowledges that there are also challenges associated with IoT. These challenges include standardization, architecture, scalability, and security. As the number of IoT devices continues to grow, it will be important to develop new standards, architectures, and security measures to ensure that this technology can be implemented effectively.

Revisiting of Elliptical Curve Cryptography for Securing Internet of Things (IOT) [2]

Description:

This paper proposes a secure communication protocol (ECIOT) for the Internet of Things (IoT) based on elliptic curve cryptography (ECC). The main concern is security for billions of resource-constrained IoT devices communicating over the internet. ECIOT uses Diffie-Hellman key exchange with an elliptic curve to establish a shared secret key between an IoT device and a server. This avoids transmitting the key over an insecure channel. The implementation uses the MIRACL library and the NIST p-192 elliptic curve for faster computations. The protocol reduces bandwidth usage by compressing the key coordinates and avoids complex modular inversion operations to save battery power on IoT devices.

Exploring LSB Steganography Possibilities in RGB Images [3]

Description: This paper introduces a new image steganography technique that hides secret messages within colored images. It combines two established methods: LSB (Least Significant Bit) steganography for embedding data and AES-128 encryption for securing the information. The unique aspect lies in how the data is hidden. Instead of a fixed order, the algorithm determines the sequence for embedding information in the red, green, and blue channels of the image based on the image's resolution and file type. This dynamic approach makes it more challenging for attackers to crack the hidden message. Tests on various images showed that the new method offers similar image quality compared to traditional LSB steganography with encryption. Overall, this technique enhances security in image steganography by adding a layer of randomization to the data embedding process.

Application of Cryptography Based on Elliptic Curves [4]

Description: This article explores using elliptic curve cryptography (ECC) to secure data transmission in wireless networks. ECC offers advantages like smaller key sizes for similar security compared to other cryptography methods. The paper discusses two ECC algorithms: ECDH for key exchange and ECDSA for digital signatures. The authors implemented a test bed using two computers to demonstrate secure communication with ECC. They used the Open SSL library and generated keys on each computer. Public keys were exchanged, and a shared secret was created using the ECDH algorithm. For data transmission, they used symmetric AES encryption with a shared secret key derived from the elliptic curve keys. Additionally, they signed the data with a digital signature using the ECDSA algorithm and the sender's private key. The receiver verified the signature using the sender's public key to ensure authenticity. The experiment successfully demonstrated secure data transmission using a combination of symmetric encryption and asymmetric encryption with ECC. This method is considered suitable for wireless networks and is expected to be increasingly used in various applications, including autonomous vehicles and Internet of Things (IoT) devices.

Secure Key Exchange for Protecting Health Data Diffie-Hellman Based Approach[5]

Description: This research proposes a method to secure health data transmission using Diffie-Hellman key exchange. The method was implemented in a Node.js application and evaluated using HL7 health data files. The Diffie-Hellman protocol facilitates a secure key exchange between the sender and receiver. The shared secret key is then used to encrypt the HL7 data for secure transmission. The evaluation focused on key exchange time, encryption time, and decryption time. The results indicate that the Diffie-Hellman protocol is effective for securing health data transmission with acceptable processing times. The study also compared the performance of Diffie-Hellman and Elliptic Curve Cryptography for key exchange. Diffie-Hellman was found to have faster key exchange times at a key length of 1024 bits. Overall, this research demonstrates a promising approach for securing health data transmission using Diffie-Hellman key exchange. Future work could explore the use of different health data sets and investigate the impact of varying key lengths and cryptographic algorithms on performance.

Image Steganography using Least Significant Bit (LSB) - A Systematic Literature Review [6]

Description: This article provides a comprehensive overview of research on Least Significant Bit (LSB) image steganography techniques published between 2016 and 2020. It utilizes a Systematic Literature Review (SLR) methodology to answer key questions about LSB steganography. Four main parameters are used to assess the quality of steganography:

- **PSNR (Peak Signal-to-Noise Ratio):** Measures the similarity between the original and stego image (higher PSNR indicates better quality).
- **MSE (Mean Squared Error):** Represents the cumulative error between the original and stego image (lower MSE indicates less error).
- **CPU Consumption:** Evaluates the computational resources required for encoding/decoding secret data.
- **Embedding Capacity:** Measures the maximum amount of secret data that can be hidden in the cover image

2.2 EXISTING SYSTEMS:

Basic LSB (Least Significant Bit) Steganography is one of the simplest methods used in image steganography. It involves embedding secret data into the least significant bits of the pixels in an image. The main advantage of this method is its simplicity and minimal distortion to the image, making it easy to implement and extract the hidden data. However, it has significant limitations, such as low security since it is highly susceptible to statistical and visual attacks. Additionally, the capacity for hidden data is limited; embedding too much data can result in noticeable changes to the image. The predictability of the pattern used for embedding also makes it easier to detect and compromise.

LSB with Pseudorandom Key enhances basic LSB steganography by using a pseudorandom number generator to determine which pixels to modify, adding a layer of security. This method is slightly more secure than basic LSB and reduces predictability. However, it is still vulnerable to statistical analysis, especially if the pseudorandom sequence is discovered. The system's security heavily depends on the strength and secrecy of the key; a weak or compromised key can lead to the system being easily defeated.

Edge-Based LSB Steganography modifies pixels in the edge regions of an image, which are less perceptible to human vision. This method offers better imperceptibility compared to basic LSB and can achieve higher capacity in complex images. However, it is more complex to implement as it requires edge detection algorithms. The computational intensity of this method can be a drawback, and its effectiveness can vary significantly based on the content and characteristics of the image.

Transform Domain Techniques such as Discrete Cosine Transform (DCT) and Discrete Wavelet Transform (DWT) embed data into the frequency domain rather than the spatial domain. These methods are more robust against common image processing operations like compression and scaling. Despite their robustness, transform domain techniques are complex to implement and require a deep understanding of the transformations involved. Additionally, improper implementation can still lead to noticeable artifacts in the image, which might reveal the presence of hidden data.

Adaptive Steganography adapts the embedding process based on the local characteristics of the image, typically aiming to minimize perceptibility. This method achieves high imperceptibility and often provides a good balance between capacity and security. However, it involves high computational complexity and is harder to standardize. Moreover, advanced statistical methods can still detect anomalies introduced by this technique, posing a risk to the hidden data.

2.3 Limitations of Existing Methods

Security is a major limitation across many steganographic methods. Basic techniques lack robustness and are easily detected through statistical analysis, visual attacks, and steganalysis tools. Key management is another concern; methods relying on weak pseudorandom keys or simple encryption can be compromised easily, rendering the steganographic method ineffective.

Capacity is another significant limitation. Many simple methods, particularly basic LSB techniques, allow only a limited amount of data to be embedded without causing visible distortion. Furthermore, the capacity often depends heavily on the nature of the image, limiting the general applicability of the method.

Complexity and Computational Load also pose challenges. Advanced methods such as transform domain techniques and adaptive steganography are complex to implement and require significant computational resources. This complexity can be a barrier to real-time applications, where speed and efficiency are critical.

Imperceptibility is crucial for effective steganography, yet some methods can introduce visible artifacts or alter the image quality, raising suspicion. Even subtle changes can be detected by advanced steganalysis methods, compromising the hidden data. This makes it essential to carefully balance the need for data hiding with the preservation of image quality.

CHAPTER-3

PROPOSED SYSTEM

PROPOSED SYSTEM

3.1 Steganography

Steganography is a technique for hiding information within images in a way that's undetectable to the human eye. It's like hiding a secret message within a picture, but instead of text, it can be used to hide various kinds of data like:

- Text messages
- Audio
- Videos
- Images

Text Steganography:

Text steganography involves hiding information within text files. This can be done in various ways, such as:

- **Format-based Methods:** Modifying the formatting of text, such as using different fonts, spacing, or invisible characters.
- **Random and Statistical Generation:** Generating text based on a statistical distribution or using random text patterns.
- **Linguistic Methods:** Embedding hidden messages using syntactic or semantic manipulation of the text.

Audio Steganography:

Audio steganography hides information within audio files. Common methods include:

- **LSB Coding:** Similar to image steganography, altering the least significant bits of audio samples.
- **Echo Hiding:** Embedding data by introducing an echo into the original signal with a short delay.
- **Phase Coding:** Altering the phase of an audio signal to encode information.

Video Steganography:

Video steganography involves embedding data within video files. Techniques include:

- **LSB Modification:** Modifying the least significant bits of video frames.
- **Motion Vector Embedding:** Hiding data within the motion vectors used for video compression.
- **Transform Domain Techniques:** Embedding information in the frequency domain of the video signal, similar to image steganography...

Image Steganography:

Image steganography is one of the most popular types and involves hiding data within digital images. Techniques include:

- **Least Significant Bit (LSB) Insertion:** Modifying the least significant bits of image pixels to encode data. It is a simple and widely used method.
- **Masking and Filtering:** Using digital watermarks or masks to embed information in an image.
- **Transform Domain Techniques:** Embedding data in the transform space of an image, such as using Discrete Cosine Transform (DCT) or Discrete Wavelet Transform (DWT).

Applications of image steganography:

- **Copyright protection:** Embedding a watermark in an image can help identify the copyright owner.
- **Covert communication:** Steganography can be used to hide messages in plain sight for secure communication.
- **Data authentication:** Steganography can be used to embed verification data within an image to ensure its authenticity.

Limitations of image steganography:

- **Capacity limitations:** The amount of data that can be hidden in an image is limited by the image size and quality.
- **Detectability:** Sophisticated steganalysis techniques can potentially detect the presence of hidden data.
- **Fragility:** Stego images can be corrupted if edited or compressed, making data recovery difficult.

3.2 LSB IMAGE STEGANOGRAPHY:

LSB imaging, which stands for Least Significant Bit imaging, is a technique used for steganography. Steganography is the practice of hiding information within another file, in this case, an image. The goal of steganography is to conceal the existence of the hidden message, making it undetectable to the human eye.

Here's how LSB imaging works:

- **Digital Images:** Digital images are typically composed of pixels. Each pixel represents a tiny square on the image and contains data that determines its color and brightness. This data is usually stored in a binary format (0s and 1s).

- **Hiding Information:** In LSB imaging, the least significant bit (LSB) of each pixel's data is replaced with a bit from the message being hidden. Since the LSB has the least impact on the overall color of the pixel, these changes are usually imperceptible to the human eye.

Advantages of LSB imaging:

- **Relatively easy to implement:** LSB is a straightforward technique that can be implemented with minimal technical expertise.
- **Large hiding capacity:** Images can hold a significant amount of hidden data depending on their size and color depth.

Disadvantages of LSB imaging:

- **Detectable with sophisticated techniques:** While invisible to the naked eye, statistical analysis or more advanced techniques can reveal the presence of hidden data.
- **Fragile to image manipulation:** LSB embedded data can be easily corrupted if the image is edited or compressed, as the manipulation can alter the LSBs.

Applications of LSB imaging:

- **Copyright protection:** LSB imaging can be used to embed copyright information within images.
- **Covert communication:** In some scenarios, LSB imaging might be used to hide secret messages within images for covert communication.

Security Considerations:

It's important to note that LSB imaging, on its own, is not a robust security solution. The hidden data can be vulnerable if someone suspects its presence and uses steganography detection techniques. For stronger security, LSB imaging can be combined with encryption to protect the confidentiality of the hidden message.

3.3 Advanced Encryption Standard (AES)

AES encryption for securing documents involves encrypting the contents of the document to protect it from unauthorized access. Here's a detailed guide, explained in paragraphs, on how AES-128 encryption can be used for document encryption, including considerations for its use in IoT and steganography.

- **Generate or Obtain a Key:** The first step in AES encryption is to have a 128-bit key. This key must be kept secret as it is the crux of the encryption's security. In an IoT context, keys might be pre-shared or exchanged securely between devices.

- **Prepare the Plaintext:** The document content (plaintext) must be prepared for encryption. This often involves converting the content to a byte array. AES operates on blocks of data, so the plaintext may need to be padded to ensure its length is a multiple of the block size (16 bytes for AES).
- **Initialization Vector (IV):** An Initialization Vector (IV) is used in certain modes of AES (e.g., CBC - Cipher Block Chaining) to ensure that the same plaintext does not result in the same ciphertext. The IV does not need to be secret, but it must be unique and random for each encryption operation.
- **Encrypt the Data:** Using the key and IV, the plaintext is encrypted to produce the ciphertext. This step transforms the document into an unreadable format. The IV is typically prepended to the ciphertext so it can be used during decryption.
- **Store or Transmit the Ciphertext:** The encrypted document (ciphertext) can then be stored or transmitted securely. Even if intercepted, without the key, the content remains protected.

3.4 Elliptic Curve Cryptography (ECC)

Elliptic Curve Cryptography (ECC) represents a modern approach to public-key cryptography, leveraging the mathematical properties of elliptic curves over finite fields. Unlike traditional methods such as RSA, ECC offers comparable security with significantly smaller key sizes, making it particularly suitable for resource-constrained environments like mobile devices and IoT. At its core, ECC relies on the manipulation of points on elliptic curves, where cryptographic operations are defined by algebraic equations within finite fields. The security of ECC is based on the difficulty of certain mathematical problems, such as the elliptic curve discrete logarithm problem (ECDLP), which underpins the security of key generation and cryptographic operations.

3.5 Elliptic Curve Diffie Hellman (ECDH)

The ECDH (Elliptic Curve Diffie–Hellman Key Exchange) is an anonymous key agreement scheme, which allows two parties, each having an elliptic-curve public–private key pair, to establish a shared secret over an insecure channel. ECDH is very similar to the classical DHKE (Diffie–Hellman Key Exchange) algorithm, but it uses ECC point multiplication instead of modular exponentiations. ECDH is based on the following property of EC points:

$$(a * G) * b = (b * G) * a$$

If we have two secret numbers a and b (two private keys, belonging to Alice and Bob) and an ECC elliptic curve with generator point G , we can exchange over an insecure channel the values $(a * G)$ and $(b * G)$ (the public keys of Alice and Bob) and then we can derive a shared secret: $\text{secret} = (a * G) * b = (b * G) * a$. Pretty simple. The above equation takes the following form:

$$\text{alicePubKey} * \text{bobPrivKey} = \text{bobPubKey} * \text{alicePrivKey} = \text{secret}$$

The ECDH algorithm (Elliptic Curve Diffie–Hellman Key Exchange) is trivial:

Alice generates a random ECC key pair: $\{\text{alicePrivKey}, \text{alicePubKey} = \text{alicePrivKey} * G\}$

Bob generates a random ECC key pair: $\{\text{bobPrivKey}, \text{bobPubKey} = \text{bobPrivKey} * G\}$

Alice and Bob exchange their public keys through the insecure channel (e.g. over Internet)

Alice calculates $\text{sharedKey} = \text{bobPubKey} * \text{alicePrivKey}$

Bob calculates $\text{sharedKey} = \text{alicePubKey} * \text{bobPrivKey}$

Now both Alice and Bob have the same $\text{sharedKey} == \text{bobPubKey} * \text{alicePrivKey} == \text{alicePubKey} * \text{bobPrivKey}$

CHAPTER-4

SYSTEM DESIGN

4. SYSTEM DESIGN

4.1 Block Diagram:

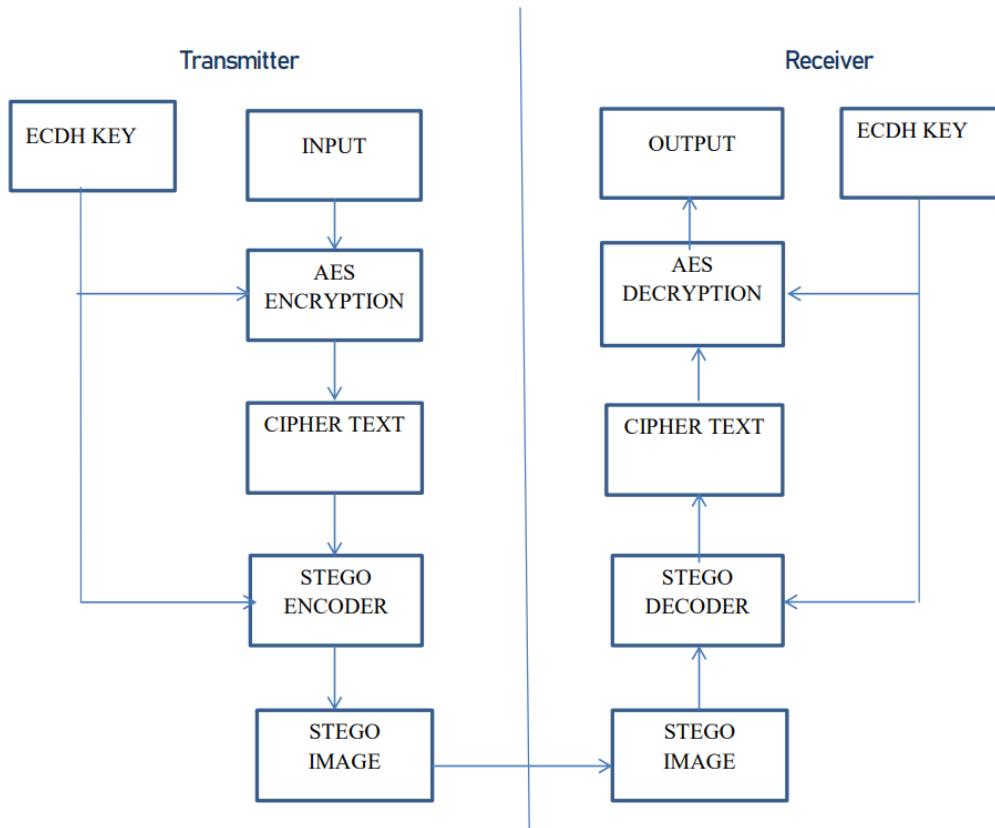


Fig:4.1 Block Diagram

Block Diagram of a system that uses image steganography and ECDH key exchange to secure IoT data. Here's a breakdown of the process:

1. **ECDH Key Exchange:** Elliptic curve Diffie–Hellman (ECDH) is a key agreement protocol that allows two parties to establish a shared secret key over an insecure communication channel. In the first step, two parties generate private keys and public keys. Then, they exchange their public keys to compute a shared secret key.
2. **AES Encryption:** The data to be hidden, which can be text or other data types is encrypted with the shared secret key using the Advanced Encryption Standard (AES) algorithm. This encrypts the data into ciphertext.
3. **Stego-Encoding:** The encrypted data (ciphertext) is hidden inside an image using steganography techniques. Steganography is the practice of hiding information within a digital carrier, such as an image, audio file, or video. In this case, the image is the carrier and the ciphertext is the information being hidden.
4. **Transmission:** The stego-image, which contains the hidden ciphertext, is transmitted over an insecure channel.
5. **Stego-Decoding:** At the receiving end, the stego-image is used to extract the hidden ciphertext.
6. **AES Decryption:** The shared secret key is used to decrypt the ciphertext back to the original data.

This system helps secure IoT data by hiding it inside an image and using a secure key exchange method to establish a shared secret key for encryption and decryption.

4.2 Encoding Algorithm for LSB Stegano

The encoding algorithm the process designed and implemented for LSB Stegano. LSB Stegano takes in four inputs from the user: 1) Data (plain strings or file(s) of any type), 2) Secret key for AES encryption and random number generation, 3) Source PNG image, and 4) Destination PNG image name. The secret key is used for three purposes: 1) The first 128 bits of the key are used for the initialization vector (IV) for AES, 2) The second 128 bits of the key are used as the secret key for AES (AES-128 is being used to encrypt the data), and 3) The key is used as the seed for the random number generator. Without the secret key, retrieving the hidden data is not possible. The receiving party must know the secret key to retrieve the hidden data inside the image. The random number generator takes in another input, the total number of encodable bits, which is calculated by multiplying the image's length by its width and subtracting one row (the length) from the product.

This step is necessary to create a random number (the hop number) of the correct size, where each pixel is represented by one bit of the random number (these random numbers are extremely large). For example, if the total number of encodable pixels is 100, a 100-bit random number would be generated using the secret key as a seed for the random number generator. The first row (row 0) is subtracted from the total number of pixels in the image to get the total number of encodable pixels because the first row is used for the header. The header comprises two values:

1. The first 10 pixels of the image contain the length of the ciphertext in binary encoded in the LSB of the RGB values, and
2. The zip file header (if the user specified a file or files as the data to hide). The length of the ciphertext is necessary because LSB Stegano needs to know where to stop the encoding and decoding process.

The zip header is also necessary because LSB Stegano creates a zip archive where all the user's files are stored, so only one type of header (zip) has to be processed instead of the hundreds of different types of file headers that exist. This makes LSB Stegano very robust because it can hide any type of file (.exe, .pdf, .jpg, .xlsx, .docx, .gif, etc.) and more than one file can be hidden in the image. Additionally, since a zip archive is being used, LSB Stegano works on any platform that supports Python 3 and zip archives, making it very versatile. Once LSB Stegano has created the header inside the image, the actual encoding of the ciphertext can begin.

The random number, now converted into binary, is the key to hiding the encrypted data randomly. Starting at the first pixel in row 1 (the image is represented as a 2-D array which is 0-bounded for both rows and columns) and working through the image, LSB Stegano encodes 3 bits of ciphertext in each pixel that maps to a 1 in the random number. If a pixel maps to a 0, that pixel is not encoded with any data and is initially skipped. If LSB Stegano reaches the end of the image and there is still ciphertext remaining to be encoded, it returns to the beginning of the image and starts encoding data in the pixels it skipped.

In other words, each pixel that maps to a 0 in the random number is now encoded with the remaining ciphertext. In this manner, all the pixels are potentially used, as needed by the ciphertext length, and more importantly, the ciphertext is more spread out through the cover image. Just by looking at the LSB of each pixel, it is very difficult, if not impossible, to determine whether or not it is encoded because the data is encrypted using AES, and the original LSB may have been the same as the cipher bit being encoded; therefore, no change would appear if the modified image was compared to the original image. Once the encoding has finished, the resulting image is saved as per the user-specified name in the current working directory. The encoding functionality includes exhaustive error checking to help the user navigate through the menu-oriented procedure and to identify and correct any mistakes.

4.3 Decoding Algorithm for LSB Stegano

The decoding algorithm mirrors the encoding process but in reverse. LSB Stegano requires two inputs for the decoding function: 1) The secret key, used for seeding the random number generator and for AES decryption, and 2) The encoded image that contains the hidden data. LSB Stegano automatically identifies whether the encoded image contains plain text or files by examining the second part of the header (the zip header portion). If the second part of the header matches the zip file header, it indicates that one or multiple files are hidden inside the image. After reading the header to determine the size of the ciphertext and whether the image contains any files, LSB Stegano reverses the encoding process, reading the LSB values instead of altering them.

The secret key is divided into two 128-bit segments: the first segment serves as the initialization vector, and the second segment is used as the secret key for AES. The entire key is used as the seed for the random number generator, which also considers the total number of encodable pixels to generate the random number. This random number, the hop number, is the same as the one generated during the encoding process and is essential for retrieving the hidden data in the correct sequence. By applying the same technique, starting at the first pixel in row 1 (the image is represented as a 2-D array with 0-bounded rows and columns) and reading every pixel that maps to a 1 in the random number, LSB Stegano extracts 3 bits of ciphertext at a time. If LSB Stegano reaches the end of the image and there is still ciphertext remaining, according to the size of the ciphertext encoded in the header, it returns to the beginning of the image and starts reading pixels that map to a 0 in the random number.

Once all the ciphertext is read, it is decrypted using AES. If the header included a zip file header, LSB Stegano writes the decrypted bytes as a zip archive and extracts the archive into a `HIDDEN_DATA` folder. If a `HIDDEN_DATA` folder already exists, LSB Stegano prompts the user to either overwrite or append to the folder for safety reasons. If the header does not include a zip file header, LSB Stegano recognizes that the hidden data is a plain string and displays it accordingly. The decoding algorithm, like the encoding algorithm, includes comprehensive error checking to assist the user and to alert them if the secret key is incorrect or if the encoded source image does not contain any data.

4.4 ECDH - ECDH Elliptic Curve Diffie Hellman

ECDH Elliptic Curve Diffie Hellman (ECDH) is used to create a shared key. In this we use the elliptic curve defined as secp256k1 to generate points on the curve. Bob will generate a public key (Q_B) and private key (d_B), as will Alice (Q_A and d_A). They then exchange their public keys, and the shared key will then be $d_A \times d_B \times G$, and where G is the generator point on the elliptic curve. Another popular curve is Curve 25519, and which is used by Tor nodes Elliptic Curve Diffie Hellman (ECDH) is used to create a shared key. In this example we use secp256k1 (as used in Bitcoin) to generate points on the curve. Its format is:

$$y^2 = x^3 + 7$$

with a prime number (p) of 0xFFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE FFFFFFFC2F

All our operations will be (mod p)

Bob will generate a public key and a private key by taking a point on the curve. The private key is a random number (d_B) and the Bob's public key (Q_B) will be:

$$Q_B = d_B \times G$$

Alice will do the same and generate her public key (Q_A) from her private key (d_A):

$$Q_A = d_A \times G$$

They then exchange their public keys. Alice will then use Bob's public key and her private key to calculate:

$$\text{SharekeyAlice} = d_A \times Q_B = d_A \times d_B \times G$$

This will be the same as:

$$\text{SharekeyAlice} = d_A \times d_B \times G$$

Bob will then use Alice's public key and his private key to determine:

$$\text{SharekeyBob} = d_B \times Q_A = d_B \times d_A \times G$$

This will be the same as:

$$\text{SharekeyBob} = d_B \times d_A \times G$$

SECP256K1: refers to the parameters of the elliptic curve used in Bitcoin's public-key cryptography, and is defined in Standards for Efficient Cryptography (SEC) (Certicom Research, Currently Bitcoin uses secp256k1 with

the ECDSA algorithm, though the same curve with the same public/private keys can be used in some other algorithms such as Schnorr.

secp256k1 was almost never used before Bitcoin became popular, but it is now gaining in popularity due to its several nice properties. Most commonly-used curves have a random structure, but secp256k1 was constructed in a special non-random way which allows for especially efficient computation. As a result, it is often more than 30% faster than other curves if the implementation is sufficiently optimized. Also, unlike the popular NIST curves, secp256k1's constants were selected in a predictable way, which significantly reduces the possibility that the curve's creator inserted any sort of backdoor into the curve.

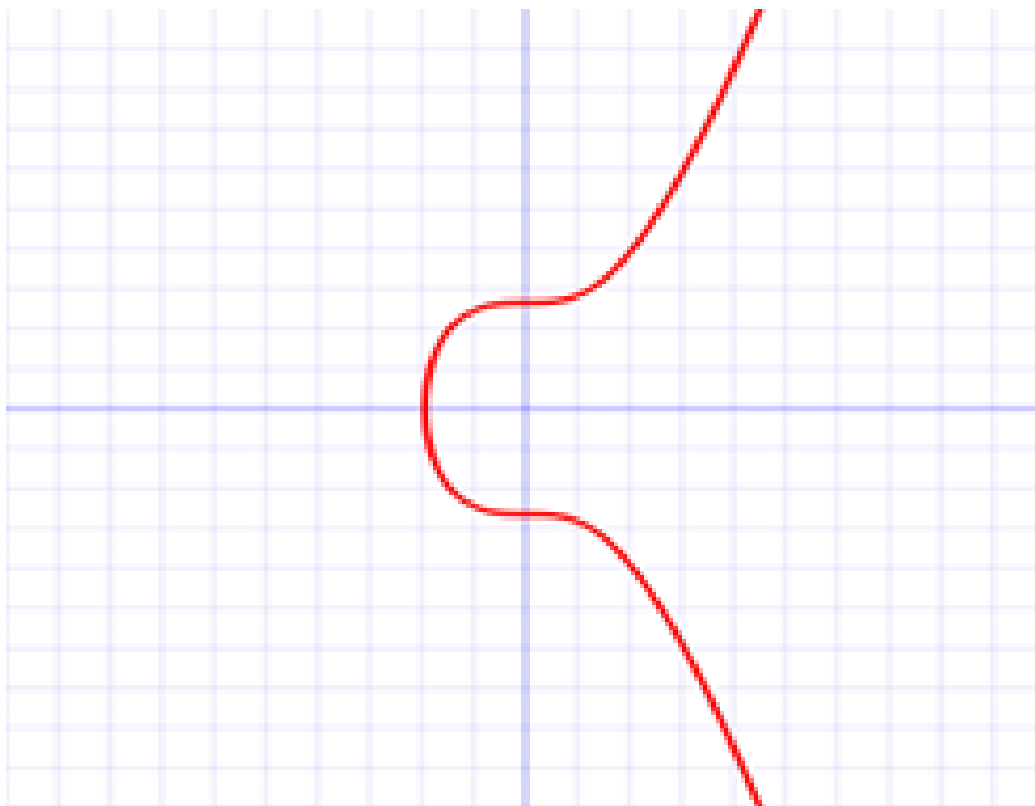


Fig:4.4.1 SECP256K1

Technical details

As excerpted from *Standards*:

The elliptic curve domain parameters over F_p associated with a Koblitz curve secp256k1 are specified by the sextuple $T = (p, a, b, G, n, h)$ where the finite field F_p is defined by:

- $p =$ FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE FFFFFFFC2F
- $= 2^{256} - 2^{32} - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1$

The curve $E: y^2 = x^3 + ax + b$ over F_p is defined by:

- $a =$ 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
- $b =$ 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000007

The base point G in compressed form is:

- $G =$ 02 79BE667E F9DCBBAC 55A06295 CE870B07 029BFCDB 2DCE28D9 59F2815B 16F81798

and in uncompressed form is:

- $G =$ 04 79BE667E F9DCBBAC 55A06295 CE870B07 029BFCDB 2DCE28D9 59F2815B 16F81798
483ADA77 26A3C465 5DA4FBFC 0E1108A8 FD17B448 A6855419 9C47D08F FB10D4B8

Finally the order n of G and the cofactor are:

- $n =$ FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE BAAEDCE6 AF48A03B BFD25E8C D0364141
- $h =$ 01

Properties

- secp256k1 has characteristic p , it is defined over the prime field \mathbb{Z}_p . Some other curves in common use have characteristic 2, and are defined over a binary Galois field $GF(2^n)$, but secp256k1 is not one of them.
- As the a constant is zero, the ax term in the curve equation is always zero, hence the curve equation becomes $y^2 = x^3 + 7$

4.5 SYSTEM IMPLEMENTATION:

PYTHON: Python is a high-level, interpreted programming language that was created in the late 1980s by Guido van Rossum. It is widely used for web development, scientific computing, data analysis, artificial intelligence, machine learning, and many other applications.

Python's syntax is designed to be easy to read and write, with a focus on code readability and simplicity. This makes it a popular choice for beginners learning to code, as well as for experienced programmers who value the ease of development and fast prototyping.

Some key features of Python include:

Interpreted: Python code is interpreted at runtime, which means there is no need to compile code before running it.

Dynamic typing: Python is dynamically typed, which means that variable types are determined at runtime rather than being declared beforehand.

Object-oriented: Python supports object-oriented programming principles such as encapsulation, inheritance, and polymorphism.

Extensible: Python has a large standard library and supports third-party libraries, making it easy to extend the language with additional functionality.

Cross-platform: Python runs on multiple platforms, including Windows, macOS, and Linux.

Python is also known for its excellent documentation and large and supportive community. There are many resources available for learning Python, including books, online tutorials, and forums where developers can ask questions and get help.

4.6 Elliptical curve cryptography (ECC) :

Elliptical curve cryptography (ECC) is a public key encryption technique based on elliptic curve theory that can be used to create faster, smaller and more efficient cryptographic keys. ECC is an alternative to the Rivest-Shamir-Adleman (RSA) cryptographic algorithm and is most often used for digital signatures in cryptocurrencies, such as Bitcoin and Ethereum, as well as one-way encryption of emails, data and software.

An elliptic curve is not an ellipse, or oval shape, but it is represented as a looping line intersecting two axes, which are lines on a graph used to indicate the position of a point. The curve is completely symmetric, or mirrored, along the x-axis of the graph. Public key cryptography systems, like ECC, use a mathematical process to merge two distinct keys and then use the output to encrypt and decrypt data. One is a public key that is known to anyone, and the other is a private key that is only known by the sender and receiver of the data.

ECC generates keys through the properties of an elliptic curve equation instead of the traditional method of generation as the product of large prime numbers. From a cryptographic perspective, the points along the graph can be formulated using the following equation: $y^2 = x^3 + ax + b$

ECC is like most other public key encryption methods, such as the RSA algorithm and Diffie-Hellman. Each of these cryptography mechanisms uses the concept of a one-way, or trapdoor, function. This means that a mathematical equation with a public and private key can be used to easily get from point A to point B. But, without knowing the private key and depending on the key size used, getting from B to A is difficult, if not impossible, to achieve.

ECC is based on the properties of a set of values for which operations can be performed on any two members of the group to produce a third member, which is derived from points where the line intersects the axes as shown with the green line and three blue dots in the below diagram labeled A, B and C. Multiplying a point on the curve by a number produces another point on the curve (C). Taking point C and bringing it to the mirrored point on the opposite side of the x-axis produces point D. From here, a line is drawn back to our original point A, creating an intersection at point E. This process can be completed n number of times within a defined max value. The n is the private key value, which indicates how many times the equation should be run, ending on the final value that is used to encrypt and decrypt data. The maximum defined value of the equation relates to the key size used.

ECC offers several benefits

- It operates on devices with low CPU and memory resources.
- It encrypts and decrypts faster.
- Larger key sizes can be used without significantly increasing the key size or CPU and memory requirements.
 - ECC is thought to be highly secure if the key size used is large enough. The U.S. government requires the use of ECC with a key size of either 256 or 384 bits for internal communications, depending on the sensitivity level of the information being transmitted.
- But ECC is not necessarily any more or less secure compared to alternatives such as RSA. The primary benefit of ECC is the inherent efficiencies gained when encrypting and decrypting data.

CHAPTER – 5

Quality Evaluation Metrics

5 Quality Evaluation Metrics

For LSB (Least Significant Bit) steganography, where information is hidden within the least significant bits of image pixels, you can use the same formulas mentioned above for evaluating image quality metrics such as PSNR, RMSE.

5.1 Mean Square Error and Peak Signal to Noise Ratio

Mean Square Error (MSE) is the square of the errors between the cover image and the modified image [6.3]. If the MSE is low, then there are a very small amount of errors, and if the MSE is high, then there are a large amount of errors. Equation 1 is used to calculate the MSE where the H and W are the height and width of the image, respectively, P(i, j) represents the original cover image, and S(i, j) represents the modified image[6.4] :

$$MSE = \frac{\sum_{i=1}^H \sum_{j=1}^W (P(i, j) - S(i, j))^2}{H * W}$$

The MSE calculated for truck.png and plain_text.png was a miniscule 0.0204. That means that there is a very small difference between the two images that can be determined by a quantitative analysis. The MSE calculated for the shuttle.png and files.png was a relatively small amount of 0.1109. Considering that there was more data hidden in the files.png image, 0.1109 is a small value to have because relatively speaking, it shows that the modified image is very similar to the original image, especially for having 1,928,920 pixels encoded with data out of 6,108,000 pixels compared to the 43,682 pixels encoded for plain_text.png.

5.2 Peak Signal to Noise Ratio (PSNR)

Peak Signal to Noise Ratio (PSNR), Equation 2, is used as a quality measurement to compute the signal-to-noise ratio in decibels between two images :

The higher the PSNR value, the better the quality of the modified image, since the noise (difference) between the pixels in the images is low. The PSNR value for the shuttle.png and files.png was calculated to be 57.68. This is a relatively high value considering that visually the images look exactly the same, but as shown in the figure, the top 66 percent of the image has data hidden in it. For truck.png and plain_text.png, the PSNR value calculated was 65.03. This means that plain_text.png has a very close resemblance to truck.png. The MSE and PSNR values are quantitative metrics used to determine how close modified images are to their original counterparts. By encoding with LSB, the modified images are almost identical to the original images.

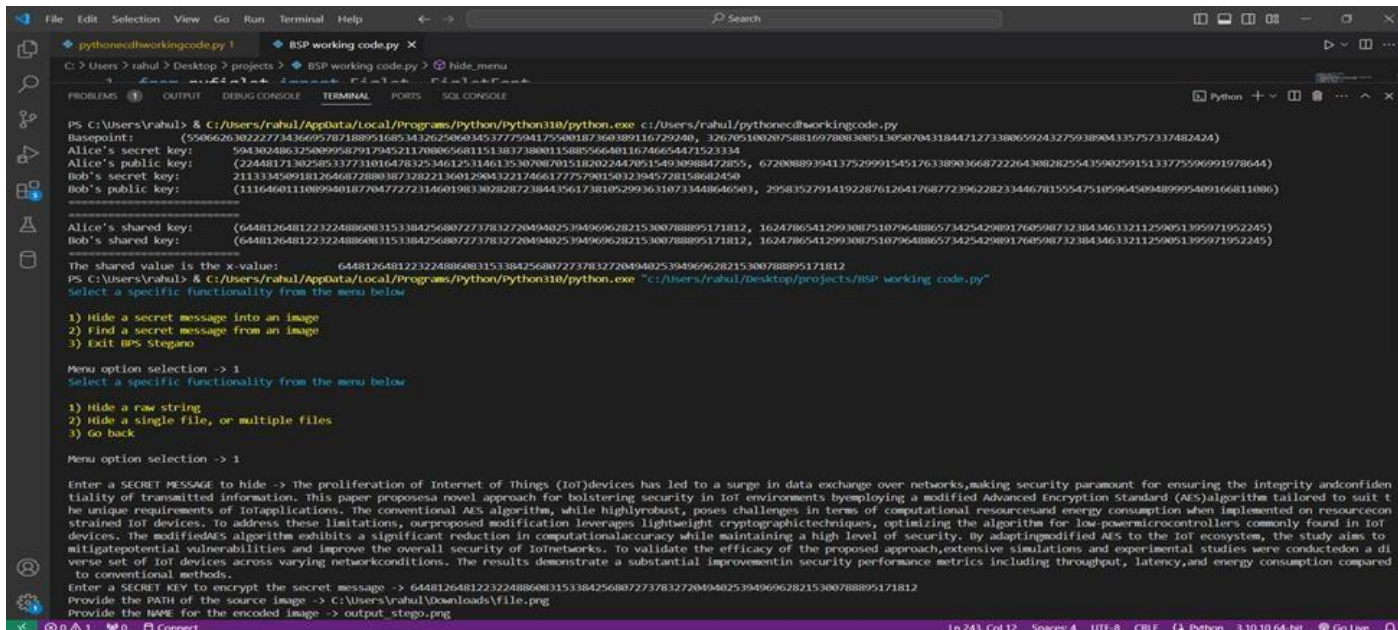
$$PSNR = 10 \log_{10} \frac{L^2}{MSE}$$

The higher the PSNR value, the better the quality of the modified image, since the noise (difference) between the pixels in the images is low. The PSNR value for the shuttle.png and files.png was calculated to be 57.68. This is a relatively high value considering that visually the images look exactly the same, but as shown in the figure, the top 66 percent of the image has data hidden in it. For truck.png and plain_text.png, the PSNR value calculated was 65.03. This means that plain_text.png has a very close resemblance to truck.png. The MSE and PSNR values are quantitative metrics used to determine how close modified images are to their original counterparts. By encoding with LSB, the modified images are almost identical to the original image

CHAPTER-6

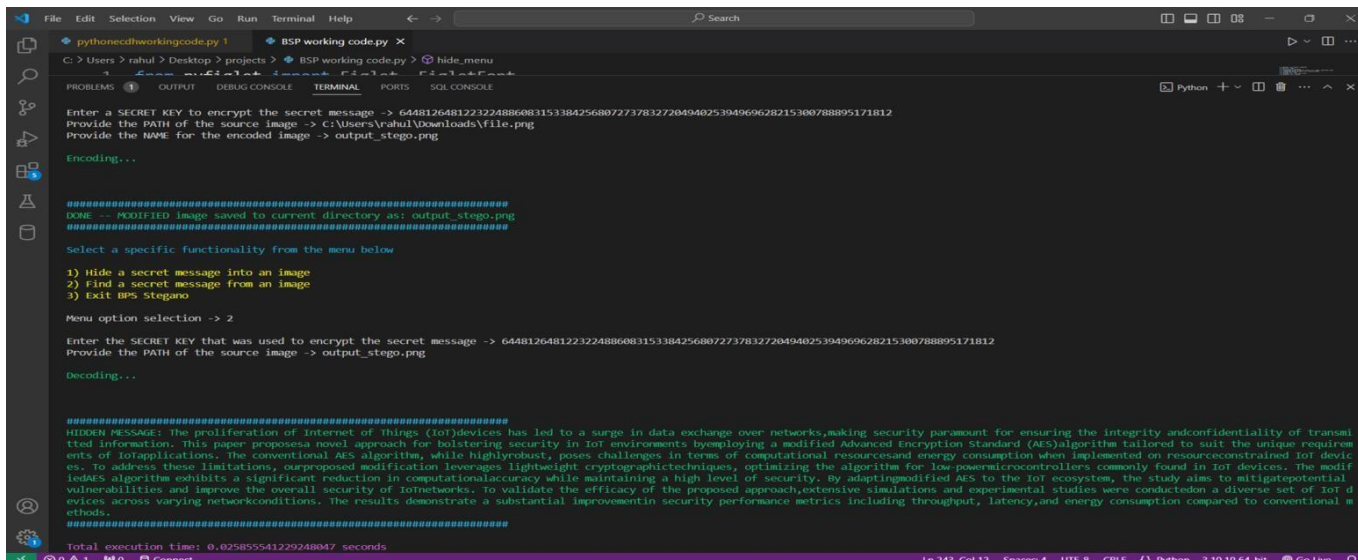
RESULTS

RESULTS



```
PS C:\Users\rahul> & C:\Users\rahul\AppData\Local\Programs\Python\Python310\python.exe c:/Users/rahul/pythonecdhworkingcode.py
Basepoint: (5506626302227734366957871889516853412625660345377294175508187360389116729280, 32679510020758816978083085130507043184471273380659243275938984335757337482424)
Alice's secret key: 5943302486325899587917045211788805681151381738801158855664011674665447152334
Alice's public key: (22448171302585337731016478325346125314613530708701518202244705154930988472855, 67200889394137529991545176338903668722264308282554359025915133775596991978644)
Bob's secret key: 2113345091812646872880387328221360129043221746617775790150323945728158684250
Bob's public key: (111646011108994018770577272314601981302828723844356173810529936310733448646503, 29583527914192287612641768772396228233446781555475105964509489995409166811086)
-----
Alice's shared key: (64481264812232248860831533842568072737832720494025394969628215300788895171812, 16247865412993087510796488657342542989176059873238434633211259051395971952245)
Bob's shared key: (64481264812232248860831533842568072737832720494025394969628215300788895171812, 16247865412993087510796488657342542989176059873238434633211259051395971952245)
-----
The shared value is the x-value: 64481264812232248860831533842568072737832720494025394969628215300788895171812
PS C:\Users\rahul> & C:\Users\rahul\AppData\Local\Programs\Python\Python310\python.exe "c:/Users/rahul/Desktop/projects/BSP_working_code.py"
Select a specific functionality from the menu below
1) Hide a secret message into an image
2) Find a secret message from an image
3) Exit BPS Stegano
Menu option selection -> 1
Select a specific functionality from the menu below
1) Hide a raw string
2) Hide a single file, or multiple files
3) Go back
Menu option selection -> 1
Enter a SECRET MESSAGE to hide -> The proliferation of Internet of Things (IoT) devices has led to a surge in data exchange over networks, making security paramount for ensuring the integrity and confidentiality of transmitted information. This paper proposes a novel approach for bolstering security in IoT environments by employing a modified Advanced Encryption Standard (AES) algorithm tailored to suit the unique requirements of IoT applications. The conventional AES algorithm, while highly robust, poses challenges in terms of computational resources and energy consumption when implemented on resource-constrained IoT devices. To address these limitations, our proposed modification leverages lightweight cryptographic techniques, optimizing the algorithm for low-power microcontrollers commonly found in IoT devices. The modified AES algorithm exhibits a significant reduction in computational accuracy while maintaining a high level of security. By adapting modified AES to the IoT ecosystem, the study aims to mitigate potential vulnerabilities and improve the overall security of IoT networks. To validate the efficacy of the proposed approach, extensive simulations and experimental studies were conducted on a diverse set of IoT devices across varying network conditions. The results demonstrate a substantial improvement in security performance metrics including throughput, latency, and energy consumption compared to conventional methods.
Enter a SECRET KEY to encrypt the secret message -> 64481264812232248860831533842568072737832720494025394969628215300788895171812
Provide the PATH of the source image -> C:\Users\rahul\Downloads\file.png
Provide the NAME for the encoded image -> output_stego.png
```

Fig :6.1 ECDH KEY EXCHANGE



```
Enter a SECRET KEY to encrypt the secret message -> 64481264812232248860831533842568072737832720494025394969628215300788895171812
Provide the PATH of the source image -> C:\Users\rahul\Downloads\file.png
Provide the NAME for the encoded image -> output_stego.png
Encoding...
#####
DONE -- MODIFIED Image saved to current directory as: output_stego.png
#####
Select a specific functionality from the menu below
1) Hide a secret message into an image
2) Find a secret message from an image
3) Exit BPS Stegano
Menu option selection -> 2
Enter the SECRET KEY that was used to encrypt the secret message -> 64481264812232248860831533842568072737832720494025394969628215300788895171812
Provide the PATH of the source image -> output_stego.png
Decoding...
#####
HIDDEN MESSAGE: the proliferation of Internet of Things (IoT) devices has led to a surge in data exchange over networks, making security paramount for ensuring the integrity and confidentiality of transmitted information. This paper proposes a novel approach for bolstering security in IoT environments by employing a modified Advanced Encryption Standard (AES) algorithm tailored to suit the unique requirements of IoT applications. The conventional AES algorithm, while highly robust, poses challenges in terms of computational resources and energy consumption when implemented on resource-constrained IoT devices. To address these limitations, our proposed modification leverages lightweight cryptographic techniques, optimizing the algorithm for low-power microcontrollers commonly found in IoT devices. The modified AES algorithm exhibits a significant reduction in computational accuracy while maintaining a high level of security. By adapting modified AES to the IoT ecosystem, the study aims to mitigate potential vulnerabilities and improve the overall security of IoT networks. To validate the efficacy of the proposed approach, extensive simulations and experimental studies were conducted on a diverse set of IoT devices across varying network conditions. The results demonstrate a substantial improvement in security performance metrics including throughput, latency, and energy consumption compared to conventional methods.
#####
Total execution time: 0.02585541229248847 seconds
```

Fig :6.2
IMAGE STEGANOGRAPHY



FIG 6.3 INPUT



Fig 6.4 OUTPUT

CHAPTER-7

CONCLUSION

CONCLUSION

This project explored a secure communication framework for the Internet of Things (IoT) by combining three cryptographic techniques: Elliptic Curve Diffie-Hellman (ECDH) key exchange, image steganography, and AES-128 encryption. ECDH established a secure, shared secret key between devices without transmitting it openly. Image steganography then provided a covert channel by hiding the encrypted message within an image, making data transmission inconspicuous. Finally, AES-128 encryption ensured the confidentiality of the message itself. This combined approach offers several advantages: enhanced security through layered encryption, covert communication through steganography, and efficient encryption suitable for resource-constrained IoT devices. Future advancements could explore lightweight cryptography for improved performance, adaptive steganography for optimized image quality, and post-quantum cryptography for long-term security against potential threats from quantum computers. By building upon this foundation, we can create an even more robust and secure communication framework for the ever-growing world of IoT

CHAPTER-8

FUTURE SCOPE

FUTURE SCOPE

Building upon this project's strong foundation, future efforts can focus on enhancing its suitability for resource-constrained IoT devices. Exploring lightweight cryptography alternatives for ECDH and AES could improve processing efficiency. Additionally, investigating adaptive steganographic techniques can optimize the trade-off between image quality and data embedding capacity. Furthermore, incorporating post-quantum cryptography would ensure long-term security against potential advancements in quantum computing. These advancements can lead to a more robust, efficient, and future-proof secure communication framework for the ever-evolving realm of the Internet of Things.

CHAPTER-9

REFERENCES

REFERENCES

1. S. A. Goswami, B. P. Padhya and K. D. Patel, "Internet of Things: Applications, Challenges and Research Issues," 2019 Third International conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), Palladam, India, 2019, pp. 47-50, doi: 10.1109/I-SMAC47947.2019.9032474. [1]
2. D. P. Shah and P. G. Shah, "Revisiting of elliptical curve cryptography for securing Internet of Things (IOT)," 2018 Advances in Science and Engineering Technology International Conferences (ASET), Dubai, Sharjah, Abu Dhabi, United Arab Emirates, 2018, pp. 1-3, doi: 10.1109/ICASET.2018.8376830.[2]
3. R. Dumre and A. Dave, "Exploring LSB Steganography Possibilities in RGB Images," 2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT), Kharagpur, India, 2021, pp. 1-7, doi: 10.1109/ICCCNT51525.2021.9579588.[3]
4. M. Koppl et al., "Application of Cryptography Based on Elliptic Curves," 2021 2nd International Conference on Electronics, Communications and Information Technology (CECIT), Sanya, China, 2021, pp. 268-272, doi: 10.1109/CECIT53797.2021.00054.[4]
5. A. Alhamada, O. O. Khalifa, F. D. B. A. Rahman and H. M. Nasir, "Secure Key Exchange for Protecting Health Data Diffie-Hellman Based Approach," 2023 IEEE 9th International Conference on Smart Instrumentation, Measurement and Applications (ICSIMA), Kuala Lumpur, Malaysia, 2023, pp. 296-301, doi: 10.1109/ICSIMA59853.2023.10373446.[5]
6. M. A. Aslam et al., "Image Steganography using Least Significant Bit (LSB) - A Systematic Literature Review," 2022 2nd International Conference on Computing and Information Technology (ICCIT), Tabuk, Saudi Arabia, 2022, pp. 32-38, doi: 10.1109/ICCIT52419.2022.9711628.

CHAPTER-10

APPENDIX

APPENDIX

Views.py:

Code for Image steganography:

```
import cv2
import string
import os
d={}
c={}

for i in range(255):
    d[chr(i)]=i
    c[i]=chr(i)

#print(c)

x=cv2.imread("C:/Users/rahul/Downloads/vignan.jpg")

i=x.shape[0]
j=x.shape[1]
print(i,j)

key=input("Enter key to edit(Security Key) : ")
text=input("Enter text to hide : ")

kl=0
tln=len(text)
z=0 #decides plane
n=0 #number of row
m=0 #number of column

l=len(text)

for i in range(l):
    x[n,m,z]=d[text[i]]^d[key[kl]]
    n=n+1
    m=m+1
    m=(m+1)%3 #this is for every value of z , remainder will be between 0,1,2 . i.e G,R,B plane will be set
    automatically.
    #whatever be the value of z , z=(z+1)%3 will always between 0,1,2 . The same concept is used for
    random number in dice and card games.
    kl=(kl+1)%len(key)

cv2.imwrite("encrypted_img.jpg",x)
os.startfile("encrypted_img.jpg")
print("Data Hiding in Image completed successfully.")
#x=cv2.imread("encrypted_img.jpg")
```



```

kl=0
tln=len(text)
z=0 #decides plane
n=0 #number of row
m=0 #number of column

ch = int(input("\nEnter 1 to extract data from Image : "))

if ch == 1:
    key1=input("\n\nRe enter key to extract text : ")
    decrypt=""

    if key == key1 :
        for i in range(1):
            decrypt+=c[x[n,m,z]^d[key[kl]]]
            n=n+1
            m=m+1
            m=(m+1)%3
            kl=(kl+1)%len(key)
        print("Encrypted text was : ",decrypt)
    else:
        print("Key doesn't matched.")
else:
    print("Thank you. EXITING.")

```

Code For AES:

```

import hashlib
from base64 import b64encode, b64decode
import os
from Cryptodome.Cipher import AES
from Cryptodome.Random import get_random_bytes

os.system("cls")

# Start of Encryption Function
def encrypt(plain_text, password):
    # generate a random salt
    salt = get_random_bytes(AES.block_size)

    # use the Scrypt KDF to get a private key from the password
    private_key = hashlib.scrypt(
        password.encode(), salt=salt, n=2**14, r=8, p=1, dklen=32

```

```

)

# create cipher config
cipher_config = AES.new(private_key, AES.MODE_GCM)

# return a dictionary with the encrypted text
cipher_text, tag = cipher_config.encrypt_and_digest(bytes(plain_text, "utf-8"))
return {
    "cipher_text": b64encode(cipher_text).decode("utf-8"),
    "salt": b64encode(salt).decode("utf-8"),
    "nonce": b64encode(cipher_config.nonce).decode("utf-8"),
    "tag": b64encode(tag).decode("utf-8"),
}

# Start of Decryption Function
def decrypt(enc_dict, password):
    # decode the dictionary entries from base64
    salt = b64decode(enc_dict["salt"])
    cipher_text = b64decode(enc_dict["cipher_text"])
    nonce = b64decode(enc_dict["nonce"])
    tag = b64decode(enc_dict["tag"])

    # generate the private key from the password and salt
    private_key = hashlib.scrypt(
        password.encode(), salt=salt, n=2**14, r=8, p=1, dklen=32
    )

    # create the cipher config
    cipher = AES.new(private_key, AES.MODE_GCM, nonce=nonce)

    # decrypt the cipher text
    decrypted = cipher.decrypt_and_verify(cipher_text, tag)

    return decrypted

def main():
    print("\t\tAES 256 Encryption and Decryption Algorithm")
    print("\t\t-----\n\n")
    password = input("Enter the Password: ")
    secret_mssg = input("\nEnter the Secret Message: ")

    # First let us encrypt secret message
    encrypted = encrypt(secret_mssg, password)
    print("\n\nEncrypted:")
    print("-----\n")
    print("\n".join("{}: {}".format(k, v) for k, v in encrypted.items()))

    # Now let us decrypt the message using our original password
    decrypted = decrypt(encrypted, password)

```

```
print("\n\nDecrypted:")
print("-----\n")
print(bytes.decode(decrypted))
```

main()

Code for ECDH:

```
import collections
import random

EllipticCurve = collections.namedtuple('EllipticCurve', 'name p a b g n h')

curve = EllipticCurve(
    'secp256k1',
    # Field characteristic.
    p=0xfffffffffffffffffffffffffffffffffffffffffffffffffffffffc2f,
    # Curve coefficients.
    a=0,
    b=7,
    # Base point.
    g=(0x79be667ef9dcbbac55a06295ce870b07029bfcdb2dce28d959f2815b16f81798,
        0x483ada7726a3c4655da4fbfc0e1108a8fd17b448a68554199c47d08ffb10d4b8),
    # Subgroup order.
    n=0xfffffffffffffffffffffffffffffffebaaedce6af48a03bbfd25e8cd0364141,
    # Subgroup cofactor.
    h=1,
)

# Modular arithmetic
#####

def inverse_mod(k, p):
    """Returns the inverse of k modulo p.
    This function returns the only integer x such that (x * k) % p == 1.
```

```

k must be non-zero and p must be a prime.
"""
if k == 0:
    raise ZeroDivisionError('division by zero')

if k < 0:
    #  $k^{-1} = p - (-k)^{-1} \pmod{p}$ 
    return p - inverse_mod(-k, p)

# Extended Euclidean algorithm.
s, old_s = 0, 1
t, old_t = 1, 0
r, old_r = p, k

while r != 0:
    quotient = old_r // r
    old_r, r = r, old_r - quotient * r
    old_s, s = s, old_s - quotient * s
    old_t, t = t, old_t - quotient * t

gcd, x, y = old_r, old_s, old_t

assert gcd == 1
assert (k * x) % p == 1

return x % p

# Functions that work on curve points #####

def is_on_curve(point):
    """Returns True if the given point lies on the elliptic curve."""
    if point is None:

```

```

    # None represents the point at infinity.
    return True

x, y = point

return (y * y - x * x * x - curve.a * x - curve.b) % curve.p == 0

def point_add(point1, point2):
    """Returns the result of point1 + point2 according to the group law."""
    assert is_on_curve(point1)
    assert is_on_curve(point2)

    if point1 is None:
        # 0 + point2 = point2
        return point2
    if point2 is None:
        # point1 + 0 = point1
        return point1

    x1, y1 = point1
    x2, y2 = point2

    if x1 == x2 and y1 != y2:
        # point1 + (-point1) = 0
        return None

    if x1 == x2:
        # This is the case point1 == point2.
        m = (3 * x1 * x1 + curve.a) * inverse_mod(2 * y1, curve.p)
    else:
        # This is the case point1 != point2.
        m = (y1 - y2) * inverse_mod(x1 - x2, curve.p)

```

```
x3 = m * m - x1 - x2
y3 = y1 + m * (x3 - x1)
result = (x3 % curve.p, -y3 % curve.p)
```

```
assert is_on_curve(result)
```

```
return result
```

```
def scalar_mult(k, point):
```

```
    """Returns k * point computed using the double and point_add algorithm."""
```

```
    assert is_on_curve(point)
```

```
    if k % curve.n == 0 or point is None:
```

```
        return None
```

```
    if k < 0:
```

```
        # k * point = -k * (-point)
```

```
        return scalar_mult(-k, point_neg(point))
```

```
    result = None
```

```
    addend = point
```

```
    while k:
```

```
        if k & 1:
```

```
            # Add.
```

```
            result = point_add(result, addend)
```

```
        # Double.
```

```
        addend = point_add(addend, addend)
```

```
        k >>= 1
```

```
    assert is_on_curve(result)
```

```

    return result

# Keypair generation and ECDSA
#####

def make_keypair():
    """Generates a random private-public key pair."""
    private_key = random.randrange(1, curve.n)
    public_key = scalar_mult(private_key, curve.g)

    return private_key, public_key

print("Basepoint:\t", curve.g)

aliceSecretKey, alicePublicKey = make_keypair()
bobSecretKey, bobPublicKey = make_keypair()

print("Alice\'s secret key:\t", aliceSecretKey)
print("Alice\'s public key:\t", alicePublicKey)
print("Bob\'s secret key:\t", bobSecretKey)
print("Bob\'s public key:\t", bobPublicKey)

print("=====")

sharedSecret1 = scalar_mult(bobSecretKey, alicePublicKey)
sharedSecret2 = scalar_mult(aliceSecretKey, bobPublicKey)

print("=====")
print("Alice\'s shared key:\t", sharedSecret1)
print("Bob\'s shared key:\t", sharedSecret2)

print("=====")
print("The shared value is the x-value: \t", (sharedSecret1[0]))

```

