**1. INTRODUCTION**

In today's interconnected world, the digital exchange of information plays a critical role in various sectors, including finance, healthcare, education, and government. As the volume of digital data transmission and storage grows exponentially, so does the potential for sensitive information to fall into the wrong hands. Traditional security measures, while necessary, are no longer sufficient on their own to protect data against increasingly sophisticated cyber threats. This scenario demands innovative approaches that not only secure data through encryption but also conceal its very existence to prevent unauthorized access and enhance overall data confidentiality.

This project, "Enhancing Data Confidentiality Through Dual-Layer Cybersecurity Tool," is born out of the need for a more comprehensive approach to data security. It recognizes that the strength of a chain lies not just in its strongest link but in the concealment of the chain itself. By integrating the robust encryption capabilities of AES Cryptography with the covert nature of LSB Image Steganography, this project proposes a novel security model that aims to offer a dual layer of protection. This model encrypts the data, making it unreadable to unauthorized parties, and then hides this encrypted data within images, making the detection of the presence of any encrypted data significantly more difficult.

The relevance of this project extends beyond academic interest, addressing a critical need in various practical applications. In environments where data breaches can have catastrophic consequences, such as in national security or personal privacy cases, the proposed model offers an additional layer of security that can be the difference between the safety and exposure of sensitive information. Thus, this project not only contributes to the academic field of data security but also has significant implications for real-world applications where the confidentiality and integrity of data are of paramount importance.

* 1. **Project Overview**

In the digital era, the protection of sensitive information has become paramount. With the advent of sophisticated cyber threats, traditional encryption methods alone are often insufficient to guarantee the absolute confidentiality of transmitted data. This project introduces an innovative security model that combines the robustness of Advanced Encryption Standard (AES) cryptography with the subtlety of Least Significant Bit (LSB) Image Steganography, aiming to enhance data confidentiality beyond conventional methods.

The integration of LSB Image Steganography and AES Cryptography leverages the strengths of both techniques: the AES algorithm ensures the encrypted data's resilience against cryptographic attacks, while LSB Steganography conceals the very existence of the encrypted data within images, making it less susceptible to interception or scrutiny. This dual-layered approach significantly complicates unauthorized access to the data, providing an enhanced level of security.

* 1. **Project Deliverables**

The project is structured to deliver a comprehensive solution that demonstrates the practical application and effectiveness of combining LSB Image Steganography with AES Cryptography for data confidentiality. The key deliverables of this project include:

* **A Functional Prototype:** A software tool that implements the proposed security model, capable of encrypting data using AES Cryptography, embedding the encrypted data into images through LSB Steganography, and extracting/decrypting the data from the images.
* **Performance Analysis Report:** An in-depth analysis of the prototype's performance, including encryption/decryption times, steganography processing times, and the impact on image quality.
* **Security Evaluation:** An evaluation of the security effectiveness, focusing on the resistance of the combined approach to various attack vectors, including cryptographic attacks against AES and steganalysis attacks aimed at detecting the presence of steganographic content.
* **Documentation and User Guide:** Comprehensive documentation detailing the system architecture, codebase, and usage instructions for the prototype tool, ensuring accessibility for future development and application.

**1.3 Project Scope**

The scope of this project encompasses the development and evaluation of a security model that combines LSB Image Steganography with AES Cryptography to enhance data confidentiality. Specific scope elements include:

* **Development of the Prototype:** Designing and coding the software tool that integrates AES encryption with LSB steganographic techniques.
* **Performance and Security Analysis:** Conducting rigorous tests to assess the prototype's efficiency, performance, and security, focusing on encryption/decryption speeds, steganography processing, image quality impact, and vulnerability to attacks.
* **Use Case Demonstrations:** Demonstrating the application of the prototype in real-world scenarios, illustrating its practicality for secure data transmission and storage.
* **Limitations and Future Work:** This project does not cover the development of new cryptographic algorithms or steganographic techniques but focuses on the innovative application and integration of existing, proven methods to create a more secure data confidentiality solution.

**2. LITERATURE SURVEY**

**[1] “A Review of Combined Effect of Cryptography & Steganography Techniques to Secure the Information”.**

This paper provides a review of the combined effect of cryptography and steganography techniques to secure the information shared over the internet. The comparison of the combined cryptography and steganography techniques evaluated on the basis of some selected parameters like their encryption technique, key length, security, and data is hidden capacity.

[<https://ieeexplore.ieee.org/document/9161128>]

**[2] “Secure Data Transfer Through Internet Using Cryptography and Image Steganography”.**

In this paper, we proposed a system that uses both cryptography and steganography to ensure two levels of security to the data. The purpose of this paper is to develop new methodology using XOR operation for encrypting the data and embedding the encrypted data into the image pseudo randomly using user chosen key.

[<https://ieeexplore.ieee.org/document/9368301>]

**[3] “CryptMe: A Cryptographic framework with Steganography for securing data”.**

This paper proposed an approach of a combination of Steganography, Cryptography, and Transposition cipher technique. In particular, it focuses on the LSB technique in another way i.e., while performing the LSB steganography technique instead of embedding data at LSB of the base image, this method takes MSB of Video Frame and LSB of Stego Image. This combination helps in providing very little distortion and higher quality of the secure image. The value of efficiency of the algorithm would be determined by the PSNR value.

[<https://ieeexplore.ieee.org/document/9445>]

**[4] “Enhanced Digital Image and Text Data Security Using Hybrid Model of LSB Steganography and AES Cryptography Technique”**

Manish Kumar; Aman Soni; Ajay Raj Singh Shekhawat; Akash Rawat

In the present innovation, for the trading of information, the internet is the most well-known and significant medium. With the progression of the web and data innovation, computerized media has become perhaps the most famous and notable data transfer tools. This advanced information incorporates text, pictures, sound, video etc moved over the public organization. The majority of these advanced media appear as pictures and are a significant part in different applications, for example, chat, talk, news, website, web-based business, email, and digital books. The content is still facing various challenges in which including the issues of protection of copyright, modification, authentication. Cryptography, steganography, embedding techniques is widely used to secure the digital data. In this present the hybrid model of LSB steganography and Advanced Encryption Standard (AES) cryptography techniques to enhanced the security of the digital image and text that is undeniably challenging to break by the unapproved person. The security level of the secret information is estimated in the term of MSE and PSNR for better hiding required the low MSE and high PSNR values.

**[5] “An Enhanced Method for Encrypting Image and Text Data Simultaneously using AES Algorithm and LSB-Based Steganography”**

Md. Simul Hasan Talukder; Md. Nahid Hasan; Rafi Ibn Sultan; Ajay Krishno Sarkar; Sharmin Akter

Secure data transmission in the virtual world is becoming increasingly difficult due to the activities of hackers on confidential and sensitive information. Data security issues have led to developing a stronger algorithm in cryptography and steganography which have also led us to develop a secure data transmission model by concealing both image and text data simultaneously through the Advanced Encryption Standard (AES) algorithm and LSB-based image steganography. On the sender side, first, the targeted image is encrypted using AES then the targeted text data is concealed within the AES encrypted image based on the LSB algorithm. The data security is increased further by masking this blended text image with a cover image using LSB-based image steganography. Finally, the Stego image is split and indexed for secure data transmission from the sender end. Histogram analysis at a different stage has shown a better realization of the effect of encryption and decryption whereas various comparison indexes such as SSIM, PSNR, MS-SSIM, MSE between the original and retrieved image provide acceptable data integrity after decryption of the input data.

**[6] “Image Steganography Using Stego with AES and LSB”**

Lalit Negi; Lokesh Negi

Nowadays mobile phone becoming one of the most popular communication system. Information shared through this medium is a very sensitive to the users. Hence it is highly needed to secure the message from the intruders. This paper proposed an android based secured system named Steg! developed by combining the cryptography and steganography. Here the algorithm used for cryptography is Advanced Encryption Standard (AES) and Least Significant Bit is used for the steganography. This hybrid approach increases the level of secretion of information from unauthorized the access by encrypting the message and hiding into the image. The application helps the user to hide/unhide the text to/from the image. The proposed system above is proven to be powerful and robust than those system which implements cryptography and steganography alone.

**[7] “Metamorphic Cryptography Using AES and LSB Method”**

Aryansh Gupta; Anwar Ali; Aditya Kumar Pandey; Ankit Kumar Gupta; Abhinandan Tripathi

Now a days, internet users has led to the significant growth of cyber attack incidents. Therefore, it is essential to secure sensitive data before transmission to guarantee the secrecy of information and prevent unauthorized data alteration or loss. This issue can be solved using cryptography and steganography techniques. This paper uses an LSB-based image steganography algorithm to hide the message. Before being put into the cover image, the secret message is encrypted using the AES cryptographic algorithm while the secret message can be a text string, pdf, image, audio, or video. As a result, only the authorized user with the decryption key can extract the actual message from the stego object. Thus, it is still challenging to get the secret message when an attacker discovers the message. It uses Python version 3.9.10 to implement the proposed algorithm.

**[8] “An improved approach for lsb-based image steganography using AES algorithm”**

Sofyane Ladgham Chikouche; Noureddine Chikouche

The steganography is the art of hidden; its main aim is to pass unnoticed data in another data. There are many types of data that used in steganography, such as message, image, and video. In this work, we are interested in hiding a message inside an image. Our work focuses on the study of three approaches based on least significant bit (LSB) techniques that mean put the bits of the message in the least significant bits in each pixel of the image. Moreover, we propose an improved approach for LSB-based image steganography. In this approach, we reduce the length of hidden message by Deflate algorithm which is a lossless data compression algorithm that combines the LZ77 algorithm and the Huffman algorithm. Another important characteristic of our approach is to protect the reduced hidden data by AES (Advanced Encryption Standard) algorithm. Our experiment results show that our improved approach is most effective compared to existing approaches.

**[9] “The Application Of LSB Steganography For Secure Text and Hiding Confidential Information Using AES Cryptography”**

Ahyuna; Sadly Syamsuddin; Hasriani Hasriani; Ardimansyah; Irmawati; Sri Wahyuni

Maintaining the confidentiality and security of a company's message are two fundamental aspects, primarily if the message is used for business purposes. It will be perilous for the company if the message file falls into the wrong hands. The encryption technique we use in this application is Advanced Encryption Standard (AES) cryptography combined with Least Significant Bit (LSB) steganography techniques in mp3 extension files. The integration of the two techniques aims to provide protection or security systems for secret file messages in the form of images or images sent. This study indicates that the application of a combination of AES and LSB techniques can be used on document files so that the application can provide information security solutions for highly confidential company documents both when the document is sent. When the document is stored, then there are no errors in the functionality of the application. As evidenced by the results of comparing the encryption and decryption process of messages through the application with manual encryption and decryption of messages.

**[10] “An efficient filtering based approach improving LSB image steganography using status bit along with AES cryptography”**

Md. Rashedul Islam; Ayasha Siddiqa; Md. Palash Uddin; Ashis Kumar Mandal

In Steganography, the total message will be invisible into a cover media such as text, audio, video, and image in which attackers don't have any idea about the original message that the media contain and which algorithm use to embed or extract it. In this paper, the proposed technique has focused on Bitmap image as it is uncompressed and convenient than any other image format to implement LSB Steganography method. For better security AES cryptography technique has also been used in the proposed method. In the proposed technique, a new Steganography technique is being developed to hide large data in Bitmap image using filtering based algorithm, which uses MSB bits for filtering purpose. It is being predicted that the proposed method will able to hide large data in a single image retaining the advantages and discarding the disadvantages of the traditional LSB method. Various sizes of data are stored inside the images and the PSNR are also calculated for each of the images tested. Based on the PSNR value, the Stego image has higher PSNR value as compared to other method. Hence the proposed Steganography technique is very efficient to hide the secret information inside an image.

**[11]“Modified Encryption Standard for Reversible Data Hiding using AES and LSB Steganography”**

B Elisha Raju; M Ravi Sankar; T Saran Kumar; K Ramesh Chandra; V Bhavani Durga

The development in the cloud computing technologies and the information security policy, the data recovery from the hidden technologies is improved and also it is most implemented methodology. We proposed encryption algorithm using Least Significant Bit (LSB) and Auxiliary syndrome Encryption Standard (AES). In this approach, we first create an auxiliary syndrome for the data using AES algorithm and then embed the data in the least bit extracted using the wavelets of the cover image.The embedded data can be effectively retrieved and the original picture can be correctly restored on the receiver side utilizing the encryption key and extracted auxiliary syndrome. The proposed method gave the best analysis in terms of Peak Signal to Noise Ratio (PSNR) and Bit Error Rate (BER).

**3. PROBLEM ANALYSIS**

**3.1 Existing System**

The current landscape of digital data protection primarily revolves around encryption techniques and protocols to secure data during transmission or while at rest. Encryption, such as AES, scrambles data into an unreadable format for anyone who does not possess the decryption key, providing a strong layer of security. However, encryption alone does not address all aspects of data security, especially concerning the visibility of encrypted data, which can be a beacon for attackers, signifying that there's something worth intercepting or attacking.

**3.1.1 Challenges**

* **Visibility of Encrypted Data:** Even when data is encrypted, its presence is often evident. This visibility makes encrypted data a target for theft, interception, or further cryptographic attacks.
* **Vulnerability to Traffic Analysis:** Encrypted data, especially when transmitted over networks, can still be subjected to traffic analysis attacks, allowing malicious actors to infer sensitive information from patterns in data transmission.
* **Resource Intensive:** High levels of encryption can be resource-intensive, requiring significant computational power for encryption and decryption processes, which can be a limiting factor for devices with limited processing capabilities.
* **Lack of Anonymity:** In systems relying solely on encryption, there's often a lack of anonymity in data handling and storage, making it easier to trace data back to its source or destination.

**3.2 Proposed System**

In response to the limitations identified in the existing systems, our project introduces a novel approach to data security by seamlessly integrating Advanced Encryption Standard (AES) cryptography with Least Significant Bit (LSB) Image Steganography. This proposed system is designed to not only secure data through robust encryption but also to obscure the data's very presence, thereby offering a comprehensive solution to the challenges of data confidentiality in the digital age.

The cornerstone of the proposed system lies in its dual-layered security protocol. Initially, sensitive information is encrypted using the AES algorithm, renowned for its cryptographic strength and resilience against brute-force attacks. This encrypted data, while secure, remains vulnerable to detection and targeted attacks, which is where the second layer of security—LSB Image Steganography—comes into play. By embedding the encrypted data into digital images through subtle modifications of the least significant bits of the image pixels, the system effectively renders the encrypted data invisible to all but the most sophisticated analyses.

This approach not only capitalizes on the strengths of AES encryption but also mitigates its vulnerabilities by concealing the encrypted data within benign-looking images. Such images can be transmitted or stored without drawing attention, significantly reducing the likelihood of interception or unauthorized access. Moreover, the proposed system is designed with practicality in mind, ensuring that the process of embedding and extracting data does not degrade the image quality perceptibly, maintaining the usability of the steganography technique for real-world applications.

**3.2.1 Advantages**

* **Enhanced Data Concealment:** By embedding encrypted data within images using LSB Steganography, the proposed system effectively masks the presence of sensitive data, reducing the risk of targeted attacks.
* **Resistance to Traffic Analysis:** With data hidden within images, the proposed system makes it considerably more challenging for attackers to perform traffic analysis, as the transmissions may appear as innocuous image transfers rather than conspicuous encrypted data exchanges.
* **Resource Efficiency:** The proposed system leverages the computational efficiency of LSB Steganography, which requires relatively less processing power compared to high-level encryption algorithms, making it suitable for a wider range of devices.
* **Improved Anonymity:** The concealment of encrypted data within images offers an additional layer of anonymity, making it difficult to trace the data back to its source or intended destination, thereby protecting the identities of the communicating parties.
* **Versatility and Scalability:** This system is versatile and scalable, capable of being implemented across various platforms and for different applications, ranging from secure messaging to confidential data storage.

**4. SYSTEM ANALYSIS**

Systems analysis is a problem-solving method that breaks down a system into its parts to examine how effectively those parts function and interact to achieve their goals. It involves studying a process to understand its objectives and purposes, and then devising systems and procedures to efficiently achieve them.

The development of a computer-based information system includes a phase of systems analysis, which creates or improves the data model essential for building or improving a database. There are various approaches to system analysis. When developing a computer-based information system, the steps of systems analysis would typically include:

Conducting a feasibility study to determine if a project is economically, socially, technologically, and organizationally viable.

Performing fact-finding activities to understand the requirements of the system's end-users. This may involve interviews, questionnaires, or observing the current system in action. Assessing how end-users would interact with the system, such as their experience with computer hardware or software, and the system's intended purpose.

**4.1 System Requirement Specifications:**

A requirement is an attribute or limitation that the system needs to meet in order for the client to accept it. The goal of requirement engineering is to specify the specifications for the system that is being built.

The two primary steps in requirement engineering are requirement elicitation, which yields a client-understood system specification, and analysis, which produces an analysis model that the developer can clearly grasp. A requirement is a declaration of what the suggested system is expected to perform. There are two main types of requirements: non-functional requirements and functional requirements.

**4.1.1 Functional requirements**

Functional requirements outline how it interacts with its surroundings. Any other external system that the system communicates with is included in the environment, along with the user. Functional requirements define the desired behavior of the system, which can be articulated as tasks services, or other necessary actions that the system must carry out. It is helpful in product development to distinguish between characteristics that set a system apart from rivals' offerings and from variations within your company's own product line or family, as well as the basic capabilities required for any system to compete in that product area. Features could be more functionality or different functionality along a quality parameter from the fundamental functionality.

Functional requirements outline the specific actions the system must perform to meet user needs:

* **Encryption Module:** The system should support AES encryption, allowing users to specify encryption keys and modes.
* **Steganography Module:** A module must be implemented to embed encrypted data within digital images using LSB Image Steganography. Users should have the option to select cover images for embedding.
* **Decryption Module:** Users should be able to decrypt data hidden within images and encrypted with AES. The decryption process must require a decryption key.
* **User Interface:** The system must feature an intuitive graphical user interface (GUI) to guide users through encryption, embedding, and decryption processes.
  + 1. **Non-Functional Requirements**

Non-functional requirements list the features of the system that have no direct bearing on how it operates functionally. Non-functional requirements cover a wide range of specifications related to many aspects of the system, such as performance and usability.

* **Performance:** The system should operate efficiently without significant latency during data embedding, extraction, encryption, and decryption processes.Response times for user interactions should be minimal to provide a seamless experience.
* **Scalability:** The system should be capable of handling a large volume of data and images without a significant decrease in performance.It should support scaling to accommodate increasing user demand or data processing requirements.
* **Security:** Ensure robust security measures to protect sensitive data from unauthorized access or tampering. Implement secure storage mechanisms for encryption keys and embedded data to prevent data breaches.
* **Reliability:** The system should be highly reliable, with minimal chances of failure during data embedding, extraction, encryption, or decryption processes.Implement error handling and recovery mechanisms to handle unexpected errors gracefully.
* **Usability:** Design the user interface to be intuitive and easy to use, requiring minimal training for users to operate the system effectively.Ensure accessibility for users with disabilities by adhering to relevant guidelines and standards.
* **Compatibility:** Ensure compatibility with different operating systems and platforms to maximize the system's accessibility and usability.The system should support a variety of image formats commonly used by users.

**4.2 System Requirements**

A system requirements specification is a comprehensive list of the goals that the system must accomplish. Without committing to any particular method, a good specification provides a clear description of what the system is expected to achieve. Usually created in response to a user requirement specification or another form of need expression, a system requirements specification serves as the foundation for system design. The scope and accuracy of the expression of needs, which may span the intended system as well as the operating environment, may be different from the system requirement specification. The latter may also leave many broad notions unpolished.

Here, the System requirements specify the software and hardware needed to support the system:

* + 1. **Software requirements**
* Python
* Visual studio
* Operating System: Compatible with Windows 10, macOS Catalina (or later), and Ubuntu 20.04 (or later).
* AES cryptography and LSB Image Steganography algorithms.
* Image Processing Library: Utilization of image processing libraries such as OpenCV or Pillow for steganographic operations.
* GUI Framework: Implementation of the GUI using frameworks like Flask Web Framework, Bootstrap 4.

**4.2.2 Hardware requirements**

* Processor: Runs smoothly on processors with a clock speed of 1.6 GHz or higher.
* RAM: Recommends a minimum of 4 GB RAM for optimal performance.
* Storage: Requires at least 100 MB of free disk space for installation and temporary files.
* Display: Supports displays with a resolution of 1024x768 pixels or higher for effective GUI presentation.

**5. SYSTEM DESIGN**

**5.1 Introduction**

System design is the process of defining the elements of a system such as the architecture, modules and components, the different interfaces of those components and the data that goes through that system. It is meant to satisfy specific needs and requirements of a business or organization through the engineering of a coherent and well-running system. Systems design mainly concentrates on defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. Systems design could be seen as the application of systems theory to product development.

Systems design implies a systematic approach to the design of a system. It may take a bottom-up or top-down approach, but either way the process is systematic wherein it takes into account all related variables of the system that needs to be created—from the architecture, to the required hardware and software, right down to the data and how it travels and transforms throughout its travel through the system. Systems design then overlaps with systems analysis, systems engineering and systems architecture.

The systems design approach first appeared right before World War II, when engineers were trying to solve complex control and communications problems. They needed to be able to standardize their work into a formal discipline with proper methods, especially for new fields like information theory, operations research and computer science in general.

The main objective of system design is to create a solution that meets the requirements of the system in terms of functionality, performance, reliability, scalability, maintainability, and usability. It involves making design decisions such as selecting appropriate technologies, defining interfaces and data structures, allocating system resources, optimizing performance, and ensuring the system's overall integrity and robustness. The System Design phase is crucial in transforming the specified requirements into a blueprint for constructing the system. This phase elaborates on the architectural modules of the product, outlining the system's structure and detailing the interaction between its components. It also defines the data flow and control flow, ensuring the system's reliability, maintainability, and security standards are met.

**5.2 Object Oriented Analysis**

Object-oriented analysis (OOA) provides a solid foundation for understanding the system's structure and behavior by modeling it as a group of interacting objects. Each object represents an instance of a class within a particular category. This approach enables modularity, flexibility, and reusability in system design.

the steps involved in the analysis phase are

* Identify the actors
* Develop a simple business process model using UML activity diagram
* Prepare interaction diagrams
* Classification- develop a static UML class diagram Identify classes, relationships, attributes and methods.

**System models**

* Scenarios
* Use Case Model

Use case is a description of the behaviour of the system. That description is written from the point of view of a user who just told the system to do something particular.

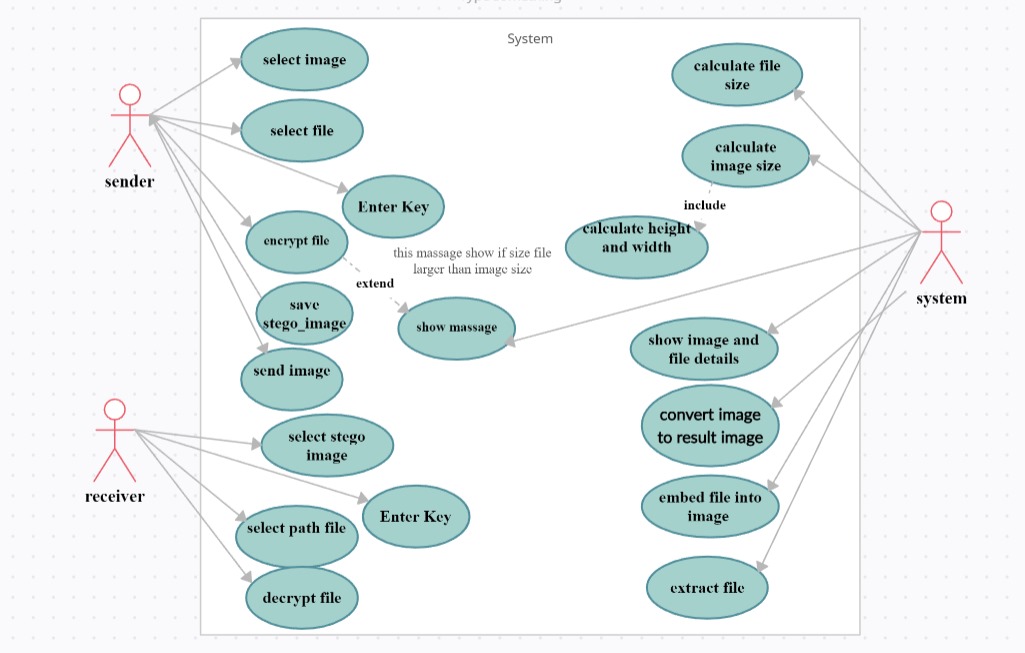
**5.2.1 Use-Case Scenario**

Consider the use case where user gives input data. The input data which is in raw format is then given for pre-processing. In pre-processing, missing values get eliminated and the input scales in between (0,1) to maintain uniformity. Then pre-processed data is given to analysing. The analysing part uses ResNet model and it gives the required prediction.

**Use-Case Diagrams**

An important part of the Unified Modelling Language (UML) is the facilities for drawing use case diagrams. Use cases are used during the analysis phase of a project to identify and partition system functionality. They separate the system into actors and use cases. Actors represent roles that can play by users of the system. Those users can be humans, other computers, pieces of hardware, or even other software systems. Use cases describe the behaviour of the system**.**

|  |  |  |
| --- | --- | --- |
| Actor | n actor as mentioned is a user of the system and is depicted using a stick figure. The role of the user is written beneath the icon. Actors are not limited to humans. If a system communicates with another application and expects input or delivers output then that application can also be considered as an actor. | Actor Role Name |
| Use case | Use Case is the functionality provided by the system typically described as verb + object |  |
| Directed Association | Associations are used to link Actors with use cases and indicates that an actor participates in the Use Case in some form. Directed Association is same as association but difference is that it represented by a line having an arrow head. |  |
| System boundary boxes | You can draw a rectangle around the use cases, called the system boundary box, to indicate the scope of your system. Anything within the box represents functionality that is in scope and anything outside the box is not. | System |

***Table 5.1 Graphical Notations for Use Case Diagram***

**Fig 5.1 Use Case Diagram of Dual layer tool**

**5.2.2 Class Diagram**

Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing and documenting17different aspects of a system but also for constructing executable code of the software application. It describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modeling of object-oriented systems because they are the only UML diagrams which can be mapped directly with object-oriented languages.

**Purpose**

The purpose of the class diagram is to model the static view of an application. The class diagrams are only diagrams which can be directly mapped with object-oriented languages and thus widely used at the time of construction. The UML diagrams like activity diagram, sequence

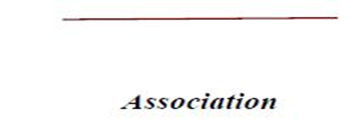
diagram can only give the sequence flow of the application but class diagram is a bit different. So, it is the most popular UML diagram in the coder community. So, the purpose of the class diagram can be summarized as:

* Analysis and design of the static view of an application.
* Describe the responsibilities of a system.
* Base for component and deployment diagrams.
* Forward and reverse engineering.

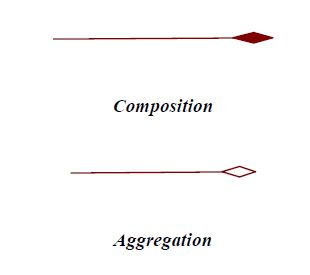
**Active Class:** Active classes initiate and control the flow of activity, while passive classes store data and serve other classes. Illustrate active classes with a thicker border.

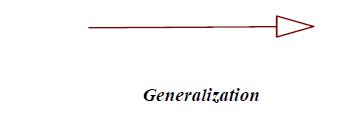
**Visibility:** Use visibility markers to signify who can access the information which is in a class. There are three visibilities. They are:

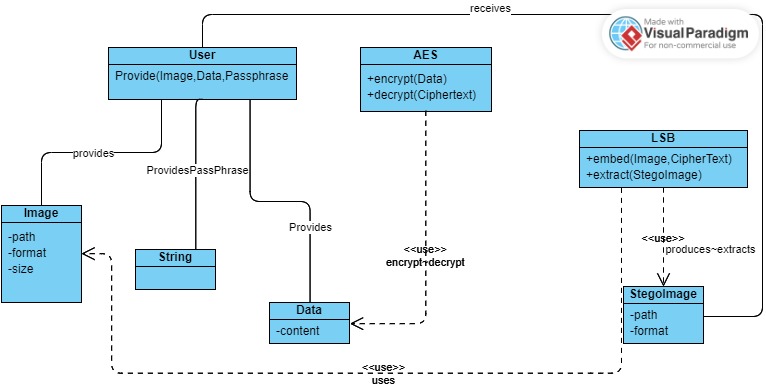
* Private visibility hides information from anything outside the class partition.
* Public visibility allows all other classes to view the marked information.
* Protected visibility allows child classes to access information which is in inherited from a parent class.

**Associations:** Associations represent static relationship between the classes. Place the association names above, on or below the association line. Use a filled arrow to indicate the direction of the relationship. Place roles at the end of an association. Roles represent how the two classes see each other.

**Multiplicity (Cardinality):** Place multiplicity notations at the ends of an association. The possible types pf multiplicity are as follows: One-to-many, Zero-or-one to one, Zero-or-one to many.

**Composition and Aggregation:** Composition and Aggregation links a semantic association between two classes in UML diagram. They are used in class diagram. They both differ in their symbols.

**Generalization**: It is a specification relationship in which objects of the specialized element (the child) are suitable for objects of the generalization element (the parent). It is used in class diagram.

**Class Diagram Scenario**

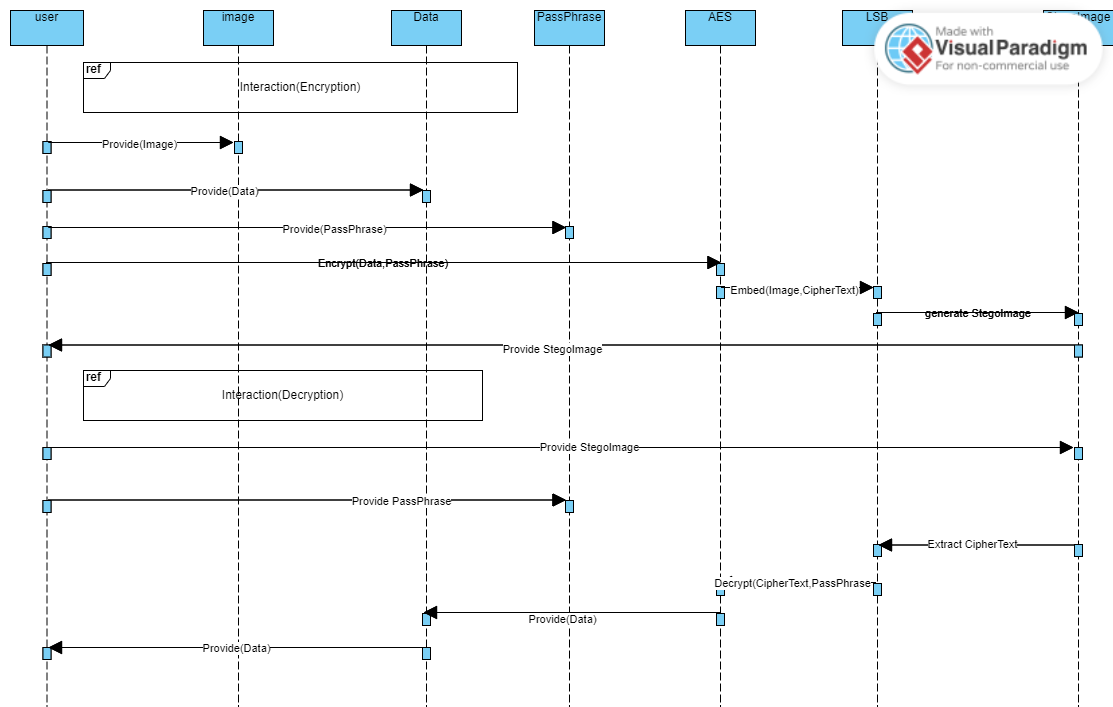
***Fig 5.2 Class Diagram for Dual layer tool***

The above class diagram depicts the recognition of emotion behind speech. The main classes participating in the system are Data aggregation, Pre-processing, Analyzing and output. Data aggregation node collects the data from different sources and stores the data. The raw data in Data aggregation node is taken as an input for Pre-processing node, which eliminates the missing values and applies different methods that maintains uniformity in data. This pre-processed data is given as an input for Analyzing node. This Analyzing node used ResNet model and predicts the required data.

**5.2.3 Sequence Diagram**

The sequence diagram represents the flow of messages in the system and is also termed as an event diagram. It helps in envisioning several dynamic scenarios. It portrays the communication between any two lifelines as a time-ordered sequence of events, such that these lifelines took part at the run time. In UML, the lifeline is represented by a vertical bar, whereas the message flow is represented by a vertical dotted line that extends across the bottom of the page. It incorporates the iterations as well as branching.

|  |  |  |
| --- | --- | --- |
| Object | Objects are instances of classes and are arranged horizontally. The pictorial representation for an Object is class (a rectangle) with the name prefixed by the object name (optional). |  |
| Actor | Actor can also communicated with objects so they too can be listed as a column. An Actor is modeled using the stick figure |  |
| Lifeline | The Lifeline identifies the existence of the object over time. The notation for a life time is a vertical dotted line extending from an object. |  |
| Activation | Activation modeled as rectangular boxes on the lifeline indicate when the object is performing an action. |  |

***0Table 5.2 Graphical Representation of Sequence Diagra******m***

***Fig 5.3 Sequence Diagram of Dual layer tool***

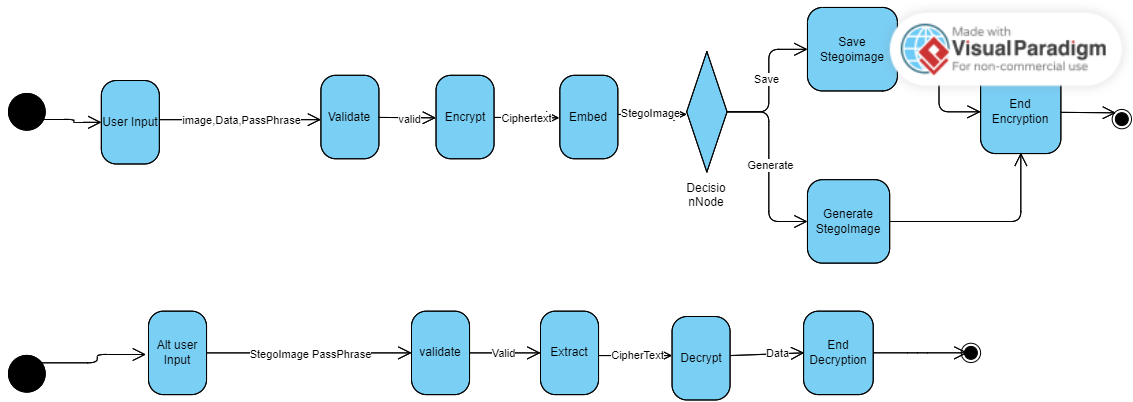
**5.2.4 Activity Diagram**

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control. Activity diagrams are constructed from a limited repertoire of shapes, connected with arrows.

The most important shape types:

* Rounded rectangle represents activities.
* Diamonds represent decisions.
* Bars represent the start (split) or end (join) of concurrent activities.
* A black circle represents the start (initial state) of the workflow.
* An encircled black circle represents the end (final state).
* Arrows run from the start towards the end and represent the order in which activities happen.

Activity diagrams can be used to model the interaction between the user and the speech emotion recognition system. For instance, the diagram can show the flow of activities for user input, system feedback, and system response. This can help in designing user-friendly interfaces and understanding the overall user experience during the emotion recognition process.

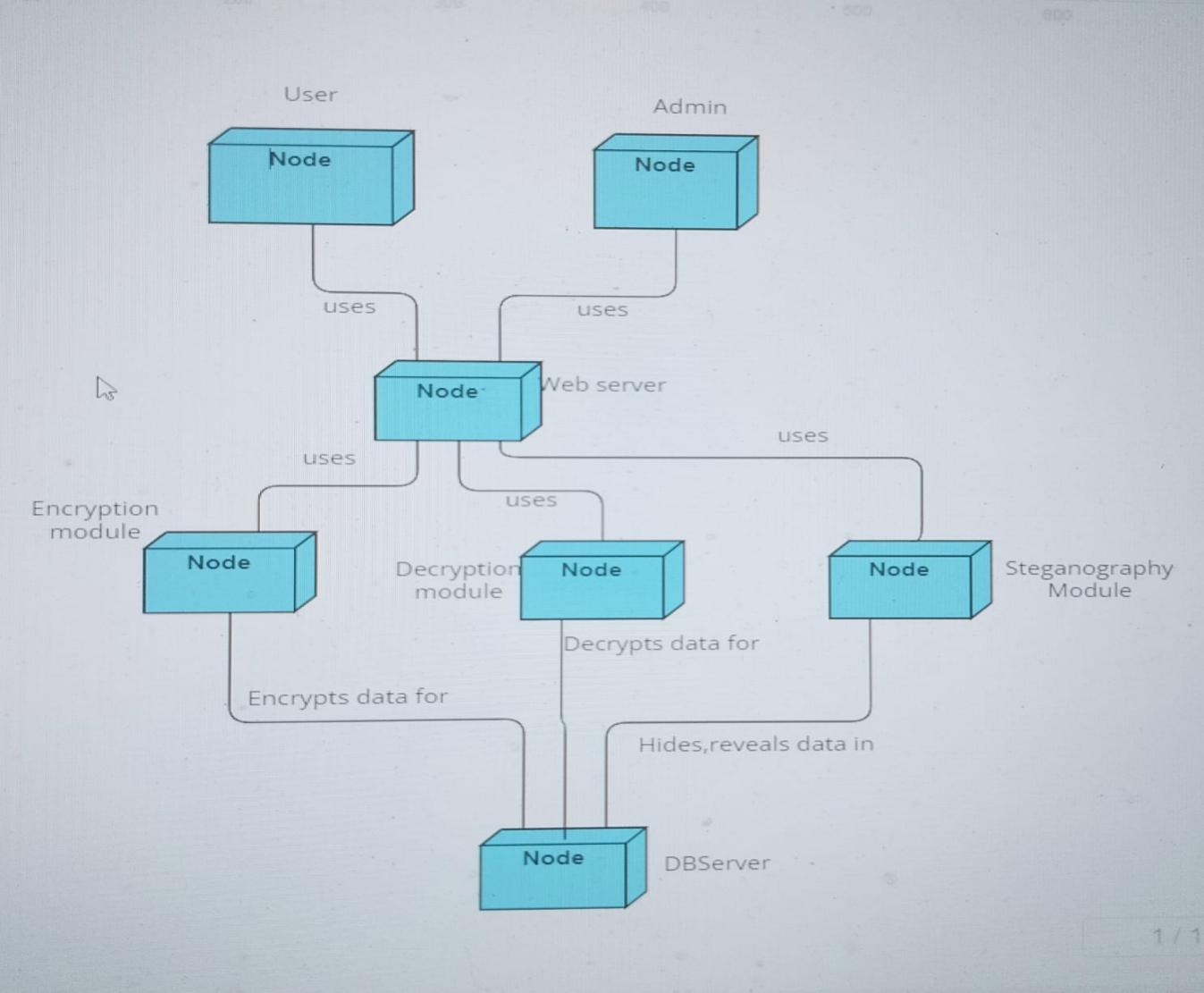
Overall, activity diagrams can provide a visual representation of the flow of activities and tasks in a speech emotion recognition system, helping to improve system understanding, design, and development.

***Fig 5.4 Activity Diagram for Dual layer tool***

**5.2.5 Deployment Diagram**

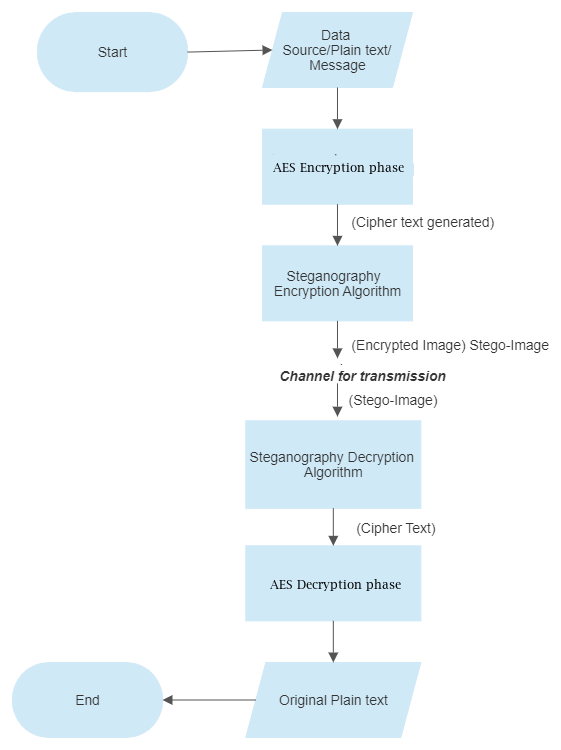
Deployment diagram is a structure diagram which shows architecture of the system as deployment (distribution) of software artifacts to deployment targets. Artifacts represent concrete elements in the physical world that are the result of a development process. Deployment Diagram is usually represented by a node which is either a hardware device or some software execution environment. Nodes could be connected through communication paths to create networked systems of arbitrary complexity.

Deployment diagrams could describe architecture at specification level (also called type level) or at instance level (similar to class diagrams and object diagrams).

**** In a speech emotion recognition project, a deployment diagram can be used to show how the various software components and hardware resources are organized and connected to facilitate speech emotion recognition. It can be used to visually represent the physical deployment and configuration of software components and hardware resources, including servers, devices, networks, and security measures. It helps in understanding the physical architecture of the system and how its components are organized and connected to facilitate speech emotion recognition.

***Fig 5.5 Deployment Diagram for Dual layer tool***

**5.2.6 System Architecture**



***Fig 5.6 System Architecture***

**5.3 Algorithm Description:**

**5.3.1 LSB Steganography Algorithm:**

Least Significant Bit (LSB) steganography is a method of hiding data within the least significant bits of an image's pixel values. This technique takes advantage of the fact that small changes to the least significant bits of pixel values are often imperceptible to the human eye. Here's an outline of the LSB steganography algorithm:

**Data Encoding:** The data to be hidden (e.g., encrypted message) is converted into binary form.

**Image Selection:** An image, known as the cover image, is selected as the carrier for the hidden data.

**Embedding Process:**

* Pixel Selection: Each pixel in the cover image is accessed sequentially.
* Data Embedding: The least significant bit (LSB) of the pixel's color channel (e.g., red, green, or blue) is replaced with a bit of the binary data to be hidden. This replacement is done in such a way that the change is imperceptible to the human eye.
* End-of-Data Marker: To signify the end of the hidden data, a special marker is appended to the binary data, ensuring proper extraction.

**Steganographed Image:** The resulting image, known as the steganographed image, contains the original cover image with the hidden data embedded within its pixel values.

**Data Extraction:** During data extraction, the LSB of each pixel in the steganographed image is extracted and concatenated to reconstruct the hidden binary data.

**Decoding:** The extracted binary data is converted back into its original form (e.g., decrypted message) for interpretation.

**5.3.2 AES Encryption Algorithm:**

The Advanced Encryption Standard (AES) algorithm is a symmetric key encryption algorithm widely adopted for securing sensitive data. It operates on fixed-size blocks of data, typically 128 bits, and uses a variable-length key, which can be 128, 192, or 256 bits. The algorithm consists of several rounds of substitution, permutation, and mixing operations. Here's a high-level overview of the AES encryption and decryption processes:

**Key Expansion:** The AES algorithm requires a key of specific lengths (128, 192, or 256 bits). If the provided key is not of the required length, it undergoes key expansion to generate a key schedule containing the necessary round keys for each round of encryption and decryption.

**Initial Round:** The input data is XORed with the first round key.

**Rounds:** The AES algorithm consists of multiple rounds, each comprising four distinct transformation steps: SubBytes, ShiftRows, MixColumns, and AddRoundKey.

**SubBytes:** Each byte of the state matrix is replaced with a corresponding byte from the S-box, a predefined substitution table.

**ShiftRows:** The rows of the state matrix are cyclically shifted by different offsets.

**MixColumns:** Each column of the state matrix is multiplied with a fixed polynomial modulo a predefined irreducible polynomial.

**AddRoundKey:** Each byte of the state matrix is XORed with a corresponding byte from the round key.

**Final Round:** The final round omits the MixColumns step to simplify the process.

**Output:** The final state matrix after the last round of transformation represents the encrypted data, known as ciphertext.

The decryption process follows a similar sequence of steps, but in reverse order, using the round keys in reverse order. Decryption involves reversing the transformations applied during encryption, including SubBytes, ShiftRows, MixColumns (with an inverse operation), and AddRoundKey.

This process ensures that the hidden data remains concealed within the image while preserving the visual integrity of the cover image. These algorithms form the core of the proposed system, enabling the encryption of data using AES and the concealment of encrypted data within images using LSB steganography.

**6.IMPLEMENTATION**

**6.1 Technology Description:**

**6.1.1 Python:**

Python is a high-level, general-purpose, interpreted programming language. Python was developed by Guido van Rossum and was originally made available in 1991. Its design philosophy places a strong emphasis on code readability and makes extensive use of whitespace. Its object-oriented methodology and language elements are designed to assist programmers in writing logical, understandable code for both small and large-scale projects.

Python uses garbage collection and dynamic typing. It is compatible with several programming paradigms, such as functional, object-oriented, and procedural programming. Because of its extensive standard library, Python is frequently referred to as a "batteries included" language.  
 The idea for Python emerged in the late 1980s as an ABC replacement language. With the release of Python 2.0 in 2000, features like list comprehensions and a garbage collection system that could gather reference cycles were included. Following the official discontinuation of the Python 2 language, or Python 2.7.x, on January 1, 2020 (originally scheduled for 2015), security updates and other enhancements will no longer be made available for it. Only Python 3.5 and later are supported due to the end of life of Python 2.

There are interpreters for Python on a wide range of operating systems. An opensource reference implementation called CPython is developed and maintained by a global community of programmers. The Python Software Foundation, a non-profit, oversees and provides resources for Python and CPython development.

**Characteristics of Python:**

* **Ease of use:** Python is an easy and uncomplicated language. Reading a well-written Python program is like to reading English, but extremely formal English! One of the best things about python is that it is essentially pseudo-code. It enables you to focus on the problem's solution rather than the language itself.
* **Simple to Learn:** Python is quite simple to begin using, as you shall discover. As was already established, Python has a very simple syntax.
* **Open Source and Free:** One type of FLOSS (Free/Libre and OpenSource Software) is Python. To put it simply, you are allowed to examine the source code, distribute copies of this software
* freely, alter it, and incorporate parts of it into other free applications.
* **Easy to Read:** As you will see, learning Python is quite simple. As was already established, Python’s syntax is really straightforward. The code block is defined by the indentations rather than by semicolons or brackets.
* **Object-Oriented Language:** One of the key features of Python is Object-Oriented programming. Python supports object-oriented language and concepts of classes, object encapsulation, etc.
* **Easy to Debug:** Excellent information for mistake tracing. You will be able to quickly identify and correct the majority of your program’s issues once you understand how to interpret Python’s error traces. Simply by glancing at the code, you can determine what it is designed to perform.

**6.1.2** **Numpy:**  
 The core Python library for scientific computing is called NumPy. A multidimensional array object, different derived objects (like masked arrays and matrices), and a variety of routines for quick array operations—like sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation, and much more—are all provided by this Python library. The ndarray object is the central component of the NumPy package.  
 This contains homogeneous data types in n-dimensional arrays, and numerous operations are carried out in compiled code for efficiency. The conventional Python sequences and NumPy arrays differ significantly in the following ways:

NumPy arrays are created with a fixed size, in contrast to Python lists, which can increase as needed. A ndarray's size changes result in the creation of a new array and the deletion of the old one.

Since every element in a NumPy array must be of the same data type, they will all have the same amount of memory space. The exception is that arrays of (Python, including NumPy) objects are allowed, so that arrays with elements of varying sizes can be created.

NumPy arrays make complex mathematical and other kinds of operations on massive amounts of data easier. These actions are typically carried out more quickly and with less code than when utilizing Python's built-in sequences.

**6.1.3 Pandas DataFrames:**

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. The name Pandas is derived from the word Panel Data – an Econometrics from Multidimensional data.

In 2008, developer Wes McKinney started developing pandas when in need of high performance, flexible tool for analysis of data. Prior to Pandas, Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data — load, prepare, manipulate, model, and analyse.

Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

**6.1.4 BitVector:**

BitVector is a Python library designed for handling bit arrays efficiently. It provides a memory-efficient packed representation of bit arrays and supports various logical and bitwise operations on these arrays. Developed with a focus on performance and ease of use, BitVector simplifies working with binary data in Python applications.

A bit array (also known as bitmask,[1] bit map, bit set, bit string, or bit vector) is an array data structure that compactly stores bits. It can be used to implement a simple set data structure. A bit array is effective at exploiting bit-level parallelism in hardware to perform operations quickly. A typical bit array stores kw bits, where w is the number of bits in the unit of storage, such as a byte or word, and k is some nonnegative integer. If w does not divide the number of bits to be stored, some space is wasted due to internal fragmentation.

**Key Features and Benefits of BitVector:**

* **Memory Efficiency:** BitVector stores bits compactly, consuming minimal memory compared to storing individual boolean values. This makes it ideal for applications that deal with large volumes of binary data, such as cryptography, data compression, and network protocols.
* **Efficient Operations:** BitVector supports a wide range of logical and bitwise operations, including AND, OR, XOR, NOT, as well as left and right shifts. These operations can be performed efficiently even on large bit arrays, enabling rapid manipulation and analysis of binary data.
* **Flexibility:** BitVector offers flexibility in accessing and manipulating individual bits within a bit array. Users can easily index and slice bit vectors, allowing for granular control over binary data.
* **Conversion Utilities:** BitVector provides utilities for converting bit vectors to other formats, such as integers, strings, hexadecimal, and binary representations. This facilitates seamless integration with existing codebases and simplifies data interchange between different formats.
* **Cross-Platform Compatibility:** BitVector is compatible with major operating systems and Python versions, ensuring consistent behavior across different environments. This makes it a reliable choice for projects targeting diverse platforms and ecosystems.

**6.1.5 Flask - A Microframework for Python Web Development:**

Flask is a lightweight and flexible web framework for Python. It's often referred to as a "microframework" because it provides the essentials for building web applications without imposing strict dependencies or project structure. Flask prioritizes simplicity, allowing developers to quickly create web applications with minimal overhead.

Flask was created by Armin Ronacher of Pocoo, an international group of Python enthusiasts formed in 2004. According to Ronacher, the idea was originally an April Fool's joke that was popular enough to make into a serious application. The name is a play on the earlier Bottle framework.

When Ronacher and Georg Brandl created a bulletin board system written in Python in 2004, the Pocoo projects Werkzeug and Jinja were developed.

In April 2016, the Pocoo team was disbanded and development of Flask and related libraries passed to the newly formed Pallets project. Since 2018, Flask-related data and objects can be rendered with Bootstrap.

Flask has become popular among Python enthusiasts. As of October 2020, it has the second-most number of stars on GitHub among Python web-development frameworks, only slightly behind Django, and was voted the most popular web framework in the Python Developers Survey for years between and including 2018 and 2022.

**Key Features and Benefits of Flask:**

* **Minimalistic Design:** Flask is designed to be lightweight and unopinionated, providing just the essentials for web development without imposing unnecessary complexity. This minimalist approach allows developers to focus on writing clean and concise code.
* **Easy to Learn:** Flask's simplicity makes it easy for beginners to learn and understand. Its intuitive API and straightforward documentation enable developers to get started quickly, making it an excellent choice for prototyping and small projects.
* **Flexible and Extensible:** Flask is highly flexible and extensible, allowing developers to customize and extend its functionality as needed. It supports a wide range of extensions and plugins, enabling integration with various third-party libraries and services.
* **Built-in Development Server:** Flask includes a built-in development server, making it convenient for testing and debugging web applications during the development process. The development server supports automatic code reloading, facilitating rapid iteration and experimentation.
* **Jinja2 Templating:** Flask integrates seamlessly with the Jinja2 templating engine, which enables developers to build dynamic and interactive web pages using templates. Jinja2 provides powerful features such as template inheritance, macros, and filters, enhancing the flexibility and maintainability of Flask applications.
* **RESTful Routing:** Flask promotes RESTful routing and follows the principles of the HTTP protocol, making it well-suited for building RESTful APIs and web services. It provides decorators for defining routes and handling HTTP methods, simplifying the implementation of CRUD operations and resource-based endpoints.
* **Werkzeug WSGI Toolkit:** Flask is built on top of the Werkzeug WSGI (Web Server Gateway Interface) toolkit, which provides low-level HTTP request and response handling. This underlying architecture gives Flask robust HTTP handling capabilities while maintaining a lightweight and modular design.
* **Community and Ecosystem:** Flask has a vibrant community of developers and enthusiasts who contribute to its ecosystem by creating extensions, tutorials, and resources. The Flask community fosters collaboration and knowledge sharing, providing support and guidance to developers at all skill levels.

**Visual Studio Code:**

Visual Studio Code is a free, lightweight but powerful source code editor that runs on your desktop and on the web and is available for Windows, macOS, Linux, and Raspberry Pi OS. It comes with built-in support for JavaScript, TypeScript, and Node.js and has a rich ecosystem of extensions for other programming languages (such as C++, C#, Java, Python, PHP, and Go), runtimes (such as .NET and Unity), environments (such as Docker and Kubernetes), and clouds (such as Amazon Web Services, Microsoft Azure, and Google Cloud Platform).

Aside from the whole idea of being lightweight and starting quickly, Visual Studio Code has IntelliSense code completion for variables, methods, and imported modules; graphical debugging; linting, multi-cursor editing, parameter hints, and other powerful editing features; snazzy code navigation and refactoring; and built-in source code control including Git support. Much of this was adapted from Visual Studio technology.

**6.2 Sample Source Code**

**App.py**

# Import necessary modules

from flask import Flask, render\_template, request, redirect, url\_for, send\_from\_directory,send\_file

from mainapp import Encryption, Decryption

import os

import io

# Initialize Flask application

app = Flask(\_\_name\_\_)

# Route for home page

@app.route('/')

def home():

return render\_template('index.html')

# Route for handling encryption

@app.route('/encrypt', methods=['POST'])

def encrypt():

if request.method == 'POST':

# Get the file data from the form

coverimage = request.files['coverimage']

data = request.form['data']

key = request.form['key']

# Read the file data into memory

coverimage\_data = coverimage.read()

# Encrypt the data and save the encrypted image to a file

encrypted\_image\_path = Encryption(io.BytesIO(coverimage\_data), data, key)

# Send the encrypted image file to the client

return send\_file(encrypted\_image\_path, mimetype='image/png', as\_attachment=True)

# Route for handling decryption

@app.route('/decrypt', methods=['POST'])

def decrypt():

if request.method == 'POST':

stego\_image = request.files['stegoImage']

key = request.form['key']

# Assuming your Decryption class requires stegoimage and key

decryption = Decryption(stego\_image, key)

decrypted\_data = decryption

# Simply return the decrypted data as a response

return decrypted\_data

else:

return 'Method Not Allowed', 405

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

**Mainapp.py  
# the mainbackend**

from AESCryptography import AESencrypt,AESdecrypt

from LSBSteganography.Steganography\_Encryption import StegoEncryption

from LSBSteganography.Steganography\_Decryption import StegoDecryption

import os

def Encryption(coverimage, data, key=0):

ciphertext = AESencrypt.encrypt(data, key)

newimage = StegoEncryption.encode(coverimage, \*ciphertext)

result\_filename = f"result\_image.png"

# result\_image\_path = os.path.join(os.path.dirname(\_\_file\_\_), 'web', 'static', result\_filename)

result\_image\_path = os.path.join(os.path.dirname(\_\_file\_\_), 'static', 'images', result\_filename)

newimage.save(result\_image\_path)

# return newimage

return result\_image\_path

def Decryption(stegoimage, key=0):

ciphertext = StegoDecryption.decode(stegoimage)

print(ciphertext)

data = AESdecrypt.decrypt(ciphertext, key)

return data

**Steganography\_Encryption.py**

from PIL import Image

import os

class StegoEncryption:

def \_\_init\_\_(cls):

cls.data = ""

@classmethod

def encode(cls,img,data):

cls.data = data

cls.image = Image.open(img, 'r')

cls.newimg = cls.image.copy()

cls.encode\_enc()

return cls.newimg

# new\_img\_name = input("Enter the name of new image(with extension) : ")

# cls.newimg.save(new\_img\_name,str(new\_img\_name.split(".")[1].upper()))

# print("Image saved!")

@classmethod

def encode\_enc(cls):

w = cls.newimg.size[0] # w is width

(x, y) = (0, 0)

for pixel in cls.modPix():

# Putting modified pixels in the new image

cls.newimg.putpixel((x, y), pixel)

if (x == w - 1):

x = 0

y += 1

else:

x += 1

@classmethod

def modPix(cls):

datalist = cls.genData()

lendata = len(datalist)

imdata = iter(cls.newimg.getdata())

for i in range(lendata):

# Extracting 3 pixels at a time

pix = [value for value in imdata.\_\_next\_\_()[:3] +

imdata.\_\_next\_\_()[:3] +

imdata.\_\_next\_\_()[:3]]

# Pixel value should be made

# odd for 1 and even for 0

for j in range(0, 8):

if (datalist[i][j] == '0' and pix[j] % 2 != 0):

pix[j] -= 1

elif (datalist[i][j] == '1' and pix[j] % 2 == 0):

if (pix[j] != 0):

pix[j] -= 1

else:

pix[j] += 1

# pix[j] -= 1

# Eighth pixel of every set tells

# whether to stop or read further.

# 0 means keep reading; 1 means the

# message is over.

if (i == lendata - 1):

if (pix[-1] % 2 == 0):

if (pix[-1] != 0):

pix[-1] -= 1

else:

pix[-1] += 1

else:

if (pix[-1] % 2 != 0):

pix[-1] -= 1

pix = tuple(pix)

yield pix[0:3]

yield pix[3:6]

yield pix[6:9]

# Convert encoding data into 8-bit binary

# form using ASCII value of characters

@classmethod

def genData(cls):

# list of binary codes

# of given data

newd = []

for i in cls.data:

newd.append(format(ord(i), '08b'))

return newd

# s = StegoEncryption()

# image = "meet.jpg"

# data = "watashiwa darksaintdesu"

# s.encode(image, data)

**Steganography\_Decryption.py**

from PIL import Image

class StegoDecryption:

@classmethod

def decode(cls,img):

cls.data = ""

image = Image.open(img, 'r')

imgdata = iter(image.getdata())

while (True):

pixels = [value for value in imgdata.\_\_next\_\_()[:3] +

imgdata.\_\_next\_\_()[:3] +

imgdata.\_\_next\_\_()[:3]]

binstr = ''

# string of binary data

for i in pixels[:8]:

if (i % 2 == 0):

binstr += '0'

else:

binstr += '1'

cls.data += chr(int(binstr, 2))

if (pixels[-1] % 2 != 0):

return cls.data

# print(self.data)

**AESencrypt.py**

from AESCryptography.AESencryptfunc import \* #import AESencryptfunc module to use functions created for this program

import math #import math module to use function such as ceiling

# #check that script is running with the two text files as the two parameters or else quit

# if len(sys.argv) is not 3:#takes in two arguments for the plaintext.txt file name and cipherhex.txt file name

# sys.exit("Error, script needs two command-line arguments. (Plaintext.txt File and cipherhex.txt File)")

def encrypt(data, key=0):

# set passphrase to be a 16 characters, 16 characters \* 8 bits = 128 bits strength

PassPhrase=""

while(len(PassPhrase)!=16):

# print("Enter in the 16 character passphrase to encrypt your text file %s" %sys.argv[1])

PassPhrase= "darksaint09" if key==0 else key #takes in user input of char, eg. "Iwanttolearnkung"

if(len(PassPhrase)<16):#check if less than 16 characters, if so add one space character until 16 chars

while(len(PassPhrase)!=16):

PassPhrase=PassPhrase+"\00"

if(len(PassPhrase)>16):#check if bigger than 16 characters, if so then truncate it to be only 16 chars from [0:16]

# print("Your passphrase was larger than 16, truncating passphrase.")

PassPhrase=PassPhrase[0:16]

#open plaintext.txt file to read and encrypt

# file=open(sys.argv[1], "r")

message=data

# print("Inside your plaintext message is:\n%s\n" % message)

# file.close()

message=BitVector(textstring=message)

message=message.get\_bitvector\_in\_hex()

replacementptr=0

while(replacementptr<len(message)):

if(message[replacementptr:replacementptr+2]=='0a'):

message=message[0:replacementptr]+'0d'+message[replacementptr:len(message)]

replacementptr=replacementptr+4

else:

replacementptr=replacementptr+2

message=BitVector(hexstring=message)

message=message.get\_bitvector\_in\_ascii()

#set up some parameters

start=0#set starting pointer for the part to encrypt of the plaintext

end=0#set ending pointer for the part to encrypt of the plaintex

length=len(message)#check the entire size of the message

loopmsg=0.00#create a decimal value

loopmsg=math.ceil(length/16)+1#use formula to figure how long the message is and how many 16 character segmentss must be encrypted

outputhex=""#setup output message in hex

#need to setup roundkeys here

PassPhrase=BitVector(textstring=PassPhrase)

roundkey1=findroundkey(PassPhrase.get\_bitvector\_in\_hex(),1)

roundkey2=findroundkey(roundkey1,2)

roundkey3=findroundkey(roundkey2,3)

roundkey4=findroundkey(roundkey3,4)

roundkey5=findroundkey(roundkey4,5)

roundkey6=findroundkey(roundkey5,6)

roundkey7=findroundkey(roundkey6,7)

roundkey8=findroundkey(roundkey7,8)

roundkey9=findroundkey(roundkey8,9)

roundkey10=findroundkey(roundkey9,10)

roundkeys=[roundkey1,roundkey2,roundkey3,roundkey4,roundkey5,roundkey6,roundkey7,roundkey8,roundkey9,roundkey10]

#set up FILEOUT to write

FILEOUT = open("temp.txt", 'w')

# set up the segement message loop parameters

for y in range(1, loopmsg): # loop to encrypt all segments of the message

if(end+16<length): #if the end pointer is less than the size of the message, then set the segment to be 16 characters

plaintextseg = message[start:end + 16]

else: #or else if the end pointer is equal to or greator than the size of the message

plaintextseg = message[start:length]

for z in range(0,((end+16)-length),1): #run a while loop to pad the message segement to become 16 characters, if it is 16 already the loop will not run

plaintextseg = plaintextseg+"\00"

#plaintextseg2=BitVector(textstring=plaintextseg)

#print(plaintextseg2.get\_bitvector\_in\_hex())

#add round key zero/ find round key one

bv1 = BitVector(textstring=plaintextseg)

bv2 = PassPhrase

resultbv=bv1^bv2

myhexstring = resultbv.get\_bitvector\_in\_hex()

for x in range(1, 10): # loop through 9 rounds

# sub byte

myhexstring = resultbv.get\_bitvector\_in\_hex()

temp1=subbyte(myhexstring)

# shift rows

temp2=shiftrow(temp1)

# mix column

bv3 = BitVector(hexstring=temp2)

newbvashex=mixcolumn(bv3)

newbv=BitVector(hexstring=newbvashex)

#add roundkey for current round

bv1 = BitVector(bitlist=newbv)

bv2 = BitVector(hexstring=roundkeys[x-1])

resultbv = bv1 ^ bv2

myhexresult = resultbv.get\_bitvector\_in\_hex()

#start round 10

# sub byte round 10

myhexstring = resultbv.get\_bitvector\_in\_hex()

temp1=subbyte(myhexstring)

# shift rows round 10

temp2=shiftrow(temp1)

# add round key round 10

newbv = BitVector(hexstring=temp2)

bv1 = BitVector(bitlist=newbv)

bv2 = BitVector(hexstring=roundkeys[9])

resultbv = bv1 ^ bv2

myhexstring = resultbv.get\_bitvector\_in\_hex()

#set encrypted hex segement of message to output string

outputhextemp = resultbv.get\_hex\_string\_from\_bitvector()

FILEOUT.write(outputhextemp)

start = start + 16 #increment start pointer

end = end + 16 #increment end pointer

# encrypted output hex string to specified cipherhex file

FILEOUT.close()

return open("temp.txt", mode='r')

# file2=open(sys.argv[2], "r")

# print("The output hex value for the entire message is:\n%s\n" % file2.read()

# file2.close()

# encrypt(data,key)

**AESdecrypt.py**

from AESCryptography.AESdecryptfunc import \* #import AESdecryptfunc module to use functions created for this program

import math #import math module to use function such as ceiling

# #check that script is running with the two text files as the two parameters or else quit

# if len(sys.argv) is not 3:#takes in two arguments for the ciphertext.txt file name and plainhex.txt file name

# sys.exit("Error, script needs two command-line arguments. (Ciphertext.txt File and plainhex.txt File)")

# PassPhrase=""

def decrypt(ciphertext, key):

PassPhrase = ""

while(len(PassPhrase)!=16):

# print("Enter in the 16 character passphrase to decrypt your text file %s" %sys.argv[1])

PassPhrase= "darksaint09" if key==0 else key

if(len(PassPhrase)<16):#check if less than 16 characters, if so add one space character until 16 chars

while(len(PassPhrase)!=16):

PassPhrase=PassPhrase+"\00"

if(len(PassPhrase)>16):#check if bigger than 16 characters, if so then truncate it to be only 16 chars from [0:16]

# print("Your passphrase was larger than 16, truncating passphrase.")

PassPhrase=PassPhrase[0:16]

message= ciphertext

#set up some parameters

start=0#set starting pointer for the part to decrypt of the ciphertext

end=32#set ending pointer for the part to decrypt of the plaintex

length=len(message)#check the entire size of the message

loopmsg=0.00#create a decimal value

loopmsg=math.ceil(length/32)+1#use formula to figure how long the message is and how many 16 character segmentss must be decrypted

outputhex=""#setup output message segment in hex

asciioutput=""#setup compilation of output message in ascii

#need to setup roundkeys here

PassPhrase=BitVector(textstring=PassPhrase)

roundkey1=findroundkey(PassPhrase.get\_bitvector\_in\_hex(),1)

roundkey2=findroundkey(roundkey1,2)

roundkey3=findroundkey(roundkey2,3)

roundkey4=findroundkey(roundkey3,4)

roundkey5=findroundkey(roundkey4,5)

roundkey6=findroundkey(roundkey5,6)

roundkey7=findroundkey(roundkey6,7)

roundkey8=findroundkey(roundkey7,8)

roundkey9=findroundkey(roundkey8,9)

roundkey10=findroundkey(roundkey9,10) roundkeys=[roundkey1,roundkey2,roundkey3,roundkey4,roundkey5,roundkey6,roundkey7,roundkey8,roundkey9,roundkey10]

# FILEOUT = io.open(sys.argv[2], 'w', encoding='utf-8')

text = ""

# set up the segement message loop parameters

for y in range(1, loopmsg): # loop to encrypt all segments of the message

plaintextseg = message[start:end]

# add round key

bv1 = BitVector(hexstring=plaintextseg)

bv2 = BitVector(hexstring=roundkeys[9])

resultbv = bv1 ^ bv2

myhexstring = resultbv.get\_bitvector\_in\_hex()

#inverse shift row

myhexstring=invshiftrow(myhexstring)

#inverse subbyte

myhexstring=invsubbyte(myhexstring)

for x in range(8, -1, -1):

# add roundkey for current round

bv1 = BitVector(hexstring=myhexstring)

bv2 = BitVector(hexstring=roundkeys[x])

resultbv = bv1 ^ bv2

myhexstring = resultbv.get\_bitvector\_in\_hex()

# mix column

bv3 = BitVector(hexstring=myhexstring)

myhexstring=invmixcolumn(bv3)

# shift rows

myhexstring = invshiftrow(myhexstring)

# sub byte

myhexstring = invsubbyte(myhexstring)

#add initial round key

bv1 = BitVector(hexstring=myhexstring)

bv2 = PassPhrase

resultbv = bv1 ^ bv2

myhexstring = resultbv.get\_bitvector\_in\_hex()

start = start + 32 #increment start pointer

end = end + 32 #increment end pointer

replacementptr = 0

while (replacementptr < len(myhexstring)):

if (myhexstring[replacementptr:replacementptr + 2] == '0d'):

myhexstring = myhexstring[0:replacementptr] + myhexstring[replacementptr+2:len(myhexstring)]

else:

replacementptr = replacementptr + 2

outputhex = BitVector(hexstring=myhexstring)

asciioutput = outputhex.get\_bitvector\_in\_ascii()

asciioutput=asciioutput.replace('\x00','')

# FILEOUT.write(asciioutput)

# print("Result: " ,asciioutput, sep="\n")

# return asciioutput

text += asciioutput

return text

# file2=io.open(sys.argv[2], "r", encoding='utf-8')

# print("The decrypted message for the entire ciphertext is:\n%s\n" % file2.read())

# file2.close()

**7.TESTING**

**7.1 Introduction**

Testing is the major quality control measure employed for software development. Its basic function is to detect error in the software. During requirement analysis and design, the output is a document which is usually textual and non-textual. After the coding phase, computer programs are available that can be executed for testing purpose. This implies that testing has to uncover errors introduced during coding phases. Thus, the goal of testing is to cover requirement, design, or coding errors in the program. The purpose is to exercise the different parts of the module code to detect coding errors. After this, the modules are gradually integrated into subsystems, which are then integrated themselves to eventually form the entire system. During the module integration, testing is performed. The goal is to detect designing errors, while focusing the interconnecting between the modules. After the system was put together, system testing is performed. Here the system is tested against the system requirements to see if all requirements were met and the system performs as specified by the requirements. Finally, testing is performed to demonstrate to the client for the operation of the system.

For the testing to be successful, proper selection of the test case is essential. There are two different approaches for selecting test case. The software or the module to be tested is treated as a black box, and the test cases are decided based on the specifications of the system or module. For this reason, this form of testing is also called “black box testing”.The focus here is on testing the external behaviour of the system. In structural testing, the test cases are decided based on the logic of the module to be tested. A common approach here is to achieve some type of coverage of statements in the code. The two forms of testing are complementary: one tests the external behaviour, the other tests the internal structure. Often structural testing is used for lower level of testing, while functional testing is used for higher levels.

Testing is an extremely critical and time-consuming activity. It requires proper planning of the overall testing process. Frequently the testing process starts with the test plan. This plan identifies all testing related activities that must be performed and specifies the schedule, allocates the resources, and specifies guidelines for testing. Then for different test unit, a test case specification document is produced, which lists all the different test cases, together with the expected outputs, that will be used for testing. During the testing of the unit the specified test cases are executed and the actual results are compared with the expected outputs. The final output of the testing phase is the testing report and the error report are a set of such report. Each test report contains a set of test cases and the result of executing the code with the test cases. The error report describes the error encountered and the action taken to remove the error.

**Testing approaches**

Testing is a process, which reveals the error in a program. It is a major quality measure employed during software development. During testing, the program is executed with a set of conditions known as test case and output is evaluated to determine whether the program is performing as expected. In order to make sure that the system does not have errors, the different level of testing strategies is applied at differing phases of software development.

**Unit Testing**

Unit Testing is done on individual modules as they are completed and become executable. It is confined only to the designer's requirements.

**Each module can be tested using the following two strategies**

**Black Box Testing**

Internal system design is not considered in this type of testing. Tests are based on the requirements and the functionality. This testing is used to find the errors in the following categories.

* Incorrect or missing functions
* Interface errors
* Performance errors
* Initialization and termination errors.

In this testing, only the output is checked for correctness but the logical flow of the data isn’t checked.

**White Box Testing**

This testing is based on the knowledge of the internal logic of an application’s code. Also known as Glass box Testing. Internal software and code working should be known for this type of testing. Tests are based on coverage of code, statements etc. It is used to generate the test cases in the following cases:

* Guarantee that all the independent paths have been executed.
* Execute all the logical decisions on their true and false sides.
* Execute all the loops at their boundaries and within their operational
* Execute the internal data structures to ensure their validity.

**Integration Testing**

Integration testing ensures that the software and the subsystems work together as a whole. It tests the interface of all the modules to make sure that the modules behave properly or not when integrated together.

**System Testing**

It involves in-house testing of the entire system before the delivery to the user. Its aim is to satisfy the user and the system that meets all the requirements of the client's specifications.

**Acceptance Testing**

It is a pre-delivery testing in which the entire system is tested at the client’s site on the real-world data to find errors.

**Validation Testing**

The system is tested and implemented successfully and thus ensured that all the requirements as listed in the software requirements specification are completely fulfilled. In case of erroneous input corresponding error messages are displayed.

**Compiling Test**

It was a good idea to do our stress testing early, because it gives us time to fix some of the unexpected exceptions and stability problems that only occur when the components were exposed to very high transaction volumes.

**Execution Test**

Finally, the program was successfully loaded and executed.

**Output Test**

The successful output screens are placed in the output screens section.

**7.2 Test cases**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **T ID** | **Description** | **Input** | **Expected**  **value** | **Actual**  **value** | **Result** |
| 1. | Encryption | Image, Data, PassPhrase | Stegano image | Stegano image | Pass |
| 2. | Decryption | Stegano image, PassPhrase | Data | Data | Pass |
| 3 | Encryption without passphrase | Image,data | Stegano  image | Stegano  image | Pass |
| 4 | Decryption without Passphrase | Stegano Image | Data | Data | Pass |
| 5 | Decryption with wrong passphrase | Stegano Image,Wrong PassPhrase | Garbage values | Garbage values | Pass |
| 6 | Decryption with random Image | Image,passphrase | Exception | Exception | Pass |
| 7 | Encryption with Stegano Image | Stegano Image, data, passphrase | Overridden Stegano Image | Overridden Stegano Image | Pass |
| 8 | Decryption with Overridden Image | Overridden Image,PassPhrase | Data in Overridden Image | Data in Overridden Image | Pass |

***Table 7.1* Test Cases**

**8. RESULTS AND DISCUSSIONS**

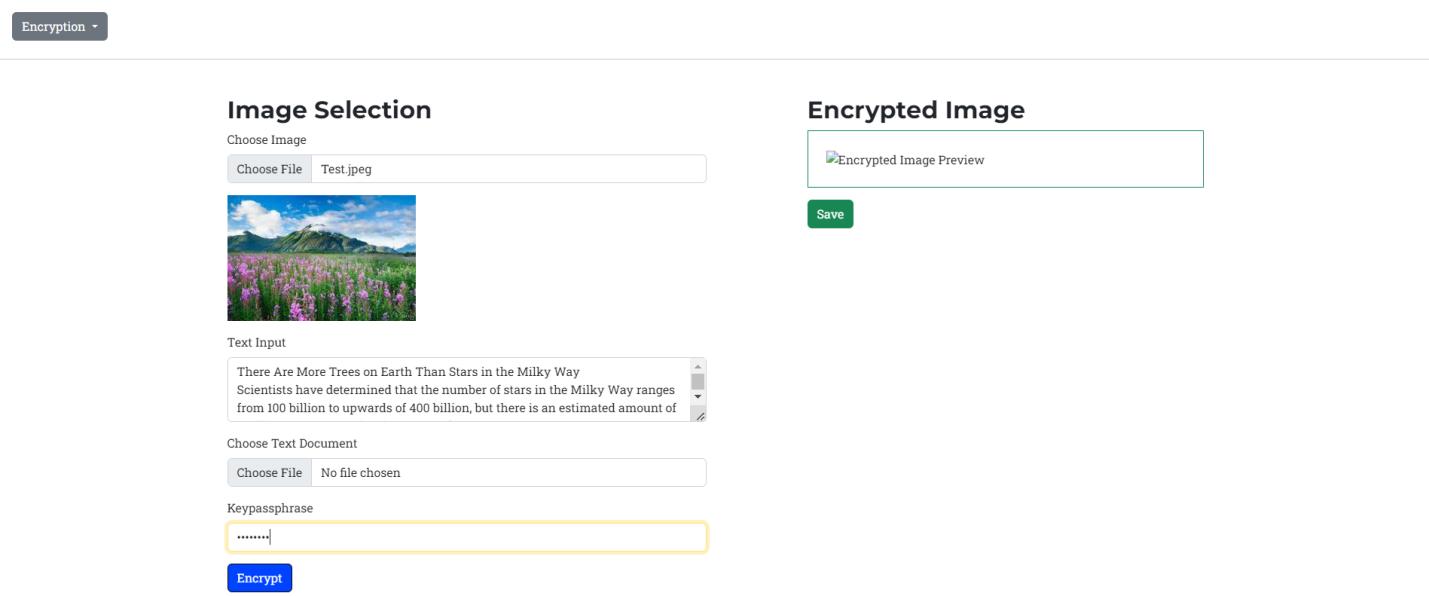
The user interface (UI) of the system is designed to be intuitive and user-friendly, guiding users through the encryption and decryption processes seamlessly. It features clear instructions and interactive elements to facilitate user interaction. The UI allows users to input data for encryption, select cover images for embedding, and initiate decryption processes. Visual feedback is provided to inform users of the progress and status of each operation.

**8.1 Encryption Process:**

The encryption process involves several phases:

**Data Input:** Users input the data they want to encrypt into the system.

**Encryption:** The system utilizes the AES encryption algorithm to encrypt the input data, generating ciphertext.

**Steganography:** The encrypted data is embedded into digital images using LSB Image Steganography, producing steganographed images.

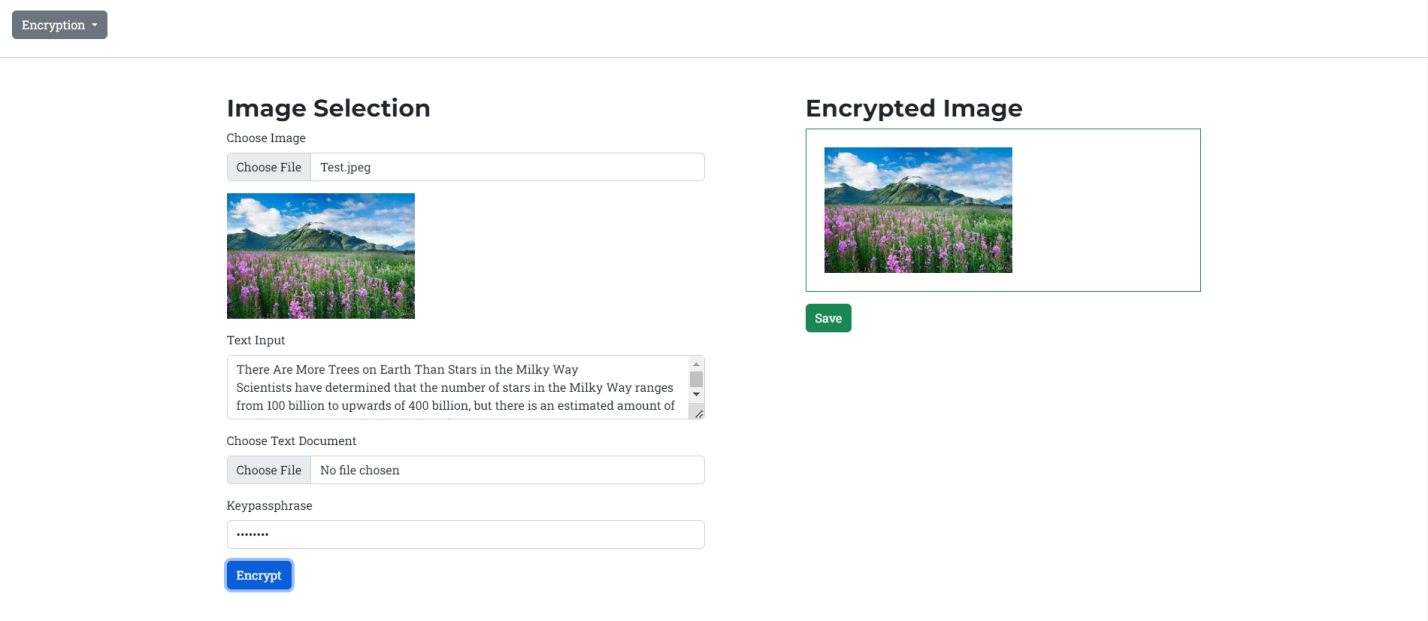
**Fig 8.1 Encryption page Input**

**8.2 Encryption Results**

After completing the encryption process, the following results are obtained:

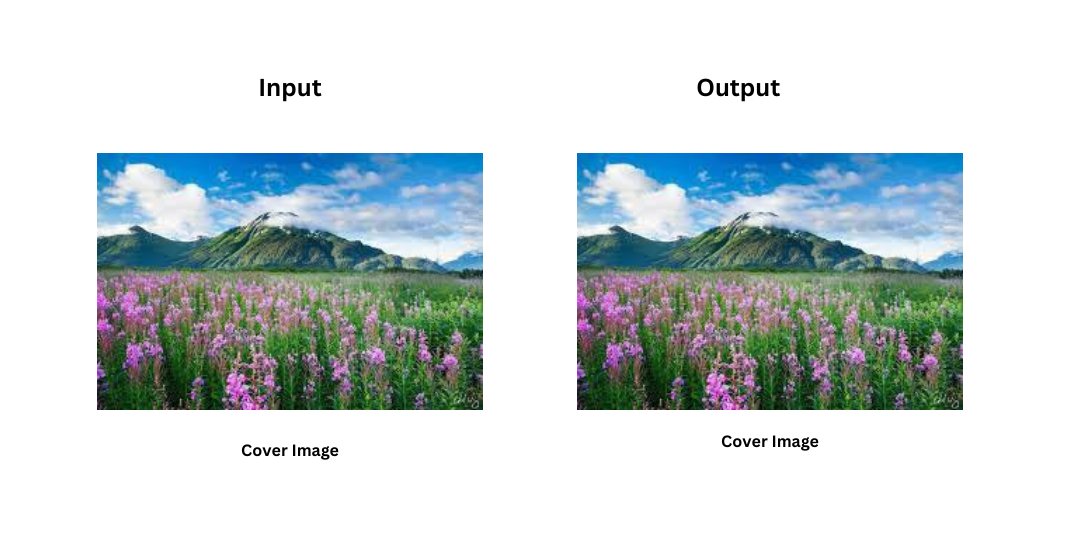
**Ciphertext:** The encrypted data in ciphertext form.

**Steganographed Images:** Digital images containing the encrypted data embedded within them.



**Fig 8.2 Encryption page Output**

The Cipher text will be generated in the backend which is not shared with anyone for the security purposes. The **Cipher text** will be in the form of   

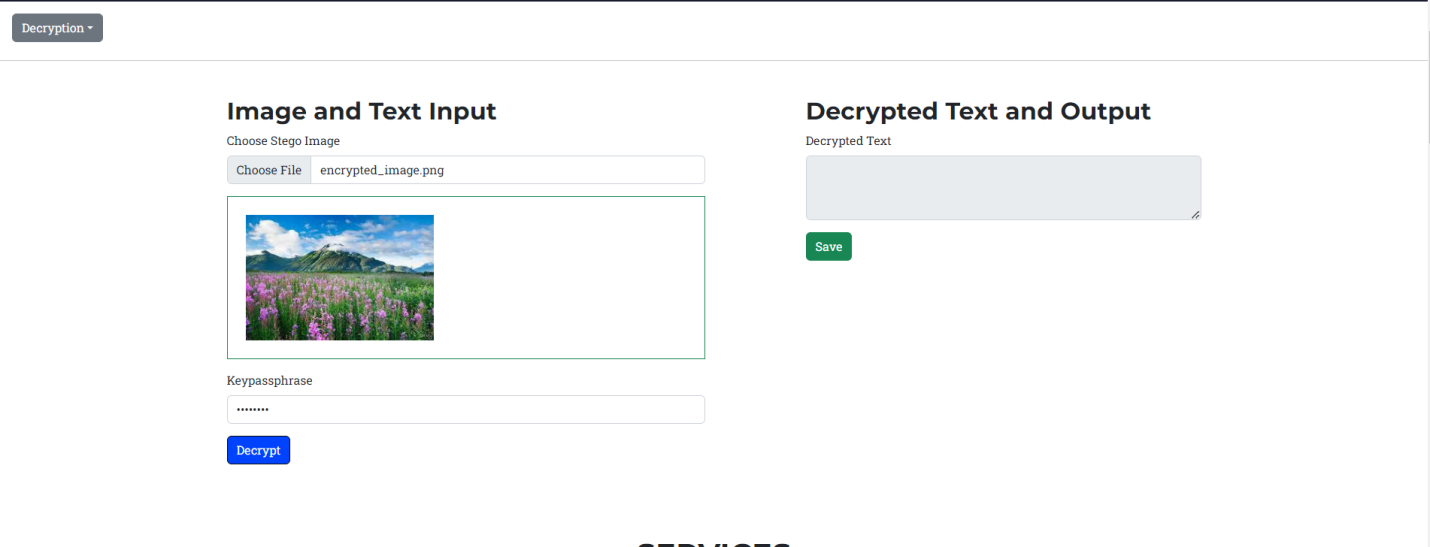

****This won’t be the same for all the time.

**Fig 8.3 Image Input and Output**

**8.3 Decryption Process:**

The decryption process reverses the encryption phases:

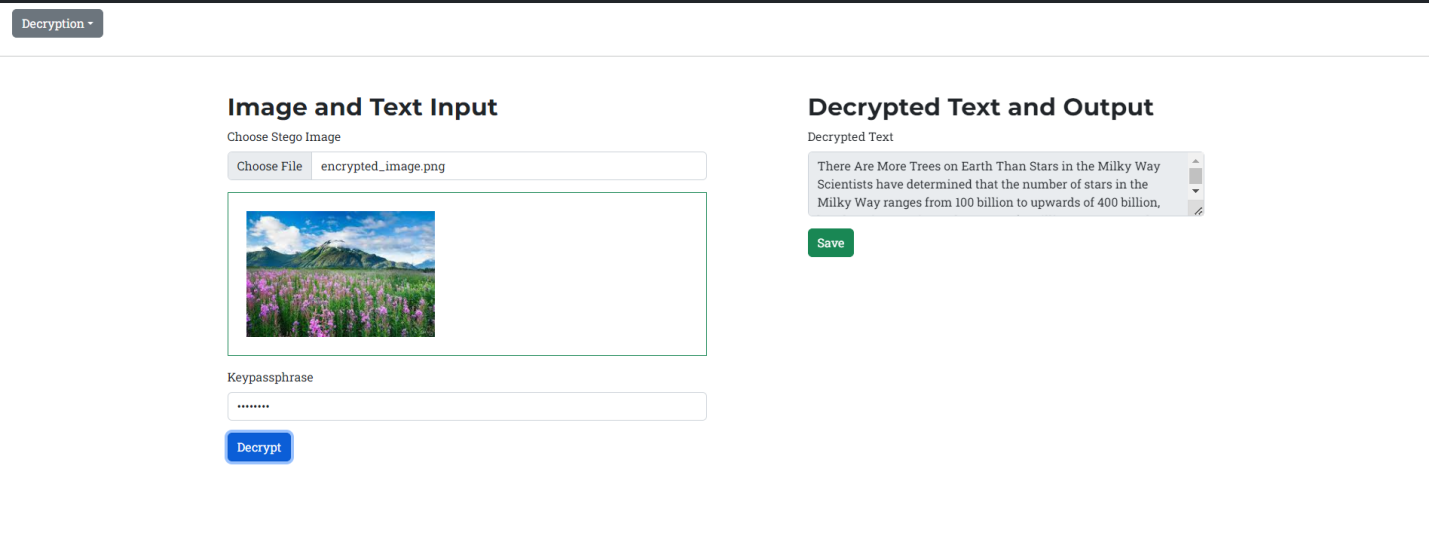
**Steganography Extraction:** The system extracts the encrypted data from steganographed images using LSB Image Steganography.

**Decryption:** The extracted encrypted data is decrypted using the AES decryption algorithm, resulting in plaintext.

**Fig 8.4 Decryption Page Input**

**8.4 Decryption Results:**

Upon completing the decryption process, the following results are obtained:

**Decrypted Data:** The original plaintext data recovered from the decrypted ciphertext.

**Fig 8.5 Decryption page Output**

**8.5 Discussion:**

The results obtained from the encryption and decryption processes demonstrate the efficacy of the proposed system in securing data confidentiality. The encryption phase ensures that sensitive data is transformed into an unreadable format, protecting it from unauthorized access. Additionally, the steganography phase conceals the encrypted data within digital images, adding an extra layer of security. The decryption phase successfully retrieves the original plaintext data from steganographed images, providing assurance that the data remains intact and accessible only to authorized parties.

**9. CONCLUSION**

In this project, titled "Enhancing Data Confidentiality Through Dual-Layer Cybersecurity Tool," we have successfully developed a system that combines AES cryptography with LSB image steganography to enhance data confidentiality and security. Through the integration of these two techniques, we aimed to provide a robust solution for safeguarding sensitive information from unauthorized access and interception.

Our system utilizes AES encryption to secure data, ensuring that it remains unreadable to unauthorized parties. Additionally, LSB image steganography is employed to conceal the encrypted data within digital images, effectively masking its presence and reducing the risk of detection. This dual-layered approach offers a comprehensive security solution, addressing both confidentiality and stealth in data transmission and storage.

Through implementation and testing, we have observed promising results in terms of security, performance, and practical applicability. The encryption and steganography processes demonstrated efficient performance, with minimal computational overhead and negligible impact on image quality. Moreover, the successful extraction and decryption of hidden data from stego images validate the reliability and robustness of our system.Our project has attained, Program Outcomes PO1, PO2, PO3, PO4, PO5, PO6, PO7, PO8, PO9, PO10, PO11, PO12 and Program Specific Outcomes PSO1, PSO2, PSO3.

While our project has shown promising results, there are opportunities for further enhancement and future research. Some potential areas for future exploration include:

Enhanced Steganographic Techniques: Investigating advanced steganographic methods to further improve data concealment and resistance against steganalysis techniques.

Scalability and Optimization: Developing strategies to optimize system performance and scalability, particularly in handling larger volumes of data and improving processing efficiency.

Integration with Machine Learning: Exploring the integratsion of machine learning algorithms to automate the process of identifying and updating steganographic models based on evolving security requirements and data patterns.

In conclusion, our project represents a significant step towards enhancing data confidentiality through innovative integration of encryption and steganography techniques. The system's effectiveness in securing sensitive information underscores its potential for practical application in various domains where data security is paramount.

**9.1 Program Outcomes**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| PO1 | PO2 | PO3 | PO4 | PO5 | PO6 | PO7 | PO8 | PO9 | PO10 | PO11 | PO12 |
| 3 | 3 | 2 | 3 | 2 | 2 | 2 | 3 | 2 | 2 | 3 | 3 |

**9.2 Program Specific Outcomes**

|  |  |  |
| --- | --- | --- |
| PSO1 | PSO2 | PSO3 |
| 3 | 2 | 2 |

**10. REFERENCES**

* Schneier, Bruce. "Applied Cryptography: Protocols, Algorithms, and Source Code in C." John Wiley & Sons, 1996.
* Ferguson, Niels, and Schneier, Bruce. "Practical Cryptography." John Wiley & Sons, 2003.
* Stallings, William. "Cryptography and Network Security: Principles and Practice." Pearson, 2016.
* Johnson, Neil F., and Jajodia, Sushil. "Steganography: Seeing the Unseen." IEEE Computer Society Press, 1998.
* Fridrich, Jessica, Goljan, Miroslav, and Du, Rui. "Detecting LSB steganography in color and grayscale images." IEEE Multimedia, Vol. 8, No. 4, 2001, pp. 22-28.
* Python Imaging Library (PIL). "Pillow: Python Imaging Library." [Online]. Available: <https://python-pillow.org/.>
* "A Novel Approach to Enhance Data Confidentiality using LSB Image Steganography and AES Cryptography". Authors: Anandhavelu P., Kavitha G. ,Source: 2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)
* "Image Steganography with AES Encryption Technique". Authors: Aparna Singh, Amit Kushwaha, Source: 2016 International Conference on Advances in Computing, Communication, & Automation (ICACCA)
* "A Hybrid Approach for Data Hiding using LSB Steganography and AES Cryptography". Authors: M. Kishore Kumar, T. Padmavathi, Source: 2018 International Conference on Inventive Research in Computing Applications (ICIRCA)
* "A Hybrid Approach for Data Hiding using LSB Steganography and AES Cryptography". Authors: M. Kishore Kumar, T. Padmavathi, Source: 2018 International Conference on Inventive Research in Computing Applications (ICIRCA)

**Books:**

* “Cryptography and Network Security: Principles and Practices" by William Stallings.
* “Steganography in Digital Media: Principles, Algorithms, and Applications" by Jessica Fridrich.
* “Applied Cryptography: Protocols, Algorithms, and Source Code in C" by Bruce Schneier.