STELWZAES: ENHANCING THE SECURITY OF STORED SPII USING 3 LAYERS OF A PROTECTION

MSc Research Project

Master of Science in Cyber Security

Sameer Surendra Ambilpure

Student ID: 22211586

School of Computing

National College of Ireland

Supervisor: Michael Prior

National College of Ireland
MSc Project Submission Sheet
School of Computing

Student Name:	Sameer Surendra Ambilpure
Student ID:	22211586
Programme:	Master of Science in Cyber Security **Year:** 2023-2024
Module:	Practicum Part 2
Supervisor:	Michael Prior
Submission Due Date:	12th August 2024
Project Title:	MSc Research Project Part 2
Word Count:	7235 **Page Count:** 20

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: Sameer Surendra Ambilpure

Date: 11th August 2024

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):	

STELWZAES: ENHANCING THE SECURITY OF STORED SPII USING 3 LAYERS OF A PROTECTION

Sameer Surendra Ambilpure
Student ID: 22211586

Abstract

The security associated with sensitive data is highly important as cyber criminals look to access the Personal Identifiable Information(PII) of people in an unauthorised manner to carry out malicious activities. These malicious activities cause many problems the PIIs can be used for accessing the bank account of a user or for accessing some other kinds of important information. The leakage of PIIs results in financial losses and may cause a negative impact on the reputation of the people who are targeted. There are many methods that can be used to secure data and two of the most popular methods is Steganography and Cryptography. The combination of these two methods helps in increasing the security associated with PIIs. In the study proposed here a system that combines Steganography, Cryptography and image compression for securing PIIs is proposed. The study proposed here uses the Least Significant Bit(LSB) technique to perform steganography and hide PII in the form, of text in a carrier image. Lempel-Ziv-Welch(LZW) is used to compress the carrier image and Advanced Encryption Standard(AES) is used for encrypting the carrier image containing the PII. A desktop application is built in this study using Python and the Steganography, Cryptography and image compression techniques are integrated into the desktop application. The desktop application is able to successfully secure a PII given as input . The encrypted image can be decrypted and the PII can be obtained. The Steganography, Cryptography and image compression techniques are successfully implemented using Python.

1 Introduction

1.1 Background

The security of sensitive and personal information or Personal Identifiable Information(PII) is highly important in a time where a large number of companies and individuals are targeted by cyber attacks. These cyber attacks have a goal of leaking or accessing the personal or sensitive information of the companies or individuals. It was found that during the fourth quarter of the year 2023, more than eight million records were exposed worldwide(Ani , 2024). It was found that a data breach on average was 4.45 million U.S. dollars(Ani , 2024). For saving the money lost due to the leak of sensitive information security measures are invaluable for all kinds of companies that handle sensitive information. There are different techniques that can be used for securing sensitive information that is being sent over a network or being handled in an organisation or by individuals. Two main kinds of methods that are used commonly are Cryptography and Steganography.

Cryptography is a method used to protect data by transforming it into an unreadable format, making it hard to interpret(Solichin and Erwin Wahyu, 2017). Cryptography makes sure that even if the data is leaked by an attacker during a cyber attack they are not able to understand or misuse the data as the data is in a form that is not understandable. The conversion of the data into an unreadable form is encryption and retrieving the data back into its original form is called decryption. There are algorithms that can be used for encrypting the data like Data Encryption Standard(DES), Advanced Standard Encryption(AES), RSA, SHA etc(Nithya and Sripriya, 2016). The algorithms encrypt the data using a cryptographic key. Cryptographic algorithms are divided into two based on the types of keys used for encrypting and decrypting data, these are asymmetric and symmetric. The cryptographic algorithms that encrypt and decrypt data using the same key are called symmetric encryption algorithms. In asymmetric encryption and decryption, the key used by the algorithms to encrypt the data is different from the key used for decrypting the data(Hamouda, 2020). The cryptographic algorithms based on symmetric keys are much faster and more secure than asymmetric key-based cryptographic algorithms(Hamouda, 2020). So the use of symmetric key based cryptographic algorithms is better for securing sensitive data. Cryptographic algorithms can be used to secure data in the forms of both images and text(Kumar et al., 2017; Mona et al., 2014).

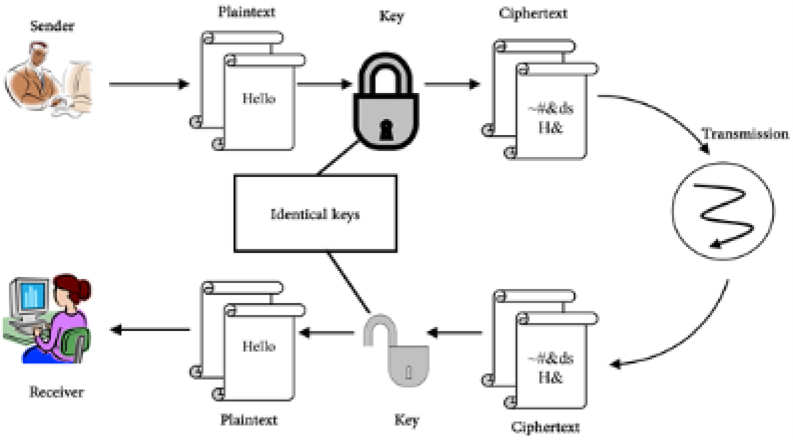


Figure (1): cryptography using symmetric key(Hamouda, 2020)

Steganography is the practice of hiding sensitive information within another medium, called the cover or carrier medium(Shetty, 2017). Steganography can be utilised for concealing information in carrier files that are multimedia files like images, or in the form of text(Shetty, 2017). The attackers who may enter a network or storage location of a company or an individual will not be able to find out that a particular file is a carrier file as the file will look like a random file containing information that is not important or relevant. The carrier files in which important information is concealed are called a stego object(Shetty, 2017).

1.2 Problem Definition

Cryptography and Steganography have been used successfully over the year for mitigating the damages caused by cyber attacks, However, as time passed the methods used by cyber criminals have also evolved to a stage where they are able to overcome the defence posed by cryptographic and steganographic methods(Packetlabs,2022). Attackers have now developed methods to generate keys that can be used to decrypt data that is in an encrypted form. Steganography also faces problems in securing data as an attacker may become suspicious of a random carrier file and this suspicion can lead to the attacker finding out that the file conceals information. The attacker can then extract the information hidden in the carrier files to get the important data and use it according to their needs.

These issues associated with Steganography and cryptography can be solved by combining the two methods. Steganography can be utilised to conceal data in a carrier file and the carrier file can be encrypted utilising a cryptographic algorithm. In the study proposed here a system for securing sensitive data in the form of text is proposed. The system proposed here hides the text in a carrier file that is in the form of an image and this image is then encrypted using a cryptographic technique. The storage of the image is made easier as an image compression technique is used for compressing the image. The study proposed here uses the Least significant Bit(LSB) method for Steganography, Lempel-Ziv-Welch(LZW) for image compression and AES for encrypting and decrypting the image. The AES is a symmetric key cryptographic algorithm.

The study has the following research question:

How would the merging of steganography, LZW compressor, and AES encryption mechanisms strengthen the confidentiality of archived PII's?

The aim of the study is to build a system that combines steganography, image compression and cryptography for securing PII’s. The objectives of the study are:

1. Implement Steganography using the LSB technique.
2. Compress an image utilising the LZW compression technique.
3. Implement AES encryption and decryption.
4. Build a desktop application that can be used to secure PII’s.
5. Integrate the steganography, image compression and cryptographic techniques.
6. Test the performance of the system that combines steganography, image compression and cryptography using attacks generated via Linux tools.

The novelty of the system proposed here is that it combines steganography, image compression and cryptography into a single system that secures PIIs. The addition of LZW based image compression technique is the contribution of the study as previous systems only combined steganography and cryptographic techniques. The image compression technique reduces the storage space required for storing the encrypted image.

The Section 1 of the report is the introduction that gives the details about steganography, cryptography and the system proposed here. Section 2 contains the details of the related work. Section 3 contains the details of the steganography, image compression and cryptographic methods used in the study. Section 4 contains details about the specifications about the different specifications in the study. Section 5 contains the details associated with the final implementation of the system as a desktop application. Section 6 contains the details of the evaluation of the results of the study. Section 7 contains the details of the conclusion and future enhancements.

2 Literature Review

This section of the report discusses previous research and the challenges encountered. It focuses on earlier encryption, steganography, and compression methods used for data security, with an emphasis on the traditional encryption techniques employed. The main goal of this research is to enhance data security by using advanced encryption techniques that offer higher security and faster encryption.

2.1 Encryption Methods

The RSA algorithm is a widely-used asymmetric cryptography technique, to encrypt and decrypt data which was proposed in this study by (Obaid, 2020). The prime numbers N and M were used to generate the public and private keys, with N and M being chosen as large prime integers to enhance security. The MATLAB library functions were utilized to handle the large prime numbers and perform the encryption and decryption processes. The results demonstrated that the RSA algorithm was successfully implemented on different sizes of message files, with the encrypted ciphertext expected to be difficult for hackers to interfere with. Specifically, the study reported that the public key consisted of the modulo n and the index e, while the private key included the modulus n and the private key d, which was kept secret. The main limitation of the RSA algorithm mentioned in the study is that it takes extra time to perform the encryption process compared to symmetric encryption techniques, although the enhanced security provided by the use of large prime numbers offsets this drawback.

Encryption scheme that combines Data Encryption Standard (DES) and network coding (NC) to achieve dynamic security protection was proposed in this study by (Tang et al., 2018). The main method involves a three-layer encryption process, where the inner and outer layers use NC, and the middle layer implements DES. The data used in the study consists of a 1 Mb plaintext, and the performance of the proposed scheme is evaluated under different parameter settings, such as the block length (L) and the divisor (D) of the block length. Other methods used in the study include theoretical justification of the full-rank property of the encryption matrices and numerical validation of the encryption and decryption complexity. The results show that the running ratio of the proposed scheme is relatively lower than or comparable to the triple DES, with the ratio of the total NC encoding and decryption time over the total encryption and decryption time of the whole scheme being 0.24 when L = 64 and 0.71 when L = 256. The study also demonstrates that the proposed scheme can effectively defend against both exhaustive and analysis attacks, and the dynamic security is achieved by combining low-frequency regular key update and high-frequency partial key update. The limitation of the study is that the simulation is conducted in MATLAB, which may not fully capture the efficiency of the proposed scheme in a real-world implementation, and the security level of the proposed scheme is not thoroughly tested.

A countermeasure that is low-cost for the AES encryption and decryption process, employing temporal redundancy and modifying the AES round architecture into three parts with pipeline registers to enhance fault coverage was discussed in the study by (Bedoui et al., 2022). Additional methods include fault injection simulations using VHDL language to assess fault coverage, with tests differentiating by the number of bits in injected faults. Results indicate that the proposed schema can detect 99.539% of injected random errors, with an area overhead that is relatively low and minimal increases in speed and power consumption. However, the study acknowledges limitations, including the potential for the same error value to be injected twice in the same place at the same time, which the detection system may not identify, highlighting the need for further validation of the fault detection capabilities under varied conditions.

2.2 Steganography Methods

Various steganographic techniques used for information hiding, including substitution, transform domain, statistical, and distortion methods were described in this study by (C.P, T and G, 2013). The data consists of digital images used as cover media to hide secret messages. The study also reviews techniques like integer wavelet transform, matrix embedding, and pixel value differencing used in the different steganographic categories. The results show that the proposed methods achieve high PSNR values, up to 66.96 dB, and low MSE values, around 0.013168, indicating good image quality and imperceptibility. The edge-adaptive scheme based on LSB matching revisited showed the best performance, with an embedding capacity of up to 50% of the cover image size and a PSNR of 54.1 dB. However, the study acknowledges that the trade-off between embedding capacity, image quality, and robustness against attacks remains a limitation, and further research is needed to address these challenges in practical steganographic applications.

A Multi-Level Steganography (MLS) algorithm that employs AES and Blow-Fish encryption algorithms to secure cover images and embed encryption keys within stego images, enhancing data security was discussed in the study by (Alanzy et al., 2023). It incorporates pixel randomization and hybrid encryption to augment the complexity of the encryption process, aiming to maintain the quality of the stego image while ensuring reliable encryption and decryption of messages that are secrets. The study also explores the utilisation of LSB steganography, where the least significant bit in the cover image is replaced with a bit from a secret message, and evaluates the performance using metrics like Mean Square Error (MSE) and Peak Signal-to-Noise Ratio (PSNR). Results indicate that the proposed algorithm effectively protects data with high PSNR and low MSE values, suggesting superior image quality and robust encryption.

2.3 Image Compression Method

A new colour image lossless compression algorithm with bit-error awareness based on a general bi-level block coding method was proposed in this study by (Peng et al., 2020). The proposed method contains three stages: conversion of the RGB colour space to the YCrCb color space, prediction of the YCrCb image using predictors, and further encoding of the residue sequences using a 2-D bi-level block coding algorithm. The study used public domain test images of "Lena", "Baboon", "Pepper", and "Airplane" each with a size of 512x512 pixels to validate the performance. Additionally, the study compared the proposed 2-D bi-level block coding algorithm with 1-D bi-level block coding, interval Huffman coding, and standard Huffman coding. The results showed that the proposed 2-D bi-level block coding algorithm achieved the highest compression ratio (CR) and when the bit-error rate (BER) was larger than 0.001, it offered the highest peak signal-to-noise ratio (PSNR) compared to the other methods. Specifically, for the "Lena" image, the CR was 4.55 and the PSNR was 48.37 dB when the BER was 0.005. The study also utilised a particle swarm optimization (PSO) algorithm to determine the best combination of colour space transformations and predictors to achieve the minimum residue entropy. The main limitation of the study is that it only considered lossless compression and did not explore the performance of the proposed method in lossy compression scenarios.

The Lempel-Ziv-Welch compression algorithm, an enhancement of the LZ78 algorithm, was used to exploit repeated patterns in medical data for efficient storage in Hadoop was discussed in the study by (Addepalli and Lakshmi, 2021). Additionally, it utilized Huffman coding and Run Length

Encoding (RLE) as comparative lossless compression techniques. The study's results demonstrated that LZW compression saved up to 40% of Hadoop storage space when applied to text and image medical data, including CT scans, X-rays, and MRI scans. The compression was lossless, ensuring complete data recovery upon decompression, which is crucial for medical applications. However, the study acknowledged limitations in the form of potential biases due to the specific datasets used, which may not represent all types of medical data, thus suggesting the need for further research to validate the algorithm's effectiveness across diverse medical datasets.

A medical image compression method called colour wavelet difference reduction (CWDR), which is an extension of the standard wavelet difference reduction (WDR) method was proposed in this study by (Matina Ch. Zerva et al., 2023). The dataset employed in the study consists of 31 slides of colorectal cancer images extracted using the Hamamatsu NanoZoomer 210 scanner, which provides 20× and 40× optical magnification options. Additionally, the study compares the performance of the proposed CWDR method with four other compression algorithms: discrete wavelet transform (DWT) technique, JPEG 2000, HEIC, and WEBP. The results indicate that the CWDR method achieved state-of-the-art compression results with a high compression ratio and slight information loss within an acceptable range. Specifically, the CWDR method achieved PSNR values ranging from 32.01 to 47.49 dB, and SSIM values ranging from 0.68 to 0.98, with 30 out of 31 images having an SSIM value of 0.92 or higher. The study also employed a Mean Opinion Score (MOS) scale to evaluate the perceived quality of the compressed images, and the results showed a statistically significant association between the MOS scores of the original uncompressed images and the compressed images using the CWDR method. However, a limitation of the study is that it only focused on the compression of histopathological microscopy images, and further evaluation on a wider range of medical image modalities would be beneficial to assess the generalizability of the proposed CWDR method.

2.4 Summary

The summary of all the existing studies analysed here is given below:

Study	Methods	Results	Limitations
Bedoui et al., (2022).	The study uses temporal redundancy and modified the AES round architecture into three parts with pipeline registers. They conducted fault injection simulations using VHDL to assess fault coverage, varying the number of injected fault bits.	The study found that their proposed schema detected 99.539% of randomly injected errors. It also reported minimal area overhead and slight increases in speed and power consumption.	A potential limitation identified was the risk of the same error value being injected twice in the same location simultaneously, which might not be detected by the system. This highlights the need for further assessment of fault detection capabilities across different conditions of operations.

Alanzy et al., (2023)	The study introduced a Multi-Level Steganography (MLS) algorithm utilizing AES and Blow-Fish encryption algorithms to secure cover images and embed encryption keys within stego images. It employed pixel randomization and hybrid encryption to enhance encryption complexity while maintaining stego image quality.	The algorithm achieved high PSNR and low MSE values, ensuring superior image quality and robust encryption.	The method's complexity in encoding highlights the need for improved encryption and compression algorithms, and exploration of AI to enhance resistance against attacks.
Peng et al., (2020)	The study proposed a new colour image lossless compression algorithm based on a general bi-level block coding method, involving stages such as RGB to YCrCb colour space conversion, YCrCb image prediction using predictors, and encoding	The proposed 2-D bi-level block coding algorithm achieved the highest compression ratio (CR) and highest peak signal-to-noise ratio (PSNR) for bit-error rates (BER) larger than 0.001. For instance, the "Lena" image achieved a CR of 4.55 and PSNR of 48.37 dB at a BER of 0.005. Particle swarm optimization (PSO) was employed to optimize colour space transformations and predictors, minimizing residue entropy.	The study focused solely on lossless compression and did not investigate the performance of the proposed method in lossy compression scenarios.

	residue sequences with a 2-D bi-level block coding algorithm. Public domain test images ("Lena", "Baboon", "Pepper", and "Airplane") of size 512x512 pixels were used for validation.		
Addepalli and Lakshmi, (2021)	The study utilized the Lempel-Ziv-Welch compression algorithm, along with Huffman coding and Run Length Encoding as comparative techniques, to compress medical data for efficient storage in Hadoop.	The study found that LZW compression saved up to 40% of Hadoop storage space when applied to various types of medical data, including text, CT scans, X-rays, and MRI scans. The compression was lossless, ensuring complete data recovery during decompression, which is critical for medical applications.	A potential limitation noted was the bias introduced by specific datasets used in the study, which may not fully represent all types of medical data. Further research is needed to validate the algorithm's effectiveness across diverse medical datasets.

Table (1): Summary table of the reference literature studies

From table(1) it can be seen that AES and LSB are effective cryptographic and steganographic techniques. The studies also showed that the LZW is an effective image compression technique. Existing studies also combined steganography and cryptographic techniques for effectively securing data. However, none of the studies proposed a system that combined AES, LSB and LZW techniques for data security. So a system that combines these methods is proposed in this study.

3 Methodology

3.1 Overall architecture

This research project introduces a holistic strategy to bolster Personally Identifiable Information security using steganography, LZW compression, and AES encryption. Initially, PII data is concealed within a carrier file using steganography to prevent unauthorized access. Following this, LZW compression efficiently reduces file size without compromising data integrity, followed by AES encryption to ensure strong protection against unauthorized interception.

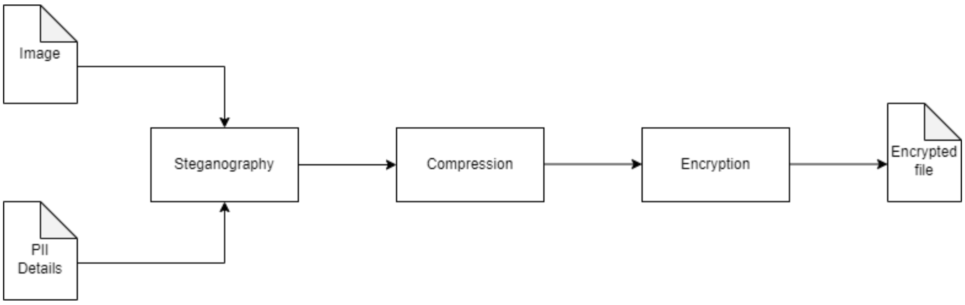


Figure (2): Encryption Process

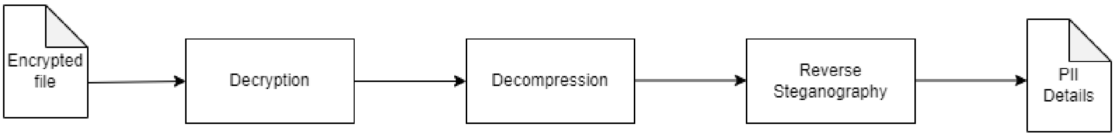


Figure (3): Decryption Process

Figure (2) is the block diagram of the encryption process, detailing how PII data is steganographically embedded, compressed using LZW, and then encrypted with AES. Figure (3), on the other hand, shows the decryption process, showing how the encrypted data is decrypted using AES, decompressed using LZW, and then using reverse steganography to extract embedded data.

3.2 Data Preparation and Initial Security Layer

The data preparation process begins with cleaning the data to ensure its accuracy and consistency. This involves removing any irrelevant information, such as new lines, and formatting the data appropriately for further processing. Steganography is then applied to the cleaned data. Steganography is a technique used to hide information within another file, making the hidden data invisible to unauthorized users. In this study, steganography is utilized to embed Personally Identifiable Information (PII) within a carrier file, such as an image file. For embedding the PII data, the Least Significant Bit technique is employed. This technique is favored for its simplicity and minimal impact on the carrier file's appearance or quality. There are several popular image steganography techniques, with the Least Significant Bit (LSB) method being a well-known example. LSB is used to hide secret messages or data within a cover image. Under LSB, there are two common methods: (i) Insertion based Method and (ii) Substitution based Method. Both are widely used for hiding data, but they have some differences. The Insertion based method increases the image size when the secret data is added, whereas the Substitution based method replaces bits of the image with the secret data without changing the image size(Aslam et al., 2022).LSB embedding is straightforward to implement and computationally efficient, requiring minimal processing overhead. It operates by altering the least significant bits of pixels or bytes in digital media, which often results in imperceptible changes to the original content. This method is compatible with a wide range of digital media formats, including images, audio, and video files. It can be applied without requiring specific modifications to file structures or formats, making it versatile for various applications. LSB embedding typically introduces minimal distortion to the carrier file's visual or auditory quality. By manipulating the least significant bits, the embedded data can be hidden effectively without noticeably affecting the overall perceptual quality of the media(Arun, Juhi and Harsh, 2023). In this study, LSB embedding involves splitting the binary representation of PII data into bits, which are then inserted into the least significant positions of the pixels in the carrier file. The changes made to the carrier file are so subtle that they are imperceptible to the human eye. A challenge arises in the coding implementation, where managing the binary manipulation and ensuring accurate bit-level operations can be complex. This process requires precise handling of binary data, and any errors in the coding logic can lead to corruption of the embedded information or unintended alterations to the carrier file. By embedding PII data using LSB steganography, the initial security layer hides the existence of the information that is sensitive effectively, making it less likely to be targeted by users who are unauthorized.

3.3 Data Compression

Data compression is a technique used to reduce the size of a file without losing any important information. This process is essential for optimizing storage space and improving the efficiency of data transmission. LZW (Lempel-Ziv-Welch) compression is utilised for compressing the carrier file that contains the embedded Personally Identifiable Information (PII). This algorithm works by finding sequences of data that are repetitive and encoding them into representations that are shorter. As the data is processed, a dictionary is built, mapping repeated sequences to shorter codes. The more repetitive the data, the more efficient the compression becomes. In this data compression phase the LZW compression begins by initializing a dictionary containing all single-character strings. As the input data is read, the algorithm searches for the longest string that matches an entry in the dictionary. Once found, it outputs the code for that string and adds a new entry to the dictionary for the string plus the next character. This process continues until the entire input data is encoded. The result is a compressed file that is smaller than the original but can be decompressed back to its original form without any loss of data. However, there are challenges associated with LZW compression. One difficulty lies in handling very large files, as the dictionary can grow significantly, leading to increased memory usage and slower processing times. This requires careful management of the dictionary to ensure that the compression remains efficient without overwhelming system resources. Using LZW compression on the carrier file not only optimizes storage and transmission but also preserves the integrity of the concealed PII data. This step ensures that the data remains intact and accessible, providing a balance between efficiency and security in the overall data protection strategy.

3.4 Data Encryption

Data encryption is a process that transforms readable data into an unreadable format using a specific algorithm and an encryption key. This transformation ensures that the data remains confidential and protected from unauthorized access. In this study, AES is employed to encrypt the carrier file containing the compressed and steganographically embedded Personally Identifiable Information (PII). AES is a symmetric encryption algorithm, meaning the same key is used for both encryption and decryption. Recommended by NIST, AES's strength lies in its ability to divide data into blocks and apply multiple rounds of intricate transformations such as substitutions, permutations, and mixing, all based on the encryption key. The number of rounds—10, 12, or 14—depends on the key size (128, 192, or 256 bits respectively). Typically, data is processed in a 4x4 byte matrix known as the state matrix, ensuring AES's effectiveness in securing digital information (Bedoui et al., 2022). AES operates in the study by initializing with a randomly generated Initialization Vector (IV) for added security. The AES cipher configured with Cipher Feedback (CFB) mode processes the data in blocks, encrypting each block sequentially. The encryptor instance applies the AES algorithm to transform the data, ensuring robust confidentiality. This approach effectively secures the PII within the carrier file, rendering it inaccessible without the correct decryption key. One of the difficulties encountered in this process is due to the robust nature of AES encryption. AES generates specific encrypted bytes for images that must be accurately managed and processed during decryption. Any mishandling of these bytes can result in decryption failures, complicating the recovery of the original data. Applying AES encryption to the compressed and steganographically embedded file provides a robust final layer of protection. This step ensures that even if the data is intercepted during transmission or storage, it remains inaccessible and unreadable to unauthorized individuals, thus significantly enhancing the security of the PII data.

3.5 Data Retrieval and Decryption

In the process of data retrieval and decryption, the journey from encrypted and compressed data to the original, usable information involves a sequence of carefully orchestrated steps. AES decryption serves as the initial key to unlocking the encrypted file, effectively reversing the encryption process to reveal the compressed data hidden within. This pivotal step ensures that the data is returned to its pre-encryption state, laying the groundwork for subsequent actions. Following AES decryption, LZW decompression steps in to expand the compressed data back to its original size and format. This process is akin to unpacking a tightly compressed suitcase, where each piece of data is reinstated to its rightful place and structure. By restoring the data to its original form, LZW decompression ensures that the integrity and completeness of the information are preserved, facilitating further analysis and utilization. Lastly, LSB decoding plays a crucial role in extracting concealed Personally Identifiable Information (PII) from the carrier file. This method delicately unveils hidden details embedded within the image or data carrier, making the recovered PII accessible for authorized purposes. LSB decoding acts like a skilled investigator, carefully extracting and presenting the concealed information without altering or compromising its accuracy.

4 Design Specification

4.1LSB Steganography

One of the most widely used techniques in Steganography today is known as least significant bit (LSB) insertion. This method involves altering the least noticeable parts of a cover image specifically, the least significant bits to encode hidden information within it(Refaat Said ,2021).

4.2 LZW Compression

LZW is a lossless data compression algorithm that is part of the dictionary coding technique. It creates a table of symbol sequences to encode data into indexes, aiming to reduce data size without losing any information. LZW helps in saving data storage space and speeds up data exchange over networks. LZW ensures that the original data can be perfectly reconstructed from the compressed data, maintaining data integrity. LZW utilizes fixed length codes for encoding, which simplifies the process but can result in sequences that are longer (Maulunida and Solichin, 2018).

4.3 AES Encryption

AES is a symmetric key block cipher developed by Joan Daemen and Vincent Rijmen in 1998. It supports key lengths of 128, 192, and 256 bits. AES is efficient in both software and hardware implementations. It operates on a block size of 128-bit and utilizes a number of rounds that are variable depending on the length of the key. Compared to other algorithms like, 3DES, DES and Blowfish, AES has a higher degree of diffusion, meaning it's more resistant to attacks due to its substitution-permutation network and use of Galois fields for transformations (Patil et al., 2016).

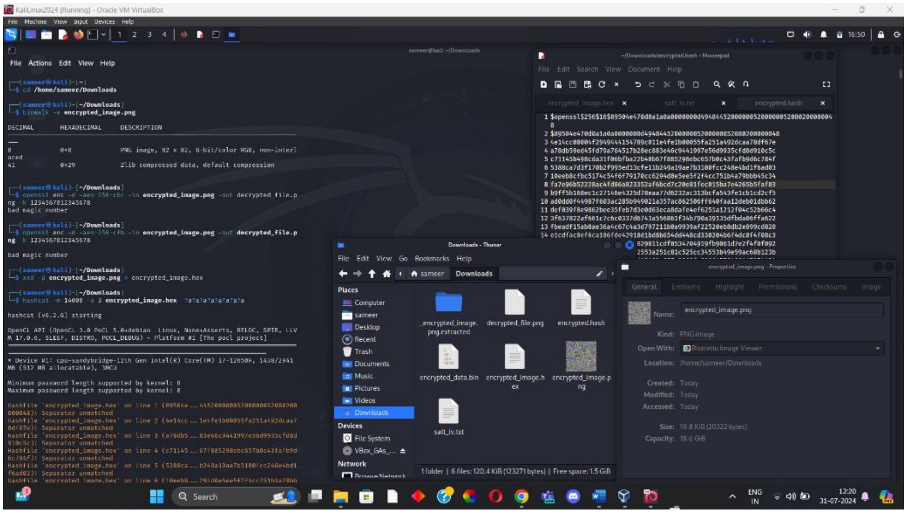
5 Implementation

The steganography, compression, and encryption methods were integrated into a desktop application. This application was developed using Python's Tkinter library, which provided a simple and effective way to create a graphical user interface (GUI). The GUI is designed with three main panels to facilitate user interaction and ensure a smooth workflow. The first panel is dedicated to encryption. Here, users can input their data and the encryption key. This panel includes text fields where the data and key are entered, and a button labelled "Encrypt Data" to initiate the encryption process. The design of this panel is straightforward, ensuring that users can easily input their information without any confusion. When the "Encrypt Data" button is clicked, the application processes the input data using the integrated steganography, compression, and encryption techniques, and the encrypted data is prepared for further use. The second panel is focused on decryption. Labelled "Result," this panel contains a single text field for users to input the decryption key. Similar to the first panel, there is a button labelled "Decrypt Data" which, when clicked, starts the decryption process. This panel is designed to be as user-friendly as the first, allowing users to quickly and easily retrieve their original data from the encrypted form. The decrypted data is then displayed, providing immediate feedback to the user. The third panel is the "Text Log Box," which plays a crucial role in providing transparency and insight into the processes occurring within the application. This log box displays detailed logs of the entire process, including steps involved in steganography, compression, encryption, and decryption. Additionally, it shows performance evaluation metrics, allowing users to see the efficiency and effectiveness of the processes. This feature ensures users are fully aware of what is happening with their data at each stage, fostering a sense of trust and reliability in the application. The development of this application using Tkinter emphasizes simplicity and functionality. Tkinter's straightforward methods for creating buttons, text fields, and other widgets made it an ideal choice for this project. The library's capabilities allowed for the creation of a user-friendly interface where users can easily input their data, initiate encryption or decryption operations, and monitor the progress and results through the text log box. The design of the GUI prioritizes ease of use. By dividing the interface into three distinct panels, users are guided through each step of the process in a logical and intuitive manner. The clear labeling of buttons and fields reduces the chances of user error and ensures that even those with limited technical knowledge can effectively use the application. The integration of the steganography, compression, and encryption techniques within a single application streamlines the process of securing data. Users do not need to rely on multiple tools or applications to achieve their security goals; instead, they can perform all necessary operations within this single, cohesive environment. This integration not only simplifies the user experience but also enhances the overall efficiency of data handling and protection.

6 Result And Evaluation

6.1 Results

Using tools like Kali Linux for vulnerability assessment and decryption attempts with brute force techniques provided valuable insights into the strength of AES encryption. By systematically trying every possible key until the correct one was found, the robustness of key management practices was measured. The results showed that AES encryption is highly showing resistant to brute force attacks.



```
┌──(sameer㉿kali)-[~]
└─$ cd /home/sameer/Downloads

┌──(sameer㉿kali)-[~/Downloads]
└─$ binwalk -e encrypted_image.png

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             PNG image, 82 x 82, 8-bit/color RGB, non-interl
aced
41            0x29            Zlib compressed data, default compression


┌──(sameer㉿kali)-[~/Downloads]
└─$ openssl enc -d -aes-256-cbc -in encrypted_image.png -out decrypted_file.p
ng -k 1234567812345678
bad magic number

┌──(sameer㉿kali)-[~/Downloads]
└─$ openssl enc -d -aes-256-cfb -in encrypted_image.png -out decrypted_file.p
ng -k 1234567812345678

bad magic number

┌──(sameer㉿kali)-[~/Downloads]
└─$ xxd -p encrypted_image.png > encrypted_image.hex

┌──(sameer㉿kali)-[~/Downloads]
└─$ hashcat -m 14000 -a 3 encrypted_image.hex  ?a?a?a?a?a?a?a?a

hashcat (v6.2.6) starting

OpenCL API (OpenCL 3.0 PoCL 5.0+debian  Linux, None+Asserts, RELOC, SPIR, LLV
M 17.0.6, SLEEF, DISTRO, POCL_DEBUG) - Platform #1 [The pocl project]
==============================================================================
============================================================================
* Device #1: cpu-sandybridge-12th Gen Intel(R) Core(TM) i7-12650H, 1438/2941
MB (512 MB allocatable), 3MCU

Minimum password length supported by kernel: 8
Maximum password length supported by kernel: 8

Hashfile 'encrypted_image.hex' on line 1 (89504e....445200000052000000520802000
000048): Separator unmatched
Hashfile 'encrypted_image.hex' on line 2 (4e14cc....1e4fe1b00055fa251a492dcaa7
8df67e): Separator unmatched
Hashfile 'encrypted_image.hex' on line 3 (a78db5....83e46c9441997e56d9935cfd8d
910c5c): Separator unmatched
Hashfile 'encrypted_image.hex' on line 4 (c71145....67f885298ebc657b0c43fafb9d
6c784f): Separator unmatched
Hashfile 'encrypted_image.hex' on line 5 (5388ca....b249a19ae7b3100fcc248e4bd1
f6ad03): Separator unmatched
```

~/Downloads/encrypted.hash - Mousepad

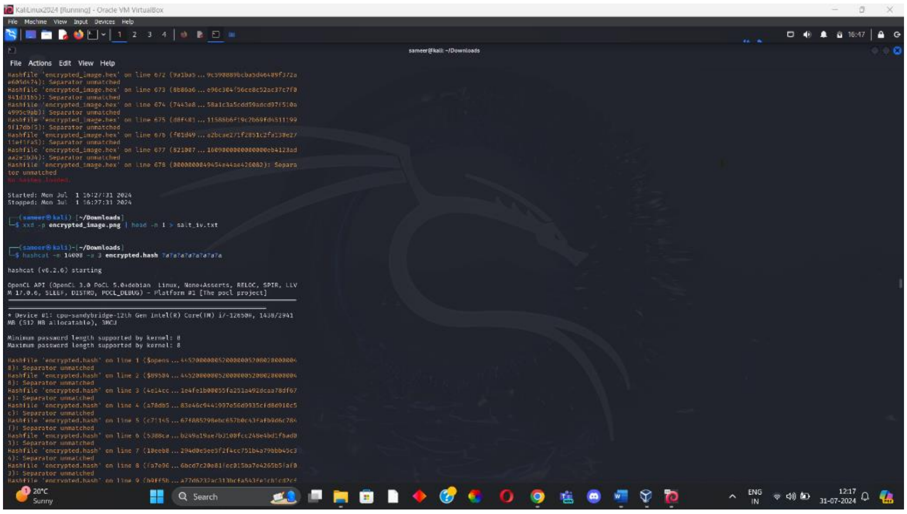
```
1 $openssl$256$16$89504e470d0a1a0a0000000d494844520000005200000052080200000004
  8
2 $89504e470d0a1a0a0000000d4948445200000052000000520802000000848
3 4e14cc00004f294944415478 9c011e4fe1b00055fa251a492dcaa78df67e
4 a78db59ed45fd79a764517b28ec883e46c9441997e56d9935cfd8d910c5c
5 c71145b498cda31f06bfba22b48b67f885298ebc657b0c43fafb9d6c784f
6 5388ca7d3f170b2f995ed13cfe11b249a19ae7b3100fcc248e4bd1f6ad03
7 10eeb8cfbc5174c54f6f79170cc6294d0e5ee5f2f4cc751b4a79bbb45c34
8 fa7e96b52228ac4fd86a823352af6bcd7c20c81fec815ba7e4265b5faf03
9 b9ff5b108ec1c27140e4325d78eaa77db232ac313bcfa543fe1cb1cd2cf5
10 ad0dd0f44987f603ac285b949021a357ac862506ff640faa12deb01dbb62
11 def039f8c9862bee35feb7d3e0d63eca8dafe4ef6251a1212f04c52b66c4
12 3f637822af661c7cbc0337d0743a5568b1f34b790a39135dfbda06ffa622
13 fbeadf15ab8ae36a4c67c4a3d797211b0a9939af22520eb8db2e899cd820
14 e1cdfac0ef6ca186fde42910d1bd8b654dd448cd330204b6f4dc0f4f08c3
   829811cdf0534704919fb9861d7e2f4f8f092
   2553a251c81c525cc34553b49e59ac68b123b
```

encrypted_image.png - Properties

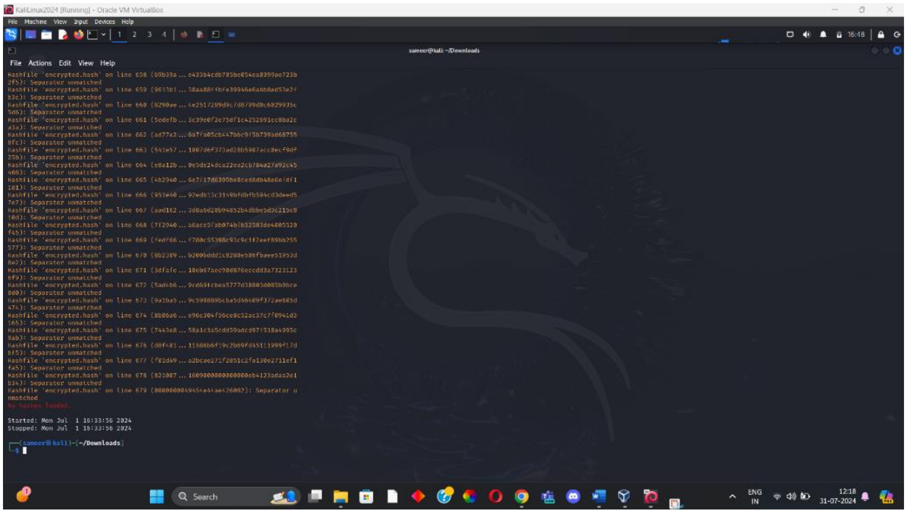
Name: encrypted_image.png
Kind: PNG image
Open With: Ristretto Image Viewer
Location: /home/sameer/Downloads
Created: Today
Modified: Today
Accessed: Today
Size: 18.8 KiB (20322 bytes)
Capacity: 18.6 GiB

Downloads - Thunar

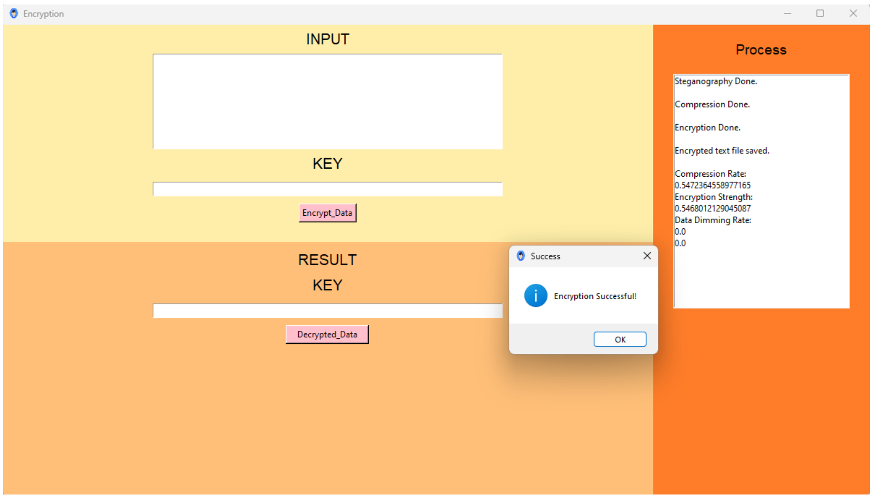
1 folder | 6 files: 120.4 KiB (123271 bytes) | Free space: 1.5 GiB



```
Hashfile 'encrypted_image.hex' on line 672 (9a1ba5....9c5980089bcba5d46409f372a
e605d474): Separator unmatched
Hashfile 'encrypted_image.hex' on line 673 (8b86a6....e96c384f56ce8c52ac37c7f8
941d3165): Separator unmatched
Hashfile 'encrypted_image.hex' on line 674 (7443a8....58a1c3a5cdd59adcd97f510a
4995c9ab): Separator unmatched
Hashfile 'encrypted_image.hex' on line 675 (d8f481....115886f19c2b69fd4511199
9f17dbf5): Separator unmatched
Hashfile 'encrypted_image.hex' on line 676 (f01d49....a2bcae271f2851c2fa130e27
11ef1fa5): Separator unmatched
Hashfile 'encrypted_image.hex' on line 677 (821007....1009800000000000eb4123ad
aa2e1b34): Separator unmatched
Hashfile 'encrypted_image.hex' on line 678 (0000000049454e44ae426082): Separa
tor unmatched
No hashes loaded.

Started: Mon Jul  1 16:27:31 2024
Stopped: Mon Jul  1 16:27:31 2024

┌──(sameer㉿kali)-[~/Downloads]
└─$ xxd -p encrypted_image.png | head -n 1 > salt_iv.txt


┌──(sameer㉿kali)-[~/Downloads]
└─$ hashcat -m 14000 -a 3 encrypted.hash ?a?a?a?a?a?a?a?a

hashcat (v6.2.6) starting

OpenCL API (OpenCL 3.0 PoCL 5.0+debian  Linux, None+Asserts, RELOC, SPIR, LLV
M 17.0.6, SLEEF, DISTRO, POCL_DEBUG) - Platform #1 [The pocl project]
==============================================================================
============================================================================
* Device #1: cpu-sandybridge-12th Gen Intel(R) Core(TM) i7-12650H, 1438/2941
MB (512 MB allocatable), 3MCU

Minimum password length supported by kernel: 8
Maximum password length supported by kernel: 8

Hashfile 'encrypted.hash' on line 1 ($opens....4452000000520000005208020000004
8): Separator unmatched
Hashfile 'encrypted.hash' on line 2 ($89504....4452000000520000005208020000004
8): Separator unmatched
Hashfile 'encrypted.hash' on line 3 (4e14cc....1e4fe1b00055fa251a492dcaa78df67
e): Separator unmatched
Hashfile 'encrypted.hash' on line 4 (a78db5....83e46c9441997e56d9935cfd8d910c5
c): Separator unmatched
Hashfile 'encrypted.hash' on line 5 (c71145....67f885298ebc657b0c43fafb9d6c784
f): Separator unmatched
Hashfile 'encrypted.hash' on line 6 (5388ca....b249a19ae7b3100fcc248e4bd1f6ad0
3): Separator unmatched
Hashfile 'encrypted.hash' on line 7 (10eeb8....294d0e5ee5f2f4cc751b4a79bbb45c3
4): Separator unmatched
Hashfile 'encrypted.hash' on line 8 (fa7e96....6bcd7c20c81fec815ba7e4265b5faf0
3): Separator unmatched
```


KaliLinux2024 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
File Actions Edit View Help
Search
ENG IN
12:18
31-07-2024

```
Hashfile 'encrypted.hash' on line 658 (69b39a ... e433b4cdb785be054ca8399ae723b
2f5): Separator unmatched
Hashfile 'encrypted.hash' on line 659 (9613b1 ... 38aa88ffbfe39946e6abb8ed53e2f
b3c): Separator unmatched
Hashfile 'encrypted.hash' on line 660 (8290ae ... 4e2517289d9c7d8799d8c6829935c
5d6): Separator unmatched
Hashfile 'encrypted.hash' on line 661 (5edefb ... 3c39e0f2c75df1c4232691cc8ba2c
a3a): Separator unmatched
Hashfile 'encrypted.hash' on line 662 (ad77a2 ... 6a7fa05cb447bbc915b739ad68755
8fc): Separator unmatched
Hashfile 'encrypted.hash' on line 663 (541e57 ... 1807d6f373ad28b5907acc8ecf9df
25b): Separator unmatched
Hashfile 'encrypted.hash' on line 664 (e8a12b ... 0e5de24dca22ea2cb784a27a92c45
408): Separator unmatched
Hashfile 'encrypted.hash' on line 665 (4b2940 ... 6e7f17d6395be8ced8db48a6e1df1
181): Separator unmatched
Hashfile 'encrypted.hash' on line 666 (953e40 ... 92edb13c3149bfdbfb594cd3deed5
7e7): Separator unmatched
Hashfile 'encrypted.hash' on line 667 (aad182 ... 3d8abd28b94852b4dbbe5d3c215c8
10d): Separator unmatched
Hashfile 'encrypted.hash' on line 668 (7f2940 ... a6ace5fab074bfb32583de4805528
f45): Separator unmatched
Hashfile 'encrypted.hash' on line 669 (fedf66 ... f780c55398c93c9c1f7eef89bb255
577): Separator unmatched
Hashfile 'encrypted.hash' on line 670 (8b2389 ... b200bddd1c8288e586fbaee51953d
8e2): Separator unmatched
Hashfile 'encrypted.hash' on line 671 (3dfafc ... 18eb67acc98d876eccdd3a7323123
6f9): Separator unmatched
Hashfile 'encrypted.hash' on line 672 (5ad4b6 ... 9cd69fcbea5777d38803d085b9bce
8d0): Separator unmatched
Hashfile 'encrypted.hash' on line 673 (9a1ba5 ... 9c598889bcba5d46489f372ae885d
474): Separator unmatched
Hashfile 'encrypted.hash' on line 674 (8b86a6 ... e96c304f56ce8c52ac37c7f0941d3
165): Separator unmatched
Hashfile 'encrypted.hash' on line 675 (7443e8 ... 58a1c3a5cdd59adcd97f518a4995c
9ab): Separator unmatched
Hashfile 'encrypted.hash' on line 676 (d8fk81 ... 11586b6f19c2b69fd45111999f17d
bf5): Separator unmatched
Hashfile 'encrypted.hash' on line 677 (f81d49 ... a2bcae271f2851c2fa130e2711ef1
fa5): Separator unmatched
Hashfile 'encrypted.hash' on line 678 (821087 ... 1609800800800800eb4123adaa2e1
b34): Separator unmatched
Hashfile 'encrypted.hash' on line 679 (8808800849454e44ae426082): Separator u
nmatched
No hashes loaded.

Started: Mon Jul  1 16:33:56 2024
Stopped: Mon Jul  1 16:33:56 2024

┌──(sameer㉿kali)-[~/Downloads]
└─$ 
```



Encryption Successfull

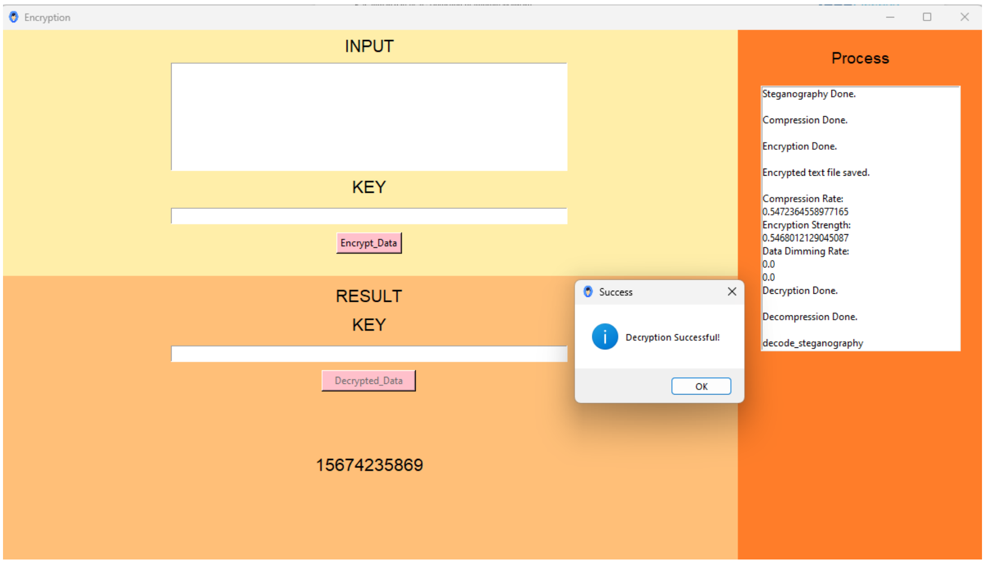
```
Steganography Done.

Compression Done.

Encryption Done.

Encrypted text file saved.

Compression Rate:
0.5472364558977165
Encryption Strength:
0.5468012129045087
Data Dimming Rate:
0.0
0.0
```

Figure (4): Performing encryption

Figure (4) illustrates the encryption process within the GUI's third panel, where real-time progress is displayed in the text log box. The log box details the stages of steganography, compression, encryption, and performance evaluation, including metrics such as compression rate, encryption strength, and data mining rate. This visual representation allows users to track each step of the encryption process and review performance metrics simultaneously, ensuring transparency and facilitating informed decisions about data security measures.

Figure (5): Encrypted image

Figure(5) shows the AES-encrypted image where the effects of the encryption process are visibly manifested in the form of grainy textures. The encryption operation alters the pixel values within the image, introducing subtle distortions that appear as grains when viewed closely. This visual transformation is a direct outcome of AES encryption, which ensures data security by transforming the image data into an encrypted format that can only be deciphered with the correct decryption key. The grainy appearance serves as a visual representation of the encryption's effectiveness in safeguarding sensitive information from unauthorized access.



Decryption Successful!

15674235869

```
Steganography Done.

Compression Done.

Encryption Done.

Encrypted text file saved.

Compression Rate:
0.5472364558977165
Encryption Strength:
0.5468012129045087
Data Dimming Rate:
0.0
0.0
Decryption Done.

Decompression Done.

decode_steganography
```

Figure (6): Performing decryption

Figure(6) represents the decryption process within the GUI's third panel, where the ongoing progress is reflected in the text log box. Users can monitor the step-by-step decryption process, including steganography extraction, decompression, and AES decryption. The log box provides real-time updates on each decryption stage, ensuring transparency and enabling users to track the procedure comprehensively. Additionally, the decrypted result is displayed in the bottom panel of the GUI, allowing immediate access to the recovered data after successful decryption. This visual feedback enhances user experience by providing clear insights into the decryption workflow and ensuring the secure retrieval of encrypted information.

6.2 Evaluation

The results of the study show that the system that is proposed here which combines steganography, image compression and cryptography was able to secure the data effectively. The system was able to successfully defend the data against the Brute force attacks generated using Kali Linux tools. The objectives set for the study were achieved successfully. Objective 1 was achieved successfully as the text data was successfully hidden in an image using the LSB technique. The stego image was then compressed successfully using the LZW compression technique and from this, it can be seen that objective 2 was achieved. The compressed was successfully encrypted and decrypted using AES which means that objective 3 was achieved. A desktop application was built using Tkinter library in Python which showed that objective 4 was achieved. Objective 5 was achieved as it was seen that the steganography, image compression and cryptographic techniques were successfully integrated into the desktop application. The performance of the system was tested by generating a Brute force attack using Kali Linux tools and objective 6 was successfully achieved.

The research question of the study was :

How would the merging of steganography, LZW compressor, and AES encryption mechanisms strengthen the confidentiality of archived PII's?

This research question was answered as from figures(4) and (5) it can be seen that the PIIs were hidden successfully in an image and the image was successfully encrypted. The Brute force attacks generated using Kali Linux tools were not able to extract the data from the encrypted files.

The system built in the study was able to successfully secure PIIs in text. However, if the amount of data to be secure is large in size then the steganography methods may have issues in properly hiding the data in a carrier image or file.

7 Conclusion and future enhancements

The study proposed here built a system that combined steganography, image compression and cryptographic techniques. The steganography was implemented using LSB steganography, image compression was implemented using the LZW compression technique and the image was encrypted using AES. The steganography, image compression and cryptographic techniques were successfully implemented and these three techniques were integrated successfully into a desktop application. The desktop application was able to successfully hide the text given as input by the user in an image and store the image after compression and encryption. The system also retrieved the image and converted the image back into its original form after decryption. The AES, LZW and LSB techniques and the desktop application were implemented successfully using methods in Python. The ability of the system proposed here to secure PIIs against attacks was tested by generating Brute force attacks using Kali Linux tools.

In future studies, other cryptographic techniques or asymmetric cryptographic techniques can be used for securing PIIs. Currently, the system built here allows users to secure PIIs in future studies the system can be transformed such that it can be used to secure the information that is sent between two individuals.

References

Addepalli and Lakshmi (2021).An Efficient Lossless Medical Data Compression using LZW compressionfor OptimalCloud Data Storage. https://www.researchgate.net/publication/353514407_An_Efficient_Lossless_Medical_Data_Compression_using_LZW_compressionfor_OptimalCloud_Data_Storage.

Alanzy, M., Alomrani, R., Alqarni, B. and Almutairi, S. (2023). Image Steganography Using LSB and Hybrid Encryption Algorithms. Applied Sciences, [online] 13(21), p.11771. doi:https://doi.org/10.3390/app132111771.

Ani , P. (2024). Data records breached worldwide Q4 2023. [online] Statista. Available at: https://www.statista.com/statistics/1307426/number-of-data-breaches-worldwide/#:~:text=Global%20number%20of%20breached%20user%20accounts%20Q1%202020%2DQ4%202023&text=During%20the%20fourth%20quarter%20of.

Arun, Juhi and Harsh (2023). Steganography in Images Using LSB Technique. https://www.researchgate.net/publication/371671984_Steganography_in_Images_Using_LSB_Technique

Aslam, M.A., Rashid, M., Azam, F., Abbas, M., Rasheed, Y., Alotaibi, S.S. and Anwar, M.W. (2022). Image Steganography using Least Significant Bit (LSB) - A Systematic Literature Review. [online] IEEE Xplore. doi:https://doi.org/10.1109/ICCIT52419.2022.9711628.

Bedoui, M., Mestiri, H., Bouallegue, B., Hamdi, B. and Machhout, M. (2022). An improvement of both security and reliability for AES implementations. Journal of King Saud University - Computer and Information Sciences. doi:https://doi.org/10.1016/j.jksuci.2021.12.012.

Bedoui, M., Mestiri, H., Bouallegue, B., Hamdi, B. and Machhout, M. (2022). An improvement of both security and reliability for AES implementations. Journal of King Saud University - Computer and Information Sciences. doi:https://doi.org/10.1016/j.jksuci.2021.12.012.

C.P, S., T, S. and G, U. (2013). A Study of Various Steganographic Techniques Used for Information Hiding. International Journal of Computer Science & Engineering Survey, [online] 4(6), pp.9–25. doi:https://doi.org/10.5121/ijcses.2013.4602.

Hamouda, B.E.H.H. (2020). Comparative Study of Different Cryptographic Algorithms. Journal of Information Security, 11(03), pp.138–148. doi:https://doi.org/10.4236/jis.2020.113009.

Kumar, S., Patidar, K., Kushwah, R. and Chouhan, S. (2017). A review and analysis on text data encryption techniques. International Journal of Advanced Technology and Engineering Exploration, 4(30), pp.88–92. doi:https://doi.org/10.19101/ijatee.2017.430003.

Matina Ch. Zerva, Christou, V., Nikolaos Giannakeas, Tzallas, A.T. and Kondi, L.P. (2023). An Improved Medical Image Compression Method Based on Wavelet Difference Reduction. IEEE access, 11, pp.18026–18037. doi:https://doi.org/10.1109/access.2023.3246948.

Maulunida, R. and Solichin, A. (2018). Optimization of LZW Compression Algorithm With Modification of Dictionary Formation. IJCCS (Indonesian Journal of Computing and Cybernetics Systems), 12(1), p.73. doi:https://doi.org/10.22146/ijccs.28707.

Mona, Ahmed, Fathi and Ayman H. Abd El-aziem (2014). Image Security With Different Techniques Of Cryptography And Coding: A Survey. IOSR journal of computer engineering, 16(3), pp.39–45. doi:https://doi.org/10.9790/0661-16313945.

Nashat, D. and Mamdouh, L. (2019). An efficient steganographic technique for hiding data. Journal of the Egyptian Mathematical Society, 27(1). doi:https://doi.org/10.1186/s42787-019-0061-6.

Nithya, B and P., Sripriya. (2016). A review of cryptographic algorithms in network security. 8. pp.324-331.
https://www.researchgate.net/publication/299186349_A_review_of_cryptographic_algorithms_in_network_security

Obaid, T.S. (2020). Study A Public Key in RSA Algorithm. European Journal of Engineering Research and Science, 5(4), pp.395–398. doi:https://doi.org/10.24018/ejers.2020.5.4.1843.

Packetlabs (2022). Cryptography Attacks: 6 Types & Prevention. [online] Packetlabs. Available at: https://www.packetlabs.net/posts/cryptography-attacks/.[Accessed 28 June 2024]

Patil, P., Narayankar, P., Narayan D.G. and Meena S.M. (2016). A Comprehensive Evaluation of Cryptographic Algorithms: DES, 3DES, AES, RSA and Blowfish. Procedia Computer Science, 78, pp.617–624. doi:https://doi.org/10.1016/j.procs.2016.02.108.

Peng, X., Jiang, J., Tan, L. and Hou, J. (2020). 2-D Bi-Level Block Coding for Color Image Compression and Transmission With Bit-Error Awareness. IEEE Access, 8, pp.110093–110102. doi:https://doi.org/10.1109/access.2020.3001073.

Refaat Said (2021). LSB technique in image steganography.
https://www.researchgate.net/publication/354543297_LSB_technique_in_image_steganography

Shetty, N. (2017). Steganography for Secure Data Transmission. International Journal of Computational Intelligence Research, [online] 13(10), pp.2289–2295. Available at: https://www.ripublication.com/ijcir17/ijcirv13n10_01.pdf [Accessed 28 Jun. 2024].

Solichin, A. and Erwin Wahyu, R. (2017). Enhancing data security using DES-based cryptography and DCT-based steganography. In: 2017 3rd International Conference on Science in Information Technology (ICSITech). IEEE. 10.1109/ICSITech.2017.8257187

Tang, H., Sun, Q.T., Yang, X. and Long, K. (2018). A Network Coding and DES Based Dynamic Encryption Scheme for Moving Target Defense. IEEE Access, 6, pp.26059–26068. doi:https://doi.org/10.1109/access.2018.2832854.

Zhang, F., Li, Z., Wen, M.C., Jia, X. and Chen, C. (2011). Implementation and Optimization of LZW Compression Algorithm Based on Bridge Vibration Data. Procedia Engineering, [online] 15, pp.1570–1574. doi:https://doi.org/10.1016/j.proeng.2011.08.292.