

BaatCheet: Android chat application coupling End-to-End encryption and LSB substitution

UJJWAL DODEJA, University of Twente, The Netherlands

In this era of rapid technological advancements, secure communication has become a paramount concern. Messaging applications like WhatsApp, Signal and Telegram offer End-to-end encryption (E2EE), which ensures that only the sender and receiver can read the messages. The persistent need for better security measures has led researchers to explore data-hiding techniques that use media files as data carriers, also known as steganography techniques. Image steganography, the hiding of data in images, has been widely used due to the high frequency of digital images on the Internet. Least Significant Bit (LSB) substitution is a technique that hides the secret data into the least significant bits of an image, and its computationally lightweight nature provides a delicate balance between security and operational efficiency compared to techniques like Discrete Cosine Transform, which provides higher embedding capacity. The study was aimed at achieving two primary goals: implementing End-to-End encryption and integrating LSB substitution for image steganography. The research looks into developing an E2EE Android-based chat application utilizing established E2EE protocols like Signal Protocol from Signal, and MTProto 2.0 from Telegram. Anonymous sign-ups and an extra security layer through LSB substitution distinguish the application from other E2EE chat applications like Telegram, Theerma, etc. This research produced an E2EE chat application incorporating LSB substitution. The research concludes with insights into involved cryptographic algorithms, the integration process with LSB, and its effects on the efficiency in terms of message exchange time, and communication capacity. The results can later be used during the development of new applications, or other educational purposes.

Additional Key Words and Phrases: End-to-end encryption (E2EE), Least Significant Bit (LSB) Substitution, Android Chat Application, Symmetric Encryption

1 INTRODUCTION

In this modern era, technological advancements have helped communication reach new heights in terms of speed, and accessibility. Email remains the leading way for businesses and individuals to communicate at a distance but has not been a very secure option[6]. By offering efficiency and enhancing user experience, messaging applications like WhatsApp, Telegram, and Snapchat have significantly made an impact on day-to-day communication among individuals[5,6]. However, this evolution in communication has also given rise to new types of security vulnerabilities including Malware and Spyware attacks, End-to-End Encryption Concerns, Data Privacy Issues, etc. When selecting a chat application, users expect applications to meet certain security requirements like minimal logging and metadata collection, authorized access, end-to-end encryption, etc.

End-to-end encryption has emerged as one of the most basic requirements to ensure secure communication. E2EE is a way to

ensure that only the parties involved in the communication can read the messages, and prevents even the service providers from reading the messages. The lack of end-to-end encryption leaves private user conversations vulnerable to third parties, organizations behind applications, and government organizations [5]. It is made possible by combining various cryptographic processes like key generation, key distribution, encryption, decryption, and key management [5]. Some popular social media applications like Twitter, Instagram, and Facebook Messenger did not meet this requirement until recently. On the other hand, applications like Signal, Telegram, and WhatsApp, have emerged by providing end-to-end encrypted messaging and prioritizing secure communication. Applications like Signal, Wire, and Telegram have also developed their custom protocols to provide End-to-End encryption namely Signal Protocol, Proteus, and MTProto respectively[6,7,8,9]. WhatsApp has also adapted the Signal Protocol to secure their messaging [6].

End-to-end encryption has proven to be very secure as the data cannot be decrypted without the key. However, the transmission of encrypted communication through communication channels draws attention. This has led to research into hiding the existence of important data during transmission. Steganography is the art of hiding data in a multimedia file for secret communication. Steganography techniques can be categorized based on the types of media files used as information carriers. These categorizations are image, text, audio, video, and protocol [4,12]. Due to the frequency of digital images on the Internet and its applicability, image steganography has drawn a good amount of attention from researchers. Image steganography can be categorized into two domains: spatial domain steganography and transform domain steganography [13]. Spatial domain steganography involves manipulating the pixel values directly, while transform domain steganography pertains to the transformation of the image data before embedding information[13].

One of the most prominent and widely utilized is the Least Significant Bit (LSB) substitution. In LSB substitution, information is hidden in the least significant bits of the pixel values in an image without significantly altering its appearance. This technique provides a high embedding capacity by allowing the use of multiple bits in each pixel for hiding information. Although frequency domain techniques like Discrete Cosine Transformation (DCT) [32] can offer higher embedding capacity, and more robustness over LSB substitution but also come with higher computational complexity [31]. This complexity can directly affect the efficiency of the communication process in a chat application. The advantage of LSB over other techniques lies in its simplicity. [14]. It is well-suited for real-time applications due to its computationally lightweight nature while also allowing optimizations to increase randomness, and tackle image distortion effectively. The embedding capacity can be varied based on these optimizations depending on the requirements. It provides a balance between security, capacity, and efficiency. As a result of its simplicity, it offers seamless integration into existing

TScIT 37, July 8, 2022, Enschede, The Netherlands

© 2022 University of Twente, Faculty of Electrical Engineering, Mathematics and Computer Science.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Version	Key Length	Strength
AES-128	128	128
AES-192	192	192
AES-256	256	256

Table 1. Comparison of Key Lengths and Strength

systems, or for enhancing the security of End-to-End encrypted communication.

This paper looks into the development process of an Android-based chat application that provides End-to-End encryption by utilizing the existing secure protocols, MTProto 2.0 and Signal Protocol, from Telegram and Signal respectively [6,7,9]. This application is further integrated with LSB Substitution to enhance the security of communication. LSB substitution acts as an extra layer of security that differentiates the system from other existing End-to-End encrypted chat applications like Telegram, WhatsApp, or Signal. The research also monitors the trade-offs resulting from this combination.

1.1 Algorithms involved

1.1.1 Diffie-Hellman (DH) Key Exchange. Diffie-Hellman is an asymmetric key agreement protocol. It allows two parties to establish a shared secret key over an insecure communication channel without the need for prior shared secrets. It ensures that the parties can jointly agree on a shared secret key. This enhances the security of communication channels by preventing eavesdroppers from easily determining the shared secret key. The exchange starts with two DH key pairs generated by both parties using a secure random and the Diffie-Hellman key generation procedure. This secure random number generation ensures that the private keys are truly unpredictable and uniformly distributed. The public keys are exchanged through the insecure communication channel. Then, both parties perform the Diffie-Hellman calculation to reach a key agreement and produce a shared secret. This shared secret is large and not fully random, and is used as the base value for deriving shared Advanced Encryption Standard (AES) keys for secure communication [22].

1.1.2 Advanced Encryption Standard (AES). AES is a symmetric key encryption algorithm which means the same key is used for both encryption and decryption. It is designed to encrypt sensitive information, commonly used to encrypt data for confidentiality. It is a block cipher operating on fixed-size blocks of data i.e. 128 bits. It can operate in various block cipher modes namely Electronic Codebook (ECB) [23], Cipher Block Chaining (CBC), Counter (CTR), Galois/Counter Mode (GCM), etc [24, 25]. The operating mode determines how the blocks of data will be processed by the algorithm. It supports various key sizes ranging from a minimum of 128 bits, and a maximum of 256 bits [17]. It supersedes DES to provide larger key lengths and more secure communication. Table 1 provides a small comparison between AES alternatives with respect to key lengths and comparison [17]. In the case of AES, the key lengths directly affect the strength of encryption. It is a widely used symmetric algorithm demanding low memory and provides higher efficiency for large amounts of data.

1.1.3 Least Significant Bit (LSB) Substitution. LSB substitution [12, 14, 21] is an image steganography technique where information is hidden in the least significant bits of the cover image pixels by replacing them with secret message bits in the spatial domain[21]. Typically, digital images use 8 bits to represent each of their Red, Green, and Blue (RGB) channels. These channels represent the color for each pixel of the image. The least significant bits of these color channels for selected or all the image pixels are replaced with the bits of the secret message. The idea is to alter the bits that contribute the least to the visual appearance of the image to keep the image imperceptible to human eyes. The ASCII codes representing the characters of a message are converted into 8-bit binary representations. These representations are hidden bit by bit in the LSBs of the RGB channels. To retrieve the message, These binary values are later concatenated to get the ASCII values and further converted back to the original message. The size of an image and the number of used LSBs directly affect the capacity of this technique. Increasing the capacity of this method often results in noticeable differences in the image's visual quality.

Various modifications and optimization techniques have been used to increase the capacity and imperceptibility of this image steganography technique. These modifications are usually based on differently choosing the LSBs to be used. The optimized LSB technique used in this application makes use of the last bit of each pixel's red color channel. This optimization minimizes the effects of embedding on the image's visual quality[21].

2 LITERATURE REVIEW

Digital communication has been a consistently growing aspect of technology in terms of efficiency, capacity, and availability. However, efficiency might often come with certain trade-offs like security, and affordability, and vice versa. When it comes to secure communication, End-to-end encryption has evolved to become the most prominent requirement while other security enhancements have also gained attention. Image steganography which involves data hiding in images, has also grown into a highly experimented method to enhance security. In recent years, there have been ongoing efforts to create combinations of cryptographic and steganographic methods that aim to encrypt data and further hide its presence in communication channels. Although these kinds of solutions can be promising, they also come with certain limitations. This paper reviewed such existing combinations to provide an efficient and feasible solution.

Initially, the paper studied the different encryption approaches used by modern-day chat applications. It was discovered that several chat applications use a hybrid encryption approach using symmetric and asymmetric cryptography together. The hybrid approach tries to combine the advantages while minimizing the disadvantages [17]. It helps in providing secure and efficient communication. For example, by using RSA for key exchange, and AES for encrypting real-time communication. Some popular examples are WhatsApp, Signal, Theerma, and iMessage. Table 2. highlights the advantages of using a hybrid encryption algorithm over symmetric and asymmetric encryption algorithms.

	Symmetric Encryption Algorithm	Asymmetric Encryption Algorithm	Hybrid Encryption Algorithm
Number of Keys	Single Secret Key	Pair of Keys	Pair of Keys derives single secret shared key
Speed	Very fast	Slow	Very fast
Security	High	Very high	Very high
Application	large amount of data	highly confidential data	large amounts of confidential data

Table 2. Comparison of Encryption Approach

It was found that Signal and Telegram are often considered to be two of the most secure applications as both provide End-to-end encrypted messaging services [5, 6]. The protocols developed by Signal and Telegram are the Signal Protocol and the MTPProto 2.0 respectively. The documentation of these communication protocols was studied extensively to produce an acceptable and up-to-date E2EE schema.

The Signal Protocol [9, 16] is Signal Messenger’s customized cryptographic protocol designed to provide secure end-to-end encryption for instant messaging and voice calls. The cryptographic algorithms used in this protocol are X3DH [26], Curve25519 [27], AES-256, and HMAC-SHA256 [28].

MTPProto 2.0 [7] is the cryptographic protocol developed by Telegram to secure and facilitate communication within their messaging platform. It uses a combination of 256-bit symmetric AES encryption, 2048-bit RSA encryption, and Diffie-Hellman key exchange.

This is also reflected by Table 3 below.

Application	Algorithms
Signal	X3DH Curve25519 AES-256 HMAC-SHA256
Telegram	DHKeyEx AES-IGE AES-256 SHA256

Table 3. Algorithms used by E2EE chat applications

Furthermore, several existing crypto-stego schemes were found that aimed at providing encryption and data hiding together. Fig. 1. provides a general overview of the combined working and the sequential processing of messages in crypto-stego schemes for better understanding.

Varghese et al.[2] proposed a secure data transmission model enhancing the strength of hybrid cryptography and steganography approaches. This model made use of an optimized substitution box

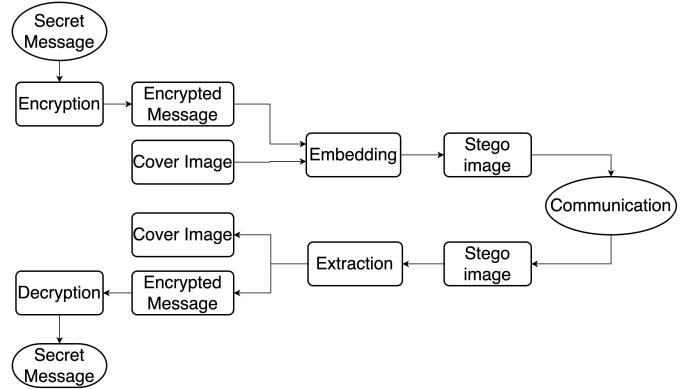


Fig. 1. Overview of Crypto-Stego schemes

(S-box) and a texture-based distortion function. The model integrates Syndrome Trellis Coding (STC) for data hiding to enhance imperceptibility and undetectability. Through means of extensive experiments and subsequent results, the model showcased high capacity and robustness against attacks. However, this model has been implemented and evaluated in MATLAB only which limits the feasibility of the model in real-world scenarios. The security and robustness of this model were not very clearly evaluated against state-of-the-art models and attacks.

Bucerzan et al.[20] developed an Android-based steganography application named SmartSteg which combined LSB steganography with stream cipher cryptography and random algorithm to provide such a desired solution. This application provided exceptional results regarding execution time practical immediately, and support for a variety of files even with large sizes while heavily depending on the secret key for security. This research did not provide any evaluation of the proposed solution in terms of security, capacity, and imperceptibility. Moreover, many vulnerabilities of stream cipher cryptography have been discovered in recent years.

Ahmed et al.[4] developed an Android application that aimed at enhancing security using steganography and cryptography based on the smartphone user’s location. This application utilized GPS technology to collect the sender and recipient’s locations which were used in an equation to obtain the algorithm key. It further used the Mersene Twister random number generator to select pixels for embedding encrypted text. The application resulted in a high level of security but suggested exploration into other image steganography methods. This application makes use of a very unconventional method of encryption which is not very suitable for day-to-day chat applications.

Jan et al.[1] evaluated various image steganography and crypto-stego techniques to find insights into the feasibility and effectiveness of double-layer schemes for future use. This evaluation presented various parameters for Key Analysis, Speed analysis, and statistical analysis of crypto-stego schemes. The results show that most of the image steganography approaches suffered from less security, low payload, poor image quality, and complexity. Furthermore, it suggests the use of edge-based image steganographic methods to

preserve the visual quality of images while providing high embedding capacity. It concluded by highlighting the need to develop efficient image steganographic schemes combined with data encryption to improve image embedding capacity and imperceptibility, and provide security with less complexity, and also ensure resistance to different attacks.

From the literature, it is observed that an intricate balance is required to achieve optimal security when using crypto-stego schemes. The previously developed crypto-stego schemes have limitations when it comes to their real-world applications. The efforts towards providing double-layer security with cryptography and steganography have highlighted the need for innovative solutions in the realm of secure digital communication.

3 RESEARCH QUESTIONS

The following research questions were answered during this research.

- (1) Which existing cryptography algorithms can be used to implement End-to-End encryption in an Android-based chat application?
- (2) How can image steganography techniques like LSB substitution be incorporated into E2E android-based chat applications to enhance secure communication?
- (3) How is the efficiency of a chat application in terms of transmission speeds and memory usage affected when integrated with LSB substitution image steganography technique to enhance security?

4 PROPOSED SYSTEM

BaatCheet is a secure E2EE chat application enabling extra security through LSB Substitution for data hiding while also providing anonymous signups. It makes use of the hybrid approach of encryption inspired by the functioning of the Signal, and MTPProto 2.0 protocols. The application uses Diffie-Hellman Key Exchange working asymmetrically to establish an AES-256 symmetric key for real-time communication. This section explains the functioning of different aspects of the system.

4.1 User Authentication

The application offers complete anonymity to users by terminating the need for phone numbers or email addresses for sign-ups. A user can simply create a user account by setting a username and password. These credentials are sent to the server, and saved by the server for future login sessions. An overview of the Sign-Up process is provided in Fig. 2.

4.2 Key Management

This key management for this application is designed to ensure that only the end devices have access to their private key and the secrets shared with other users.

4.2.1 Server Side. As reflected in Fig. 2., a user device generates a pair of Diffie-Hellman keys during the Sign-Up process. The private key is stored locally on the same device, and the public is sent to the server alongside other user credentials. Upon receiving the user

details of each user, the server stores a mapping of the user and the associated key.

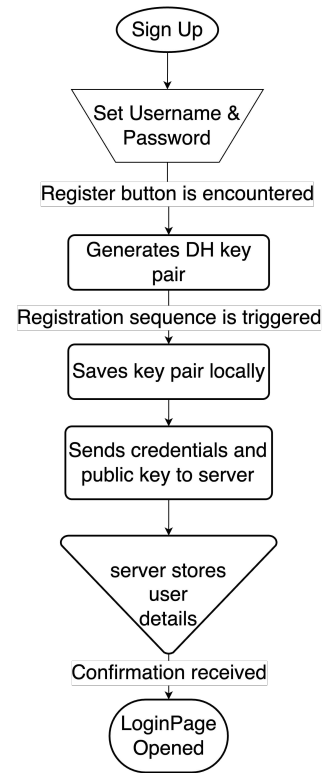


Fig. 2. Sign Up Process

4.2.2 Client Side. After a user logs in, the application retrieves a list of all the public key mappings stored on the server. For each listed user, the receiving device uses the corresponding public key and its own private key to derive a new shared secret key working with the AES algorithm with a key length of 256 bits. The device locally stores a mapping of the user and the secret derived with their public key. This shared secret is used to establish a secure communication channel between the users. This process is reflected in the Fig.3.

4.3 End-to-End encryption provided through Diffie-Hellman Exchange and AES-256

- The Diffie-Hellman key exchange is initiated when an instance of the Diffie-Hellman key pair is generated during the Sign-Up process.
- During the Login process, this exchange is completed when two parties derive a shared secret key with the AES symmetric encryption algorithm with a key length of 32 bytes i.e. 256 bits.
- As a result, both parties end up with the same shared secret key which can be used for symmetric encryption. This ensures that the symmetric key is not shared over the communication

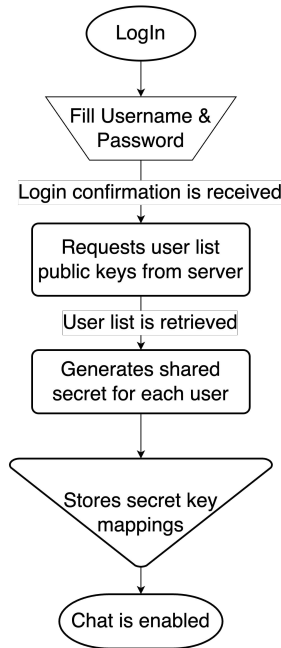


Fig. 3. Login Process

channel which might be vulnerable, and only the end devices can access or derive the secret key. The encryption achieved has the security of asymmetric encryption with the efficiency of symmetric encryption.

4.4 Incorporation of LSB Substitution

One of the main reasons for selecting the LSB substitution over other image steganography techniques is the advantage of seamlessly integrating it with existing systems [20].

- In the encryption phase, the original message is encrypted using the secret key (AES-256) shared with the selected recipient. The encrypted message is passed to the LSB embedding process entering the stego phase.
- Stego Phase
 - The LSB embedding process takes the encrypted message and cover image as inputs. Each character of this message is represented by an ASCII code which is further converted into an 8-bit binary encoding. It iterates through each pixel of the cover image to hide a message bit into the least significant bit of the pixel’s red color channel until the entire message is embedded. As a result, the stego image is generated.
 - The stego image is shared with the recipient over the communication channel.
 - The length of the hidden cipher text is also encrypted with the same secret key and shared right after the image.
 - At the recipient’s side, the length is decrypted to find out the number of hidden bits.

- The extraction process iterates over each pixel to collect the least significant bit of its red color channel until the collected number of bits is equal to the expected number of bits. These bits are converted back into ASCII encodings, and subsequently into the encrypted secret message.
- This extracted message is passed on to the decryption phase.
- In the decryption phase, the extracted message is decrypted using the secret key shared with the sender to retrieve the original message. The secret key shared between the sender and receiver is a symmetric key.

Fig. 4. highlights the architecture of the system concerning the sequential processing of a message and the integration between crypto and stego techniques. This figure provides a real-time example of the process explained above.

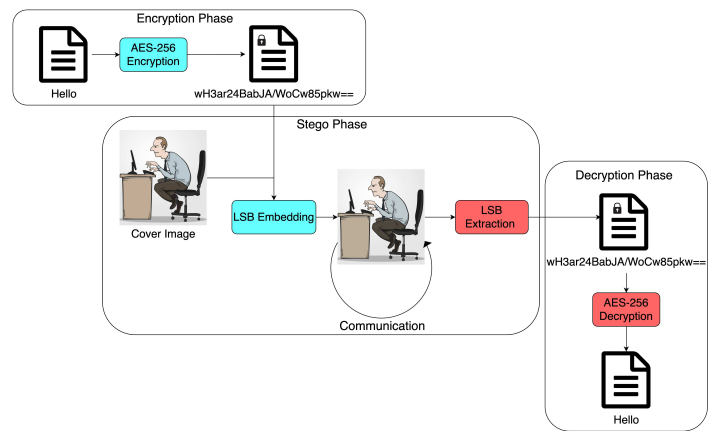


Fig. 4. Integration of Crypto and Stego phases

The proposed solution achieved the following goals in terms of functionality to produce significant results.

- **Goal 1:** End-to-end encrypted chat application on an Android smartphone
- **Goal 2:** An additional security layer using LSB substitution image steganography which embeds messages in images during communication

5 RESULTS AND DISCUSSIONS

The section documents the results of the achieved system concerning the workflow of the application, and evaluations of the system based on different criteria.

- This research resulted in the development of an Android-based chat application that was implemented in Java and Android Studio. The workflow of this application is provided through the flowchart in Fig. 5. When a user initiates the app, they can either log in or sign up depending on whether they are registered. Once the user is logged in, they can retrieve a list of online users and type a message to a user of their choice. This message is passed

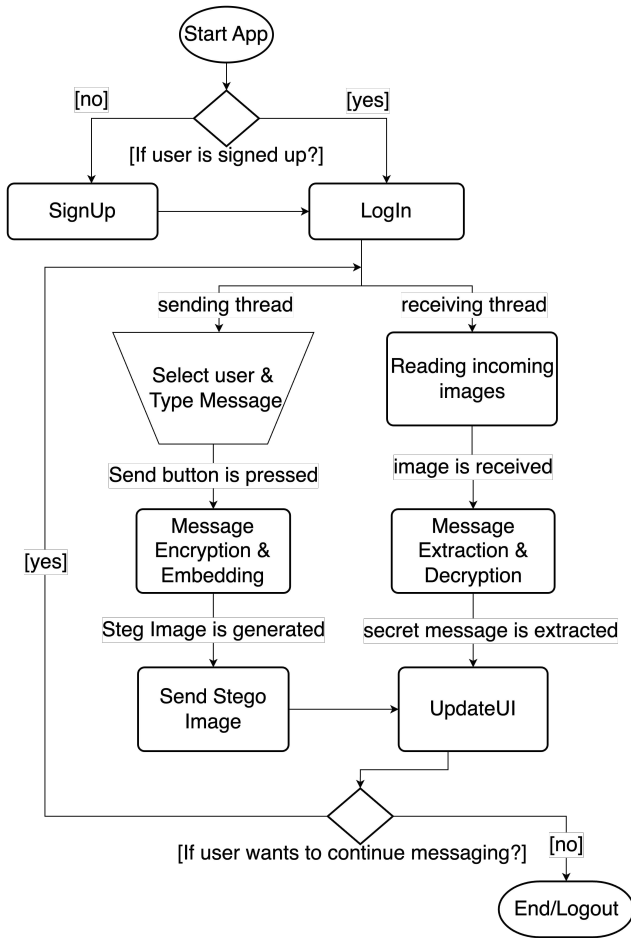


Fig. 5. Workflow of proposed system

through the encryption and embedding process respectively. The generated stego image is sent to the recipient, and the UI is updated. Simultaneously, the receiving thread starts looking for incoming stego images. The received images are passed through the extraction and decryption processes respectively. The received message and the stego image are updated on the UI. Based on their choice to continue the conversation, the user can choose to terminate the app which will automatically log out the user.

- With insights from the documentation of different protocols and the results produced from the development process, it can be summarised that End-to-End encryption was achieved as a result of using a hybrid encryption approach combining asymmetric and symmetric encryption. DH Key exchange and AES are the two existing cryptographic algorithms that are commonly utilized to provide E2EE messaging by most of the secure messengers.
- Using the overview of LSB incorporation provided in section 4.4, it can be summarized that the incorporation of image steganography techniques into E2EE is rather smooth. This

can be said because the encryption, stego, and decryption phases of the system do not interfere with each other’s functionality, and work sequentially. However, it was identified that the encryption process slightly increases the functional complexity of the stego phase due to the increase in the length of messages due to encryption.

The achieved system has been evaluated for performance and capacity as these are the most important aspects to ensure efficient communication. Furthermore, the security of the implemented steganography technique is evaluated for imperceptibility, and using histogram analysis.

5.1 Performance Evaluation

The embedding and extraction of messages using LSB increases the computational complexity which directly affects the efficiency of communication. A set of test messages with different lengths was defined to evaluate the overhead introduced to the communication process after integration with LSB. Table 4. compares the time taken to exchange each of these messages in both application versions with and without LSB substitution. In the case of the E2EE version, the recorded time includes the encryption time, transmission time, and decryption time. For the E2EE with LSB version, this time includes the encryption time, embedding time, transmission time of the image, and the extraction time followed by the decryption time. This evaluation used Fig. 6. as the cover image. The efficiency

Message	Length (bits)	E2EE exchange time (secs)	E2EE with LSB exchange time (secs)
Hello	40	0.786	2.780
Let’s chat	80	0.785	2.658
Welcome to BaatCheet	160	1.656	2.729
This is an E2EE chat application	240	1.614	4.049
-----	>400	1.773	16.954

Table 4. Comparison for the efficiency of communication

achieved was promising but showed signs of increasing computational complexity for bigger messages.



Fig. 6. posture.png

5.2 Embedding Capacity

The LSB implementation hides one bit of information in each pixel which makes the embedding capacity in terms of the number of bits directly equal to the number of pixels in the cover image. Each character of secret information is represented in 8-bit binary values. If the average word length is considered to be 10 characters, then the smallest image with 717K pixels can embed somewhere around 8962 words.

Embedding Capacity(in bits) = Number of pixels

- 1 pixel = 1 bit
- 1 character = 8 bits
- 8 pixels = 1 character

The embedding capacity achieved from the current LSB implementation is almost inexhaustible for basic messaging applications.

5.3 Imperceptibility Evaluation

The metrics chosen to evaluate the security of the implemented LSB method are peak signal-to-noise ratio (PSNR), mean square error (MSE), and the Structural Similarity Index (SSIM) [4, 21]. The execution times for the embedding and extraction methods for the LSB implementation are also measured to test efficiency.

5.3.1 Mean Squared Error (MSE). MSE is used to find the squared difference between the pixels of the original image and the stego image.

$$MSE = \frac{1}{N} \sum_{i=1}^N (I[i] - K[i])^2 \tag{1}$$

where:

- N - Total number of pixels
- $I[i]$ - Intensity of the original pixel
- $K[i]$ - Intensity of the corresponding pixel in the stego image

5.3.2 Peak signal-to-noise ratio (PSNR). PSNR is a measurement of the image quality.

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

where:

- MAX - Maximum possible pixel intensity (e.g., 255 for an 8-bit image)
- MSE - Mean Squared Error

5.3.3 Structural Information Change (SSIM). SSIM is a method that is used to determine the likeness between two images. SSIM index's range is [-1,1].

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{3}$$

where:

- x and y - the cover image and the stego image
- μ_x and μ_y - the average luminance values of x and y
- σ_x and σ_y - the standard deviations of x and y
- σ_{xy} - the covariance of x and y
- C_1 and C_2 - constants to stabilize the division with weak denominator

The measurements for these performance metrics and execution times were evaluated for images of various sizes. The evaluation was performed in the following steps:

- A random 9400-bit message was embedded into each image using the LSB implementation.
- After the embedding process, the measurements for PSNR and MSE between the cover image and stego image to find the impact of the LSB implementation on the image quality.
- The SSIM index between the cover and stego image was calculated for each of the test images.

Table 5. reflects the results of the evaluation for a set of images with varying sizes. The images provided are real-time stego images produced after the embedding process.







Image	Pixels	MSE	PSNR (dB)	SSIM	Embedding time (ms)	Extraction time (ms)
	717.660K	0.0064	70.0024	0.99	176	29
	768K	0.0060	70.3379	0.989	170	35
	2073.6K	0.0022	74.5993	0.988	291	36
	4096K	0.0011	77.5195	0.989	444	58
	6193.95K	7.5347E-4	79.3601	0.986	712	34
	9981.8K	5.0265E-4	81.1180	0.991	637	29

Table 5. Results for a random 9400-bit message in different images

These results can conclude that the image distortion caused by the embedding process is not very significant as all the observed values point towards high image quality, and indicate high similarity between the cover and stego images.

5.4 Histogram Analysis

Histogram analysis is a method used to evaluate the security of image steganography techniques very often. The LSB implementation was evaluated by visualizing the histograms of the cover and stego images. The image used for this evaluation is Fig. 6 with a size of 717K pixels. The visual inspection shows these histograms to be completely identical as no visible anomalies were introduced after embedding the message into the cover image. The research performed a statistical analysis of these histograms to quantify their differences.

The histograms for the cover image and the stego images embedding texts with variable lengths are provided below.

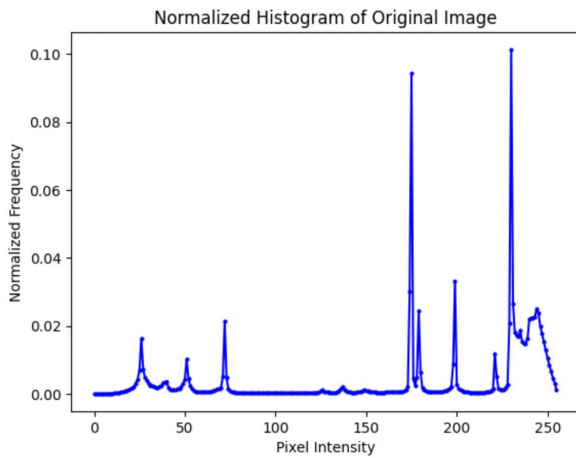


Fig. 7. Histogram for Cover Image

Calculations from the histogram in Fig. 7.

- Mean Intensity: 0.0039062495
- Median Intensity: 0.00063888193
- Standard Deviation Intensity: 0.010335826
- Skewness Intensity: 6.363142731473249
- Kurtosis Intensity: 51.643895749497325

Calculations from the histogram in Fig. 8.

- Mean Intensity: 0.00390625
- Median Intensity: 0.0006374885
- Standard Deviation Intensity: 0.010332786
- Skewness Intensity: 6.358829191188631
- Kurtosis Intensity: 51.57916871799228

Calculations from the histogram in Fig. 9.

- Mean Intensity: 0.00390625
- Median Intensity: 0.0006374885
- Standard Deviation Intensity: 0.010329557
- Skewness Intensity: 6.354418763655995
- Kurtosis Intensity: 51.513740379082485

The closeness of these statistical measures suggested that the embedding process did not significantly alter the overall distribution and characteristics of pixel intensities compared to the original cover image. It can be concluded that the LSB implementation is

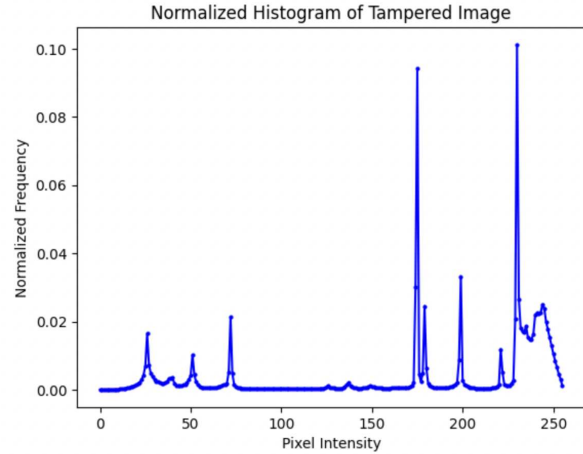


Fig. 8. Histogram for Stego Image hiding 9408-bit text

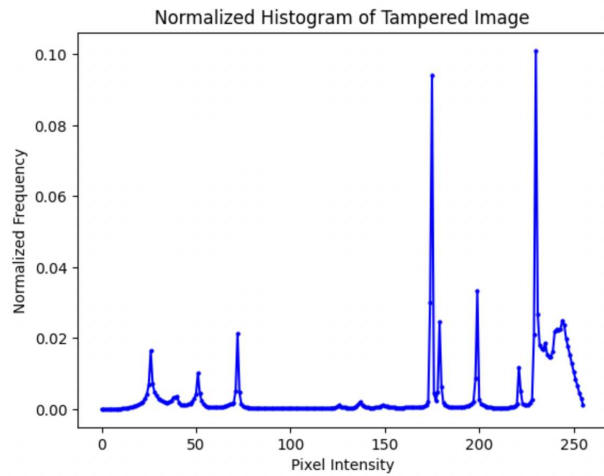


Fig. 9. Histogram for Stego Image hiding 18816-bit text

secure against histogram analysis as these measures do not provide significant evidence to prove the presence of hidden data.

5.5 UI Overview

- Fig. 10 shows the initial page of the application which is the login page, the user can either choose to login or signup. Once logged in, the user is navigated to the chat page displayed in Fig. 11.
- Fig. 12 shows the case where a user has sent a message. The red box displays the sent message 'Hey' preceded by the recipient named 'User'.
- In Fig. 13, the case where of receiving a message is displayed. The stego image received from the sender is displayed at the top of the screen. The yellow box displays the encrypted message when the user presses the 'EXTRACT' button after

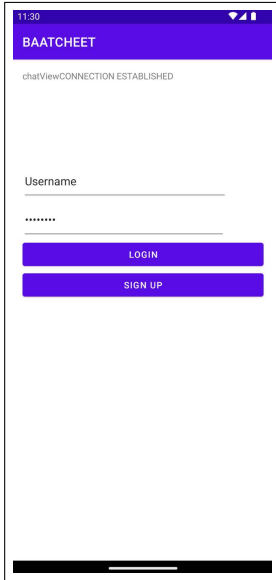


Fig. 10. Login Page

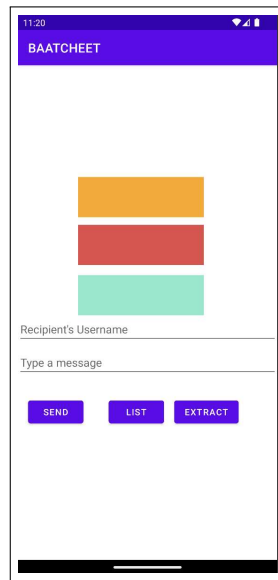


Fig. 11. Chat Page

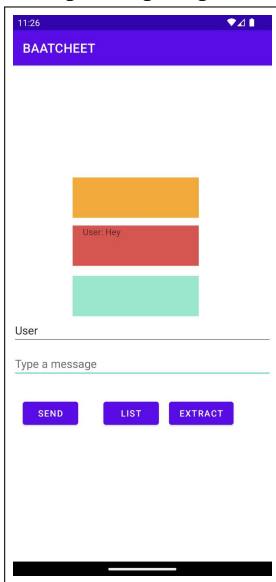


Fig. 12. Message Sent

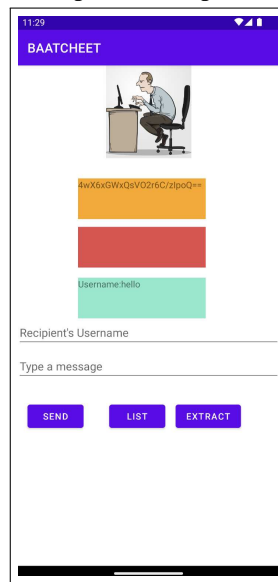


Fig. 13. Message Receiving

the stego image is displayed. The blue box displays the message 'hello' as decryption preceded by the name of the sender 'Username'.

- What if the stego image is compromised?**
 In case the presence of a stego image is realized, the LSB optimization offers randomness as it only chooses to substitute bits from red color channels. When it comes to color channels, the blue color channel offers the highest embedding capacity by causing the least image distortion. The red color channel

provides the second highest embedding capacity, also causing less distortion. The pixels are also selected randomly and with a limit based on message lengths. This process offers randomness.

- What if the LSB algorithm is known?**
 The purpose behind the addition of LSB substitution into the proposed system is security enhancement with an additional security layer over End-to-End encryption. However, the robustness of E2EE cannot be undermined as it is the most secure and acceptable approach for secure communication in the current age. Therefore, the attacker can only extract the encrypted messages if the LSB algorithm is known. They would still need to break the encryption to gain access to the secret information.

6 CONCLUSION AND FUTURE WORK

The research studied the architecture of E2EE protocols developed by popular messaging applications to find the existing cryptographic algorithms used to provide End-to-end encryption and developed an E2EE chat application that can be acceptable alongside other modern-day applications. Moreover, the research explored security enhancements using LSB substitution for image steganography with the primary goal of hiding the presence of encrypted data while prioritizing efficient communication to ensure potential real-world applications of the proposed solution. The research concluded with insights into the incorporation of image steganography techniques into End-to-End applications and also evaluated its effects on the efficiency of communication. Furthermore, the research evaluated the embedding capacity, imperceptibility, and security of the implemented image steganography technique.

The future extensions of this application can look to incorporate Session Perfect Forward Secrecy to ensure the confidentiality of past communication sessions. The LSB implementation can also be optimized by experimenting with different pixel selection techniques for embedding to increase the robustness and randomness of the algorithm.

REFERENCES

- (1) Jan, A., Parah, S. A., Hussan, M., & Malik, B. A. (2021). Double layer security using crypto-stego techniques: a comprehensive review. *Health and Technology*, 12(1), 9–31. <https://doi.org/10.1007/s12553-021-00602-1>
- (2) Varghese, F., Sasikala, P. (2023). Secure data transmission using optimized cryptography and steganography using Syndrome-Trellis Coding. *Wireless Personal Communications*, 130(1), 551–578. <https://doi.org/10.1007/s11277-023-10298-3>
- (3) Ambika, Virupakshappa, Veerashetty, S. (2022). Secure communication over wireless sensor network using image steganography with generative adversarial networks. *Measurement: Sensors*, 24, 100452. <https://doi.org/10.1016/j.measen.2022.100452>
- (4) A. M. Ahmed and A. S. Nori, "Improve Security Using Steganography and Cryptography Based on Smartphone Users Locations," 2022 Second International Conference on Advances in

- Electrical, Computing, Communication and Sustainable Technologies (ICAECT), Bhilai, India, 2022, pp. 1-7, <https://doi.org/10.1109/ICAECT54875.2022.9808046>.
- (5) G, T. (2022b, March 8). *10 Most Secure Messaging Apps – The Best Platforms & Solutions*. <https://getstream.io/blog/most-secure-messaging-apps/>
 - (6) Florence, E. (2022, January 7). *Best secure and encrypted messaging apps | SecurityTech*. SecurityTech. <https://securitytech.org/secure-encrypted-messaging-app/best/>
 - (7) *MTProto Mobile Protocol*. (n.d.). <https://core.telegram.org/mtproto>
 - (8) Florence, E. (2021, September 30). *Wire Messenger Review (The Good and the Bad)*. SecurityTech. <https://securitytech.org/secure-encrypted-messaging-app/wire/>
 - (9) *Documentation*. (n.d.). Signal Messenger. <https://signal.org/docs/>
 - (10) Ermoshina, K., Musiani, F., & Halpin, H. (2016). End-to-End Encrypted Messaging Protocols: An Overview. In *Lecture Notes in Computer Science* (pp. 244–254). https://doi.org/10.1007/978-3-319-45982-0_22
 - (11) Sutikno, T., Handayani, L., Stiawan, D., Riyadi, M.A., & Subroto, I. (2016). WhatsApp, Viber, and Telegram: which is the Best for Instant Messaging?
 - (12) Morkel, T., Eloff, J. H., & Olivier, M. S. (2005, June). An overview of image steganography. In *ISSA* (Vol. 1, No. 2, pp. 1-11).
 - (13) Mandal, P. C., Mukherjee, I., Paul, G., & Chatterji, B. N. (2022). Digital image steganography: A literature survey. *Information Sciences*, 609, 1451–1488. <https://doi.org/10.1016/j.ins.2022.07.120>
 - (14) *Image Steganography using Least Significant Bit (LSB) - A Systematic Literature Review*. (2022, January 25). IEEE Conference Publication | IEEE Xplore. <https://ieeexplore.ieee.org/document/9711628>
 - (15) Kolla, A. (2020, June 11). *List of 10 best steganography tools to hide Data*. Geek Dashboard. <https://www.geekdashboard.com/best-steganography-tools/>
 - (16) Alwen, J., Coretti, S., & Dodis, Y. (2019). The double ratchet: security notions, proofs, and modularization for the signal protocol. In *Lecture Notes in Computer Science* (pp. 129–158). https://doi.org/10.1007/978-3-030-17653-2_5
 - (17) *An overview and analysis of hybrid encryption: the combination of symmetric encryption and asymmetric encryption*. (2021, January 1). IEEE Conference Publication | IEEE Xplore. <https://ieeexplore.ieee.org/abstract/document/9463286>
 - (18) Avoine, G., Canard, S., & Ferreira, L. (2020). Symmetric-Key Authenticated Key Exchange (SAKE) with Perfect Forward Secrecy. In *Lecture Notes in Computer Science* (pp. 199–224). https://doi.org/10.1007/978-3-030-40186-3_10
 - (19) Arneson, T. (2022, January 7). *Threema Messenger Review (Everything You Need to Know)*. SecurityTech. <https://securitytech.org/secure-encrypted-messaging-app/threema/>
 - (20) Bucerzan, D., Rațiu, C., & Manolescu, M. (2013b). SmartSteg: a new Android based steganography application. *International Journal of Computers Communications & Control*, 8(5), 681. <https://doi.org/10.15837/ijccc.2013.5.642/>
 - (21) AbdelRaouf, A. (2021). A new data hiding approach for image steganography based on visual color sensitivity. *Multimedia Tools and Applications*, 80(15), 23393–23417. <https://doi.org/10.1007/s11042-020-10224-w>
 - (22) Nan Li, "Research on Diffie-Hellman key exchange protocol," 2010 2nd International Conference on Computer Engineering and Technology, Chengdu, China, 2010, pp. V4-634-V4-637, <https://doi.org/10.1109/ICCET.2010.5485276>.
 - (23) Implementation of Low Bit Coding Algorithm and Cipher Block with Electronic Code Book Mode for Data Legality in Audio Steganographic Streaming. (2018, October 1). IEEE Conference Publication | IEEE Xplore. <https://ieeexplore.ieee.org/abstract/document/8695921>
 - (24) Preneel, B. (2006). Modes of operation of a block cipher. In Springer eBooks (pp. 386–391). https://doi.org/10.1007/0-387-23483-7_258
 - (25) McGrew, D., Viega, J. (2004). The security and performance of the Galois/Counter Mode (GCM) of operation. In *Lecture Notes in Computer Science* (pp. 343–355). https://doi.org/10.1007/978-3-540-30556-9_27
 - (26) Hashimoto, K., Katsumata, S., Kwiatkowski, K., Prest, T. (2022). An efficient and generic construction for Signal's handshake (X3DH): post-quantum, state leakage secure, and deniable. *Journal of Cryptology*, 35(3). <https://doi.org/10.1007/s00145-022-09427-1>
 - (27) Bernstein, D. J. (2006). Curve25519: New Diffie-Hellman speed records. In *Lecture Notes in Computer Science* (pp. 207–228). https://doi.org/10.1007/11745853_14
 - (28) Implementation of HMAC-SHA256 algorithm for hybrid routing protocols in MANETs. (2015, January 1). IEEE Conference Publication | IEEE Xplore. <https://ieeexplore.ieee.org/document/7060558>
 - (29) BaatCheet client | GitHub repository. <https://github.com/ujjwaldodeja/BaatCheet-Basic.git>
 - (30) BaatCheet server | Github repository. <https://github.com/ujjwaldodeja/BaatCheet-Server.git>
 - (31) IEEE Xplore Full-Text PDF: (n.d.). <https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9335027>
 - (32) Murthy, N. K., Sharma, S., Priyadarsini, M. J. P., Ranjan, R., Sarkar, S., Basha, N. S. (2021). Image steganography using discrete cosine transform algorithm for medical images. In *Lecture notes in electrical engineering* (pp. 2349–2358). https://doi.org/10.1007/978-981-15-8221-9_219