**Image Based Encryption System**

***A PROJECT REPORT***

***Submitted by***

**Roshan VS**

**Kumaran KM**

**Kishore Rajan K**

***in partial fulfillment for the award of the degree***

***of***

**BACHELOR OF ENGINEERING**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

A picture containing logo

Description automatically generated

**PANIMALAR ENGINEERING COLLEGE**

Banglore Trunk Road, Varadharajapuram, Nasarathpettai, Poonamalle, Chennai 600 123

Logo

Description automatically generated

**ANNA UNIVERSITY : CHENNAI 600 025**

**MAY 2022**

A picture containing graphical user interface

Description automatically generated

**ANNA UNIVERSITY : CHENNAI 600 025 BONAFIDE CERTIFICATE**

Certified that this project report “Image Based Encryption System” is the bonafide work of “Roshan VS, Kumaran KM, Kishore Rajan K” who carried out the project work under my supervision.

**SIGNATURE SIGNATURE**

**HEAD OF THE DEPARTMENT SUPERVISOR**

**ABSTRACT**

In today’s era of digitalisation and internet, user authentication, the use of passwords and use of secret messages has been a centrepiece for decades. There is substantial research in the field of password security. Unfortunately, the methods of storing the passwords and its mechanism have failed to fetch enough attention from the researchers. In the past few years, a lot of researchers recommended the use of a combination of security methods like steganography and cryptography to elevate the security of the system. However, it still does not assure the security of the password. In this paper, we propose a system that implements multi-level encryption with decentralized and distributed storage of enterprises’ sensitive information and offers multi-layered security to the system. In this design, we introduce a password security equation, *Data Protection = Encrypt + Hide + Split*. According to our equation, we encrypt the password using AES-128, split it into pieces and hide it behind the images using steganography adding another layer of encryption and store it in a decentralized manner. When the user logs in to the system all the pieces of the password are retrieved from all the steganographic images, merged and decrypted to form a legitimate password in one piece.

***Keywords****: Cryptography, Encryption, Steganography, and Decentralization*

**TABLE OF CONTENTS**

| **CHAPTER NO.** | **TITLE** | **PAGE NO.** |
| --- | --- | --- |
|  | **ABSTRACT** | 3 |
| **1.** | **INTRODUCTION** | 6 |
|  | 1.1 Overview |  |
|  | 1.2 Problem Statement |  |
| **2.** | **LITERATURE SURVEY** | 9 |
|  | 2.1 Introduction |  |
|  | 2.2 Purpose of Literature Survey |  |
|  | 2.3 Objectives of Literature Survey |  |
|  | 2.4 Technology Stack |  |
|  | 2.5 Scope of the Study |  |
|  | 2.6 Methodology |  |
|  | 2.7 Bibliography |  |
| **3.** | **SYSTEM ANALYSIS** | 15 |
|  | 3.1 Existing System |  |
|  | 3.2 Proposed system |  |
|  | 3.3 Feasibility Study |  |
| **4.** | **SYSTEM DESIGN** | 18 |
|  | 4.1. UML Diagrams |  |
|  | 4.2 Diagrams |  |
|  | 4.21 Use Case Diagram |  |
|  | 4.22 Sequence Diagram |  |
|  | 4.23 Activity Diagram |  |
|  | 4.24 Flowchart |  |
|  | 4.25 ER Diagram |  |
| **5.** | **SYSTEM ARCHITECTURE** | 31 |
|  | 5.1 Architechture Overview |  |
|  | 5.2 Algorithms |  |
|  | 5.21 Advanced Encryption Standard - 128 |  |
|  | 5.22 Base-64 Encoding |  |
|  | 5.23 Image Steganography |  |
|  | 5.24 Decentralization of Steg Image |  |
| **6.** | **SYSTEM IMPLEMENTATION** | 48 |
|  | 6.1 Software Coding |  |
| **7.** | **SYSTEM TESTING** | 54 |
|  | 7.1 Unit Testing |  |
|  | 7.2 Integration Testing |  |
|  | 7.3 Test Cases |  |
| **8.** | **CONCLUSION** | 66 |
|  | 8.1 Results & Discussion |  |
|  | 8.2 Future Enhancements |  |
|  | **APPENDICES** | 67 |
|  | A.1 Sample Screens |  |
|  | **REFERENCES** | 70 |

**CHAPTER I**

**INTRODUCTION**

**1.1 OVERVIEW**

Every single day the technology is advancing however, the password remains unchanged as the most popular way of user authentication. To access any private and ‘secure’ system a user needs to go through a user authentication process. The user authentication process can involve the verification of one or more of some knowledge (like a password), something in possession (like an RFID access card) or inherence (like the fingerprint) of the entity.Over the past few years, the primary mean of user authentication has been the passwords in the form of text. Users register on the systems by putting username and password to login to the system and gain access over it. The user is expected to remember this secret information to pass the user authentication process. Considering the number of user accounts people have these days on social media, work-related emails, drives, and websites, etc. it could be very difficult for the user to remember all those passwords. Then users tend to use the same password for different systems putting its security on stake and then come RFID access cards and biometric user authentication in the picture, which are unique. But at the end of the day, the question arises on the security of the password storage. No matter what guidelines have been followed to create an ideally secure password, the question remains the same, is it being stored on a secure system in a secure manner?

Different systems store passwords in their database in their own style. It also depends on the type of database system being used. Having confidential information stored at the centralized database system has proven to be highly insecure from time to time. However, storing this confidential information in a distributed manner makes it difficult for the attacker to track it down to every single location and perform the attack. The maintenance and security of the password file in the database is still a point of concern in various fields. Many websites are not following the good storage practice which would ideally include encryption, good hashing algorithms, salting, and a large number of rounds. In the past, many renowned organizations like Yahoo, LinkedIn, United Nations and Sony Online Entertainment were attacked for ignoring these security measures. Sensitive business information is at risk when it falls into the hands of unauthorized people with malicious intentions. Considering the easy availability of hacking tables and tools on the internet, to interrupt the primary level of security with attacks such as Dictionary attack, Rainbow table attack, Hybrid attack, Brute force attack and Smart brute force attack it is very difficult for these security measure to defend the system against such attacks when they are considered to be safe. It would be a mistake to rely on these measures completely assuming that the system is unbreakable. Also, looking at the advanced hacking techniques trending in the market these days, white hat hackers are coming up with innovative solutions to protect the system against them. To increase the security of the systems and keep the data transfer safe throughout the internet, various techniques have been designed such as Digital Watermarking, Cryptography, and Steganography. However, the attackers are successfully being able to penetrate the system in some or the other way. Hence, it has become very crucial for the security researchers and professionals to come up with a stronger solution that is rigid enough to be penetrated and flexible enough to be accommodated in varied environments. The solution must be something that blends the existing security solutions and enhances the security even more. It is possible with multi-layered security implementation using the decentralized password storage system.

This research revolves around a combined methodology that merges these security techniques with respect to the threshold cryptography system. Performing the combination of steganography and cryptography guarantees a high level of data security.

**1.2 PROBLEM STATEMENT**

In today's world of modernization and the web, client confirmation and security have been the foremost imperative need in many organizations and companies.

There's a considerable inquiry within the field of information privacy. Unfortunately, the strategies of putting away and covering up touchy data like passwords and personal data have fizzled to fetch enough consideration from the analysts.

Within the last decade, a parcel of analysts recommended the utilization of a combination of security strategies like steganography and cryptography to elevate the security of the framework. In any case, it still does not guarantee sufficient information security.

In this paper, we propose a framework that executes numerous encryption strategies with a decentralized and distributed capacity of delicate data of clients and offers multi-layered security to the system.

**CHAPTER II**

**LITERATURE SURVEY**

**2.1 INTRODUCTION:**

We can classify passwords as the most important and crucial resource being stored in the database. Hence, in the past extensive research has been done and a lot of researchers from around the world have published their research about its security, attacks on the database system, password attacks, innovative ways of performing those attacks and defense systems against them. Some recommended the use of images or separate use of steganography and cryptography for user authentication, while some suggested the combination of steganography and cryptography for the same. Considering the fact that each one of them had some strengths and weaknesses that sometimes they significantly shielded the system or failed due to some gaps in their method of implementation.

**2.2 PURPOSE OF THE LITERATURE SURVEY**

The literature review plays a very important role in the research process. It is a source from where research ideas are drawn and developed into concepts and finally theories. It also provides the researcher a bird’s eye view about the research done in that area so far. Depending on what is observed in the literature review, a researcher will understand where his/her research stands. Here in this five-literature survey, all primary, secondary and tertiary sources of information were searched.

The study of literature on “Image Based Encryption System” in general and in the field of library and information science particularly revealed several efforts made by the scholars in different discipline. The purpose of the literature survey is to collect a plethora of journal’s article about stock movements and stock volatility on “Image Based Encryption System” with abstract. The main aim of this collection is to provide a guideline and brief information of researcher, user and other individuals who want information about stock investments.

**2.3 OBJECTIVES OF THE LITERATURE SURVEY**

1. Know who writes, what and where about “Image Based Encryption System.”
2. Identify the tools and sources of “Image Based Encryption System.”
3. Prepare the relevant bibliographic entries with abstract of the related topic.

**2.4 TECHNOLOGY STACK**

**1. Programming Language ::** Python

**2. Algorithms ::** AES-128 , Steganography

**3. Platform ::** Windows

**4. Other’s ::** Pillow, PyInstaller

**2.5 SCOPE OF THE STUDY**

The literature survey is confuted though an extensive period, the information is collected from various sources like the IEEE and other technology related articles and journals which is then analysed to produce the desired results. And other databases are used to complete review of literature for the proposed study on cryptography, including search on e-journals websites and other internet sources. In addition to above searches, bibliography of journal articles is also reviewed for more sources as well as websites and consulted various report from eminent experts in the said field of study.

This information is pooled together so as to expedite the results and see through the stock fluctuations.

**2.6 METHODOLOGY**

For preparing of this literature survey following steps are taken to collect the articles about the security features and various cryptography algorithms. To begin with, data collection started with us defining what the literature survey is and what are the steps involved for preparing it. We chose the domain and the topic “A novel image-based implicit password authentication system (IPAS) for

mobile and non-mobile devices” for the base survey of the project. Polled by the collection of numerous articles from journals on the chosen topic. Then for further reference, electronic resources were consulted for collecting articles such as emerald, JCCC@UGC In fonet, Open-Jgate, Directory of Open Access

Journals (DOAJ), etc. After the cross-referencing of the entries of the articles to the considered databases final report on the literature is produced.

**2.7 BIBLIOGRAPHY**

**1 : Multi-factor authentication using threshold cryptography**

Authors :: V. VENUKUMAR and V. PATHARI

Abstract :

MFA plans require all middle of road One Time passwords(OTPs) to be put away for the lifetime of the verification process. They include security dangers at whatever point a confirmation prepare requires the user’s secret word at an open put like a Point-of-Sale terminal or an open ATM booth.

**2 : Advanced password authentication protection by hybrid cryptograph & audio steganography**

Authors :: E. S. I. HARBA

Abstract :

The proposed strategy is utilized sound steganography and AES Cryptography. The authentication key (watchword) is spilled into two parts, the first half is utilized as input content to the stego-crypto handle whereas the second half is utilized as the crypto the key is scrambled utilizing the HMAC-SHA256 hash algorithm.

**3 : Image steganography based on AES algorithm with huffman coding**

Authors :: J. H. KENNEDY, M. T. A. KHAN, M. J. AHMED and M. RASOOL

Abstract :

This paper presents an algorithm, in which the picture is input to AES Encryption to urge the scrambled picture, at that point, it is compressed with Huffman Coding, and the encrypted image is uncompressed and given as the input to AES Decryption to urge the initial image using AES (Advanced Encryption Standard)the algorithm, LSB algorithm, and compression and decompression of that image using Huffman Coding

**4 : Image encryption using Huffman Coding for steganography**

Authors : L. SHARMA and A. GUPTA

Abstract :

Encrypt the message by Huffman code and apply the In Steganography. We separate the picture into multiple pieces which are non-overlapped in nature and the block estimate is 3×3 pixel which is able to consider a framework. For every value from the network can be spoken to in eight-bit where two-bit will utilize as the slightest noteworthy bit (LSB) substitution and quotient esteem differencing (QVD) is connected to other bits.

**5 : Secure image steganography algorithm based on DCT**

Authors : D. R. I. M. SETIADI, E. H. RACHMAWANTO and C. A. SARI

Abstract :

The combination of steganography utilizing discrete cosine change (DCT) and cryptography using the one-time cushion or vernal cipher executed on a computerized picture. The measurement method utilized to decide the quality of stego picture is the crest flag to clamor proportion (PSNR) and normalized cross relationship (NCC) to degree the quality of the extraction of the decrypted message.

**CHAPTER III**

**SYSTEM ANALYSIS**

**3.1 EXISTING SYSTEM**

The existing system in the field of hiding secret data, a lot of researchers recommended the use of a combination of security methods like steganography and cryptography to elevate the security of the system. Over the past few years, the primary mean of user authentication has been the passwords in the form of text. Users register on the systems by putting username and password to login to the system and gain access over it. The user is expected to remember this secret information to pass the user authentication process. So storing confidential information in centralized manner is the today’s existing system.

**3.2 PROPOSED SYSTEM**

In this paper, we propose a system that implements multi-level encryption with decentralized and distributed storage of enterprises’ sensitive information and offers multi-layered security to the system. In this design, we introduce a password security equation, *Password Protection = Encrypt + Hide + Split*. According to our equation, we encrypt the password using AES-128, split it into pieces and hide it behind the images using steganography using AES-128 adding another layer of encryption and store it in a decentralized manner. When the user logs in to the system all the pieces of the password are retrieved from all the steganographic images, merged and decrypted to form a legitimate password in one piece.

**3.3 FEASIBILITY STUDY**

* **Economic Feasibility**:

Here, we find the total cost and benefit of the proposed system over current system. The main cost of this project is the cost of building and maintaining a database. Other than that, the computational power required to run our program is feasible and efficient.

* **Technical Feasibility**:

In consideration to the technologies used in this project, all the software modules are open sourced Even Large-Scale Implementation requires less financial support.

* Social **Feasibility**:

There is always an on-demand for Security and Privacy. Since our product proves to be faster and safer than other proposed systems, the end-users, companies and business ventures will adopt to our project.

**CHAPTER IV**

**DESIGN**

**4.1 UML DIAGRAMS**

A UML diagram is a partial graphical representation (view) of a model of a

system under design, implementation, or already in existence. UML

diagram contains graphical elements (symbols) - UML nodes connected

with edges (also known as paths or flows) - that represent elements in the

UML model of the designed system. The UML model of the system might

also contain other documentation such as use cases written as templated texts.

The kind of the diagram is defined by the primary graphical symbols shown on the diagram. For example, a diagram where the primary symbols in the contents area are classes is class diagram. A diagram which shows use cases and actors is use case diagram. A sequence diagram shows sequence of message exchanges between lifelines.

UML specification does not preclude mixing of different kinds of diagrams, e.g. to combine structural and behavioral elements to show a state machine nested inside a use case. Consequently, the boundaries between the various kinds of diagrams are not strictly enforced.

At the same time, some UML Tools do restrict set of available graphical elements which could be used when working on specific type of diagram.

UML specification defines two major kinds of UML diagram: structure diagrams and behavior diagrams.

Structure diagrams show the static structure of the system and its parts on different abstraction and implementation levels and how they are related to each other.

The elements in a structure diagram represent the meaningful concepts of a system, and may include abstract, real world an implementation concepts.

Behavior diagrams show the dynamic behavior of the objects in a system, which can be described as a series of changes to

the system over time.

**4.2 DIAGRAMS**

**4.21 USE CASE DIAGRAM**

In the Unified Modelling Language (UML), a use case diagram can summarize the details of your system's users (also known as actors) and their interactions with the system. To build one, you'll use a set of specialized symbols and connectors. An effective use case diagram can help your team discuss and represent:

* Scenarios in which your system or application interacts with people,

organizations, or external systems.

* Goals that your system or application helps those entities (known as

actors) achieve.

* The scope of your system.

Diagram

Description automatically generated

**4.22 SEQUENCE DIAGRAM**

A sequence diagram is a type of interaction diagram because it describes how

and in what order a group of objects works together. These diagrams are used

by software developers and business professionals to understand requirements

for a new system or to document an existing process. Sequence diagrams are

sometimes known as event diagrams or event scenarios.

Sequence diagrams can be useful references for

businesses and other organizations.

Try drawing a sequence diagram to:

• Represent the details of a UML use case.

• Model the logic of a sophisticated procedure, function, or operation.

• See how objects and components interact with each other to complete a

process.

• Plan and understand the detailed functionality of an existing or future

scenario.

Graphical user interface

Description automatically generated**4.23 ACTIVITY DIAGRAM**

The basic purposes of activity diagrams is similar to other four diagrams. It captures the dynamic behavior of the system. Other four diagrams are used to show the message flow from one object to another but activity diagram is used to show message flow from one activity to another.

Activity is a particular operation of the system. Activity diagrams are not only used for visualizing the dynamic nature of a system, but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in the activity diagram is the message part.

It does not show any message flow from one activity to another. Activity diagram is sometimes considered as the flowchart. Although the diagrams look like a flowchart, they are not. It shows different flows such as parallel, branched, concurrent, and single.

The purpose of an activity diagram can be described as −

* Draw the activity flow of a system.
* Describe the sequence from one activity to another.
* Describe the parallel, branched and concurrent flow of the system.

Diagram

Description automatically generated

Diagram

Description automatically generated

**4.24 FLOW CHART**

A flowchart is a type of diagram that represents a workflow or process. A flowchart can also be defined as a diagrammatic representation of an algorithm, a step-by-step approach to solving a task. The flowchart shows the steps as boxes of various kinds, and their order by connecting the boxes with arrows.

Diagram

Description automatically generated

**4.25 ENTITY RELATIONSHIP**

An Entity Relationship(ER) Diagram is a type of flowchart that illustrates how “entities” such as people, objects or concepts relate to each other within a system. ER Diagrams are most often used to design or debug relational databases in numerous fields.

ER Models use a defined set of symbols such as rectangles, diamonds, ovals and connecting lines to depict the interconnectedness of entities, relationships and their attributes. They mirror grammatical structure, with entities as

nouns and relationships as verbs.

Diagram

Description automatically generated

**CHAPTER V**

**SYSTEM ARCHITECHTURE**

**5.1 ARCHITECHTURE OVERVIEW**

**OVERALL ARCHITECHTURE:**

**A picture containing graphical user interface

Description automatically generated**

**ENCRYPTION/DECRYPTION PLATFORM:**

**Graphical user interface, application

Description automatically generated**

**5.2 ALGORITHMS**

**5.21 ADVANCED ENCRYPTION STANDARD – 128**

Advanced Encryption Standard (AES) is a specification for the encryption of electronic data established by the U.S National Institute of Standards and Technology (NIST) in 2001. AES is widely used today as it is a much stronger than DES and triple DES despite being harder to implement.

* AES is a block cipher.
* The key size can be 128/192/256 bits.
* Encrypts data in blocks of 128 bits each.

**Working of the cipher :**  
AES performs operations on bytes of data rather than in bits. Since the block size is 128 bits, the cipher processes 128 bits (or 16 bytes) of the input data at a time.

The number of rounds depends on the key length as follows :

* 128 bit key – 10 rounds
* 192 bit key – 12 rounds
* 256 bit key – 14 rounds

**Creation of Round keys :**  
A Key Schedule algorithm is used to calculate all the round keys from the key. So the initial key is used to create many different round keys which will be used in the corresponding round of the encryption.

Chart, box and whisker chart

Description automatically generated

**Encryption :**  
AES considers each block as a 16 byte (4 byte x 4 byte = 128 ) grid in a column major arrangement.

**[ b0 | b4 | b8 | b12 |**

**| b1 | b5 | b9 | b13 |**

**| b2 | b6 | b10| b14 |**

**| b3 | b7 | b11| b15 ]**

Each round comprises of 4 steps :

* SubBytes
* ShiftRows
* MixColumns
* Add Round Key

The last round doesn’t have the MixColumns round.

The SubBytes does the substitution and ShiftRows and MixColumns performs the permutation in the algorithm.

**SubBytes  :**  
This step implements the substitution.

In this step each byte is substituted by another byte. Its performed using a lookup table also called the S-box. This substitution is done in a way that a byte is never substituted by itself and also not substituted by another byte which is a compliment of the current byte. The result of this step is a 16 byte (4 x 4 ) matrix like before.

The next two steps implement the permutation.

**ShiftRows :**  
This step is just as it sounds. Each row is shifted a particular number of times.

* The first row is not shifted
* The second row is shifted once to the left.
* The third row is shifted twice to the left.
* The fourth row is shifted thrice to the left.

(A left circular shift is performed.)

**[ b0 | b1 | b2 | b3 ] [ b0 | b1 | b2 | b3 ]**

**| b4 | b5 | b6 | b7 | -> | b5 | b6 | b7 | b4 |**

**| b8 | b9 | b10 | b11 | | b10 | b11 | b8 | b9 |**

**[ b12 | b13 | b14 | b15 ] [ b15 | b12 | b13 | b14 ]**

**MixColumns :**  
This step is basically a matrix multiplication. Each column is multiplied with a specific matrix and thus the position of each byte in the column is changed as a result.

**This step is skipped in the last round.**

**[ c0 ] [ 2 3 1 1 ] [ b0 ]**

**| c1 | = | 1 2 3 1 | | b1 |**

**| c2 | | 1 1 2 3 | | b2 |**

**[ c3 ] [ 3 1 1 2 ] [ b3 ]**

**Add Round Keys :**  
Now the resultant output of the previous stage is XOR-ed with the corresponding round key. Here, the 16 bytes is not considered as a grid but just as 128 bits of data.

Diagram

Description automatically generated

After all these rounds 128 bits of encrypted data is given back as output. This process is repeated until all the data to be encrypted undergoes this process.

**Decryption :**  
The stages in the rounds can be easily undone as these stages have an opposite to it which when performed reverts the changes.Each 128 blocks goes through the 10,12 or 14 rounds depending on the key size.

The stages of each round in decryption is as follows :

* Add round key
* Inverse MixColumns
* ShiftRows
* Inverse SubByte

The decryption process is the encryption process done in reverse so i will explain the steps with notable differences.

**Inverse MixColumns :**  
 This step is similar to the MixColumns step in encryption, but differs in the matrix used to carry out the operation.

**[ b0 ] [ 14 11 13 9 ] [ c0 ]**

**| b1 | = | 9 14 11 13 | | c1 |**

**| b2 | | 13 9 14 11 | | c2 |**

**[ b3 ] [ 11 13 9 14 ] [ c3 ]**

**Inverse SubBytes :**  
Inverse S-box is used as a lookup table and using which the bytes are substituted during decryption.

**Key Size : 256-bit vs 192-bit vs 128-bit**

There are three different sizes: 256-bit AES, 192-bit AES and 128-bit AES. The largest size, 256-bit AES, is the most secure, while 128-bit is conversely the least secure of the three. That said, all three key sizes are strong enough to repel even the most dedicated brute-force attack, but the two smaller key sizes are theoretically easier to crack (the time it would take to crack 128-bit AES through a brute-force attack is still billions of years).

The middle-grade 192-bit keys are seldom used, with most AES encryption and decryption using either a 256-bit key or 128-bit key. Although 256-bit encryption is more secure, a 128-bit key uses less computing power. Thus, it’s useful for less sensitive data or when the encryption process has limited resources available to it.

**Key Takeaways:**

* AES stands for “Advanced Encryption Standard.”
* The AES algorithm is the industry-standard encryption protocol that protects sensitive information from traditional brute-force attacks.
* The two most common versions are 256-bit AES (providing greater security) and 128-bit AES (providing better performance during the encryption and decryption process).
* With current technology, AES is uncrackable through straightforward, brute-force attacks, and it is used in countless applications, from protecting top-secret or classified information in government agencies to keeping your personal data safe when stored on the cloud.

**5.22 BASE-64 ENCODING**

**Base64** is a *binary-to-text* encoding scheme. It represents binary data in a printable [ASCII](https://asciitable.xyz/) string format by translating it into a radix-64 representation.

Base64 encoding is commonly used when there is a need to transmit binary data over media that do not correctly handle binary data and is designed to deal with textual data belonging to the 7-bit US-ASCII charset only.

One example of such a system is Email (SMTP), which was traditionally designed to work with plain text data in the 7-bit US-ASCII character set. Although, It was later extended to support non US-ASCII text messages as well as non-text messages such as audio and images, It is still recommended to encode the data to ASCII charset for backward compatibility.

Base64 encoding encodes any binary data or non-ASCII text data to printable ASCII format so that it can be safely transmitted over any communication channel.

**WORKING**

Base64 encoding works with a 65-character subset of the US-ASCII charset. The first 64 characters out of the 65-character subset are mapped to an equivalent 6-bit binary sequence (26 = 64). The extra 65th character (=) is used for padding.

Each of the 6-bit binary sequences from 0 to 63 are assigned a Base64 alphabet. This mapping between the 6-bit binary sequence and the corresponding Base64 alphabet is used during the encoding process. Following is the mapping table which is also called the Base64 index or alphabet table -

# The Base64 Alphabet

Value Encoding Value Encoding Value Encoding Value Encoding

0 A 17 R 34 i 51 z

1 B 18 S 35 j 52 0

2 C 19 T 36 k 53 1

3 D 20 U 37 l 54 2

4 E 21 V 38 m 55 3

5 F 22 W 39 n 56 4

6 G 23 X 40 o 57 5

7 H 24 Y 41 p 58 6

8 I 25 Z 42 q 59 7

9 J 26 a 43 r 60 8

10 K 27 b 44 s 61 9

11 L 28 c 45 t 62 +

12 M 29 d 46 u 63 /

13 N 30 e 47 v

14 O 31 f 48 w (pad) =

15 P 32 g 49 x

16 Q 33 h 50 y

The Base64 encoding algorithm receives an input stream of 8-bit bytes. It processes the input from left to right and organizes the input into 24-bit groups by concatenating three 8-bit bytes. These 24-bit groups are then treated as 4 concatenated 6-bit groups. Finally, each 6-bit group is converted to a single character in the Base64 alphabet by consulting the above Base64 alphabet table.

When the input has fewer than 24 bits at the end, zero bits are added (on the right) to form an integral number of 6-bit groups. Then, one or two pad (=) characters are output depending on the following cases -

* **The last chunk of input contains exactly 8 bits:** Four zero bits are added to form two 6-bit groups. Each 6-bit group is converted to the resulting Base64 encoded character using the Base64 index table. After that two pad (=) characters are appended to the output.
* **The last chunk of input contains exactly 16 bits:** Two zero bits are added to form three 6-bit groups. Each of the three 6-bit groups is converted to the corresponding Base64 alphabet. Finally a single pad (=) character is appended to the output.

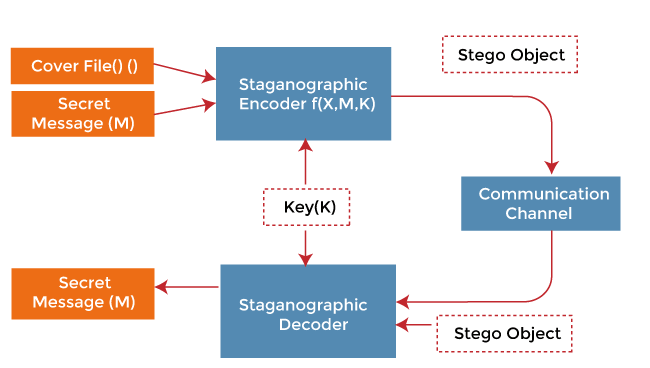
**5.23 IMAGE STEGANOGRAPHY**

The modern world of computation revolves around the word "Data". However, why is data so fascinating? In this modern world, people are starting to realize the significance of the data in order to expand the scope of their businesses. Business owners used data to potentially predict customer trends, increase sales, and push the organization to newer heights.

With the help of the rapid advancement in technology and data for nonstop revolution, it has become the chief importance to secure data. Data sharing has also increased as thousands of messages and data are being transmitted on the internet day-to-day from one place to another. Data protection is the key concern of the sender, and, significantly, we encrypt the message in a secret method that only the receiver can understand.

Steganography refers to the process of hiding a secret message within a larger one in such a manner that someone cannot know the presence or contents of the hidden message. The objective of Steganography is to maintain secret communication between two parties. Unlike Cryptography, where we can conceal the contents of a secret message, Steganography conceals the fact that a message is transmitted. Even though Steganography differs from Cryptography, there are various analogies between the two. Some authors classify Steganography as a form of Cryptography as hidden communication seems like a secret message.

**BASIC MODEL OF STEGANOGRAPHY**

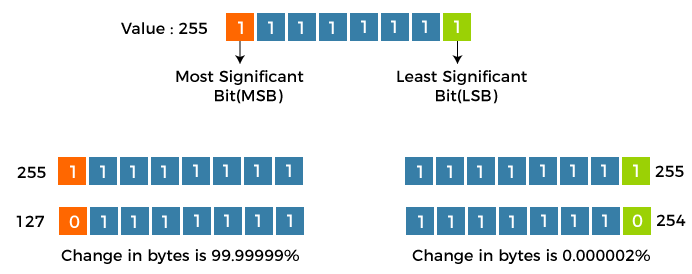


As we can observe in the above image, both the actual file (X) and secret message (M) that must be hidden are fed into a steganographic encoder as input. Steganographic Encoder function, f(X, M, K) embeds the secret message into a cover image file using techniques like least significant bit encoding. The resulting stego image looks very similar to your cover image file, with no visible changes. This completes encoding. To retrieve the secret message, the stego object is fed into Steganographic Decoder.

**UNDERSTANDING THE LEAST SIGNIFICANT BIT STEGANOGRAPHY**

We can describe a digital image as a finite set of digital values, known as Pixels. Pixels are the smallest individual element of an image, holding values representing the brightness of a particular colour at any specific point. Thus, we can think of an image as a matrix (or a two-dimensional array) of pixels consisting of a static number of rows and columns.

LSB (or Least Significant Bit) is a method where we can modify the last bit of every pixel, replacing them with the data bit of the secret message.



From the above image, we can observe that if we alter the Most Significant Bit (or MSB), it will have a larger impact on the final values; however, if we alter the Least Significant Bit (or LSB), the impact on the final value is minimal. Hence, we use Least Significant Bit (LSB) Steganography.

Each pixel consists of three values - Red, Green, Blue. These values range from 0 to 255, which means that they are 8-bit values. Let us understand the working of this technique using an example. Suppose that we want to hide the message "hi" into an image of size 4x4, which has the pixel values as shown below:

[(225, 12, 99), (155, 2, 50), (99, 51, 15), (15, 55, 22), (155, 61, 87), (63, 30, 17), (1, 55, 19), (99, 81, 66), (219l, 77, 91), (69, 39, 50), (18, 200, 33), (25, 54, 190)]

With the help of the ASCII Table, we can convert the secret message into decimal values and then into a binary form: **0110100 0110101**. Now, we can iterate through the pixel values one after one.

Once we convert them into binary, we can replace each least significant bit with that message bits in a sequential manner (For instance, the binary of **225** is **11100001**, we can then replace the last bit, the bit in the right (1), with the initial data bit (0) and so on).

This will allow us to modify the values of the pixel by +1 or -1 only, which is not perceptible at all. The output values of the pixels after performing LSBS is as follows:

[(224, 13, 99), (154, 3, 50), (98, 50, 15), (15, 54, 23), (154, 61, 87), (63, 30, 17), (1, 55, 19), (99, 81, 66), (219, 77, 91), (69, 39, 50), (18, 200, 33), (25, 54, 190)]

**5.24 DECENTRALIZATION OF STEG IMAGE**

First to understand how decentralization of a single image works, we have to understand the basic properties of a image.

**Basics of images**

**Pixels**

An image is made of tiny, square-like elements called *pixels*. Even a small image can contain millions of such pixels of different colors.

**Image properties**

Every image has three main properties:

* **Size** — This is the height and width of an image. It can be represented in centimeters, inches or even in pixels.
* **Color space** — Examples are **RGB** and **HSV** color spaces.
* **Channel** — This is an attribute of the color space. For example, RGB color space has three types of colors or attributes known as ***R****ed*, ***G****reen* and ***B****lue* (hence the name RGB).

**Pixel values**

The colors of an image are denoted by its pixel values. A pixel can have only one color but can be merged to create multiple colors.

In a grayscale image where there is only one channel, a pixel value has just a single number ranging from 0 to 255 (both inclusive). The pixel value 0 represents black and the pixel value 255 represents white. A pixel value in a grayscale image can be represented as follows:

[40]

In an RGB image where there are three color channels, a pixel value has three numbers, each ranging from 0 to 255 (both inclusive). For example, the number 0 of a pixel in the red channel means that there is no red color in the pixel while the number 255 means that there is 100% red color in the pixel. This interpretation is also valid for pixels in the other two channels. A pixel value in an RGB image can be represented as follows:

[255, 255, 0]

This pixel value represents the yellow color.

So the next layer of security we add is converting the steg image into a three dimensional array and splitting it into parts and storing it separately.

After conversion and splitting, the final data to be stored in the database will look like this,

[[[178 204 231] [173 199 226] [174 200 227] ... [153 188 218] [153 188 218] [154 189 219]] [[174 200 227] [171 197 224] [175 201 228] ... [151 186 216] [149 184 214] [147 182 212]] [[171 197 224] [170 196 223] [175 201 228] ... [150 185 215] [147 182 212] [144 179 209]]

[[130 94 46] [142 105 53] [182 143 86] ... [ 56 55 51] [ 52 50 53] [ 44 41 48]] [[137 96 66] [126 83 48] [154 111 66]

[ 43 42 37] [ 38 36 39] [ 37 34 43]] [[ 98 56 32] [110 67 35] [130 87 44] ... [ 40 39 34] [ 37 35 40] [ 44 41 52]]]

**CHAPTER VI**

**SYSTEM IMPLEMENTATION**

**6.1 SOFTWARE CODING**

from email.mime import message

from cryptosteganography import CryptoSteganography

from Crypto.Cipher import AES

from secrets import token\_bytes

from tkinter import \*

from tkinter.ttk import \*

from PIL import Image

from numpy import asarray

import numpy as np

import time

key = token\_bytes(16)

def splitImgArray():

image = Image.open('img/output\_image\_file.png')

data = asarray(image)

arr = np.array(data)

arr1,arr2,arr3,arr4 = np.array\_split(arr, 4)

#arr1, arr2, arr3, arr4 are the segments stored on database

firstcon = np.concatenate((arr1,arr2))

secondcon = np.concatenate((firstcon,arr3))

final = np.concatenate((secondcon,arr4))

pil\_image=Image.fromarray(final)

return pil\_image

# pil\_image.show()

def steganographyCipher(ciphertextsteg):

global secret

crypto\_steganography = CryptoSteganography(keyy)

crypto\_steganography.hide('img/input\_image\_name.jpg', 'img/output\_image\_file.png', ciphertextsteg)

imgArr = splitImgArray()

secret = crypto\_steganography.retrieve(imgArr)

def steganographyNonce(ciphertextsteg):

global Noncee

crypto\_steganography = CryptoSteganography(keyy)

crypto\_steganography.hide('img/nonce.jpg', 'img/nonce\_output.png', ciphertextsteg)

Noncee = crypto\_steganography.retrieve('img/nonce\_output.png')

def steganographyTag(ciphertextsteg):

global Tagg

crypto\_steganography = CryptoSteganography(keyy)

crypto\_steganography.hide('img/tag.jpg', 'img/tag\_output.png', ciphertextsteg)

Tagg = crypto\_steganography.retrieve('img/tag\_output.png')

def encrypt(msg):

cipher = AES.new(key, AES.MODE\_EAX)

nonce = cipher.nonce

ciphertext, tag = cipher.encrypt\_and\_digest(msg.encode('ascii'))

return nonce, ciphertext, tag

def decrypt(nonce, ciphertext, tag):

cipher = AES.new(key, AES.MODE\_EAX, nonce=nonce)

plaintext = cipher.decrypt(ciphertext)

try:

cipher.verify(tag)

return plaintext.decode('ascii')

except:

return False

def splitCheck(ciphersplit):

leng = len(ciphersplit) // 4

st = ''

for i in range(0,len(ciphersplit)):

if i % leng == 0 and i != 0:

st += '-'

st += ciphersplit[i]

neww = st.split('-')

arr = []

if len(neww) > 4:

for i in range(0, 3):

arr.append(neww[i])

arr.append(neww[3] + neww[4])

return arr

else:

return neww

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

try:

print(" ")

print("\*\*\*\*\*\*\*\*\*\*\*Welcome to Image based encryption System\*\*\*\*\*\*\*\*\*\*\*")

print(" ")

Message = input("Enter a message: ")

print(" ")

print("Doing encryption....")

time.sleep(1.2)

nonce, ciphertext, tag = encrypt(Message)

print("Encryption Done")

time.sleep(1.2)

print(" ")

print("Cipher text is: ", ciphertext)

print(" ")

inp = input("Do you wanna proceed with steganography? y/n : ")

if(inp == "y" or inp == "Y"):

global keyy

print(" ")

keyy = input("Enter a key to be hidden on image: ")

print(" ")

print("Doing steganography....")

time.sleep(1.2)

steganographyCipher(ciphertext)

steganographyNonce(nonce)

steganographyTag(tag)

print("Done with steganography")

print(" ")

inpp = input("Do you wanna proceed with decyption? y/n : ")

time.sleep(1.2)

print(" ")

if(inpp == "y" or inpp=="Y"):

plaintext = decrypt(Noncee, secret, Tagg)

if not plaintext:

print('Message is corrupted')

else:

print(f'Plain text is: {plaintext}')

finally:

print(" ")

print("Thank you for using our software!")

print(" ")

**CHAPTER VII**

**SYSTEM TESTING**

**7.1 UNIT TESTING**

**7.11 AES-128 MODULE**

The AES-128 module is the first part of the encryption system. This module is coded separately and tested in this first part of unit testing. The coding and the output has been attached.

**CODING:**

from Crypto.Cipher import AES

from secrets import token\_bytes

key = token\_bytes(16)

def encrypt(msg):

cipher = AES.new(key, AES.MODE\_EAX)

nonce = cipher.nonce

ciphertext, tag = cipher.encrypt\_and\_digest(msg.encode('ascii'))

print(nonce, ' ', ciphertext, ' ', tag )

return nonce, ciphertext, tag

def decrypt(nonce, ciphertext, tag):

cipher = AES.new(key, AES.MODE\_EAX, nonce=nonce)

plaintext = cipher.decrypt(ciphertext)

try:

cipher.verify(tag)

return plaintext.decode('ascii')

except:

return False

nonce, ciphertext, tag = encrypt(input('Enter a message: '))

plaintext = decrypt(nonce, ciphertext, tag)

print(f'Cipher text: {ciphertext}')

if not plaintext:

print('Message is corrupted')

else:

print(f'Plain text: {plaintext}')

**OUTPUT:**

**Text

Description automatically generated**

**7.12 STEGANOGRAPHY MODULE**

The steganography module is the second part of the encryption system. This module is coded separately and tested in this second part of unit testing. The coding and the output has been attached.

**CODING:**

from cryptosteganography import CryptoSteganography

crypto\_steganography = CryptoSteganography('My secret password key')

# Save the encrypted file inside the image

crypto\_steganography.hide('img/input\_image\_name.jpg', 'img/output\_image\_file.png', 'My secrett')

secret = crypto\_steganography.retrieve('img/output\_image\_file.png')

print(secret)

**OUTPUT:**

**Text

Description automatically generated**

**7.13 DECENTRALIZATION MODULE**

The decentralization module is the third part of the encryption system. This module is coded separately and tested in this second part of unit testing. The coding and the output has been attached.

**CODING:**

from PIL import Image

from numpy import asarray

import numpy as np

image = Image.open('img/output\_image\_file.png')

data = asarray(image)

arr = np.array(data)

arr1,arr2,arr3,arr4 = np.array\_split(arr, 4)

firstcon = np.concatenate((arr1,arr2))

secondcon = np.concatenate((firstcon,arr3))

final = np.concatenate((secondcon,arr4))

print(final)

pil\_image=Image.fromarray(final)

pil\_image.show()

**OUTPUT:**

**Text

Description automatically generated**

**7.2 INTEGRATION TESTING**

Since, every modules is confirmed to be working via unit testing, we have to integrate all the modules and test it to confirm its working status. The integrated code and its respective output has been attached below.

**CODING:**

from email.mime import message

from cryptosteganography import CryptoSteganography

from Crypto.Cipher import AES

from secrets import token\_bytes

from tkinter import \*

from tkinter.ttk import \*

from PIL import Image

from numpy import asarray

import numpy as np

import time

key = token\_bytes(16)

def splitImgArray():

image = Image.open('img/output\_image\_file.png')

data = asarray(image)

arr = np.array(data)

arr1,arr2,arr3,arr4 = np.array\_split(arr, 4)

#arr1, arr2, arr3, arr4 are the segments stored on database

firstcon = np.concatenate((arr1,arr2))

secondcon = np.concatenate((firstcon,arr3))

final = np.concatenate((secondcon,arr4))

pil\_image=Image.fromarray(final)

return pil\_image

# pil\_image.show()

def steganographyCipher(ciphertextsteg):

global secret

crypto\_steganography = CryptoSteganography(keyy)

crypto\_steganography.hide('img/input\_image\_name.jpg', 'img/output\_image\_file.png', ciphertextsteg)

imgArr = splitImgArray()

secret = crypto\_steganography.retrieve(imgArr)

def steganographyNonce(ciphertextsteg):

global Noncee

crypto\_steganography = CryptoSteganography(keyy)

crypto\_steganography.hide('img/nonce.jpg', 'img/nonce\_output.png', ciphertextsteg)

Noncee = crypto\_steganography.retrieve('img/nonce\_output.png')

def steganographyTag(ciphertextsteg):

global Tagg

crypto\_steganography = CryptoSteganography(keyy)

crypto\_steganography.hide('img/tag.jpg', 'img/tag\_output.png', ciphertextsteg)

Tagg = crypto\_steganography.retrieve('img/tag\_output.png')

def encrypt(msg):

cipher = AES.new(key, AES.MODE\_EAX)

nonce = cipher.nonce

ciphertext, tag = cipher.encrypt\_and\_digest(msg.encode('ascii'))

return nonce, ciphertext, tag

def decrypt(nonce, ciphertext, tag):

cipher = AES.new(key, AES.MODE\_EAX, nonce=nonce)

plaintext = cipher.decrypt(ciphertext)

try:

cipher.verify(tag)

return plaintext.decode('ascii')

except:

return False

def splitCheck(ciphersplit):

leng = len(ciphersplit) // 4

st = ''

for i in range(0,len(ciphersplit)):

if i % leng == 0 and i != 0:

st += '-'

st += ciphersplit[i]

neww = st.split('-')

arr = []

if len(neww) > 4:

for i in range(0, 3):

arr.append(neww[i])

arr.append(neww[3] + neww[4])

return arr

else:

return neww

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

try:

print(" ")

print("\*\*\*\*\*\*\*\*\*\*\*Welcome to Image based encryption System\*\*\*\*\*\*\*\*\*\*\*")

print(" ")

Message = input("Enter a message: ")

print(" ")

print("Doing encryption....")

time.sleep(1.2)

nonce, ciphertext, tag = encrypt(Message)

print("Encryption Done")

time.sleep(1.2)

print(" ")

print("Cipher text is: ", ciphertext)

print(" ")

inp = input("Do you wanna proceed with steganography? y/n : ")

if(inp == "y" or inp == "Y"):

global keyy

print(" ")

keyy = input("Enter a key to be hidden on image: ")

print(" ")

print("Doing steganography....")

time.sleep(1.2)

steganographyCipher(ciphertext)

steganographyNonce(nonce)

steganographyTag(tag)

print("Done with steganography")

print(" ")

inpp = input("Do you wanna proceed with decyption? y/n : ")

time.sleep(1.2)

print(" ")

if(inpp == "y" or inpp=="Y"):

plaintext = decrypt(Noncee, secret, Tagg)

if not plaintext:

print('Message is corrupted')

else:

print(f'Plain text is: {plaintext}')

finally:

print(" ")

print("Thank you for using our software!")

print(" ")

**OUTPUT:**

**Graphical user interface, application

Description automatically generated**

**7.3 TEST CASES**

|  |  |  |  |
| --- | --- | --- | --- |
| S. No | Test Case | Test Step | Test Result |
| 1 | Doing Encryption by getting secret message | Taking secret message and it will do base 64 encoding and it gives encoded ciphertext | Enter the Message:  Secret  Cipher text is:  b’\bb\ew\323\’ |
| 2 | Doing Steganography with ciphertext | Takes a ciphertext and it will do steganography with by hiding inside image | Enter a key to be hidden: key  Doing Steganography…  Steganography is Done…. |
| 3 | Doing Image Segmentation | Converting Image to Array and Splitting Array into 3 different Part | Carried out on BackEnd |
| 4 | Doing Decryption | Takes a ciphertext and converts it to plain text | Enter a key to retrieve: key  Plain text is:  Secret |

**CHAPTER VIII**

**CONCLUSION**

**9.1 RESULTS AND DISCUSSION**

The main objective of this research was to determine if the security of the system can be enhanced by combining the security methods such as, steganography, encryption and splitting the confidential data which is password in our case. Also, by doing this, we wanted to test whether storing the confidential data in decentralized fashion helps to improve security. Referring to the results that we obtained by implementing the proposed solution on various platforms, we conclude that the solution is highly adaptable as Python programming is cross-platform. In addition, implementing multi-layered encryption with the use of stronger encryption methods like AES-128 boosts the difficulty level for the hackers to penetrate through the system.

**9.2 FUTURE ENHANCEMENTS**

For future enhancement, we would need to optimize the code and pick random passwords from the password list and store the pieces randomly in the images to bolster the level of the security in the system and make penetrating through our system a strenuous task for attackers. Also, work on the latency as code optimization would improve the performance and condense the latency rate.

**APPENDICES**

**A.1 SAMPLE SCREENS**

**Text

Description automatically generated**

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Graphical user interface, application

Description automatically generated

**REFERENCES**

* V. Venukumar and V. Pathari, “Multi-factor authentication using threshold cryptography,” in 2016 International Conference on Advances in Computing, Communications, and Informatics (ICACCI), Jaipur, India, Sept 21-24, 2016
* L. Sharma and A. Gupta, “Image encryption using Huffman Coding for steganography,,” International Journal of Advance research , Ideas and Innovations in Technology, vol. 2, no. 5, pp. 1-10, 2016
* E. S. I. Harba, “Advanced password authentication protection by hybrid cryptography & audio steganography,” Iraqi Journal of Science, vol. 59, no. 1C, pp. 600-606, 2018
* J. H. Kennedy, M. T. A. Khan, M. J. Ahmed and M. Rasool, “Image steganography based on AES algorithm with huffman coding,” International Journal of Engineering Science and Computing, April 2017, vol. 7, no. 4, pp. 6352-6355, 2017
* D. R. I. M. Setiadi, E. H. Rachmawanto2 and C. A. Sari, “Secure image steganography algorithm based on DCT,” Journal of Applied Intelligent System, vol. 2, no. 1, pp. 1-11, April 2017