National College of Ireland

A Secure Web Application for Data Storage and Retrieval using Multi-Level Protection

MSc Research Project
MSc Cybersecurity TopUp (MSCCYBETOP)

Rajni Priya
Student ID: x24241946

School of Computing
National College of Ireland

Supervisor: Mark Monaghan

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Rajni Priya		
Student ID:	x24241946		
Programme:	MSc Cybersecurity TopUp (MSCCYBETOP)	**Year:**	2024-25
Module:	MSc (Research) Practicum		
Supervisor:	Mark Monaghan		
Submission Due Date:	31-07-2025		
Project Title:	A Secure Web Application for Data Storage and Retrieval Using Multi-Level Protection		
Word Count:	7874	**Page Count:**	18

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Rajni Priya

Date: 31-07-2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

A Secure Web Application for Data Storage and Retrieval Using Multi-Level Protection

[Cryptography]

Rajni Priya
x24241946

Abstract

Providing security to data is very important in today's age were the instances of cyber attack and data breaches are increasing rapidly. There many techniques that provide security to data. However, combining different methods for data security can enhance the security of data that is transferred over a network or stored in database. In the study proposed here a multi-level data security system is proposed. The system proposed in the study combines data security techniques like cryptography, steganography, and differential privacy. The Advanced Encryption Standard(AES), and RSA are the cryptographic algorithms used in the study. The Least Significant Bit(LSB) method was used for performing steganography. Differential privacy was also added to the data before the data is stored. The system proposed in the study was implemented as a prototype web app which receives data in the form of text as the input. The prototype web app received the text to be secured and image as input. The prototype web app was tested and it was found that all the functionalities of a multi-level data security system was implemented successfully, as the prototype web app was able to securely store data, and makes it accessible to the authorized users, when they need the data. The results also showed that the RSA and AES algorithms were able to encrypt the data in an effective manner.

1 Introduction

1.1 Background

Data security mechanisms have the objective to keep data secure from any unauthorized access or damage, respectively while keeping its usability for legitimate proposes. This layer of security to the data being transferred over digital networks or over the internet. It was found that cyber criminals stole 68 records every second, which shows that the occurrences of cyber attacks are very frequent(Vojinovic, 2023). Every data breach is costly and the average cost of a data breach is 3.92 million dollars. It projected that in 2025 the global cost of cyber crime is expected to reach 10.5 trillion dollars(Bonnie, 2024). These statistics show the significance of data security in digital networks. The data transferred over networks in the communication between digital systems can be secured in a number of ways. Three of the main techniques that have been used for securing data that is transferred over networks is, Cryptography, steganography, and encryption. Cryptography is the foundation of information security, encryption is the process of making message in an unreadable format through mathematical formulas (Archana and Niranjana, 2023). It makes sure that an intercepted data is also incoherent without the correct key. Steganography is a complementary technique to cryptography but instead of trying to secure the privacy of the message, the idea is to conceal the fact that the message exists at all (in a communication channel) (Dutta and Chakraborty, 2020). In contrast to encryption, which is designed to make the content of a message

unreadable, steganography has this higher layer of secrecy that is about making the existence of a message known just as much as its data content.
Differential privacy promises a solution to an important problem in privacy of data: how to keep the data of an individual private when the data are part of a large dataset (Jiang *et al.*, 2022). As there is a rising demand for data analytics for information-based decision-making, the sharing or storage of datasets may lead to the disclosure of sensitive personal information. Differential privacy adds calibrated noise to the data or query response such that the presence or absence of an individual's data records has limited impact on the response. It comes with a mathematical proof of privacy, so is suitable for use in applications that respect the data in a sensitive manner—e.g., healthcare data or finance. However, the use of these methods individually cannot ensure security to the data being transferred over networks or saved in a database. For example, encrypted data can be the target of attack if the keys needed to retrieve the information are subjected to attack and file-based steganography can fail to work if the carriers are analysed for patterns used to hide information (Du *et al.*, 2021 ; Mohamed *et al.*, 2021). So, multiple methods that can secure data needs to be combined for enhancing the security of data.
A system that combines multiple levels of security can be used for securing data by enterprises from various industries -- healthcare, finance, and government, etc., can leverage a solid foundation to safeguard their sensitive data and meet privacy regulations. For instance, hospitals can store and share patients records securely, financial institutions can secure transaction information etc.
In the study proposed here, a multi-level data security system is proposed. The system combines cryptography, steganography, and differential privacy for securing data. The system proposed here is implemented as a prototype web app. The data that is considered in the study is data in the form of text ie., text is uploaded to the prototype web app, so that it is stored in a secure manner. Files can also be uploaded to the prototype web app, here also the text data from the file is read and stored securely.

1.2 Aim

The aim of the study is to build a web app prototype that provides security to textual data using a multi-level security system that combines cryptography, steganography, and differential privacy.

1.3 Objectives

The objectives of this study are:

- Secure the textual data by combining RSA cryptography, LSB steganography, and differential privacy.
- Retrieve the data that was stored securely.
- Integrate the multi-level data security system with a prototype web app.

1.4 Research Question

Does the web app proposed in the study that combines techniques like cryptography, steganography and differential privacy show a good performance in securing the data?

1.5 Report Structure

The report begins with the 'Introduction' section. Followed by the 'Introduction' section, there is 'Literature Review', and it contains the details of the literature associated with the security of data using, cryptography, steganography, and differential privacy, and existing works related to multi-level data security using different techniques. The third section of the report is 'Methodology, and it contains the details of how the cryptographic algorithm, steganography, and differential privacy was implemented, to secure data. The fourth section is 'Design Specification'. The fifth section is 'Implementation', and it contains the details of the details of the prototype web app which is shows how the data is secured in the multi-level security system in an interactive manner. The sixth section is 'Evaluation', and it contains the details of the results of the study proposed here, ie., how successful the cryptographic algorithm, steganography, and differential privacy, methods were in securing data, and The seventh section is 'Conclusion and Discussion, and it contains the details of the objectives achieved in the study that was done, and the limitations of the study.

2 Literature Review

2.1 Data security using encryption

One of the most popular symmetric-key encryption standards is the Advanced Encryption Standard (AES). Launched in 2001 by the U.S(Mouha, 2021). AES has key sizes of 128, 192 and 256 bits according to the number of internal rounds (10, 12 or 14 respectively) (Muttaqin and Rahmadoni, 2020).

Data was secured by adding a modification to the AES in the study by (Al-Khafaji and Rahma, 2022). This study shows that AES operates on a substitution-permutation network structure which makes it effective in providing security against cryptanalysis techniques like linear and differential cryptanalysis. AES was used with 128-bit key length in the study by(Lu, 2023). In the study safety analysis was done to assess the AES's resistance against square attacks, differential cryptanalysis, and brute force attacks. The results of the study showed that the AES algorithm was able to provide security to data against a number of different kinds of cyber attacks. The studies by Al-Khafaji and Rahma(2022), and Lu(2023) showed that AES is effective in providing security to data against different kinds of cyber attacks. The insights from these studies show that the AES is an effective data encryption technique.

VLSI architecture was used for AES algorithm, emphasizing hardware optimization for high speed and low consumption of power in the study by (Saritha *et al.*, 2023). This study showed that AES is the standard for encrypting data that is recognised by organizations such as the National Institute of Standards and Technology (NIST). The AES algorithm was used for securing sensitive data generated in a tractor repair enterprise in the study by (Golovko and Tolochyn, 2022). It was shown in the study that AES was the US governments security standard. The findings from the studies by Saritha *et al.*(2023), and Golovko and Tolochyn, (2022) showed that AES is widely regarded as an effective method of encryption and providing security to data.

Three digital signature algorithms, the ECDSA, DSA, and RSA were used in blockchain technology in the study by (Xu, 2023). The study showed that the RSA algorithm is able to encrypt digital data effectively and provide digital signatures. An encrypted chat application was built using the RSA algorithm ensuring the secure messaging between users in the study by (Namassivaya *et al.*, 2023). This study shows that secure communication between a web

app and users can be done using RSA algorithm as the RSA algorithm is able to generate a public-private key pair which can be used for authenticating users. The studies by Xu(2023), and Namassivaya *et al.*(2023), showed that RSA is an effective technique for securing data and establishing communication between a web app and a user.
From the studies by Al-Khafaji and Rahma(2022), Lu(2023), Saritha *et al.*(2023), and Golovko and Tolochyn, (2022), showed that it was seen that seen that AES is an effective encryption algorithm that is finally used. Since , the study proposed here focuses on combining different data security techniques, including encryption a highly effective encryption algorithm was needed and from the studies it was seen that AES was highly effective, so it can be used in the study proposed here. Two encryption algorithms are needed, the AES can be used for encrypting the stego-image, however the text given as input need to be encrypted and a communication need to be maintained with the user has to be maintained as the user needs to retrieve data that was previously uploaded to the web app proposed in the study, so based on the findings from the studies by studies by Xu(2023), and Namassivaya *et al.*(2023), it was decided that RSA can be used in the proposed study.

2.2 Text Compression

Text compression is the process that can be used to minimize storage and transmission costs for text data by removing redundancy from the text or by encoding it more efficiently(Syahputra, 2020). Both lossy and lossless text compression are possible; lossy text compression sacrifices some information, but achieves a much higher compression ratio (Hung and Platoš, 2023). As this study focused on the security of text, it was the most important aspect, as the information in the text cannot be lost. So, the lossless compression method was used in the study proposed here.
The text compression helped in reducing the size of the data that is hidden in the cover or stego-image (Al-Okaily and Tbakhi, 2020). This is significant as the pixel count of an image limits its embedding capacity (Kumar, Kumar and Jung, 2022). So, if the text is smaller in size then data can be hidden in the image in a manner that the visual quality of the image is affected.

2.3 Data Security using Steganography

The Least Significant Bit (LSB) technique was combined with the secret key and the XOR operator for performing image steganography in the study by (Jebur *et al.*, 2023). This study shows that the LSB technique is simple and it can be used for effectively securing data. A web-based LSB image steganographic technique that used two layers of AES encryption to transmit secret messages was used in the study by (Baldha, 2023). The insights from this study shows that the AES algorithm can be used along with the LSB steganographic technique for securing data in an effective manner. From the insights gained from the studies by Jebur *et al.*(2023), and Baldha(2023) it can be seen that LSB can be used for securing the data effectively and it can be used along with the AES.
The study proposed here needs a multi-level security system that combines different data security techniques like steganography, and cryptography. From the findings by Jebur *et al.*(2023), and Baldha(2023), it can be seen that LSB techniques simple and it can also be smoothly used along with the AES. It has been decided that AES will be used in the study and as multi-level security is needed the use of a simple technique for steganography benefits the system, so the LSB technique is used for steganography in the study proposed here.

2.4 Data Security using differential privacy

Differential privacy introduced by Dwork et al. (2006), is a mathematical method to preserve privacy of the individuals in the data, while maintaining sufficiently large data to support statistical analysis of the aggregates (Guo and Barrientos, 2022). It guarantees that there is no way to include or exclude a personal record in a query that would change the query's results in any significant way, making re-identification quite difficult. This is done by perturbing the result of a query or the content of a data point with carefully controlled noise, most often sampled from a Laplace or Gaussian distribution.

Studies have shown that differential privacy is used for adding security to data in a number of domains. Differential privacy has been used in a Privacy-preserving machine learning (PPML) when handling sensitive datasets like medical records for protecting individual data points (Ganadily and Xia, 2024). Differential privacy has been used in providing security to personal information in large-scale systems, like Google's Chrome browser telemetry. The usage of differential privacy in these large-scale systems, and popular systems show that it is effective in providing security to data.

The study by Wagh et al. (2020), has shown that differential privacy can be combined with cryptography techniques for securing data. This shows that differential privacy can be used along with other techniques for securing data. However, there was no existing study that combined the differential privacy, cryptography, and steganography techniques for securing data.

2.5 Multi-level Data Security

Differential privacy and cryptography techniques were combined to enhance data security in the study by (Wagh *et al.*, 2021). The cryptography technique is used along with the Local Differential Privacy (LDP) in this study. The results of the study showed that the integration of cryptography and differential privacy techniques enhanced the security that can be given to data. A combination of steganography and cryptography was used for data security in the study by (Elden *et al.*, 2021). The AES and adaptive pixel value differencing (PVD) were used in the study to provide security to the data. The data considered in the study was colour images that were in BMP format. Double steganography was used in the study. The secret data was embedded in a mask image using adaptive PVD, the data was then encrypted using AES. The results of the study show that the image quality of the mask image was maintained and the data is secured in an effective manner. The findings from Wagh *et al.*(2021) showed that differential privacy can be used for securing data, effectively. The findings from Elden *et al.*, (2021), and Wagh *et al.*(2021), showed that different techniques for securing can be combined to enhance data security. However, these studies did not combine three data security techniques like cryptography, steganography , and differential privacy.

An Efficient Algorithm for Secure Transmission (EAST) was used for securing the communication between vehicles in Internet of Vehicles (IoV) in the study by (Rathore *et al.*, 2022). Image data is considered in the study. The EAST algorithm combines both encryption and steganography methods. The results of the study shows that the EAST is effective in securing data in an efficient manner. A hybrid multistage data encryption framework that combined steganography and cryptography techniques were used for securing data in the study by (Osman *et al.*, 2022). Image data is considered in the study. The sequential and pseudo-random encoding/decoding algorithms were used in the study along with a pre-stage text encryption using the One-Time Pad (OTP). The results of the study showed that the hybrid steganography and cryptography approach was more time-effective and efficient. The steganography and cryptography techniques did not affect the quality of the data.

Image data transferred over wireless networks in IoT environments was secured by using a dual security approach that involved steganography, and cryptography in the study by (M, P and G, 2021). This study shows that steganography, and cryptography can be combined to secure image data transferred over an IoT network. A Dual Level Security Scheme (DLSS) that combined steganography and cryptography was used to provide double level security to data in the study by (Chithra and Aparna, 2023). Audio steganography was used in the study and the image that was encrypted was used for hiding the audio signal. An adaptive stego key LSB substitution framework that integrates several techniques, like a Bit Inverting Map (BIM) approach combined with a Bit Swapping Function (BSF) to find how the secret bits are hidden into the LSB layers of the cover image, random pixel selection based on the Henon Map Function for unpredictable embedding, and Data Encryption Standard(DES) encryption for securing the secret data, in the study by(Alia, Al-Tamimib, and Ahmed, 2020). Images from the USC-SIPI image database was used as the cover images for embedding the secret data. This study shows that steganography using the LSB method can be combined with a cryptographic method.
The studies by Rathore *et al.*(2022), Osman *et al.*(2022), M, P and G(2021), Alia, Al-Tamimib, and Ahmed, (2020) and Chithra and Aparna(2023), showed that encryption and steganography methods can be combined in an effective manner for securing data. However, none of these studies combined the steganography and cryptography, with another level of security like differential privacy.

2.6 Research Niche

From the existing studies it was learned that multiple data security techniques can be used for securing data. It was also found from existing studies that methods like AES, RSA,LSB steganography, and differential privacy are effective techniques for securing data. It was also seen multi-level data security methods used either cryptography-steganography or Differential privacy-cryptography. However, there were no studies that combined the cryptography, steganography , and differential privacy for providing multi-level security to data.
In the study performed here a system that secures data using multi-level security that combines techniques like the cryptography, steganography , and differential privacy, is proposed. The methods like AES, RSA, LSB steganography, and differential privacy are combined in the system proposed in the study.

3 Methodology

3.1 Tools Used

Python was used in the study for implementing the AES, RSA, LSB, data compression, and differential privacy techniques. Python was chosen as the programming language for implementing the multi-level data security system as it offered libraries that can be used for implementing AES, RSA, LSB, data compression, and differential privacy techniques, smoothly (Lobo-Vesga, Russo and Gaboardi, 2021). Python offered the implementation of differential privacy via its libraries, and this could not be offered by programming languages like Java or C++, as differential privacy could only be implemented in these languages by extensively customising the libraries available in these languages (Prediger *et al.*, 2022 ; Yousefpour *et al.*, 2021).

The Django framework was used in the study for implementing the web app prototype. Django was used in the study because it offered faster development than other Python based web frameworks, and quick development was needed in the study as only a prototype was needed (Kumar *et al.*, 2023). The web app prototype was designed using HTML, CSS, and Javascript, as these were popular, and effective methods used for designing web apps (Celi-Párraga, Boné-Andrade and Olivero, 2023). The SQLite database was used as the database of the web app prototype as it was the default database of the Django framework, and as the database was needed only for storing the data, and demonstrating the storage of secure data (Shyam and Mukesh, 2020).

3.2 Data Storage

The multi-level data security system is designed in a manner that data in the form of text uploaded to the system was encrypted, added to the image uploaded along with the text to the system, the image was encrypted, and differential privacy was added. After this the data was stored in the database of the multi-level data security system. Here, the different steps involved in securing the data before storing the data in the database is defined.

Encryption using RSA

The RSA algorithm was implemented for encrypting the textual data given as input to the multi-level data security system. The RSA key-pair ie., the private key and public key corresponding to the RSA algorithm was generated. This was done because these keys were needed for encrypting and decrypting the data that was uploaded to the multi-level data security system. The key length was set to 2048, this key length was chosen because it created a balance between performance and security, as recommended by NIST (Gulen and Baktir, 2023 ; RSA — PyCryptodome 3.23.0 documentation, 2025). A public key and private key was generated.

The encryption of text was done using the public key that was generated. The Optimal Asymmetric Encryption Padding(PKCS1_OAEP) was used in the study for encrypting the code. Before encryption the text given as input to the multi-level data security system was encoded using UTF-8. The UTF-8 encoding technique was done in this study because it supported all Unicode characters, including text from different languages, symbols, and special characters (Keiser and Lemire, 2020). The use of PKCS1_OAEP for encrypting the text meant that the data to be encrypted had to be in the form of bytes, so the text data, given as input, had to be converted to bytes, and this was done using the UTF-8 encoding that transforms the text string into a byte sequence (Jodayree, He and Janicki, 2023). The OAEP technique was used for encrypting the data because it added security by adding randomness, reducing ciphertext attacks (Ebrahimi, 2022). After encryption the data was again encoded using base64, which was done to ensure that the data was compatible for the steganography method as the LSB technique was used in the study, and LSB needed data in the form text (R and Marlina, 2022 ; Alher, Imran and Ali, 2024).

A text undergoing RSA encryption is shown in figure(1).

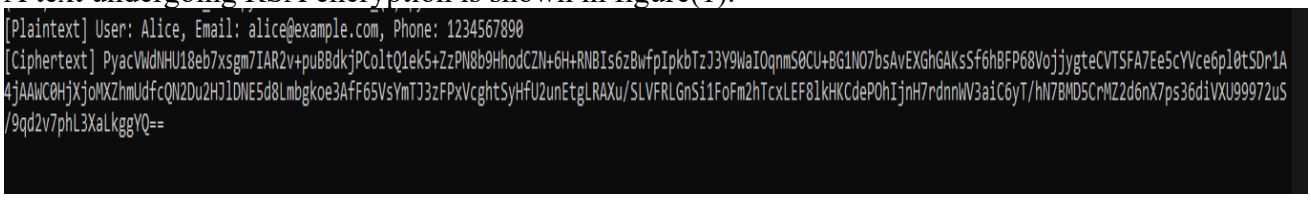


Figure (1): RSA encrypted text

Text Compression

The image or the stego-image given as input, along with the textual data, was used in this step. It was ensured that the encrypted text was in the form of bytes, and this was done by encoding the textual data using UTF-8. If the text was already in byte form then this encoding was not done. The number of bytes associated with text was found. The number of bytes helped in finding the dimensions of the square image that is able to hold all the data. A square shape made sure that optimal space was utilized and the reshaping operations were simplified.

The number of bytes is used to determine the dimensions of a square image that can hold all the data (Jebur *et al.*, 2023). A square shape ensures optimal space utilization and simplifies reshaping operations. Now for creating the image, it was checked if the total number of bytes was exactly equal to the pixel count of a square image, and if it was equal no extra operation was needed. However, if the total number of bytes was not exactly equal to the pixel count of a square image, then , zero-padding was added to fill the empty space that was remaining. This padding made sure that data can be reshaped into a perfect square without any loss to the data. The data in the form of bytes was then reshaped into a 2D NumPy array, that represented pixel values of a grayscale image. This conversion was done as the grayscale format is optimal for storing binary data and can be manipulated easily for doing steganography (Song, Song and Qu, 2021 ; Li *et al.*, 2020).

The generated image was saved in PNG format into an in-memory buffer and was then returned as data in the binary form. The PNG format was chosen as it was a lossless format — important for preserving exact pixel values that contain the message that is hidden. This made sure that data was not lost or modified during compression or storage, maintaining the quality of the information that is embedded in the image.

The image is saved in PNG format into an in-memory buffer and then returned as binary data. PNG is chosen because it is a lossless format — essential to preserve exact pixel values which contain the hidden message(Bae *et al.*, 2022). This ensures that no data is lost or altered during compression or storage, maintaining the fidelity of the embedded information.

The compression of text is shown in figure(2).

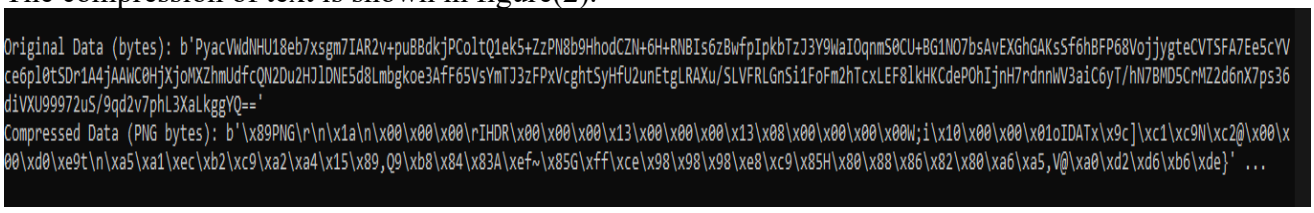


Figure (2): text compression

Steganography

In this study, steganography played a crucial role in concealing the encrypted and compressed text within an image, thereby introducing an additional layer of security by hiding the existence of the data itself. The Least Significant Bit (LSB) technique was adopted due to its simplicity, efficiency, and minimal impact on the visual quality of the image (Alqadi, 2022). The implementation begins with selecting a suitable image (referred to as the "mask image") which acts as the cover medium for hiding the data. Before embedding, the input text underwent encryption using RSA and compression to reduce its size, making it more suitable for embedding without noticeably altering the image. The data is then converted into a byte stream, and its total length is calculated. To ensure reliable extraction during the retrieval phase, the length of the data is stored within the first few bytes of the image. This enables accurate reconstruction of the original hidden content.

The byte stream, including the data length, was then embedded into the image by modifying the least significant bits of the image's pixel values. By using the LSB method, the alteration to each pixel was negligible and often imperceptible to the human eye. This ensured that the

visual quality of the image remained intact while effectively hiding the sensitive data. The final image, now carrying the hidden information, became the stego image, which was further secured using AES encryption and differential privacy techniques as part of the multi-level security architecture. This step of steganography was essential to the study as it transitions the focus from traditional encryption to secure communication, ensuring that even if the data is intercepted, its presence remains undetected, enhancing the overall security of the system.

AES Encryption

In the proposed multi-level data security system, AES encryption was employed to secure the image that carried the embedded and encrypted text. This step was vital to ensure that even if the stego image was intercepted, its content remained unintelligible to unauthorized users. The AES algorithm was implemented in Cipher Feedback (CFB) mode. CFB mode was chosen for its ability to handle data of arbitrary length without the need for padding, which aligned well with encrypting images where block alignment was not always guaranteed (Zhang *et al.*, 2023). This mode also introduced an element of chaining, making each ciphertext block dependent on all previous blocks, thereby enhancing security.

The encryption process began by generating a random 16-byte Initialization Vector (IV). This IV ensured that identical plaintext inputs produced different ciphertext outputs across different encryption sessions, thereby preventing pattern recognition and improving confidentiality (Xia, Li and Chen, 2022). The AES cipher was then initialized using the provided symmetric key along with the generated IV. The image data, which at this point contains the steganographically hidden text, was encrypted to produce a secure binary ciphertext. The IV was prepended to the ciphertext, as it was required during the decryption process to restore the original data.

The combined encrypted data (IV + ciphertext) was then base64 encoded for safer handling, storage, or transmission within the multi-level data security system. This encoding made the binary data compatible with systems that handle text-based formats and ensured consistent and reliable processing across different components of the system. AES encryption in this step adds another critical security layer, ensuring that even if the existence of the hidden text was discovered, the contents of the image remain inaccessible without the appropriate decryption key. This aligns with the study's goal of implementing a robust, multi-layered data protection framework.

Differential privacy

Differential privacy was integrated into the system as the final layer of the multi-level data security architecture to protect against potential inference attacks even after encryption and steganography have been applied. In this implementation, differential privacy was achieved by adding Laplace noise to the image data, specifically after the image had been encrypted using AES. The primary purpose of introducing Laplace noise was to mask any identifiable patterns or statistical traces in the encrypted image that could potentially be exploited by an attacker. Even if the encryption were somehow compromised, the presence of controlled noise would make it significantly harder to extract meaningful information from the image data.

The Laplace mechanism is a widely used approach in differential privacy due to its mathematical properties that guarantee a level of indistinguishability between individual data points (Shahmiri, Ling and Li, 2023). In this study, Laplace noise was generated based on a defined mean and scale (with the scale controlling the magnitude of the noise). The noise was then added to each pixel of the encrypted image. To maintain the integrity of the image format and avoid overflow or underflow, the resulting noisy data was clipped within the valid pixel value range of 0 to 255. This ensured that the image remains valid and can still be stored or processed by image-handling libraries without errors. By applying differential

privacy at this stage, the system ensured that even if an attacker gained access to the encrypted and steganographically modified image, any attempt to analyse or reverse-engineer its content will be significantly obstructed. This step aligned with the core objective of the study—to construct a robust, multi-layered data security system that preserves both the confidentiality and privacy of sensitive textual information. The noise added to the data using differential privacy is shown in figure(3).

Figure (3): Noise added to data

The encrypted image with the added image can now be stored in the database of the multi-level data security system. The final image or cover image that was obtained ie., that was stored in the database is shown in figure(4).

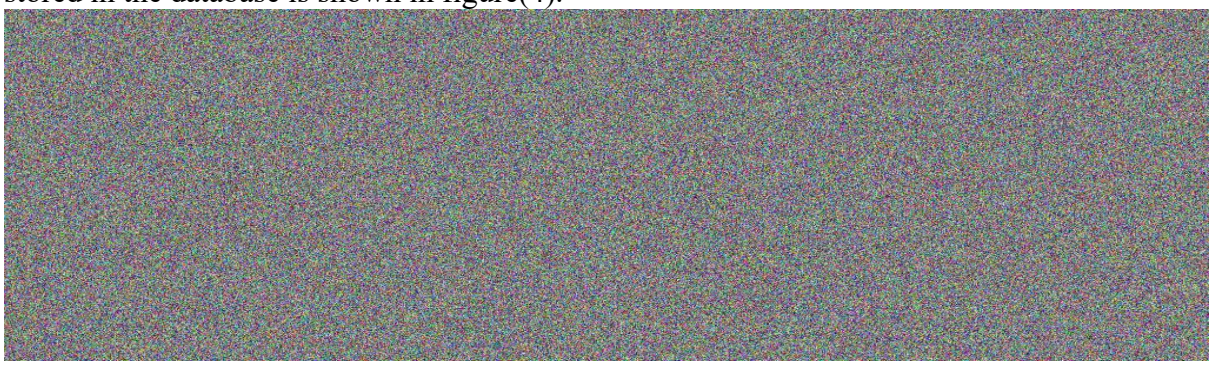


Figure (4): The final image

3.3 Data Retrieval

The data can be retrieved by reversing the process that were applied to the data when the data was being stored in the multi-level data security system. The different steps in retrieving the data are defined, here.

AES Decryption

The data retrieved from the database underwent AES decryption. The cover image containing the secure data was decrypted as it was previously encrypted. The first 16 bytes of the encrypted data, which was the IV was extracted. The encrypted data and the IV were separated. The AES cipher object was created and it was used for decrypting the encrypted data that was retrieved. Any extra padding added to the image was also removed, as this padding would prevent the data to be in its original form.

The Laplace noise and noisy data (the image with noise) are both explicitly cast into 16-bit signed integers (int16). This conversion is important because pixel values in images are typically represented using 8-bit unsigned integers (i.e., uint8) meaning the values will range from [0, 255] and cannot include negative numbers. As subtracting Laplace noise may result in negative intermediate quantities, adopting a signed data type avoids problems of underflow when performing subtraction. It is then necessary to remove the added noise by

subtracting the original Laplace noise from the noisy data. As the pixel values need to be between 0 to 255 to remain valid image data, the subtraction output is piped to ensure that the values are kept within this range ie., the data was clipped. Clipping is performed on the removed noise to make sure that pixel values lie in the range that can allow image processing.
Then the data was rescaled back to an 8-bit unsigned integer range, in order to be read and manipulated as a regular image. This step was necessary to obtain compatibility with methods of decryption (AES) and steganographic decoding (LSB) as 8-bit unsigned was a standard image format(T and A, 2022).

Retrieving Hidden Data
The image in the dataset was obtained as a multi-dimensional NumPy array. This was done since the hiding process in the system is LSB steganography and for retrieving the hidden data on an image, the precise pixel data must be obtained. The image was then transformed into a one-dimensional array. In LSB steganography, the hidden message is embedded byte by byte in the pixels of the image. By flattening the image into a single array, the pixel values can be read out linearly, enabling a simple extraction of data. For storing the length of the hidden data, the first four bytes of the image were reserved. This was done because if the length of the hidden message was unknown, it would be difficult to properly extract only the portion that was needed from the image data. This approach ensured proper extraction, and the retrieval of irrelevant data was avoided.
The byte sequence from the previous step was transformed into an integer using big-endian format for representing the length of the message that was hidden (Kápl and Parízek, 2020). The data was encoded and decoded with length as a 4-byte integer, this was done as it helped in accessing metadata needed for proper data recovery. From the decoded length the flattened array was sliced to obtain the proper number of bytes corresponding to the text data that was hidden, encrypted and compressed. The extracted portion was then transformed to a byte sequence. This operation properly isolated the hidden message from the image.

Decompressing the text
The data in compressed byte stream form was read as a PNG image as the text was represented a s a PNG image in the text compression phase. After this, the image was transformed into a NumPy array, that represented the image pixels in the form of a matrix. The pixel data was flattened into a one-dimensional array and then transformed back into raw bytes. The trailing null bytes in the data were stripped from the byte stream to remove any padding that was added during compression.

RSA Decryption
The private key of the RSA algorithm that was initially generated was loaded, and a cipher object was created using OAEP. The encrypted data was decoded from Base64 format to it's original byte format. Then, the data was decrypted using the OAEP object and the private key. The decrypted data was obtained and the data in the form of bytes(UTF-8-encoded) was converted back into a text string, which was the original input given to the multi-level data security system. So, finally the data was obtained in the form identical to the input given by the user.

3.4 Testing

The multi-level data security system was assessed to determine its performance. The performance metrics for assessing the encryption the performance were found, and the

functionalities of the multi-level data security system implemented as the web app prototype was tested. The metrics for assessing the encryption performance were:

- **Length of plaintext:** The number of bytes of the plaintext or text to be encrypted.
- **Length of Ciphertext:** The number of bytes of the plaintext after it was encrypted. If the length of the ciphertext was greater than the plaintext, then it can be considered that encryption was properly done.
- **Ciphertext Entropy:** Ciphertext entropy quantifies the unpredictability associated with the encrypted data, specifying the resilience it shows towards attacks(Zolfaghari, Bibak and Koshiba, 2022). Ciphertext entropy values close to 8 bits/byte can be considered an optimal value that shows good encryption (Rodríguez *et al.*, 2021).

The metrics length of plaintext, length of ciphertext, and Ciphertext entropy were used for assessing the performance of encryption in the study as these metrics provided clear numerical values that gave an idea about the encryption performances of different cryptographic algorithms.
The functionalities of the web app prototype was tested using unit testing. Unit testing was used in the study as the web app prototype had different functionalities and these needed to be tested. The functionalities of the web app prototype were individual components and thus unit testing was apt as the testing method (Tufano *et al.*, 2022).

4 Design Specification

Figure(5), shows the system architecture of the multi-level data security system that was proposed here. The RSA algorithm was chosen in this study compared to alternate methods like, Eliptic curve cryptography (ECC). ECC offers keys with smaller key sizes, however this algorithm was not supported by standard Python libraries, which was the reason RSA was chosen for this study instead of the ECC algorithm (Belyavsky *et al.*, 2020). The AES was used instead of algorithms like, the Blowfish algorithm, which had a small block size (64 bits), which made it insecure against modern attacks (Assa-Agyei and Olajide, 2023). The LSB technique was used instead of steganography methods like, Discrete Cosine Transform (DCT) which were not that effective in handling lossless images, so if lossless images were encountered then there were chances of issues arising (Yang and Liao, 2023). Spread spectrum methods provide better robustness but is more complex than the LSB, which was easier to implement compared to the spread spectrum methods (Zeeshan and Khan, 2022).

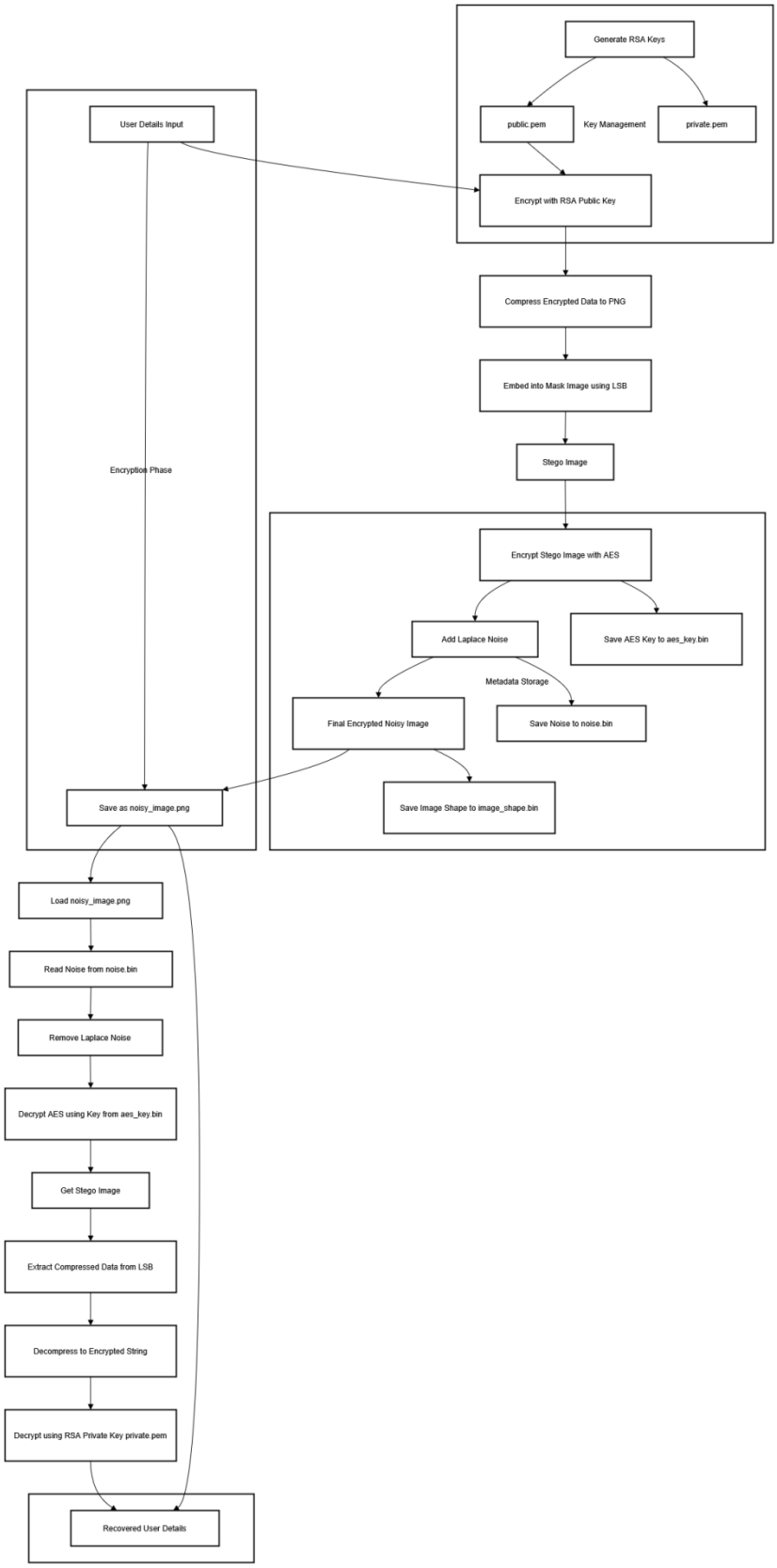


Figure (5): system architecture of the multi-level data security system

5 Implementation

5.1 Prototype Web App

A prototype web app was built in the study for demonstrating the working of the multi-level data security system in an interactive manner. This study has the aim of building a multi-level data security system, and in this system the data given as input(in text form), along with an image. The text given as input was considered as the data to be secured by the system, and the image was the cover image for performing steganography.
The web app prototype was designed in a way that it had a login and home page. The web page for securing the data in the web app prototype was designed in a way that it had an input area for receiving the text input. The web app prototype had an area that allowed the users to select an image from their PC or mobile phone and upload it to the web app. The web page had an option that allowed the user to get the public key generated for the text given as input. The web page had a button that was programmed to start the process of securing the data beginning with RSA encryption. The web page for retrieving the data from the database was designed in a manner that all the data stored in the SQLite database was displayed on a web page. The web page allowed the user to choose the data record that they wanted to download, and the web page had a button that was designed to retrieve the chosen data from the database after decryption, recovery of text, and the removal of differential privacy, so that the text was obtained in the form in which it was uploaded. The web page for securing data was shown in figure(6).

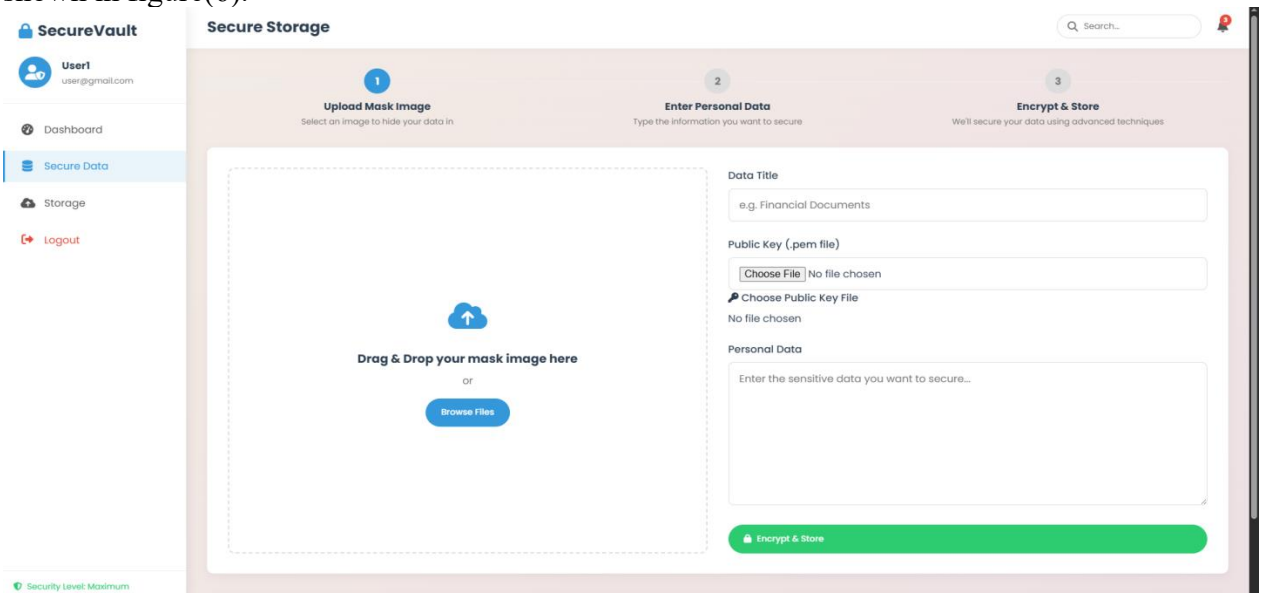


Figure (6): web page for securing data

6 Evaluation

6.1 Encryption Performance

Algorithm	Ciphertext Entropy
RSA	7.1898 bits/byte
AES	7.9999 bits/byte

Table (1): the ciphertext entropy for RSA and AES

From table(1), it can be seen that both the RSA and AES techniques show a good performance here as both algorithms were able to achieve good values for ciphertext entropy, which were close to the optimal value of ciphertext entropy which was 8bits/byte.
The encryption performances of the RSA and AES algorithms were assessed based on a dummy text value as shown in figure(7).

Figure (7): the plaintext and ciphertext lengths

From figure(7), it can be seen that a random text was used as the plain text for assessing the performances of AES, and RSA. From figure(7), it can be seen that the length of the ciphertexts for the AES and RSA were greater in length than the length of the plaintext, for the dummy text given as input to the algorithms. The results in figure(7), show that AES, and RSA were effective in the encryption of data as it was seen that length of the ciphertext was greater than the plaintexts in both cases.

6.2 Results of Unit testing

After building the web app prototype it was tested using Unit testing. The result of the Unit testing is shown in table(2). The data was stored in a SQLite database.

Operation	Test Case Description	Input	Expected Output	Actual Output	Pass/Fail
Registration	Register with valid details	Name, Email, Password	Account created, RSA keys sent via email	Account created, RSA keys sent via email	Pass
Registration	Register with duplicate email	Existing Email	Error: Email already registered	Error: Email already registered	Pass
Login	Login with correct credentials	Valid Email/Password	Redirect to dashboard	Redirect to dashboard	Pass
Login	Login with incorrect credentials	Invalid Email/Password	Show "Invalid credentials" message	Show "Invalid credentials" message	Pass
Upload	Upload mask image and personal details	Image, Details, RSA key	Data encrypted, processed, and stored in DB	Data encrypted, processed, and stored in DB	Pass
Upload	Upload without mask image	Personal Details only	Error: Image required	Error: Image required	Pass
RSA Encryption	Encrypt personal	Personal data + RSA Public	Encrypted text	Encrypted text	Pass

	details using RSA	Key			
Compression	Apply JPEG compression	Encrypted data, mask image	Compressed files	Compressed files	Pass
Steganography	Embed encrypted data in image	Compressed data + mask image	Stego-image generated	Stego-image generated	Pass
AES Encryption	Encrypt stego-image using AES	Stego-image + AES key	Encrypted image	Encrypted image	Pass
Differential Privacy	Add noise to AES-encrypted image	Encrypted image	Image with DP noise	Image with DP noise	Pass
Database	Store encrypted data in SQLite	Final image data	Data saved with metadata	Data saved with metadata	Pass
Retrieve	Retrieve with correct RSA key	Correct RSA Private Key	Decrypted personal data shown	Decrypted personal data shown	Pass
Retrieve	Retrieve with wrong RSA key	Incorrect Private Key	Error, alert email sent	Error, alert email sent	Pass
Email Alert	Unauthorized access attempt	Wrong key usage	Email sent to user	Email sent to user	Pass

Table (2): Results of Unit testing

From table(2), it can be seen that all the operations that were done in the web app prototype was implemented successfully, and the expected output was obtained for all operations. The working of all the operations of the web app prototype shows that the multi-layer data security system also works in an effective manner as the web app prototype was an implementation of the multi-layer data security system.

7 Conclusion and Discussion

7.1 Discussion

The aim of the study proposed here was to build a web app prototype that provides security to textual data using a multi-level security system that combines cryptography, steganography, and differential privacy. A system that offered multi-level security by combining cryptography, steganography, and differential privacy was built, and was implemented in an interactive manner, successfully as a web app prototype.
From the study of existing literature it was seen from the studies by Elden et al., (2021), Wagh et al.(2021), Rathore et al.(2022), Osman et al.(2022), M, P and G(2021), Alia, Al-Tamimib, and Ahmed, (2020) and Chithra and Aparna(2023), that different techniques for securing data, like cryptography, steganography, and differential privacy can be used together for securing data, even though there was no studies that used all of the cryptography, steganography, and differential privacy, techniques for securing data. These findings from existing studies were supported by the results of the study carried out here, as it was seen that the different techniques for securing data, like cryptography, steganography, and differential

privacy, can be combined effectively for securing data. The objectives of the study were achieved successfully.
The research question of the study was also answered successfully. The research question was:

- Does the web app proposed in the study that combines techniques like cryptography, steganography and differential privacy show a good performance in securing the data?

  From table(2), it was seen that the web app prototype that was built here was able to secure the data given as input to it in a secure manner. Finding the values of performance metrics like the ciphertext entropy also helped in understanding the effectiveness of the AES, and RSA encryption techniques that were used in the study that was done, here.

The implementation of the RSA, AES, text compression, differential privacy and LSB steganography were all difficult. However, as there was time for implementing the different techniques, each of the methods were implemented by over coming the difficulties. The main difficulties rose from the implementation of the RSA, AES, text compression, differential privacy and LSB steganography, which were all unfamiliar. The study helped in understanding and implementing the RSA, AES, text compression, differential privacy and LSB steganography, which were effective methods for providing security to the data. Understanding and implementing these techniques ensured that, these techniques can be used in the future for securing data.
The study that was done here had a limitation, and this was that only text data was considered in the study. There are other kinds of data like, images or files, however none of these kinds of data was used in the study. The study could have considered data in the form of images or files, etc,.

7.2 Conclusion

The study proposed here built a multi-level security system that combines cryptography, steganography, and differential privacy. The system was implemented successfully as a web app prototype. The cryptographic algorithms used in the study included, the RSA and AES. The steganography was done using LSB. Providing security to data in the form of text was done in the study that was done. The system was designed in a manner that it received text data, and a cover image as the input. The data was encrypted using RSA, and the encrypted text was compressed. The compressed and encrypted text was then added to the image given as input using the LSB technique. The text was added successfully into the image, and the image was encrypted using AES. Differential privacy was implemented and noise was added to the encrypted data. The data was then stored in the database.
A web app prototype was then built , and the multi-level data security system was integrated with the web app prototype. Since, the multi-level data security system was implemented as a web app prototype, the working of the web app prototype was tested using Unit testing. The results of Unit testing showed that all the functionalities of the web app prototype ie., the multi-level data security system were working properly.
Currently, only text data was considered in this study. In future studies other kind of data, like images or files can be considered for providing security using the multi-level data security system. In the future, other encryption techniques other than the RSA, like the DES algorithm can be used for encrypting the data received as input.

8 References

Alher, Z.A., Imran, B.M.A. and Ali, I.A. (2024) *LSB as a Steganography Tool in Information Security*, pp. 348–355. https://doi.org/10.24086/cocos2024/paper.1157.

Alia, A.S., Al-Tamimib, M.S.H. and Ahmed, A.(2020). Secure image steganography through multilevel security. *Image*, *11*(1). https://www.ijicc.net/images/vol11iss1/11111_Ali_2020_E_R.pdf

Al-Khafaji, N.B.J. and Rahma, N.A.M.S. (2022) 'Proposed new modification of AES algorithm for data security,' *Global Journal of Engineering and Technology Advances*, 12(3), pp. 117–122. https://doi.org/10.30574/gjeta.2022.12.3.0165.

Al-Okaily, A. and Tbakhi, A. (2020) 'A novel lossless encoding algorithm for data compression - genomics data as an exemplar,' *bioRxiv (Cold Spring Harbor Laboratory)* [Preprint]. https://doi.org/10.1101/2020.08.24.264366.

Alqadi, Z. (2022) 'Two PKs to protect LSB method of data steganography,' *International Journal of Computer Science and Mobile Computing*, 11(8), pp. 45–66. https://doi.org/10.47760/ijcsmc.2022.v11i08.004.

Archana, B.U. and Niranjana, V. (2023) 'Overview of cryptography,' *2*, 3(2), pp. 74–80. https://doi.org/10.46632/daai/3/2/15.

Assa-Agyei, K. and Olajide, F. (2023) 'A comparative study of twofish, blowfish, and advanced encryption standard for secured data transmission,' *International Journal of Advanced Computer Science and Applications*, 14(3). https://doi.org/10.14569/ijacsa.2023.0140344.

Bae, J. *et al.* (2022) 'L3: Accelerator-Friendly lossless image Format for High-Resolution, High-Throughput DNN training,' *arXiv (Cornell University)* [Preprint]. https://doi.org/10.48550/arxiv.2208.08711.

Baldha, N. (2023) 'A web-based least significant bit (LSB) image steganographic technique,' *Science Journal of Circuits Systems and Signal Processing* [Preprint]. https://doi.org/10.11648/j.cssp.20231101.12

Belyavsky, D. *et al.* (2020) 'Set it and forget it! Turnkey ECC for instant integration,' *Annual Computer Security Applications Conference*, pp. 760–771. https://doi.org/10.1145/3427228.3427291.

Bonnie, E. (2024) '110 of the latest data breach statistics [Updated 2024],' *Secureframe*, 3 December. https://secureframe.com/blog/data-breach-statistics. [Accessed on : 27 Jan. 2025]

Celi-Párraga, R.J., Boné-Andrade, M.F. and Olivero, A.P.M. (2023) *Programación Web del Frontend al Backend*. https://doi.org/10.55813/egaea.l.2022.18.

Chithra, P.L. and Aparna, R. (2023) 'Dual Level Security Scheme (DLSS) using RGB layer cryptography and audio steganography for secret image transmission in unsecure medium,' *Indian Journal of Science and Technology*, 16(23), pp. 1733–1744. https://doi.org/10.17485/ijst/v16i23.924.

Du, H. *et al.* (2021) 'Cryptographic Secrecy Analysis of Adaptive Steganographic Syndrome-Trellis codes,' *Security and Communication Networks*, 2021, pp. 1–16. https://doi.org/10.1155/2021/5495941.

Dutta, P.S. and Chakraborty, S. (2020) 'Image based Steganography in Cryptography implementing different Encryption-Decryption Algorithm,' *International Journal of Scientific Research in Computer Science Engineering and Information Technology*, pp. 745–748. https://doi.org/10.32628/cseit2063191.

Ebrahimi, E. (2022) 'Post-quantum security of plain OAEP transform,' in *Lecture notes in computer science*, pp. 34–51. https://doi.org/10.1007/978-3-030-97121-2_2.

Elden, M.S. *et al.* (2021) 'Optimizing data security by using Integration between double stegging by pixel value difference and advanced encryption standard,' *IOP Conference Series*

Materials Science and Engineering, 1051(1), p. 012024. https://doi.org/10.1088/1757-899x/1051/1/012024
Ganadily, N.A. and Xia, H.J. (2024) 'Privacy Preserving Machine Learning for Electronic Health Records using Federated Learning and Differential Privacy,' *arXiv (Cornell University)* [Preprint]. https://doi.org/10.48550/arxiv.2406.15962.
Golovko, G. and Tolochyn, M. (2022) 'USING THE AES ENCRYPTION METHOD IN PRACTICE,' *Системи Управління Навігації Та Зв'язку Збірник Наукових Праць*, 4(70), pp. 71–74. https://doi.org/10.26906/sunz.2022.4.071.
Gulen, U. and Baktir, S. (2023) 'Side-Channel resistant 2048-bit RSA implementation for wireless sensor networks and internet of things,' *IEEE Access*, 11, pp. 39531–39543. https://doi.org/10.1109/access.2023.3268642.
Guo, Q. and Barrientos, A.F. (2022) 'Differentially private methods for compositional data,' *arXiv (Cornell University)* [Preprint]. https://doi.org/10.48550/arxiv.2211.06337.
Hung, T.D. and Platoš, J. (2023) 'Mathematical preliminaries in the case of lossless compression Markov models,' *Saudi Journal of Engineering and Technology*, 8(05), pp. 98–102. https://doi.org/10.36348/sjet.2023.v08i05.003.
Jebur, N.S.A. *et al.* (2023) 'Hiding information in digital images using LSB steganography technique,' *International Journal of Interactive Mobile Technologies (iJIM)*, 17(07), pp. 167–178. https://doi.org/10.3991/ijim.v17i07.38737.
Jiang, H. *et al.* (2022) 'Differential Privacy in Privacy-Preserving Big Data and Learning: Challenge and opportunity,' in *Communications in computer and information science*, pp. 33–44. https://doi.org/10.1007/978-3-030-96057-5_3.
Jodayree, M., He, W. and Janicki, R. (2023) 'Preventing image data poisoning attacks in federated machine learning by an encrypted verification key,' *Procedia Computer Science*, 225, pp. 2723–2732. https://doi.org/10.1016/j.procs.2023.10.264.
Kápl, R. and Parízek, P. (2020) 'ENDICheck: Dynamic Analysis for Detecting endianness Bugs,' in *Lecture notes in computer science*, pp. 254–270. https://doi.org/10.1007/978-3-030-45237-7_15.
Keiser, J. and Lemire, D. (2020) 'Validating UTF-8 in less than one instruction per byte,' *Software Practice and Experience*, 51(5), pp. 950–964. https://doi.org/10.1002/spe.2920.
Küchler, N. *et al.* (2023) 'CoHERE: Managing differential privacy in large scale systems,' *arXiv (Cornell University)* [Preprint]. https://doi.org/10.48550/arxiv.2301.08517.
Kumar, R., Kumar, N. and Jung, K.-H. (2022) 'Reversible data hiding using an improved pixel value ordering and complementary strategy,' *Symmetry*, 14(12), p. 2477. https://doi.org/10.3390/sym14122477.
Kumar, V. *et al.* (2023) 'BUILDING a SECURE AND EFFICIENT "AUCTION SYSTEM USING PYTHON BASED DJANGO TECHNOLOGY",' *International Research Journal of Modernization in Engineering Technology and Science* [Preprint]. https://doi.org/10.56726/irjmets39510.
Li, Q. *et al.* (2020) 'A novel grayscale image steganography scheme based on chaos encryption and generative adversarial networks,' *IEEE Access*, 8, pp. 168166–168176. https://doi.org/10.1109/access.2020.3021103.
Lobo-Vesga, E., Russo, A. and Gaboardi, M. (2021) 'A Programming Language for Data Privacy with Accuracy Estimations,' *ACM Transactions on Programming Languages and Systems*, 43(2), pp. 1–42. https://doi.org/10.1145/3452096.
Lu, Z. (2023) 'Analysis on AES encryption standard and safety,' *Third International Symposium on Computer Engineering and Intelligent Communications (ISCEIC 2022)* [Preprint]. https://doi.org/10.1117/12.2662564.
M, S.R.P., P, S.R. and G, T. (2021) 'Dual Security based on Crypto Steganography Using Two Level Learning In Assist with Dual Offbeat Shielding Design,' *International Journal of*

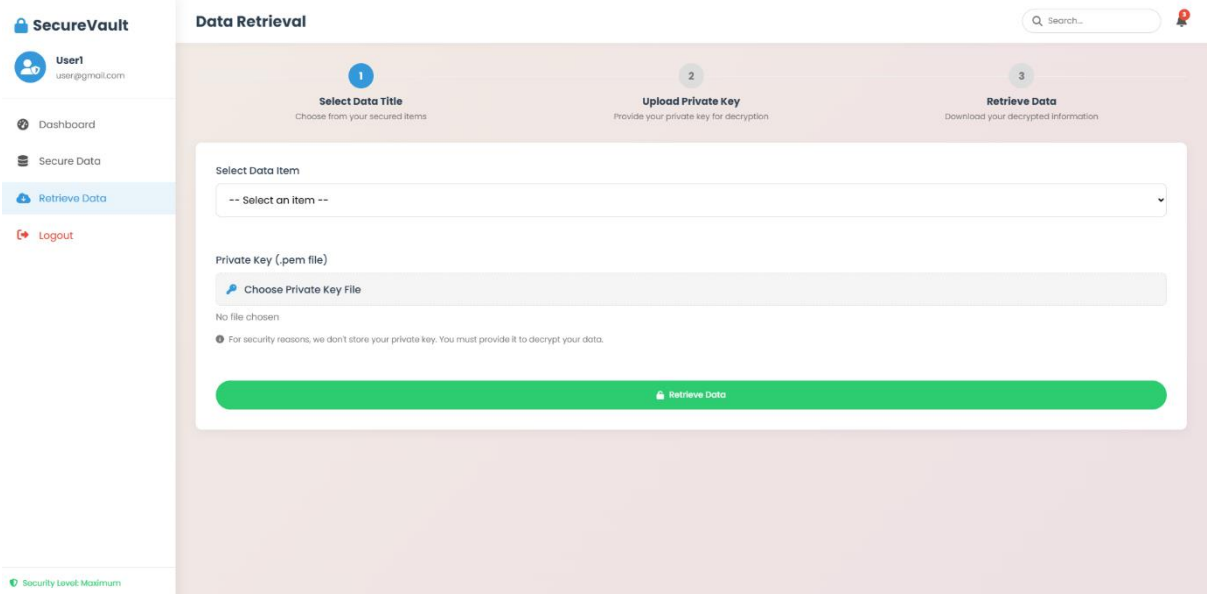
Scientific Research in Computer Science Engineering and Information Technology, pp. 541–548. https://doi.org/10.32628/cseit12173125.
Mohamed, H. *et al.* (2021) *Information-Theoretic Limits for Steganography in Multimedia*. https://doi.org/10.36227/techrxiv.14867241.v1.
Mouha, N. (2021) *Review of the advanced encryption standard*. https://doi.org/10.6028/nist.ir.8319.
Muttaqin, K. and Rahmadoni, J. (2020) 'Analysis and design of file security system AES (Advanced Encryption Standard) Cryptography based,' *Journal of Applied Engineering and Technological Science (JAETS)*, 1(2), pp. 113–123. https://doi.org/10.37385/jaets.v1i2.78.
Namassivaya, N.N. *et al.* (2023) 'Encrypted chat application using RSA Algorithm,' *International Journal of Engineering Technology and Management Sciences*, 7(2), pp. 854–859. https://doi.org/10.46647/ijetms.2023.v07i02.095.
Osman, O.M. et al. (2022) 'Hybrid multistage framework for data manipulation by combining cryptography and steganography,' Bulletin of Electrical Engineering and Informatics, 11(1), pp. 327–335. https://doi.org/10.11591/eei.v11i1.3451.
Prediger, L. *et al.* (2022) 'd3p - A Python Package for Differentially-Private Probabilistic Programming,' *Proceedings on Privacy Enhancing Technologies*, 2022(2), pp. 407–425. https://doi.org/10.2478/popets-2022-0052.
R, S. and Marlina, N. (2022) 'Aplikasi perpaduan enkripsi Base64 dengan metode steganografi Distrete Cosine Transform (DCT),' *Jurnal Sintaks Logika*, 2(2), pp. 37–45. https://doi.org/10.31850/jsilog.v2i2.1737.
Rathore, M.S. *et al.* (2022) 'A novel trust-based security and privacy model for Internet of Vehicles using encryption and steganography,' *Computers & Electrical Engineering*, 102, p. 108205. https://doi.org/10.1016/j.compeleceng.2022.108205
Rodríguez, L.C. *et al.* (2021) 'Selecting an Effective Entropy Estimator for Short Sequences of Bits and Bytes with Maximum Entropy,' *Entropy*, 23(5), p. 561. https://doi.org/10.3390/e23050561.
RSA — PyCryptodome 3.23.0 documentation (2025). https://pycryptodome.readthedocs.io/en/latest/src/public_key/rsa.html.
Saritha, N.Mrs.M. *et al.* (2023) 'VLSI implementation of AES algorithm in Cryptography developed in Xilinx,' *International Journal of Advanced Research in Science Communication and Technology*, pp. 558–562. https://doi.org/10.48175/ijarsct-9617.
Shahmiri, A.M., Ling, C.W. and Li, C.T. (2023) 'Communication-Efficient LAPlace Mechanism for Differential privacy via random quantization,' *arXiv (Cornell University)* [Preprint]. https://doi.org/10.48550/arxiv.2309.06982.
Shyam, A. and Mukesh, N. (2020) 'A Django based educational resource sharing website: Shreic,' *Journal of Scientific Research*, 64(01), pp. 138–152. https://doi.org/10.37398/jsr.2020.640134.
Song, Y., Song, J. and Qu, J. (2021) 'Optimal Image Based Information Hiding with One-dimensional Chaotic Systems and Dynamic Programming,' *The International Arab Journal of Information Technology*, 19(1). https://doi.org/10.34028/iajit/19/1/1.
Syahputra, R. (2020) 'Analysis of Even-Rodeh code for text compression,' *The IJICS (International Journal of Informatics and Computer Science)*, 4(1), p. 1. https://doi.org/10.30865/ijics.v4i1.1348.
T, N.T.S. and A, N.A.J.A.R. (2022) 'A low power Nano AES security algorithm for image cryptography,' *International Journal of Engineering Technology and Management Sciences*, pp. 301–310. https://doi.org/10.46647/ijetms.2022.v06i04.0050.
Tufano, M. et al. (2022) Generating accurate assert statements for unit test cases using pretrained transformers. https://doi.org/10.1145/3524481.3527220.

Vojinovic, I. (2023) '49 Eye-Opening Data Breach Statistics & Facts | DataProt,' *Sirisha*, 5 May. https://dataprot.net/statistics/data-breach-statistics/#:~:text=We%E2%80%99ve%20compiled%20a%20list%20of%20the%20latest%20stats,cost%20of%20a%20data%20breach%20is%20$3.92%20million.
Wagh, S. et al. (2020) DP-Cryptography: Marrying differential privacy and cryptography in emerging applications. https://arxiv.org/abs/2004.08887.
Wagh, S. *et al.* (2021) 'DP-Cryptography,' *Communications of the ACM*, 64(2), pp. 84–93. https://doi.org/10.1145/3418290
Xia, R., Li, M. and Chen, S. (2022) 'Encryption Modes Identification of Block Ciphers based on Machine Learning,' *International Journal of Network Security & Its Applications*, 14(5), pp. 1–10. https://doi.org/10.5121/ijnsa.2022.14501.
Xu, Z. (2023) 'The Advance of Digital Signature with Quantum Computing,' *Highlights in Science Engineering and Technology*, 39, pp. 1111–1121. https://doi.org/10.54097/hset.v39i.6716.
Yang, J. and Liao, X. (2023) 'Exploiting Fine-Grained DCT Representations for Hiding Image-Level Messages within JPEG Images,' *arXiv (Cornell University)* [Preprint]. https://doi.org/10.48550/arxiv.2305.06582.
Yousefpour, A. et al. (2021) OPACuS: User-Friendly Differential Privacy Library in PyTorch. https://arxiv.org/abs/2109.12298.
Zeeshan, M. and Khan, S.A. (2022) 'A Robust Carrier Frequency Offset Estimation Algorithm in Burst Mode Multicarrier CDMA based Ad Hoc Networks,' *International Journal of Communication Networks and Information Security (IJCNIS)*, 4(3). https://doi.org/10.17762/ijcnis.v4i3.221.
Zhang, X. *et al.* (2023) 'LILP: a lightweight enciphering algorithm to encrypt Arbitrary-Length messages,' *Symmetry*, 15(1), p. 177. https://doi.org/10.3390/sym15010177.
Zolfaghari, B., Bibak, K. and Koshiba, T. (2022) 'The Odyssey of Entropy: Cryptography,' *Entropy*, 24(2), p. 266. https://doi.org/10.3390/e24020266.

9 Appendices

9.1 Appendix A

The web page for retrieving the data.



9.2 Appendix B

The code for the multi-level data detection system.

```
from Crypto.PublicKey import RSA
from Crypto.Cipher import PKCS1_OAEP, AES
from Crypto.Util.Padding import unpad
import base64
import cv2
import numpy as np
from PIL import Image
import io
import struct
def load_image(image_path):
image = Image.open(image_path).convert('RGB')
return np.array(image)
def load_shape(file_path):
with open(file_path, "rb") as f:
shape_data = f.read()
return struct.unpack('III', shape_data)
def aes_decrypt(data, key):
iv = data[:16]
encrypted_data = data[16:]
cipher = AES.new(key, AES.MODE_CFB, iv=iv)
return unpad(cipher.decrypt(encrypted_data), AES.block_size)
def remove_laplace_noise(noisy_data, noise):
noisy_data = noisy_data.astype(np.int16)
noise = noise.astype(np.int16)
clean_data = np.clip(noisy_data - noise, 0, 255).astype(np.uint8)
```

```
return clean_data
def extract_data_from_image(stego_image_path):
stego_image = cv2.imread(stego_image_path)
if stego_image is None:
raise ValueError("Could not load stego image")
flat_image = stego_image.flatten()
length_bytes = flat_image[:4].tobytes()
data_length = int.from_bytes(length_bytes, byteorder='big')
data_bytes = flat_image[4:4+data_length].tobytes()
return data_bytes
def decompress_data(compressed_data):
img_pil = Image.open(io.BytesIO(compressed_data))
img_array = np.array(img_pil)
data_bytes = img_array.flatten().tobytes()
return data_bytes.rstrip(b'\x00')
def decrypt_data(encrypted_data, private_key):
rsa_key = RSA.import_key(private_key)
cipher = PKCS1_OAEP.new(rsa_key)
encrypted_bytes = base64.b64decode(encrypted_data)
decrypted_data = cipher.decrypt(encrypted_bytes)
return decrypted_data.decode('utf-8')
def decryption_phase():
with open("aes_key.bin", "rb") as f:
aes_key = f.read()
with open("noisy_data.bin", "rb") as f:
```

```
noisy_data = np.frombuffer(f.read(), dtype=np.uint8)
with open("noise.bin", "rb") as f:
noise = np.frombuffer(f.read(), dtype=np.int16)
original_shape = load_shape("image_shape.bin")
clean_data = remove_laplace_noise(noisy_data, noise)
decrypted_data = aes_decrypt(clean_data.tobytes(), aes_key)
decrypted_image = np.frombuffer(decrypted_data, dtype=np.uint8)
decrypted_image = decrypted_image[:original_shape[0] * original_shape[1] *
original_shape[2]]
decrypted_image = decrypted_image.reshape(original_shape)
save_image(decrypted_image, "decrypted_stego.png")
return "decrypted_stego.png"
def steganography_extraction_phase(stego_image_path):
compressed_data = extract_data_from_image(stego_image_path)
encrypted_data = decompress_data(compressed_data)
with open("private_key.pem", "rb") as f:
private_key = f.read()
decrypted_data = decrypt_data(encrypted_data.decode('utf-8'), private_key)
return decrypted_data
if __name__ == "__main__":
decrypted_stego_path = decryption_phase()
user_data = steganography_extraction_phase(decrypted_stego_path)
print("Successfully retrieved hidden data:")
print(user_data)
```

```
from Crypto.Cipher import PKCS1_OAEP, AES
from Crypto.Util.Padding import pad, unpad
import base64
import cv2
import numpy as np
from PIL import Image
import io
import os
import struct
def generate_rsa_keys():
key = RSA.generate(2048)
private_key = key.export_key()
public_key = key.publickey().export_key()
return private_key, public_key
def encrypt_data(data, public_key):
rsa_key = RSA.import_key(public_key)
cipher = PKCS1_OAEP.new(rsa_key)
encrypted_data = cipher.encrypt(data.encode('utf-8'))
return base64.b64encode(encrypted_data).decode('utf-8')
def compress_data(data):
if isinstance(data, str):
data_bytes = data.encode('utf-8')
else:
data_bytes = data
```

```
array_length = len(data_bytes)
width = int(np.ceil(np.sqrt(array_length)))
height = width
padded_data = np.pad(np.frombuffer(data_bytes, dtype=np.uint8),
(0, width * height - array_length),
mode='constant')
img = padded_data.reshape((height, width))
img_pil = Image.fromarray(img, mode='L')
buffer = io.BytesIO()
img_pil.save(buffer, format="PNG")
return buffer.getvalue()
def embed_data_into_image(data, mask_image_path, output_image_path):
mask_image = cv2.imread(mask_image_path)
if mask_image is None:
raise ValueError("Could not load mask image")
data_bytes = np.frombuffer(data, dtype=np.uint8)
data_length = len(data_bytes)
if data_length + 4 > mask_image.size:
raise ValueError("Data too large for the mask image")
flat_image = mask_image.flatten()
length_bytes = data_length.to_bytes(4, byteorder='big')
flat_image[:4] = np.frombuffer(length_bytes, dtype=np.uint8)
flat_image[4:4+data_length] = data_bytes
stego_image = flat_image.reshape(mask_image.shape)
cv2.imwrite(output_image_path, stego_image)
```

```
return stego_image
def aes_encrypt(data, key):
iv = os.urandom(16)
cipher = AES.new(key, AES.MODE_CFB, iv=iv)
encrypted_data = cipher.encrypt(pad(data, AES.block_size))
return iv + encrypted_data
def add_laplace_noise(data, mean=0, scale=0.25):
noise = np.random.laplace(mean, scale, data.shape).astype(np.int16)
noisy_data = np.clip(data + noise, 0, 255).astype(np.uint8)
return noisy_data, noise
def load_image(image_path):
image = Image.open(image_path).convert('RGB')
return np.array(image)
def save_image(image, path):
Image.fromarray(image).save(path)
print(f"Image saved at: {path}")
def save_shape(shape, file_path):
with open(file_path, "wb") as f:
f.write(struct.pack('III', *shape))
def steganography_phase():
user_details = "User: Alice, Email: alice@example.com, Phone: 1234567890"
private_key, public_key = generate_rsa_keys()
with open("private_key.pem", "wb") as f:
f.write(private_key)
with open("public_key.pem", "wb") as f:
```

```
f.write(public_key)
encrypted_data = encrypt_data(user_details, public_key)
compressed_data = compress_data(encrypted_data)
embed_data_into_image(compressed_data, "mask.jpg", "stego_image.png")
print("Stego image created successfully!")
return "stego_image.png"
def encryption_phase(image_path):
aes_key = os.urandom(32)
output_image = 'noisy_image.png'
image = load_image(image_path)
original_shape = image.shape
image_data = image.flatten()
# Encrypt the image data with AES
aes_encrypted = aes_encrypt(image_data.tobytes(), aes_key)
# Add noise and save
aes_encrypted_array = np.frombuffer(aes_encrypted, dtype=np.uint8)
noisy_data, noise = add_laplace_noise(aes_encrypted_array, scale=0.1)
# Reshape and save the noisy image
noisy_image = noisy_data[:original_shape[0] * original_shape[1] *
original_shape[2]].reshape(original_shape)
save_image(noisy_image, output_image)
# Save auxiliary files
save_shape(original_shape, "image_shape.bin")
with open("aes_key.bin", "wb") as key_file:
key_file.write(aes_key)
with open("noisy_data.bin", "wb") as data_file:
        data_file.write(noisy_data.tobytes())
        with open("noise.bin", "wb") as noise_file:
        noise_file.write(noise.tobytes())
        print("Image encryption completed successfully!")
        # %%
        if __name__ == "__main__":
        stego_image_path = steganography_phase()
        encryption_phase(stego_image_path)
```