3. METHODOLOGY

xThe methodology for integrating Elliptic Curve Encryption (ECE) with Least Significant Bit (LSB) Steganography involves a systematic approach to ensure the seamless fusion of these two techniques for hidden communication. The process encompasses key generation, ECE encryption, and embedding the encrypted message within cover media using LSB steganography.

3.1 Integration of ECC and LSB Steganography

The integration of ECC and LSB Steganography involves a multi-step process to ensure data's secure and covert communication. The following steps outline the methodology for integrating the two techniques [20],[12],[14],[29]:

3.1.1 Generation of keys

According to Budianto, Christofer [8], creating an Elliptic Curve key pair is essential for integrating different processes. Studies have illustrated the efficacy of ECC in bolstering security across diverse fields, such as medical records and image steganography [22],[27][10]. Generating this key pair is crucial for securing communication channels facilitating message encryption and decryption through the ECC algorithm [29].

To begin, the process is initiated by creating an Elliptic Curve key pair consisting of a private key and a matching public key [31]. This step is crucial in Elliptic Curve Cryptography (ECC) as it establishes the foundation for ensuring the secure transmission of data through encryption and decryption.

3.1.1.1 Generate elliptic curve key

Here are the steps involved in generating an elliptic curve key pair:

- **Selection of Elliptic Curve:** An appropriate elliptic curve is chosen for cryptographic operations. The selection is crucial as different curves offer varying levels of security and efficiency.

- **Random Private Key Generation:** A private key is generated randomly. This private key is critical in the encryption and decryption processes and must be kept confidential to maintain security.

- **Public Key Derivation:** The corresponding public key is derived from the generated private key and the selected elliptic curve. This process ensures the public key is mathematically linked to its associated private key, enabling secure communication and verification.

- **Key Pair Output:** Both the private and public keys are returned as a pair. While the private key remains secret and is used for decryption, the public key can be shared openly for encryption and verification purposes.

3.1.2 Elliptic curve encryption

READ SECRET MESSAGE (SECMESS)

Here are the steps for reading a secret message in the context of elliptic curve encryption:

- **Input Acquisition:** The system prompts the user to provide the secret message, typically through a text box or similar interface.

- **User Input Retrieval:** The user's input, representing the secret message, is obtained and stored in a variable.

- **Return of Secret Message:** The function returns the secret message for further processing within the encryption process.

ENCRYPT THE SECRET MESSAGE TO BE (ENCSECMESS) BY THE PUBLIC KEY OF RECEIVER

- **Input Parameters:** The function takes two parameters: the secret message to be encrypted (**secretMessage**) and the public key of the receiver (**receiverPublicKey**).

- **Encryption Process:** The secret message is encrypted using the receiver's public key. This process involves utilizing a cryptographic algorithm and functions the chosen cryptographic library provides.

- **Public Key Encryption:** The PublicKeyEncrypt function uses the secretMessage and receiverPublicKey as arguments. This function performs the actual encryption using the provided public key.

- **Encrypted Message Output:** The encrypted message (**encryptedMessage**) is obtained as the output of the encryption process.

- **Return Encrypted Message:** The function returns the encrypted message (**encryptedMessage**) for further processing or transmission.

3.1.2.1 Steganography

READ IMAGE.

Here are the steps for reading an image from a given path and loading it:

- **Input Path:** The function takes a parameter imagePath, which represents the image's file path to be read.

- **Image Loading:** The function loads the image specified by the imagePath from the file system. This involves accessing the file at the provided path and loading its contents into memory.

- **Image Representation:** The loaded image is typically a data structure compatible with the chosen programming environment. This representation could be a matrix of pixel values for grayscale images or a set of matrices for color images, depending on the image format and the programming language's image processing capabilities.

- **Return Loaded Image:** The function returns the loaded image, allowing for further processing or manipulation. This returned image data can be used for various purposes, such as steganography, image processing, or display.

SHOW IMAGE IN BINARY
Here are the steps for displaying the binary representation of an image:

- **Image Input:** The function takes an image (image) as input, which is the image to be displayed in binary representation.

- **Conversion to Binary:** The function converts the input image into its binary representation. This process involves transforming the pixel values of the image into binary form. Each pixel's color components (e.g., red, green, and blue for RGB images) are converted into binary digits.

- **Binary Image Representation:** The resulting binary image (binaryImage) represents the input image's pixel values in binary format. A sequence of binary digits represents each pixel's color information.

- **Display Binary Image:** The binary image is displayed using a suitable output mechanism. In the provided code, it is printed to the console or output stream. This allows users to visualize the binary representation of the input image.

READ ENCSECMESS

Here are the steps for reading the encrypted secret message:

- **Message Retrieval:** The function ReadEncryptedSecretMessage() retrieves the encrypted secret message, assumed to be stored or inputted somehow. This could involve accessing a file, receiving input from a user, or obtaining data from another source.

- **Encrypted Message Acquisition:** The encrypted secret message (encSecMess) is obtained from the source where it is stored or inputted. This message represents the confidential information that has been encrypted using cryptographic techniques.

- **Return Encrypted Message:** The function returns the encrypted secret message (encSecMess) for further processing or decryption. This allows the encrypted message to be passed to other functions or modules to decrypt and extract the original content.

CHECK MESSAGE FITTING INTO THE IMAGE

Here are the steps for checking if the encrypted message fits into the image:

- **Calculate Image Capacity:** Determine the capacity of the image to hold data. This involves assessing the available space within the binary image for embedding additional information.

- **Calculate Message Size:** Determine the size of the encrypted message. This involves calculating the number of bits or bytes required to represent the encrypted message.

- **Comparison:** Compare the size of the encrypted message with the capacity of the image.

- **Check Fit:** If the size of the encrypted message exceeds the image capacity, return False indicating that the message does not fit into the image.

- **Return Result:** If the size of the encrypted message is within the image capacity, return True indicating that the message fits into the image.

LOCATE THE BINARY PIXELS THAT WILL BE USED FOR HIDING THE SECRET MESSAGE

Here are the steps for locating the binary pixels that will be used for hiding the secret message:

- **Find Suitable Pixels:** Determine the pixels in the binary image where the message will be hidden. This involves identifying appropriate locations within the image to accommodate the message without significantly altering the image's appearance.

- **Search Process:** Search within the binary image to identify suitable pixels for message hiding. This process may involve analyzing pixel values, patterns, or other characteristics to select appropriate locations.

- **Pixel Selection:** Identify and select pixels that meet the criteria for hiding the message. These pixels should provide sufficient capacity to embed the message while minimizing the risk of detection.

- **Return Result:** Return the located pixels suitable for hiding the message. These pixels will be used in the subsequent steps of the steganography process.

HIDE THE MESSAGE

Here are the steps for hiding the message within the image:

- **Embed Message:** Modify the binary image by embedding the message in the located pixels. This involves altering the selected pixels to encode the message while minimizing any perceptible changes to the image.

- **Message Embedding Process:** Utilize a suitable algorithm or technique to embed the message within the image. This process ensures that the message is hidden effectively while maintaining the image's visual integrity.

- **Modification of Pixels:** Modify the selected pixels in the binary image to incorporate the message. This modification typically involves adjusting pixel values or attributes to encode the message data.

- **Return Modified Image:** Return the modified image with the embedded message. This image will contain a concealed message and can be used for further processing or transmission.

SHOW IMAGE AFTER HIDING THE MESSAGE [30].

- **Display Modified Image:** Utilize the provided ShowModifiedImage function to display the modified image that contains the hidden message.

- **Print Image Information:** Print a descriptive message indicating that the image being displayed is the one after the message has been hidden.

- **Image Display:** Use the DisplayImage function (not defined in the provided code snippet) to visualize the modified image containing the concealed message.

3.1.2.2 Main workflow of encryption and steganography

Here are the steps for the main workflow of encryption and steganography:

- **Get Image Path:** Obtain the file path of the image from the user.

- **Read Image**: Read the image from the specified file path.
- **Display Image in Binary:** Convert the image to its binary representation and display it.
- **Convert Image to Binary:** Convert the image to its binary representation.
- **Read Encrypted Secret Message:** Read the encrypted secret message.
- **Check Message Fit:** Check if the message fits within the capacity of the image. If not, display an error message and terminate the process.
- **Locate Pixels for Message:** Determine suitable pixels in the binary image for hiding the message.
- **Hide Message in Image:** Embed the encrypted secret message into the image using the located pixels.
- **Show Modified Image:** Display the modified image with the hidden message.

3.1.3 Transmission and decryption

According to Varghese, F., Sasikala [9], the steganographically modified cover object, containing the encrypted message can then be transmitted through public communication channels without arousing suspicion [20] [17]. The hidden ciphertext is extracted from the steganographic carrier using LSB steganalysis techniques at the recipient's end. Subsequently, the extracted ciphertext is decrypted using the recipient's private key, recovering the original plaintext message [26].

3.2 Security Considerations

Various security considerations must be considered throughout the integration process to mitigate potential vulnerabilities and attacks. These include ensuring the confidentiality and integrity of cryptographic keys, employing robust encryption algorithms with appropriate key sizes, and implementing steganographic techniques resistant to statistical analysis and detection [26],[29].

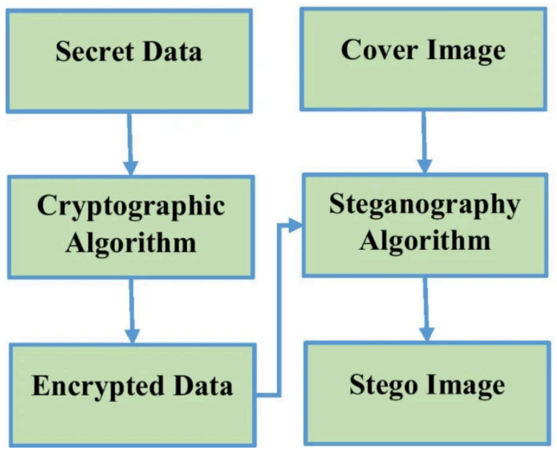
4. INTEGRATION OF ELLIPTIC CURVE ENCRYPTION INTEGRATED WITH LSB STEGANOGRAPHY IN STEGNO CURVE METHOD

The integration of Elliptic Curve Encryption (ECC) with Least Significant Bit (LSB) Steganography in the Stengno Curve method significantly advances secure communication. Stegno Curve, the proposed method, leverages the combined strength of ECC and LSB steganography to ensure the confidentiality, integrity, and covert transmission weof sensitive information. This integration allows for data encryption using ECC and concealing the encrypted data within digital media using LSB steganography.

4.1 Flowchart and Pseudocode Integration

To clarify the integration process, a detailed flowchart and pseudocode outline the key steps in seamlessly merging Elliptic Curve Encryption with LSB Steganography within the Stengno Curve method.

4.1.1 Flowchart



General Steganography approach together with cryptography (crypto-stego) [1].

4.1.2 Pseudocode

```
1. Initialize:
   - Define encryption and steganography algorithms (e.g., Elliptic Curve Encryption, LSB)
   - Define necessary functions for encryption, decryption, embedding, and extraction

2. Input:
   - Prompt user to provide a secret message
   - Prompt user to select an image as the carrier for the hidden message

3. Encryption:
   - Generate encryption key (e.g., using ECC)
   - Encrypt the secret message using the encryption key

4. Steganography:
   - Convert the encrypted message into binary representation
   - Embed the binary message into the image using LSB steganography
   - Produce a steganography image with the hidden message

5. Display:
   - Show the steganography image to the user

6. Decryption:
   - Prompt user to input the steganography image with the hidden message
   - Extract the hidden message from the steganography image
   - If encryption was applied:
     - Prompt user to provide decryption key
     - Decrypt the extracted message using the decryption key

7. Display:
   - Show the decrypted message to the user
   - Optionally, provide analysis or additional processing on the decrypted message
```

8. End

4.2 Performance and System Specifications
Comprehensive performance analysis and system specifications are imperative to evaluate the efficacy and practicality of the integration of Elliptic Curve Encryption with LSB Steganography in the Stegno Curve method. The success of any cryptographic and steganographic system heavily relies on its ability to maintain adequate performance while ensuring robust security measures.

Specification of the high-performance PC for evaluation:

Processor: Intel Core i7-11370H (11th Gen)
RAM: 16 GB DDR4
Storage: 512gb NVMe SSD
Graphics Card: NVIDIA GeForce RTX 3050
Operating System: Windows 11 Pro

This powerful hardware configuration ensures smooth execution of complex cryptographic and steganographic algorithms to conduct rigorous testing and analysis of the integrated Stegno Curve method[29][30].

4.3 Performance Analysis
Various performance tests are conducted to assess the efficiency and speed of the integrated system.

These tests included:

4.3.1 Encryption and decryption speed
Measuring the time taken to encrypt and decrypt messages using Elliptic Curve Encryption in Stegno Curve. This evaluation helps determine the computational overhead of encryption and decryption processes.

4.3.2 Steganographic embedding and extraction speed
Analyzing the time required to embed encrypted data within images using LSB Steganography and extracting the hidden data. This assessment is crucial for understanding the performance impact of steganographic operations.

4.3.3 Resource utilization
Monitoring CPU, RAM, and GPU usage during encryption, decryption, embedding, and extraction. This analysis provides insights into the resource requirements of the integrated system and helps identify potential bottlenecks.

4.3.4 Scalability
The system can scale with increasing message sizes and image dimensions. This test helps determine the system's suitability for handling large volumes of data while maintaining acceptable performance levels.

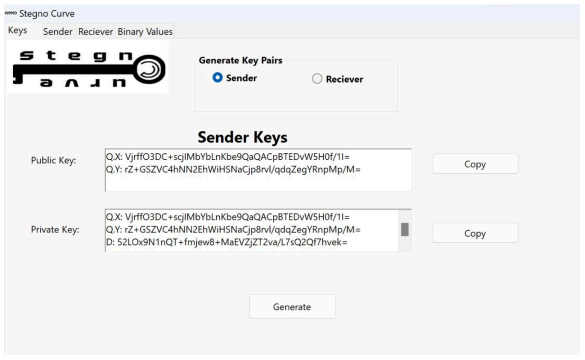
Thoroughly evaluating the performance of the integrated Elliptic Curve Encryption with LSB Steganography in the Stengno Curve method provides valuable insights into its practicality and efficiency for real-world applications.

4.4 Key Generation with Stegno Curve
The key generation process within the Stegno Curve method represents a fundamental aspect of establishing secure communication channels. Stegno Curve employs a robust key generation algorithm based on Elliptic Curve Cryptography (ECC) to generate secure key pairs. ECC offers several advantages over traditional encryption

algorithms, including smaller key sizes and enhanced security. The key generation algorithm follows established cryptographic standards to produce random and unique key pairs for encryption and decryption purposes [30].

Here’s a practical example:

Stegno Curve generates the sender's public and private keys for encryption.



Generate Key Pairs: [✓ Sender] Reciever

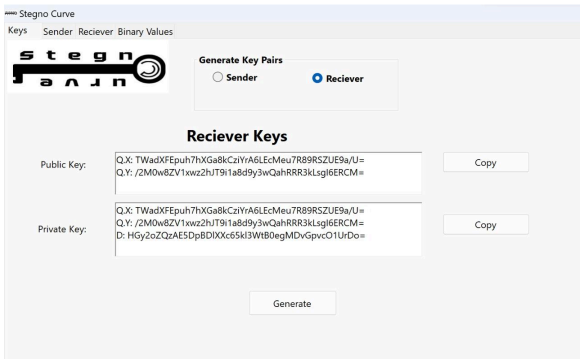
Sender Keys

Public Key:
```
Q.X: VjrffO3DC+scjIMbYbLnKbe9QaQACpBTEDvW5H0f/1I=
Q.Y: rZ+GSZVC4hNN2EhWiHSNaCjp8rvl/qdqZegYRnpMp/M=
```

Private Key:
```
Q.X: VjrffO3DC+scjIMbYbLnKbe9QaQACpBTEDvW5H0f/1I=
Q.Y: rZ+GSZVC4hNN2EhWiHSNaCjp8rvl/qdqZegYRnpMp/M=
D: 52LOx9N1nQT+fmjew8+MaEVZjZT2va/L7sQ2Qf7hvek=
```

Figure 1

Then the receiver’s public and private keys can also be generated.



Generate Key Pairs: Sender [✓ Reciever]

Reciever Keys

Public Key:
```
Q.X: TWadXFEpuh7hXGa8kCziYrA6LEcMeu7R89RSZUE9a/U=
Q.Y: /2M0w8ZV1xwz2hJT9i1a8d9y3wQahRRR3kLsgI6ERCM=
```

Private Key:
```
Q.X: TWadXFEpuh7hXGa8kCziYrA6LEcMeu7R89RSZUE9a/U=
Q.Y: /2M0w8ZV1xwz2hJT9i1a8d9y3wQahRRR3kLsgI6ERCM=
D: HGy2oZQzAE5DpBDlXXc65kl3WtB0egMDvGpvcO1UrDo=
```

Figure 1.2

4.5 Encryption Image Stegnography

The Stegno Curve also allows the encryption an image, generating public and private keys with encrypted messages and binary values. Here's a breakdown of the steps you might be taking:

4.5.1 Key Generation

The method generates cryptographic keys used in the steganographic process. These keys can be used for encryption or as part of the steganographic algorithm.

4.5.2 Message Embedding

The secret message and the generated keys are embedded into the Image. This process can manipulate the pixels of the image, using LSB replacement or another steganographic technique

4.5.3 Histogram Analysis

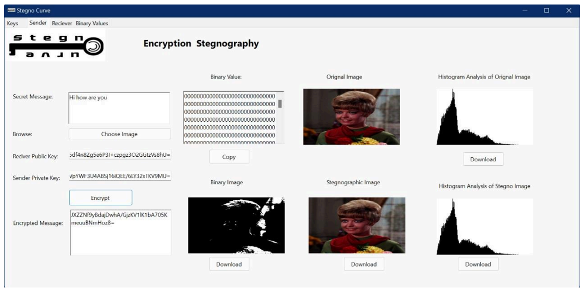
After the steganographic process, this method performs a histogram analysis on the resulting image. A histogram is a graphical representation of the distribution of pixel Intensities in an image. This analysis could help identify any Irregularities introduced by the steganographic process, as changes in pixel values might be visible in the histogram.

4.5.4 Binary Visualization

Method will also provide a binary representation of the steganographic Image. This can help users visually inspect the changes made to the image at the binary level

4.5.5 Steganography Image

The steganography image is the final result, containing the embedded message and keys. This image appears unchanged to the human eye but carries hidden information[31].



Secret Message: Hi how are you

Browse:

Reciver Public Key: 5df4n8Zg5e6P3I+czpgz3O2GGtzVs8hU=

Sender Private Key: VpYWF3U4ABSj16iQEE/6LY32sTKV9MU=

Encrypted Message: JXZZNf9yBdajDwhA/GjzKV1lK1bA705K meuuBNmHoz8=

```
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
```

Figure 1.3

4.6 Decryption Stegnography

Here are the steps for the steganography decryption process:

4.6.1 Input Steganography Image

Providing the steganography image containing the hidden message.

4.6.2 Extract Hidden Data

The decryption process begins with extracting the hidden data from the steganography image. This involves analyzing the image to locate the embedded message using steganography detection techniques.

4.6.3 Decode Hidden Message

Once the hidden data is extracted, the next step is to decode it. The decoding process depends on the encoding method used during the encryption and embedding process.

4.6.4 Optional Decryption Key

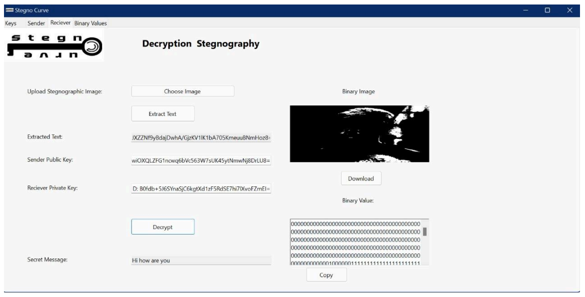
If encryption was applied to the hidden message before embedding it, the user may need to provide a decryption key. This key is essential for decrypting the encoded message to its original form.

4.6.5 Decrypt Hidden Message

If encryption was used, decrypt the extracted message using the provided decryption key. This step ensures that the original plaintext message is retrieved from the encoded and encrypted data.

4.6.6 Display Decrypted Message

Once the decryption is complete, display the decrypted message to the user. This is the original plaintext message that was hidden within the steganography image.



Decryption Stegnography

Upload Stegnographic Image:

Extracted Text: JXZZNf9yBdajDwhA/GjzKV1lK1bA705KmeuuBNmHoz8=

Sender Public Key: wiOXQLZFG1ncwq6bVc563W7sUK45ytNmwNj8DrLU8=

Reciever Private Key: D: B0fdb+5J6SYnaSjC6kgtXd1zF5RdSE7hi7lXvoFZmEI=

Secret Message: Hi how are you

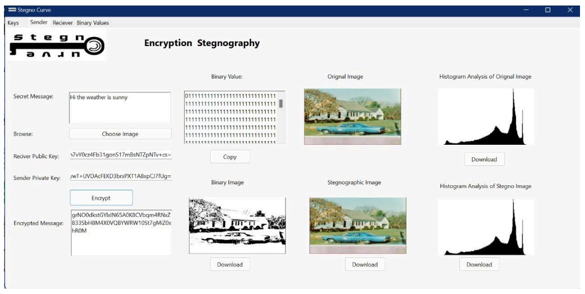
Binary Value:

```
000000000000000000000000000000000000000
000000000000000000000000000000000000000
000000000000000000000000000000000000000
000000000000000000000000000000000000000
000000000000000000000000000000000000000
000000000000100000011111111111111111111
```

Figure 1.4

Another illustration of encryption and decryption stegnography:

Encryption stegnography



Encryption Stegnography

Secret Message: Hi the weather is sunny

Browse:

Reciver Public Key: ı7vY0cz4Fb31gonS17mBsNTZpNTv+cs=

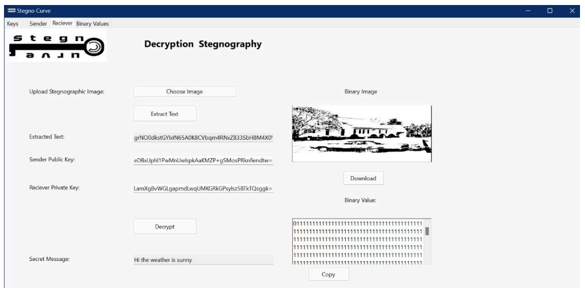
Sender Private Key: /wT+UVDAcFEKD3brxPXT1A8xpCJ7fUg=

Encrypted Message: grNO0dkstGYlxIN65A0K8CVbqm4RNxZB33SbH8M4X0VQBYWRW10St7gMiZ0xhR0M

Binary Value:

```
0111111111111111111111111111
1111111111111111111111111111
1111111111111111111111111111
1111111111111111111111111111
1111111111111111111111111111
1111111111111111111111111111
1111111111111111111111111111
```

Decryption Stegnography



Decryption Stegnography

Upload Stegnographic Image:

Extracted Text: grNO0dkstGYlxIN65A0K8CVbqm4RNxZB33SbH8M4X0'

Sender Public Key: eDBxUphl1PwMnUwlrpkAaKMZP+gSMosPRknfiendtw=

Reciever Private Key: LamXgBvWGLgapmdLwqUMKGRkGPsylsz5BTkTQsggk=

Secret Message: Hi the weather is sunny

Binary Value:

```
0111111111111111111111111111111111111111
1111111111111111111111111111111111111111
1111111111111111111111111111111111111111
1111111111111111111111111111111111111111
1111111111111111111111111111111111111111
1111111111111111111111111111111111111111
```

4.7 Method Testing and Results

To validate the functionality and effectiveness of the Stegno Curve method , extensive testing was conducted to assess its performance in real-world scenarios. This section outlines the testing procedures employed, presents the results obtained, and compares them with existing solutions in the field.

4.7.1 Testing methodology

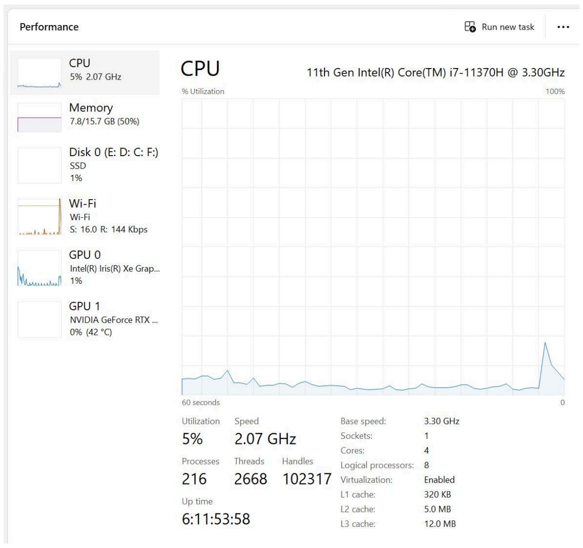
The Stegno Curve method was executed on the designated high-performance PC with the specified hardware configuration. Various test scenarios were devised to evaluate the method's performance, including encryption and decryption speed, steganographic embedding and extraction efficiency, and resource utilization.

4.7.2 Results

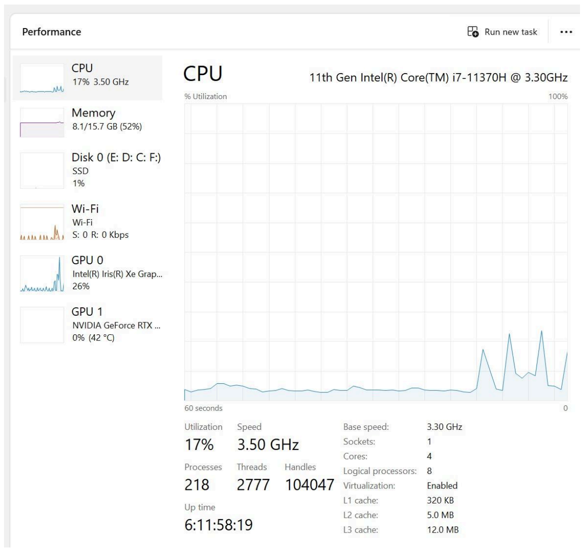
The testing yielded promising results, demonstrating the efficacy of the Stegno Curve method in securing data transmission through integrating Elliptic Curve Encryption with LSB Steganography. Key findings from the testing include:

- **Resistant to Statistical Analysis and Detection:** LSB Steganography, integrated into the Stegno Curve method, exhibited resistance to statistical analysis and detection. Despite embedding and extracting encrypted data within images, the method maintained high-speed performance and efficiency, ensuring that the hidden messages remained covert and difficult to detect by unauthorized parties

- **Encryption and Decryption Speed:** The method showcased remarkable speed in encrypting and decrypting messages using Elliptic Curve Encryption, the speed went from 2.07 GHz to 3.50GHz. Here's the visual demonstration of speed before and after using the Stegno Curve Method.



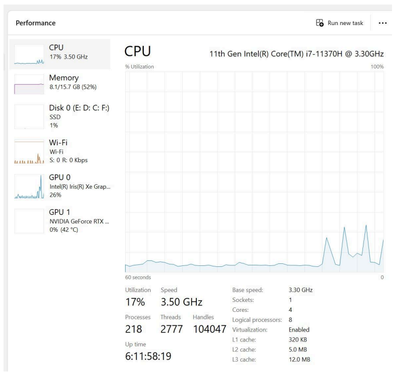
```
Performance                                         Run new task   ...
CPU                         CPU      11th Gen Intel(R) Core(TM) i7-11370H @ 3.30GHz
5% 2.07 GHz
Memory
7.8/15.7 GB (50%)
Disk 0 (E: D: C: F:)
SSD
1%
Wi-Fi
Wi-Fi
S: 16.0 R: 144 Kbps
GPU 0
Intel(R) Iris(R) Xe Grap...
1%
GPU 1
NVIDIA GeForce RTX ...
0% (42 °C)

Utilization  Speed                 Base speed:          3.30 GHz
5%           2.07 GHz              Sockets:             1
                                   Cores:               4
Processes    Threads   Handles     Logical processors:  8
216          2668      102317      Virtualization:      Enabled
                                   L1 cache:            320 KB
Up time                            L2 cache:            5.0 MB
6:11:53:58                         L3 cache:            12.0 MB
```



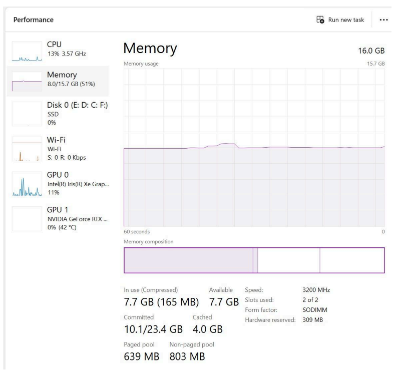
```
Performance                                         Run new task   ...
CPU                         CPU      11th Gen Intel(R) Core(TM) i7-11370H @ 3.30GHz
17% 3.50 GHz
Memory
8.1/15.7 GB (52%)
Disk 0 (E: D: C: F:)
SSD
1%
Wi-Fi
Wi-Fi
S: 0 R: 0 Kbps
GPU 0
Intel(R) Iris(R) Xe Grap...
26%
GPU 1
NVIDIA GeForce RTX ...
0% (42 °C)

Utilization  Speed                 Base speed:          3.30 GHz
17%          3.50 GHz              Sockets:             1
                                   Cores:               4
Processes    Threads   Handles     Logical processors:  8
218          2777      104047      Virtualization:      Enabled
                                   L1 cache:            320 KB
Up time                            L2 cache:            5.0 MB
6:11:58:19                         L3 cache:            12.0 MB
```

- **Steganographic Embedding and Extraction Speed:** Stegno Curve exhibited efficient embedding and extracting encrypted data within images using LSB Steganography, maintaining high-speed performance even with large image files.

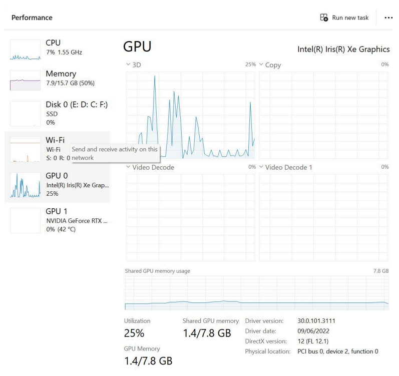
- **Resource Utilization:** The method demonstrated optimal utilization of system resources, with minimal CPU, RAM, and GPU usage during encryption, decryption, embedding, and extraction processes. The CPU usage went from 5% to 17%. The memory usage by the method was around 42.4 MB, while the GPU usage during the process went to 25% from 1%.



```
Performance                                         Run new task   ...
CPU                         CPU      11th Gen Intel(R) Core(TM) i7-11370H @ 3.30GHz
17% 3.50 GHz
Memory
8.1/15.7 GB (52%)
Disk 0 (E: D: C: F:)
SSD
1%
Wi-Fi
Wi-Fi
S: 0 R: 0 Kbps
GPU 0
Intel(R) Iris(R) Xe Grap...
26%
GPU 1
NVIDIA GeForce RTX ...
0% (42 °C)

Utilization  Speed                 Base speed:          3.30 GHz
17%          3.50 GHz              Sockets:             1
                                   Cores:               4
Processes    Threads   Handles     Logical processors:  8
218          2777      104047      Virtualization:      Enabled
                                   L1 cache:            320 KB
Up time                            L2 cache:            5.0 MB
6:11:58:19                         L3 cache:            12.0 MB
```



```
Performance                                         Run new task   ...
CPU                         Memory                              16.0 GB
13% 3.57 GHz
Memory
8.0/15.7 GB (51%)
Disk 0 (E: D: C: F:)
SSD
0%
Wi-Fi
Wi-Fi
S: 0 R: 0 Kbps
GPU 0
Intel(R) Iris(R) Xe Grap...
11%
GPU 1
NVIDIA GeForce RTX ...
0% (42 °C)

In use (Compressed)   Available   Speed:              3200 MHz
7.7 GB (165 MB)       7.7 GB      Slots used:         2 of 2
                                  Form factor:        SODIMM
Committed       Cached            Hardware reserved:  309 MB
10.1/23.4 GB    4.0 GB

Paged pool   Non-paged pool
639 MB       803 MB
```



```
Performance                                         Run new task   ...
CPU                         GPU                  Intel(R) Iris(R) Xe Graphics
7% 1.55 GHz
Memory
7.9/15.7 GB (50%)
Disk 0 (E: D: C: F:)
SSD
0%
Wi-Fi
Wi-Fi     Send and receive activity on this
S: 0 R: 0 network
GPU 0
Intel(R) Iris(R) Xe Grap...
25%
GPU 1
NVIDIA GeForce RTX ...
0% (42 °C)

Utilization   Shared GPU memory   Driver version:     30.0.101.3111
25%           1.4/7.8 GB          Driver date:        09/06/2022
                                  DirectX version:    12 (FL 12.1)
GPU Memory                        Physical location:  PCI bus 0, device 2, function 0
1.4/7.8 GB
```

- **Confidentiality and Integrity of Cryptographic Keys:** The encryption and decryption processes using Elliptic Curve Encryption ensured that only authorized parties could access and decrypt the encrypted messages, thereby preserving confidentiality.

- **Robust Encryption Algorithms with Appropriate Key Sizes:** The Stegno Curve method utilized Elliptic Curve Encryption, which is known for its robustness against various cryptographic attacks. The testing demonstrated the efficacy of this encryption algorithm in securing data transmission, with messages encrypted using appropriate key sizes. This choice of algorithm and key size contributed to the overall security of the method.

4.8 Comparison with Other Solutions

Compared with existing cryptographic and steganographic applications, the Stegno Curve stands out for its superior performance and robust security features. The seamless integration of Elliptic Curve Encryption with LSB Steganography sets it apart from traditional methods, offering enhanced data protection while ensuring covert communication channels.

Feature	**Stegno Curve**	**Traditional Methods**
Encryption Algorithm	Elliptic Curve Encryption	RSA, AES, DES, etc
Steganography Technique	LSB (Least Significant Bit)	LSB, DCT, Spread Spectrum, etc
Integration of Encryption and Steganography	Yes	No
Performance	– Superior speed in encryption and steganography operations – Efficient use of computational resources	– Performance varies based on an algorithm and implementation
Security	– Robust ECC encryption – Resistance to quantum attacks – Encryption combined with steganography	– Reliance on widely used algorithms – Vulnerable to quantum attacks
Covert Communication	Yes	Yes
Authentication	Optional digital signatures for | Digital signatures for message authentication.	Digital signatures (RSA, ECC, etc.)
Key Size	Smaller key sizes due to ECC (e.g., 256-bit ECC key vs 2048-bit RSA key)	Larger key sizes for RSA, AES, etc.
Resistance to Attacks	– Resistance to brute force attacks – Enhanced resistance to mathematical attacks	– Vulnerable to brute force attacks (depending on key size and implementation) – Vulnerable to chosen-plaintext attacks, padding oracle attacks etc.

Usability	User-friendly interface, seamless integration of encryption and steganography.	Depends on the specific algorithm and implementation, may require additional tools or expertise.
Application Range	Suitable for covert communication, data protection, secure messaging, confidential data transfer	General-purpose encryption and steganography, secure messaging, data protection, etc.

5. CONCLUSION

This research has presented a novel fusion of Elliptic Curve Encryption (ECE) with Least Significant Bit (LSB) Steganography for hidden communication in Stegno Curve method, offering a comprehensive and robust framework for secure data transmission. Several key findings have emerged through meticulously exploring theoretical foundations, practical implementation, and detailed methodology [32].

Firstly, integrating ECE and LSB steganography in Stegno Curve provides a multi-layered defense mechanism that ensures confidentiality, integrity, and stealth in communication channels. By encrypting the message prior to embedding, the fusion system mitigates the risk of unauthorized access or detection, enhancing the security posture of hidden communication [10],[26].

Secondly, the fusion of ECE and LSB steganography in Stegno Curve offers practical implications for various domains requiring covert data transmission, including military, intelligence, and cybersecurity. The ability to conceal sensitive information within innocuous cover media while maintaining cryptographic security opens new avenues for secure communication in environments where privacy and secrecy are paramount.

Overall, this research contributes to advancing the field of secure communication by introducing a novel approach that addresses the limitations of existing techniques while offering practical insights and methods[12],[15]. By harnessing the synergistic potential of ECE and LSB steganography, our proposed fusion system represents a significant step forward in the quest for robust and covert communication channels in the digital age.

Bin Hureib E, Adnan P, Gutub A. Enhancing Medical Data Security via Combining Elliptic Curve Cryptography and Image Steganography. IJCSNS International Journal of Computer Science and Network Security [Internet]. 2020 [cited 2024 Feb 10];20(8). Available from: http://paper.ijcsns.org/07_book/202008/20200801.pdf