DESIGN DOCUMENTATION

for

A Secure method to send Emails using - Compression, Encryption and Pixel Value Differencing

Prepared by
CSU 152 24 MDL15CS047 Joby Mathew
CSU 152 34 MDL15CS070 Merin Francis
CSU 152 39 MDL15CS087 Ria Rajan Alappat
CSU 152 56 MDL15CS113 Tony Josi

November 26, 2018

Contents

1 Introduction

1.1 Purpose

Electronic mail (email or e-mail) is a method of exchanging messages ("mail") between people using electronic devices. Email security refers to the collective measures used to secure the access and content of an email account or service. It allows an individual or organization to protect the overall access to one or more email addresses/accounts. An email service provider implements email security to secure subscriber email accounts and data from hackers - at rest and in transit.

This project aims at creating a platform for securing the sending of mails using Compression, Encryption and Pixel Value Differencing

1.2 Overview

The rest of the document includes the system architecture, the functional modules, the data description and the algorithms to be used for the implementation. Reading through these sections clearly, gives an idea of the various challenges in the development of this project as well as the standards and guidelines that are to be adhered to. The sequence of reading for the best understanding of the working of the system would be :-

- System Architecture
- Data Flow Diagram
- Algorithms

2 System Architecture

The whole system is divided into four major phases:

- Input phase from user
- Compression of secret massage
- Encryption of compressed data
- Steganography using PVD
- Sending the message across the network
- Retrieval of Cover Image by the receiver
- Processing to obtain the secret message

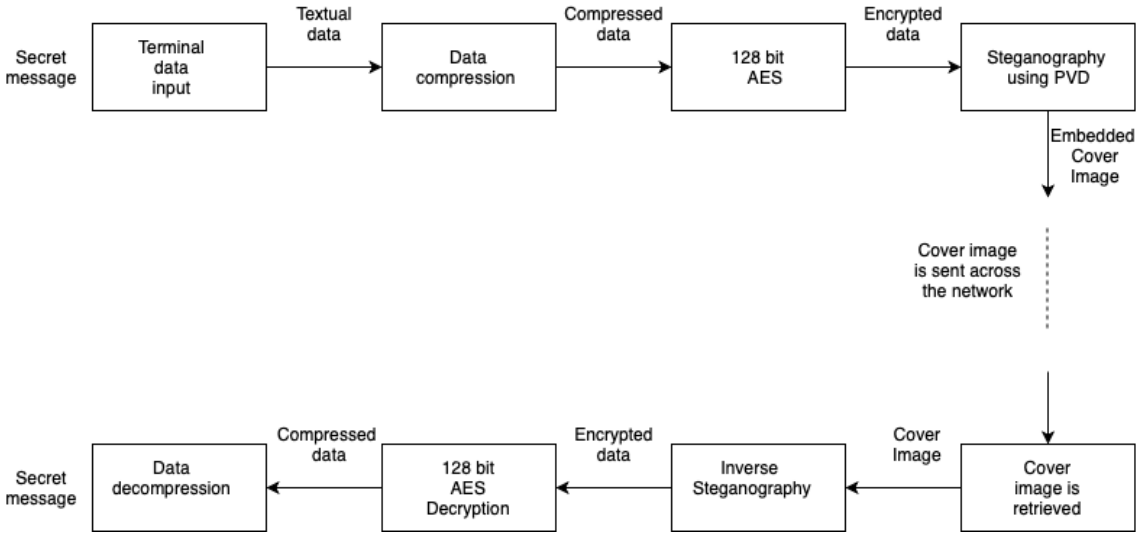

Figure 2.1: Architecture Diagram

2.1 Input phase from user

In this phase data is taken from the user along with the user log-in details and secret message that needs to be sent across using the system. The input is taken from the user via terminal or command line prompts.

2.2 Compression of secret massage

In this phase the secret message from the user input is compressed using arithmetic coding. The capacity of system can be improved by using this phase. Arithmetic coding is a form of entropy encoding used in lossless data compression. Normally, a string of characters such as the words "hello there" is represented using a fixed number of bits per character, as in the ASCII code. When a string is converted to arithmetic encoding, frequently used characters will be stored with fewer bits and not-so-frequently occurring characters will be stored with more bits, resulting in fewer bits used in total.

On an average, about 22% higher embedding capacity was achieved through Arithmetic Coding.

2.3 Encryption of compressed data

During this phase the encoded data from the previous phase is encrypted using 128-bit AES Symmetric Encryption algorithm. The fundamental function of this phase is to provide additional security for the secret message prior to hiding it in cover image using steganography.

AES is based on a design principle known as a substitution–permutation network, and is efficient in both software and hardware. AES operates on a 4 4 column-major order array of bytes, termed the state. Most AES calculations are done in a particular finite field.

2.4 Steganography using PVD

The encrypted data is then embedded in the cover image during this phase. The cover image should be a color image.

After compression and encryption of secret message, modified approach of Khodaei and Faez's LSB+PVD method is applied to embed the message in cover image. In the proposed embedding method, the cover image is divided into non-overlapping pixel blocks of 3x3 or 3x3 + 2x2 pixel blocks as per the cardinality of the cover image. That is, if the dimensions of the cover image are not divisible by 3x3 pixel blocks, the remaining pixels are taken 2x2 pixel blocks.

2.5 Sending the message across the network

The cover image is send to the receiver using the existing mailing technology. The mailing system uses Simple Mail Transfer Protocol (SMTP) for sending mails. The mail is sent using SMTPLIB - library available in Python.

The smtplib module defines an SMTP client session object that can be used to send mail to any Internet machine with an SMTP or ESMTP listener daemon.

2.6 Retrieval of Cover Image by the receiver

The cover image along with the embedded message is retrieved by the receiver by using the same SMTPLIB - library the cover image is downloaded to the receiver machine.

2.7 Processing to obtain the secret message

The cover image is then undergone inverse steganography, AES decryption and Decompression to obtain the orginal secret message sent by the sender. The process of decryption of an AES ciphertext is similar to the encryption process in the reverse order. Each round consists of the four processes conducted in the reverse order Add round key, Mix columns, Shift rows and Byte substitution

Also inverse arithmetic coding is applied to decompress the compressed message after the decryption phase of AES.

3 Data Description

3.1 Data Flow Diagram

Data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modelling its process aspects. A DFD is often used as a preliminary step to create an overview of the system without going into great detail, which can later be elaborated.This section illustrates the data flow diagram of the system in different levels or phases.

3.1.1 Level 0 DFD

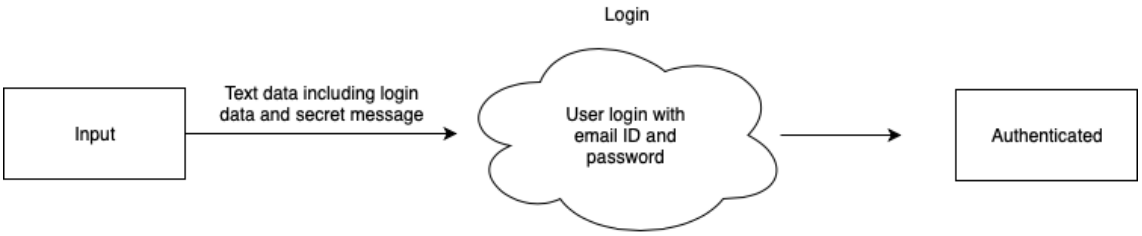

Figure 3.1: DFD Level 0

3.1.2 Level 1 DFD

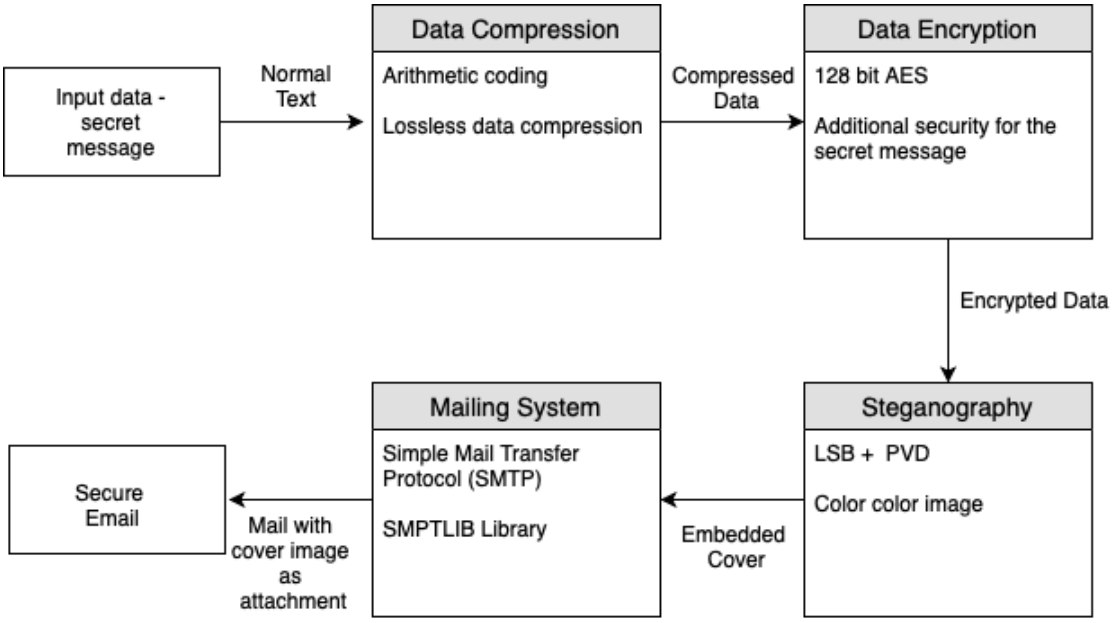

Figure 3.2: DFD Level 1

3.1.3 Level 2 DFD

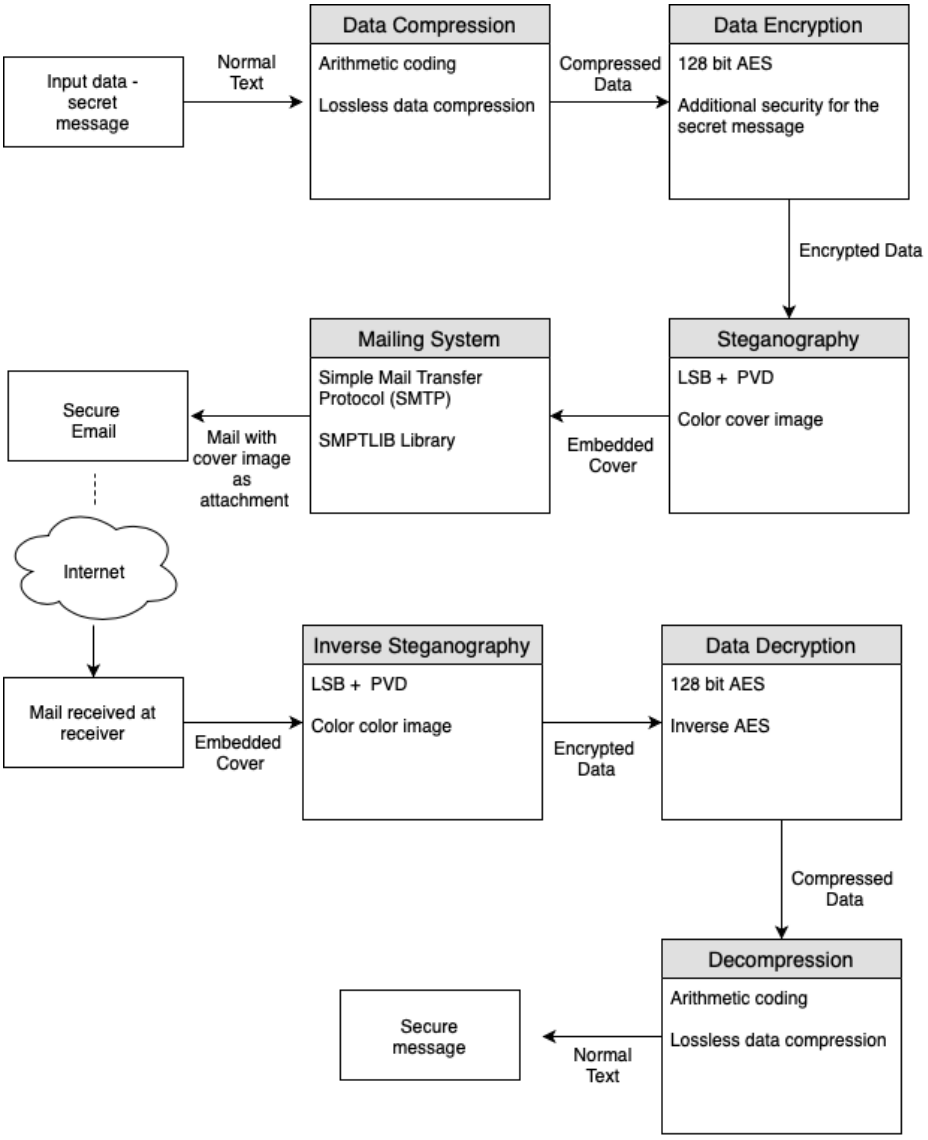

Figure 3.3: DFD Level 2

3.2 Use case diagram

Use case diagrams are usually referred to as behavior diagrams used to describe a set of actions (use cases) that some system or systems (subject) should or can perform in collaboration with one or more external users of the system (actors). Each use case should provide some observable and valuable result to the actors or other stakeholders of the system.

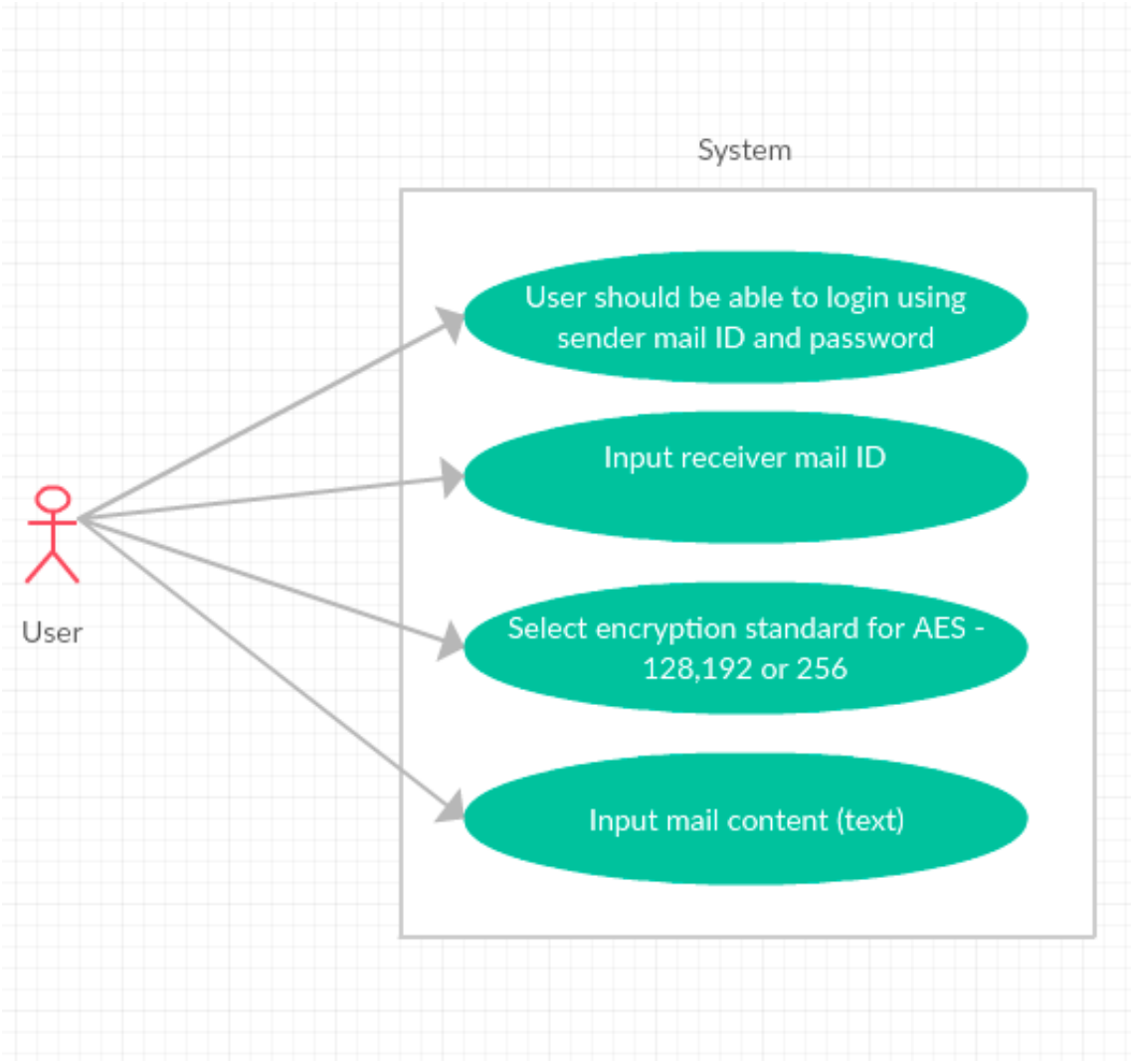

Figure 3.4: Use case diagram

3.3 Class diagram

A class diagram is an illustration of the relationships and source code dependencies among classes in the Unified Modeling Language (UML). In this context, a class defines the methods and variables in an object, which is a specific entity in a program or the unit of code representing that entity. Class diagrams are useful in all forms of object-oriented programming (OOP). The concept is several years old but has been refined as OOP modeling paradigms have evolved.

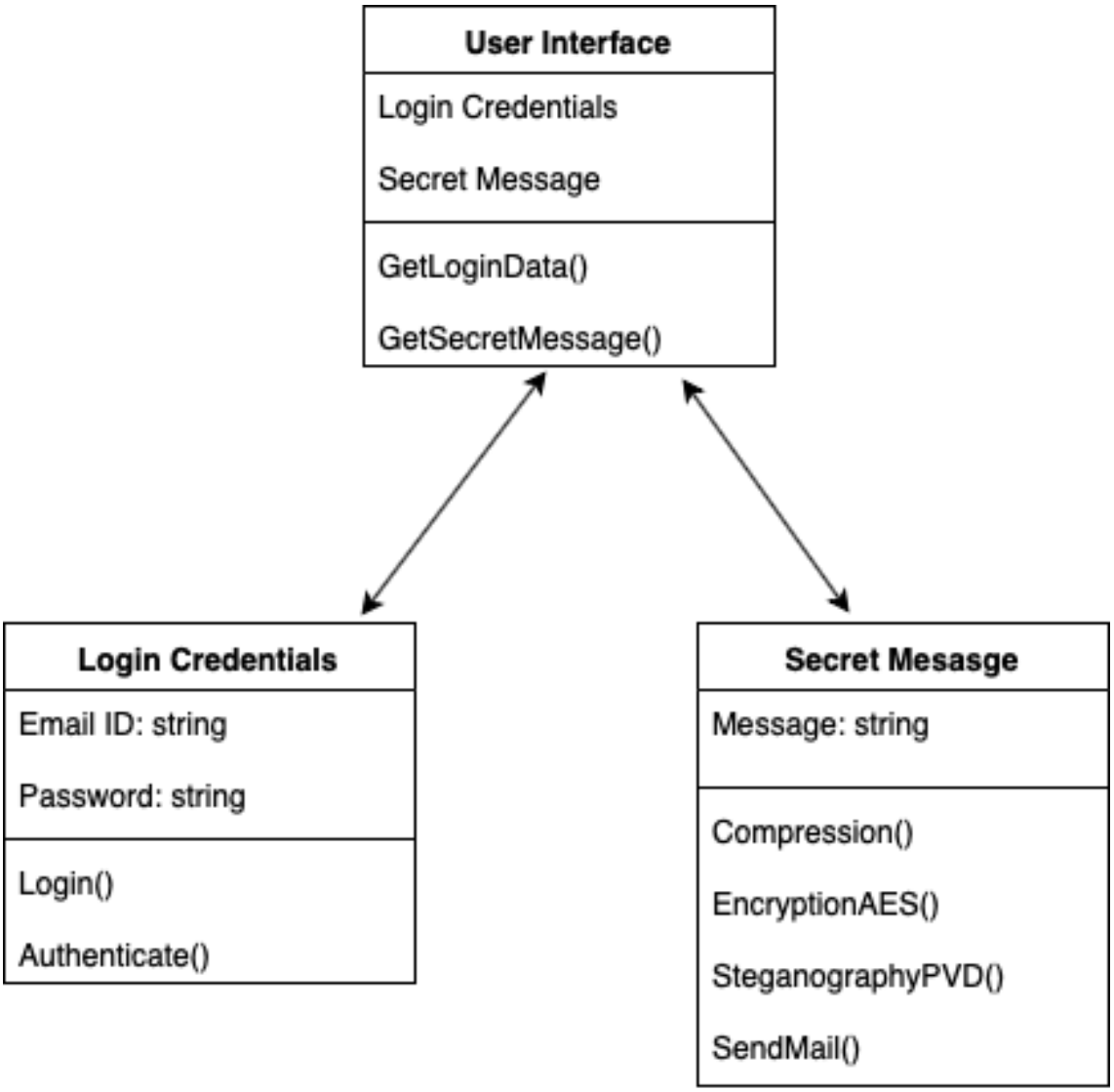

Figure 3.5: Class diagram

3.4 Activity diagram

Activity diagram is another important diagram in UML to describe the dynamic aspects of the system. Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another.

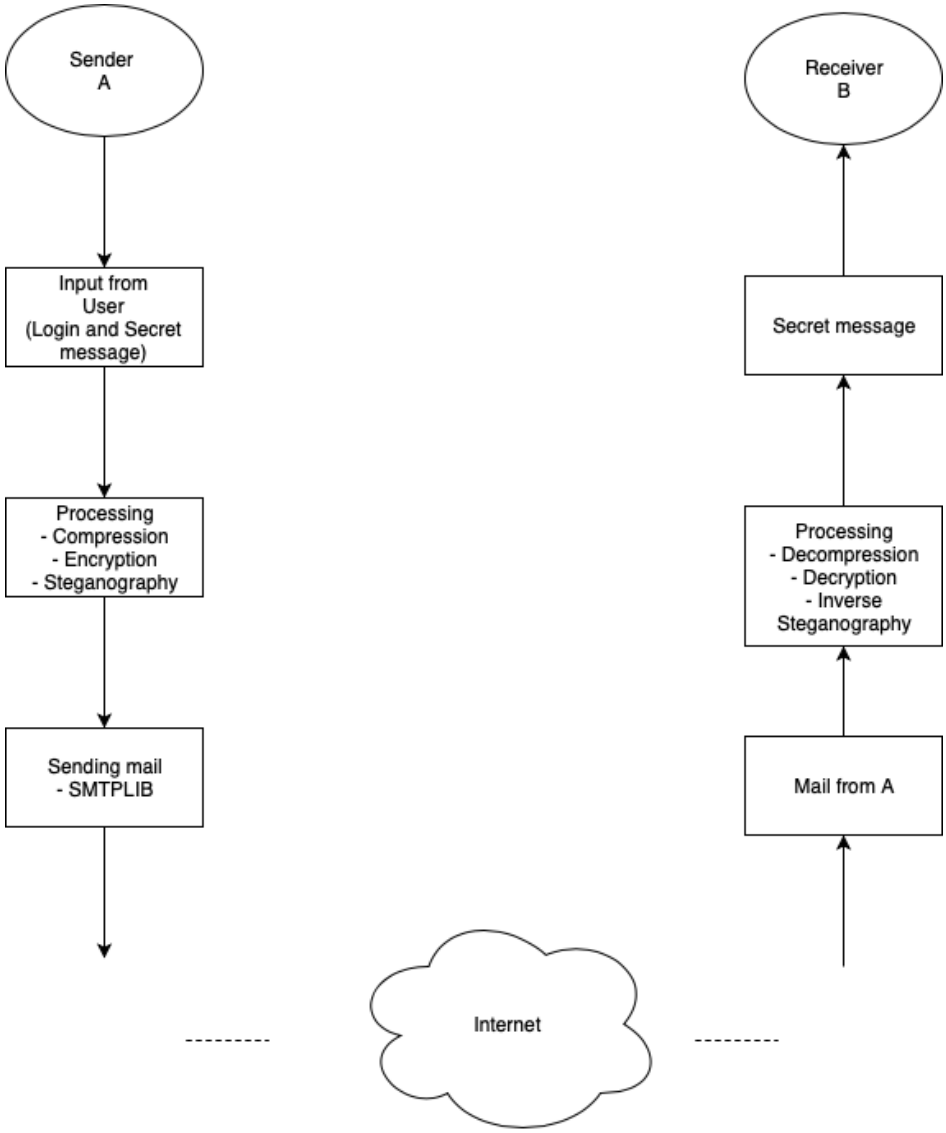

Figure 3.6: Activity diagram

4 Algorithms

4.1 Data Compression

Arithmetic coding is a common algorithm used in both lossless and lossy data compression algorithms. It is an entropy encoding technique, in which the frequently seen symbols are encoded with fewer bits than rarely seen symbols. It has some advantages over well-known techniques like Huffman coding.

Algorithm 1: Compress - Arithmetic Coding

---

1. /* ARITHMETIC ENCODING ALGORITHM. */

   /* Call encode-symbol repeatedly for each symbol in the message. */ /* Ensure that a distinguished "terminator" symbol is encoded last, then */ /* transmit any value in the range [low, high). */

2. encode-symbol(symbo1, cum-freq)

3. range = high - low

4. high = low + range * cum-freq[symbol-1]

5. low = low + range*cum-freq[symbol]

   /* ARITHMETIC DECODING ALGORITHM. */

   /* "Value" is the number that has been received. */ /* Continue calling decode-symbol until the terminator symbol is returned. */

6. decode-symbol(cum-freq)

7. find symbol such that cum-freq[symbol] ¡= (value-low)/(high-low) ¡ cum-freq[symbol-1]

   /* This ensures that value lies within the new */ /* (low, high) range that will be calculated by */ /* the following lines of code. */

8. range = high - low

9. high = low + range*cum-freq[symbol-1]

10. low = low + range*cum-freq[symbol]

11. return symbol

4.2 Encryption

AES is based on a design principle known as a substitution–permutation network, and is efficient in both software and hardware. AES operates on a 4 4 column-major order array of bytes, termed the state. Most AES calculations are done in a particular finite field.

Algorithm 2: Encryption - AES

---

1. KeyExpansion—round keys are derived from the cipher key using Rijndael's key schedule. AES requires a separate 128-bit round key block for each round plus one more.
2. Initial round key addition:
3. AddRoundKey—each byte of the state is combined with a block of the round key using bitwise xor.
4. 9, 11 or 13 rounds:
   a) SubBytes—a non-linear substitution step where each byte is replaced with another according to a lookup table.
   b) ShiftRows—a transposition step where the last three rows of the state are shifted cyclically a certain number of steps.
   c) MixColumns—a linear mixing operation which operates on the columns of the state, combining the four bytes in each column.
   d) AddRoundKey
5. Final round (making 10, 12 or 14 rounds in total):
   a) SubBytes
   b) ShiftRows
   c) AddRoundKey

Algorithm 3: Decryption - AES

---

1. Step 1: Inverse_SubBytes()
2. Step 2: Inverse_ShiftRows()
3. Step 3: Inverse_MixColumns()
4. Step 4: Inverse_AddRoundkey()
5. Step 3 is not performed for last round. All the encryption steps should be performed in reverse to achieve decryption.

4.3 Steganogrphy

After compression and encryption of secret message, modified approach of Khodaei and Faez's LSB+PVD method is applied to embed the message in cover image. In the proposed embedding method, the cover image is divided into non- overlapping pixel blocks of 3x3 or 3x3 + 2x2 pixel blocks as per the cardinality of the cover image.

Algorithm 4: Embedding

---

1. Step 1: Read the cover image I and divide it into 3x3 non-overlapping blocks. However, if the height and width of the image are not divisible by 3x3 blocks, the image is divided into 3x3 plus 2x2 blocks. For example, a 512x512 pixel image is partitioned into 28900 blocks of 3x3 pixels and 511 blocks of 2x2 pixels.
2. Step 2: Read block Bi and name the pixels as a1, a2, a3, a4, ac, a6, a7, a8, a9. For 2x2 pixel blocks, the pixels would be named as ac, a2, a3, and a4
3. Step 3: Select ac as base pixel or reference pixel.
4. Step 4: For reference pixel, embed 3 bits directly into 3 LSBs of ac to get ac'
5. Step 5: Calculate the difference di for all pixel values except for ac. ie, di = mod(ac - ai)
6. Step 6: Compute the ranges Ri to which di belongs. As stated in the range table, compute Ci i.e. number of bits to be hidden.
7. Step 7: Read the Ci bits continuously from S secret message as Si.
8. Step 8: Replace Ci bits of ai with Si to obtain ai'.

Algorithm 5: Inverse Steganography

1. Read the cover image I and divide it into 3x3 non overlapping blocks. However, if the height and width of the image are not divisible by 3x3 blocks, the image is partitioned into 3x3 plus 2x2 blocks. For example, a 512x512 pixel image is partitioned into 28900 blocks of 3x3 pixels and 511 blocks of 2x2 pixels.
2. Read block Bi and name the pixels as a1,a2, a3, a4, ac, a6, a7, a8, a9. For 2x2 pixel the pixels would be named as ac, a2, a3, a4
3. Extract k- secret bits from k- LSB's of a and name it as Sc.
4. Calculate the difference di
5. Compute the ranges R to which d belongs. As stated in range table obtain the C.
6. Then extract Cth rightmost LSB's of and i name it as Si for all the pixels of Bi.
7. Now, concatenate Sc and all Si to obtain secret message.

Bibliography

[1] A. Cheddad, et al., "Digital image steganography: Survey and analysis of current methods," Signal processing

[2] S. Atawneh, et al., "Steganography in digital images: Common approaches and tools,"