A Project Report

on

CRYPTO WEB

Submitted in partial fulfillment of the
requirement for the award of the degree
of

Bachelor of Technology in Computer Science and Engineering
CSE 4TH YEAR

Under The Supervision of
Dr. Ganga Sharma

Designation :
Associate Professor

Submitted By :

Vaibhab Kumar Yadav -

18SCSE1010301

Shivam Singh -

18SCCSE1010221

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA
INDIA
DECEMBER , 2021

DECEMBER , 2021

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING ,
GALGOTIAS UNIVERSITY,
GREATER NOIDA

CANDIDATE'S DECLARATION

I/We hereby certify that the work which is being presented in the thesis/project/dissertation, entitled " CRYPTO WEB " in partial fulfillment of the requirements for the award of the Bachelor of Technology in Computer Science and Engineering submitted in the School of Computing Science and Engineering of Galgotias University, Greater Noida, is an original work carried out during the period of November of 2021 to December of 2021, under the supervision of Dr. Ganga Sharma Designation Associate Professor , Department of Computer Science and Engineering of School of Computing Science and Engineering , Galgotias University, Greater Noida . The matter presented in the report has not been submitted by us for the award of any other degree of this or any other places.

VAIBHAB KUMAR YADAV ,
18SCSE1010301

SHIVAM SINGH ,
18SCSE1010221

This is to certify that the above statement made by the candidates is correct to the best of my
knowledge.

Supervisor Name : Dr. Ganga Sharma
Associate Professor ,Galgotias university

CERTIFICATE

The Final Thesis/Project Viva-Voce examination of Vaibhab Kumar Yadav - 18scse1010301 and Shivam Singh - 18scse1010221 has been held on _____ and their work is recommended for the award of B.TECH Degree .

Signature of Examiner(s) **Signature of Supervisor(s)**

Signature of Dean

Date: December , 2021
Place: Greater Noida

TABLE OF CONTENTS

Abstract

In this digital world with everything is online , all the bodies from individuals to organization store their sensitive content online including passwords , addresses etc. But some organizations are storing sensitive data as it is. They are either storing it at some shared places or uploading it to some server. This is very risk as any one can see their sensitive data. The sensitive data can be either organizations data or clients data. Both are need to be saved securely. This is where Encryption comes into the picture .

Encryption is the encoding of information so that only those who have access to a password or encryption key can access it. Encryption protects data content, rather than preventing unauthorized interception of or access to data transmissions. Due to recent developments in stego analysis, providing security to personal contents, messages, or digital images using steganography has become difficult. By using stego analysis, one can easily reveal existence of hidden information in carrier files. This project introduces a novel steganographic approach for communication between two private parties. The approach introduced in this project makes use of both steganographic as well as cryptographic techniques. In Cryptography we are using RSA. In Steganography we are using Image Steganography for hiding the data. And we also use Mutual Authentication process to satisfy all services in Cryptography i.e., Access Control, Confidentiality, Integrity, Authentication. In this way we can maintain the data more securely. Since we use RSA algorithm for securing the data and again on

this we perform Steganography to hide the data in an image. Such that any other person in the network cannot access the data present in the network. Only the sender and receiver can retrieve the message from the data.

By this project what we are trying to achieve is :

1. We will encrypt the secure file with one secret key
2. Save the file
3. When they want to download it again, then they need to decrypt it with thesecret key

Now the problem is solved. User can encrypt the data with a secret key and no one can be able to view the file without secret key.The only thing is that the key has to bekept secret. This will help the organization to keep their sensitive data securely .
Encryption is already a essential part of the digital world and is only going to grow from here .

Keywords : Rivest-Shamir-Adelman(RSA), Crptography, Steganograp

Introduction

1.1 Introduction :

With the increase in number of security breaches and theft of user data , most people are exposed to a lot of cyber attacks like identity , money theft , black mailing , extortion and many more . To avoid any more personal attack we need to adapt few practices which keep us safe in the digital world . One of Cryptographies is encryption which encrypts/protects a data/file with a password so that only a person with a password can access it . Encryption is the transformation of plain data (known as plain text) into unintelligible data (known as cipher text) through an algorithm referred to as cipher. Encryption is the transformation of data into a form that is as close to impossible as possible to read without the appropriate knowledge (a key). Itspurpose is to ensure privacy by keeping information hidden from anyone for whom it is not intended, even those who have access to the encrypted data . Cryptography has a main role in embedded systems design. This prevent unauthorized person from accessing the file or message and thus safeguarding the important data and the user .

Digital communication witnesses a noticeable and continuous development in many applications in the Internet. Hence, secure communication sessions must be provided. The security of data transmitted across a global network has turned into a key factor on the network performance measures. So, the confidentiality and the integrity of

data are needed to prevent eavesdroppers from accessing and using transmitted data. Steganography and Cryptography are two important techniques that are used to provide network security. The aim of this project is to develop a new approach to hiding a secret information in an image, by taking advantage of benefits of combining cryptography and steganography.

1.1.1 Cryptography -

Cryptography is one of the traditional methods used to guarantee the privacy of communication between parties. This method is the art of secret writing, which is used to encrypt the plaintext with a key into ciphertext to be transferred between parties on an insecure channel. Using a valid key, the cipher text can be decrypted to the original plaintext. Without the knowledge of the key, nobody can retrieve the plaintext. Cryptography plays an essential role in many factors required for secure communication across an insecure channel, like confidentiality, privacy, non repudiation, key exchange, and authentication .

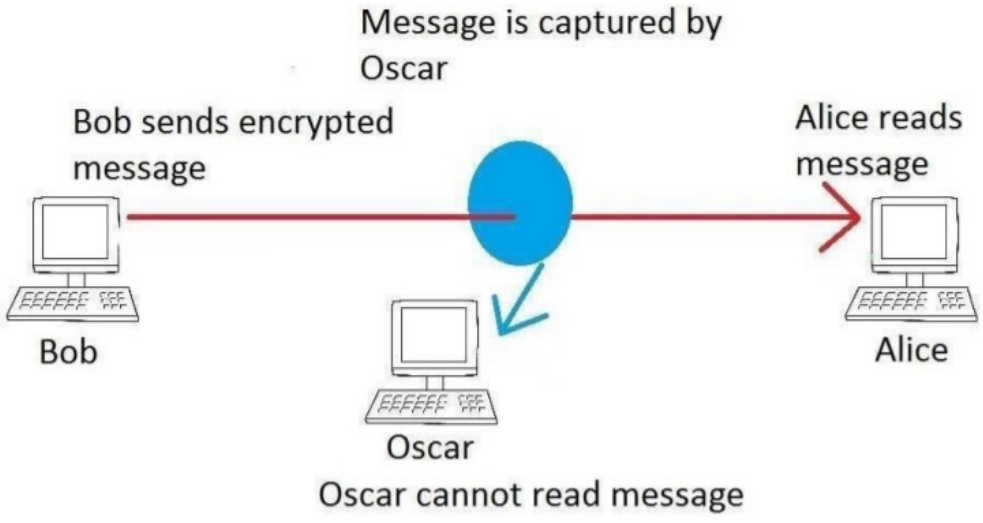


Fig 1.1.1 : Cryptography as a flow model.

1.1.1.1 Symmetric / Secret Key Cryptography -

The technique of Secret key encryption can also be known as the symmetric-key, shared key, single-key, and eventually private-key encryption. The technique of private key uses for all sides encryption and decryption of secret data. The original information or plaintext is encrypted with a key by the sender side also the similarly key is used by the receiver to decrypt a message to obtain the plaintext. the key will be known only by a people who are authorized to the encryption/decryption. However, the technique affords the good security for transmission but there is a difficulty with the distribution of the key. If one stole or explore the key he can get whole data without any difficulty. An example of Symmetric-Key is DES Algorithm.

1.1.1.2 Asymmetric / Public Key Cryptography -

We can call this technique as asymmetric cryptosystem or public key cryptosystem, this technique use two keys which are mathematically associated, use separately for encrypting and decrypting the information. In this technique, when we use the private key, there are no possibilities to obtain the data or simply discover the other key. The key used for encryption is stored public therefore it's called public key, and the decryption key is stored secret and called private key. An example of Asymmetric-Key Algorithm is RSA.

1.1.2 Steganography -

It can be defined as the science of hiding and communicating data

through apparently reliable carriers in attempt to hide the existence of the data. So, there is no knowledge of the existence of the message in the first place. If a person views the cover which the information is hidden inside, he or she will have no clue that there is any covering data, in this way the individual won't endeavour to decode the data. The secret information can be inserted into the cover media by the stego system encoder with using certain algorithm. A secret message can be plaintext, an image, ciphertext, or anything which can be represented in form of a bitstream. after the secret data is embedded in the cover object, the cover object will be called as a stego object also the stego object sends to the receiver by selecting the suitable channel, where decoder system is used with the same stego method for obtaining original information as the sender would like to transfer .

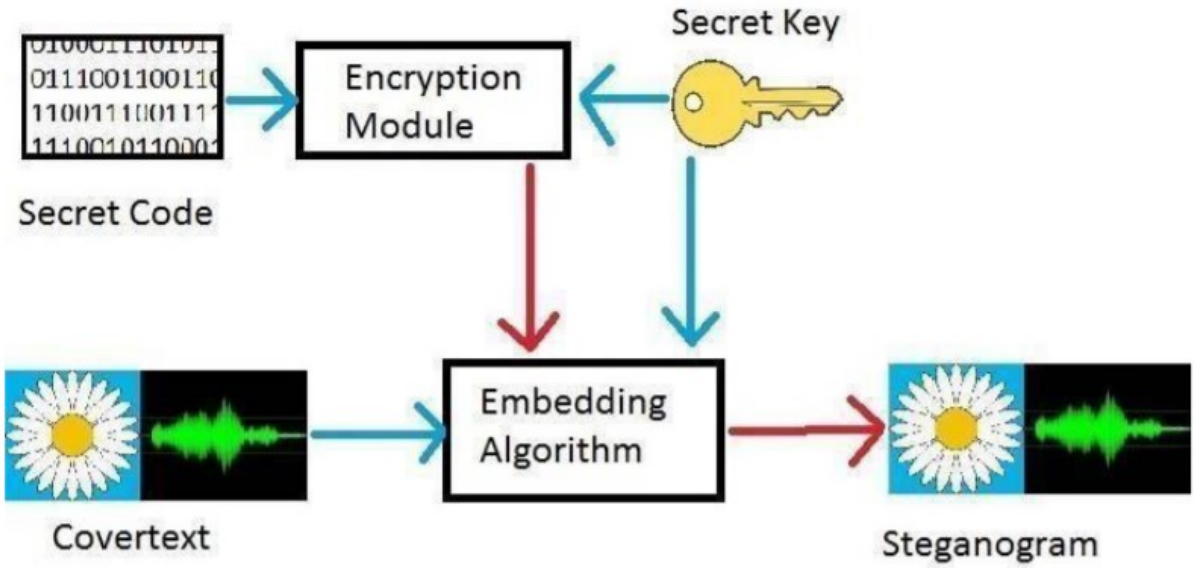


Fig 1.1.2 : Stegonography as a flow model.

1.1.3 Types of Steganography -

There are various types of steganography.

A. Text Files

The technique of embedding secret data inside a text is identified as text stego. Text steganography needs a low memory because this type of file can only store text files. It affords fast transfer or communication of files from a sender to receiver.

B. **Image Files**

It is the procedure in which we embed the information inside the pixels of image. So, that the attackers cannot observe any change in the cover image. LSB approach is a common image steganography algorithm.

C. **Audio Files**

It is the process in which we hide the information inside an audio. There are many approaches to hide secret information in an audio files for examples Phase Coding, LSB .

D. Video Files

It is the process of hiding some secret data inside the frames of a video.

1.1.4 **Steganography versus Cryptography -**

Steganography and cryptography are used for the purpose of data transmission over an insecure network without the data being exposed to any unauthorized persons. Steganography embeds the data in a cover image while cryptography encrypts the data. The advantage of Steganography is that, the look of the file isn't changed and this it will not raise any doubt for the attacker to suspect that there may be some data hidden unlike cryptography that encrypts the data and sends it to network.

1.1.5 **Benefits of Steganography and Cryptography** -

It is noted that steganography and cryptography alone is insufficient for the security of information, therefore if we combine these systems, we can generate more reliable and strong approach. The combination of these two strategies will improve the security of the information. This combined will fulfill the prerequisites, for example, memory space, security, and strength for important information transmission across an open channel. Also, it will be a powerful mechanism which enables people to communicate without interferes of 6 eavesdroppers even knowing there is a style of communication in the first place.

1.1.6 **Applications of Steganography** -

(i) Secret Communication : Steganography does not advertise secret communication and therefore avoids scrutiny of the sender message. A

trade secret, blueprint, or other sensitive information can be transmitted without alerting potential attackers.

(ii) Feature Tagging : Elements can be embedded inside an image, such as the names of the individuals in a photo or location in a map. Copying the stego image also copies all of the embedded features and only parties who possess the decode stego key will be able to extract and view the features.

(iii) Copyright Protection : Copy protection mechanisms that prevent the data, usually digital data from being copied. The insertion and analysis of water marks to protect copyrighted material is responsible for the percent rise of interest digital steganography and data embedding.

1.2 PROBLEM STATEMENT

The purpose of this project is to provide the correct data with security to the users. For some of the users the data might be lost during the transmission process in the network and for some, the data might be changed by the unauthorized person in the network and there are some other security problems in the network. Our application will give you more Security to the data present in the network and there will be able to reduce the loss of data in the network which will be transmitted from the sender to the receiver using the latest technologies. Only the Authorized persons i.e., who are using our application will be 8 there in the Network. The proposed algorithm is to hide the audio data effectively in an image without any suspicion of the data being

hidden in the image. It is to work against the attacks by using a distinct new image that isn’t possible to compare. The aim of the project is to hide the data in an image using steganography and ensure that the quality of concealing data must not be lost. We used a method for hiding the data in a distinct image file in order to securely send over the network without any suspicion the data being hidden. This algorithm, though requires a distinct image which we can use as a carrier and hide the data which is well within the limits of the threshold that the image can hide, that will secure the data.

1.3 The tools and technologies used in the project :

1. Cryptography techniques - Encryption using AES (Advance Encryption Standard)

2. Web development technologies - Html , CSS , Javascript etc
3. Tools - Sublime Text , Browser Sync , GitHub , Git
4. Hosting - Github Pages

Literature Review

Encryption is one of the techniques that ensure the security of images used in various domains like military intelligence, secure medical imaging services, intranet and internet communication, e-banking, social networking image communication like Facebook, WhatsApp, Twitter etc. All these images travel in a free and open networkeither during storage or communication; hence their security turns out to be a crucial necessity in the grounds of personal privacy and confidentiality. This article reviews and summarizes various image encryption techniques so as to promote development of advanced image encryption methods that facilitate increased versatility and security.

There is much skepticism surrounding cryptography. The National Institute of Standards andTechnology (NIST) has joined forces with the National Security Agency (NSA) to form the "Common Criteria" process known as the Common Criteria for Information Technology Security Evaluation whose aim it is to increase the confidence in cryptographic and information-related security products. Additionally, the Department of Defense (DoD) hasenacted policy directives requiring Information Assurance (IA) professionals to receive information security training in addition to basic IA training for all of its DoD employees . Bhargav-Spantzel contends that there is a recent paradigm in identity management called user-centricity identity management. . Bhargav-Spantzel et al. indicates that the most predominant identity management model on the Internet today is the silo model where users

handle their own data and provide it to organizations separately. One solution to this dilemma offered by Bhargav-Spantzel et al. is the centralized federation model, such as Microsoft's Passport, which removes the inconsistencies and redundancies of the silo model and provides the Web users a seamless experience.

3. METHODOLOGY

3.1 Sender Side -

The Sender side consists of cryptographic and steganography stages. This method starts with cryptographic then steganography.

Cryptography Stage :

In encryption stage, we use RSA (Rivest Shamir Adelson) algorithm. This technique takes two prime numbers. The Encryption can be done using the Plain Text and with “e” values which was generated using the two prime numbers. Then we will get a cipher text, which is communicated to the receiving end for decryption. This encrypted data will be used in steganography stage.

Input= Message + Two Prime Numbers.

Output= Encrypted Message.

3.2 Receiver side

Receiver side consists of steganography and cryptography stages. In receiver side we will first extract embedded data then decrypt it.

In cryptography stage, we use the data which is extracted from stego file and use RSA. We will use the same steps which are used in sender side. The

Decryption can be done using the Encrypted message, receivers private key and senders public key. Input= Encrypted Message + 2 Prime Numbers. Output= Plain Text. Now the Plain Text is in the form of Base-64. After getting the plain text apply Base64 coversion to change the Plain-text to given input, which can be Text, Image, Video, Audio.

3.3 MODULE DIVISION

3.3.1 RSA

The RSA algorithm is the basis of a cryptosystem a suite of cryptographic algorithms that are used for specific security services which enables public key encryption and is widely used to secure sensitive data, particularly when it is being sent over an insecure network 18 such as the internet. RSA was first publicly described in 1977 by Ron Rivest, Adi Shamir and Leonard Adleman of the Massachusetts Institute of Technology, though the 1973 creation of a public key algorithm by British mathematician Clifford Cocks was kept classified by the U.K.'s GCHQ until 1997. In RSA cryptography, both the public and the private keys can encrypt a message; the opposite key from the one used to encrypt a message is used to decrypt it. This attribute is one reason why RSA has become the most widely used asymmetric algorithm, It provides a method to assure the confidentiality, integrity, authenticity, and non-repudiation of electronic communications and data storage.

RSA derives its security from the difficulty of factoring large integers that are the product of two large prime numbers. Multiplying these two numbers is easy, but determining the original prime numbers from the total or factoring is considered infeasible due to the time it would take using even today's supercomputers.

3.3.1.1 Why RSA Algorithm is used ?

The public and private key generation algorithm is the most complex part of RSA cryptography. Two large prime numbers, p and q, are selected. N is calculated by multiplying p and q. This number is used by both the public and private keys and provides the link between them. Its length, usually expressed in bits, is called the key length.

The public key consists of the modulus n and a public exponent e. The e doesn't have to be a secretly selected prime number, as the public key is shared with everyone.

The private key consists of the modulus n and the private exponent d, which is calculated using the Extended Euclidean algorithm to find the multiplicative inverse with respect to the totient of n.

3.3.1.2 RSA Security

RSA security relies on the computational difficulty of factoring large integers. As computing power increases and more efficient factoring algorithms are discovered, the ability to factor larger and larger numbers also increases. Encryption strength is directly to key size, and doubling key length can deliver an exponential increase in strength, although it does impair performance. RSA keys are typically 1024-bits or 2048-bits long, but experts believe that 1024-bit keys are no longer fully secure against all attacks. This is why the government and some industries are moving to a minimum key length of 2048- bits.

Barring an unforeseen breakthrough in quantum computing, it will be many years before longer keys are required, but elliptic curve cryptography (ECC) is gaining with many security experts as an alternative to RSA to implement public key cryptography. It can create faster, smaller and more efficient cryptographic keys.

Modern hardware and software are ECC-ready, and its popularity is likely to grow, as it can deliver equivalent security with lower computing power and battery resource usage, making it more suitable for mobile apps than RSA. Finally, a team of researchers, which included Adi Shamir, a co-inventor of RSA, has successfully created a 4096-bit RSA key using acoustic cryptanalysis; however, any encryption algorithm is vulnerable to attack.

3.3.1.3 Description of Algorithm

- Plaintext is taken from a specified file and then encrypted using RSA Algorithm.
- Encryption and decryption are of following form for same plaintext M and ciphertext C.
- C=(M^e)modn
- M=(C^d)modn
- M=((M^e)^d)modn
- M=(M^ed)modn
- Both sender and receiver must know the value of n.
- The sender knows the value of e, and the receiver knows the value of d. Thus this is a public key encryption algorithm with a public key of PU = {c, n} and private key of PR= {d, n}.

3.3.1.3 RSA algorithm

a) Key Generation :

- Select p and q such that both are the prime numbers, p≠q.
- Calculate n=p×q
- Calculate q(n) = (p-1) (q-1)
- Select an integer e such that : g(d ((n), e)) =1 & 1< e < (n)
- Calculate d; de = 1 mod (q(n))
- Public Key, PU= {e, n}
- Private Key, PR ={d,n} 21

b) Encryption : □

- Plaintext : M □
- Ciphertext: C = (M^e) mod n

c) Decryption: □

- Ciphertext: C □
- Plaintext : M= (C^d) mod n □
- Note 1 : (n) -> Euler's totient function □
- Note 2: Relationship between C and d is expressed as:

  ed (mod (n)) =1

  ed = 1 mod (n)

  d =e−1 mod (n)

3.4 ALGORITHM ILLUSTARTION

Encryption :

Inputs : Message, 2 Prime Numbers, Image, Secret Key

Step 1 : Consider an Input , It can be :

a) Text

b) age

c) Audio

d) Video

Step 2 : Convert the input to Base-64 using Base-64 conversion Algorithm.

Step 3 : After converting into Base-64 we will be getting a String.

Step 4 : Store the entire string in a Text File and save the file.

Step 5 : From that file consider each character and apply RSA.

Step 6 : By Using RSA we will be getting Cipher Text (cm).

Step 7 : Let the Cipher Text (cm) be encrypted message.

Step 8 : Consider an image, And hide the encrypted message(cm) in the given image with the secret key Using Stegnography Algorithm.

Step 9: Now send the Stego-Image to the Receiver.

DECRYPTION :

Inputs : Cipher Text, 2 Prime Numbers, Image, Secret Key

Step 1 : Consider the input be Stego-Image.

Step 2 : Using the Secrert Key , Obtain the hidden message from the StegoImage.

Step 3 : And the obtained message is a Cipher Text. We must decrypt the message.

Step 4 : The Decyption of the message can be done using RSA Algorithm.

Step 5: By Using RSA we will be getting Plain Text.

Step 6 : And thus the receiver will decrypt the message and it is in the form of Base-64.

Step 7 : Finally by using Base-64 algorithm the Base-64 text is converted into the original input, Which can be Text, Image, Audio, Video.

4. DESIGN

Project design is a major step towards a successful project. A project design is a strategic organization of ideas, materials and processes for the purpose of achieving a goal. Project managers rely on a good design to avoid pitfalls and provide parameters to maintain crucial aspects of the project. Project design is an early phase of the project where a project's key features, structure, criteria for success, and major deliverables are all planned out. The point is to develop one or more designs which can be used to achieve the desired project goals. Stakeholders can then choose the best design to use for the actual execution of the project. The project design phase might generate a variety of different outputs, including sketches, flowcharts, HTML screen designs, and more.

So, the design can be implemented using Unified Modeling Language. diagrams such as class diagram, use case diagram, sequence diagram, activity diagrams. UML offers a way to visualize a system’s architectural blueprints in a diagram, including elements such as : □

- Any activities □
- Individual components of the system □
- How the system will run □
- How entities interact with others □
- External user interface

UML is a common language for business analysts, software architects and developers used to describe, specify, design, and document existing or new business processes, structure and behaviour of artifacts of software systems. The key to making a UML diagram is connecting shapes that represent an object or class with other shapes to illustrate relationships and the flow of information and data.

4.2 Use Case Diagram

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams as well. The use cases are represented by either circles or ellipses.

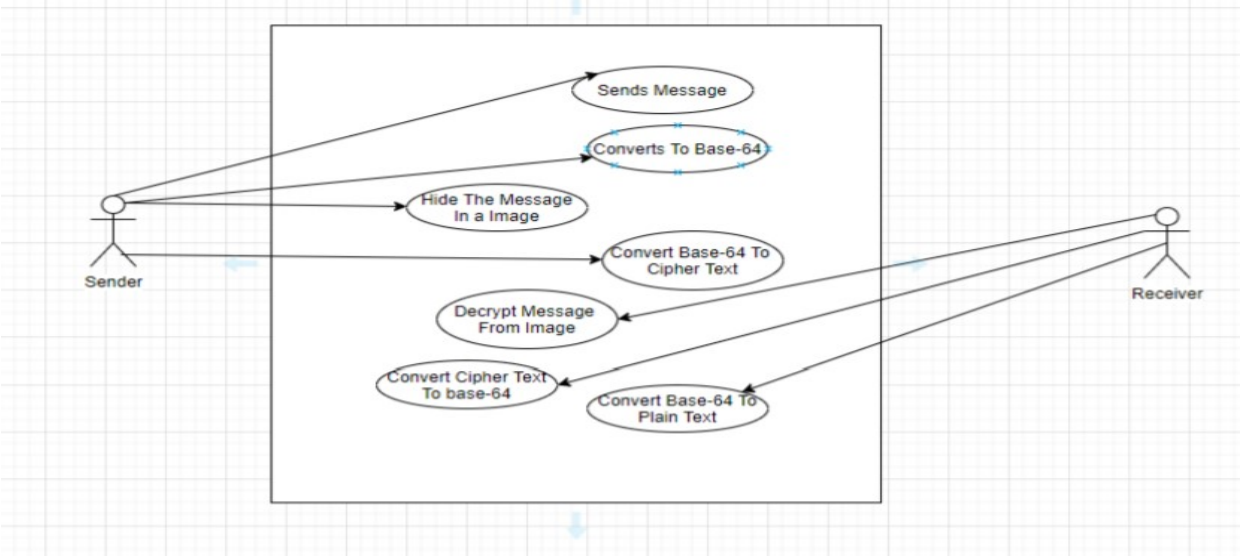


Fig 4.1 Use Case Diagram

4.4 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 are intended to model both computational and organizational processes (i.e., workflows) as well as the data flows intersecting with the related activities. Although activity diagrams primarily show the overall flow of control, they can also include elements showing the flow of data between activities through one or more data stores.

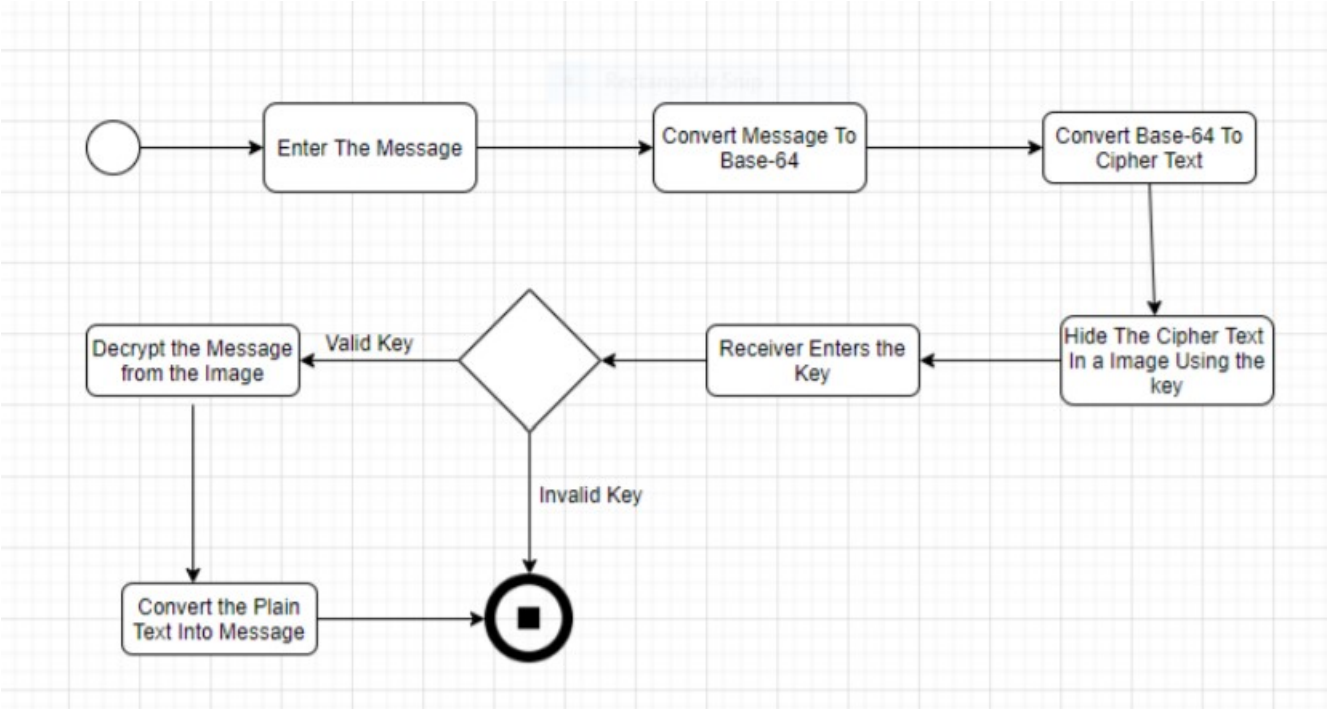


Fig 4.2 Activity Diagram

5. EXPERIMENTAL ANALYSIS AND RESULTS

5.1 SYSTEM CONFIGURATION

5.1.1 Software Requirements:

The software configurations used are -

Operating System: Windows 11

Programming Language : Html , Css , JavaScript

Audio file format: any file format is accepted

5.1.2 Hardware Requirements:

Device - Laptop

RAM: Minimum of 4GB

Storage: SSD preferred

5.2 Source Code

HTML CODE :

```
<!DOCTYPE html>
<html lang="en">
	<head>
		<meta charset="utf-8">
		<title>CRYPTOWEB</title>
		<link rel="stylesheet" type="text/css" href="css.css">
		<link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/5.15.3/css/all.min.css"/>
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
	</head>
	<style>
		body {
			font-family: 'Festive', cursive;
			color: black;
			font-size: 11pt;
		}

		a, a:link, a:visited, a:active {
			color: blue;
			text-decoration: underline;
		}

		a:hover {
			cursor:pointer;
			color: red;
		}

		.black10pointcourier {
			font-family: 'courier';
			color: black;
			font-size: 10pt;
		}
```

```
.container {
        width: 80%;
        margin: 0 auto;
}

button{
         width: 150px;
         height: 40px;
         font-family: 'Festive', cursive;
        border-radius: 15px;
}

button:hover{
        color:white;
        background-color:black;
        border: 8px;
        border-radius: 15px;
}

h1
{
        color: darkmagenta;

}
h4{
        color: rgb(226, 43, 202);
}

.dropzone {
        border: 10px dashed rgb(71, 8, 172);
        width: 100%;
        padding: 6% 2% 6% 2%;
        text-align: center;
        margin: 5px 0 5px 0;
}

.divTablefullwidth{
```

```
            display: table;
            width: 100%;
        }

        .divTable{
            display: table;
        }

        .divTableRow {
            display: table-row;
        }
        .divTableCell {
            display: table-cell;
            padding: 3px 3px;
        }
        .divTableBody {
            display: table-row-group;
        }

        .greenspan {
            color: green;
        }

        .redspan {
            color: red;
        }
    </style>
    <body>
        <div class=container>
            <div class="divTablefullwidth">
                <div class="divTableBody">
                    <div class="divTableRow">
                        <div class="divTableCell" style="float:
left;">
                            <h1>CRYPTOWEB</h1>
                            <h4>Use your web browser to encrypt
and decrypt files.</h4>
                        </div>
```

```
                                        <div class="divTableCell" style="float:
right;">
                                            <h1>
                                            <button id="btnRefresh"
onClick="javascript:location.reload();">Refresh Page</button>
                                            <button id="btnDivEncrypt"
onClick="javascript:switchdiv('encrypt');">Encrypt a File</button>
                                            <button id="btnDivDecrypt"
onClick="javascript:switchdiv('decrypt');">Decrypt a File</button>
                                            </h1>
                                        </div>
                                    </div>
                                </div>
                            </div>


                            <div class=container>
                                <hr>
                            </div>

                            <div class="container" id=divEncryptfile>
                                <h2>Encrypt a File</h2>
                                <p>To encrypt a file, enter a password and drop the file to be
encrypted into the dropzone below.  The file will then be encrypted using the
password, then you'll be given an opportunity to save the encrypted file to
your system.</p>
                            </br></br>
                                <div class="divTable">
                                    <div class="divTableBody">
                                        <div class="divTableRow">
                                            <div
class="divTableCell">Password :</div>

                                            <div class="divTableCell"><input
id=txtEncpassphrase type=password size=30
onkeyup=javascript:encvalidate(); value=""></div>
                                            <div class="divTableCell">(minumum
```

```
length eight characters, make sure it strong!)</div>
                                </div>
                                <div class="divTableRow">
                                        <div class="divTableCell">Password :
(retype)</div>
                                        <div class="divTableCell"><input
id=txtEncpassphraseretype type=password size=30
onkeyup=javascript:encvalidate(); value=""></div>
                                        <div class="divTableCell"><span
class=greenspan id=spnCheckretype></span></div>
                                </div>
                        </div>
                </div>

                <p> </p>

                <div>
                        <div class=dropzone id="encdropzone"
ondrop="drop_handler(event);" ondragover="dragover_handler(event);"
ondragend="dragend_handler(event);">
                                <p>Drag and drop the file to be encrypted into
this dropzone, or click <a onclick=javascript:encfileElem.click();>here</a> to
select file.</p>
                                <p><span id=spnencfilename></span></p>
                        </div>
                        <input type="file" id="encfileElem"
style="display:none" onchange="selectfile(this.files)">
                </div>

                <p> </p>

                <div class="divTable">
                        <div class="divTableBody">
                                <div class="divTableRow">
                                        <div class="divTableCell"><button
id=btnEncrypt onclick=javascript:encryptfile(); disabled>Encrypt
File</button></div>
                                        <div class="divTableCell"><span
```

```
id=spnEncstatus></span></div>
                                </div>
                        </div>
                </div>

                <p> </p>

                <div>
                        <a id=aEncsavefile hidden><button>Save Encrypted
File</button></a>
                </div>

                <p> </p>
        </div>

        <div class="container" id=divDecryptfile>
                <h2>Decrypt a File</h2>
                <p>Decrypt a file using the password that was previously
used to encrypt the file.  After the file is decrypted, you'll be given an
opportunity to save the decrypted file to your system.</p>

                <div class="divTable">
                        <div class="divTableBody">
                                <div class="divTableRow">
                                        <div class="divTableCell">Password</div>
                                        <div class="divTableCell"><input
id=txtDecpassphrase type=password size=30
onkeyup=javascript:decvalidate(); value=""></div>
                                </div>
                        </div>
                </div>

                <p> </p>

                <div>
                                <div class=dropzone  id="decdropzone"
ondrop="drop_handler(event);" ondragover="dragover_handler(event);"
ondragend="dragend_handler(event);">
```

```
                                        <p>Drag and drop file to be decrypted into
this dropzone, or click <a role=button
onclick=javascript:decfileElem.click();>here</a> to select file.</p>
                                        <p><span id=spndecfilename></span></p>
                                    </div>
                                    <input type="file" id="decfileElem"
style="display:none" onchange="selectfile(this.files)">
                        </div>

                        <p> </p>

                        <div class="divTable">
                            <div class="divTableBody">
                                <div class="divTableRow">
                                    <div class="divTableCell"><button
id=btnDecrypt onclick=javascript:decryptfile(); disabled>Decrypt
File</button></div>
                                    <div class="divTableCell"><span
id=spnDecstatus></span></div>
                                </div>
                            </div>
                        </div>

                        <p> </p>

                        <div>
                            <a id=aDecsavefile hidden><button>Save Decrypted
File</button></a>
                        </div>

                        <p> </p>
                    </div>

                    <div class="container">
                        <hr>
                        <h3>Usage</h3>
                        <p>
                        Use this web page to encrypt a file using a password, then
```

use the same password later to decrypt the file. IMPORTANT: The same password that was used to encrypt the file must be used to decrypt the file later. If you loose or forget the password, it cannot be recovered!

<h3>Operation and privacy</h3>

This page uses javascript running within your web browser to encrypt and decrypt files client-side, in-browser.
This page makes no network connections during this process, to ensure that your files and keys never leave the web browser during the process.
This can be independently verified by reviewing the source code for this page, or by monitoring your web browser's networking activity during operation of this page.
This page can also be downloaded and run locally on your system offline.

<h3>Cryptography</h3>

All client-side cryptography is implemented using the Web Crypto API.
Files are encrypted using AES-CBC 256-bit symmetric encryption. The encryption key is derived from the password and a random salt using PBKDF2 derivation with 10000 iterations of SHA256 hashing.

<h3>Compatibility with openssl</h3>

The encryption used by this page is compatible with openssl.

Files encrypted using this page can be decrypted using

openssl using the following command:
openssl aes-256-cbc -d -salt -pbkdf2 -iter 10000 -in encryptedfilename -out plaintextfilename

Files encrypted using the following openssl command can be decrypted using this page:
openssl aes-256-cbc -e -salt -pbkdf2 -iter 10000 -in plaintextfilename -out encryptedfilename

<h3>Running this page offline</h3>

This web page is self-contained. The page does not require any supporting files; all javascript and css for this page is contained in the source code of this page.
To run this page locally on your system offline, simply save this page to your system as a .html file, then open the file from your system in your web browser (optionally with networking disabled).

<footer>

<h2>About us</h2>

CRYPTO WEB is created by us (Shivam & Vaibhab) for the people to encrypt and decrypt the files on the internet. We are B.Tech CSE students from Galgotias University who try to implement what they know in the real world.


```
            <a
href="https://www.instagram.com/p/CJTXP2hhsgo5WlUv3pcdmUCWEwdm
Hqdtldn8Qo0/?utm_medium=copy_link"><span class="fab fa-
instagram"></span></a>
            <a
href="https://instagram.com/shivamsinghrajpoot99?utm_medium=copy_link"
><span class="fab fa-instagram"></span></a>
          </div>
        </div>
      </div>
      <div class="center box">
        <h2>Address</h2>
        <div class="content">
          <div class="place">
            <span class="fas fa-map-marker-alt"></span>
            <span class="text">Galgotias University , U.P</span>
          </div>

          <div class="email">
            <span class="fas fa-envelope"></span>
            <span class="text">vaibhavkumarvky@gmail.com</span>
            <br>
            <span class="fas fa-envelope"></span>
            <span class="text">shivamsingh199907@gmail.com</span>
          </div>
        </div>
      </div>
      <div class="right box">
        <h2>Contact us</h2>
        <div class="content">
          <form action="#">
            <div class="email">
              <div class="text">Email *</div>
              <input type="email" required>
            </div>
            <div class="msg">
              <div class="text">Message *</div>
              <textarea rows="2" cols="25" required></textarea>
```

```
            </div>
            <div class="btn">
              <button type="submit">Send</button>
            </div>
          </form>
        </div>
      </div>
    </div>
    <div class="bottom">
      <center>
        <span class="credit">Created By VAIBHAB & SHIVAM</a></span>
      </center>
    </div>
  </footer>




            <BR>
        </body>
</html>


<script type="text/javascript">
        var mode=null;
        var objFile=null;
        switchdiv('encrypt');

        function switchdiv(t) {
                if(t=='encrypt') {
                        divEncryptfile.style.display='block';
                        divDecryptfile.style.display='none';
                        btnDivEncrypt.disabled=true;
                        btnDivDecrypt.disabled=false;
                        mode='encrypt';
                } else if(t=='decrypt') {
                        divEncryptfile.style.display='none';
                        divDecryptfile.style.display='block';
```

```
                btnDivEncrypt.disabled=false;
                btnDivDecrypt.disabled=true;
                mode='decrypt';
            }
        }

        function encvalidate() {
            if(txtEncpassphrase.value.length>=8 &&
txtEncpassphrase.value==txtEncpassphraseretype.value) {
              spnCheckretype.classList.add("greenspan");
              spnCheckretype.classList.remove("redspan");
              spnCheckretype.innerHTML='&#10004;';
            } else {
              spnCheckretype.classList.remove("greenspan");
              spnCheckretype.classList.add("redspan");
              spnCheckretype.innerHTML='&#10006;';
            }

            if( txtEncpassphrase.value.length>=8 &&
txtEncpassphrase.value==txtEncpassphraseretype.value && objFile )
{ btnEncrypt.disabled=false; } else { btnEncrypt.disabled=true; }
        }

        function decvalidate() {
            if( txtDecpassphrase.value.length>0 && objFile )
{ btnDecrypt.disabled=false; } else { btnDecrypt.disabled=true; }
        }

        //drag and drop functions:
        //https://developer.mozilla.org/en-
US/docs/Web/API/HTML_Drag_and_Drop_API/File_drag_and_drop
        function drop_handler(ev) {
            console.log("Drop");
            ev.preventDefault();
            // If dropped items aren't files, reject them
            var dt = ev.dataTransfer;
            if (dt.items) {
                // Use DataTransferItemList interface to access the file(s)
```

```
                for (var i=0; i < dt.items.length; i++) {
                        if (dt.items[i].kind == "file") {
                                var f = dt.items[i].getAsFile();
                                console.log("... file[" + i + "].name = " + f.name);
                                objFile=f;
                        }
                }
        } else {
                // Use DataTransfer interface to access the file(s)
                for (var i=0; i < dt.files.length; i++) {
                        console.log("... file[" + i + "].name = " +
dt.files[i].name);
                }
                objFile=file[0];
        }
        displayfile()
        if(mode=='encrypt') { encvalidate(); } else if(mode=='decrypt')
{ decvalidate(); }
    }

    function dragover_handler(ev) {
        console.log("dragOver");
        // Prevent default select and drag behavior
        ev.preventDefault();
    }

    function dragend_handler(ev) {
        console.log("dragEnd");
        // Remove all of the drag data
        var dt = ev.dataTransfer;
        if (dt.items) {
                // Use DataTransferItemList interface to remove the drag
data
                for (var i = 0; i < dt.items.length; i++) {
                        dt.items.remove(i);
                }
        } else {
                // Use DataTransfer interface to remove the drag data
```

```
            ev.dataTransfer.clearData();
        }
    }

    function selectfile(Files) {
        objFile=Files[0];
        displayfile()
        if(mode=='encrypt') { encvalidate(); } else if(mode=='decrypt')
{ decvalidate(); }
    }

    function displayfile() {
        var s;
        var sizes = ['Bytes', 'KB', 'MB', 'GB', 'TB'];
        var bytes=objFile.size;
        var i = parseInt(Math.floor(Math.log(bytes) / Math.log(1024)));
        if(i==0) { s=bytes + ' ' + sizes[i]; } else { s=(bytes /
Math.pow(1024, i)).toFixed(2) + ' ' + sizes[i]; }

        if(mode=='encrypt') {
            spnencfilename.textContent=objFile.name + ' (' + s + ')';
        } else if(mode=='decrypt') {
            spndecfilename.textContent=objFile.name + ' (' + s + ')';
        }
    }

    function readfile(file){
        return new Promise((resolve, reject) => {
            var fr = new FileReader();
            fr.onload = () => {
                resolve(fr.result )
            };
            fr.readAsArrayBuffer(file);
        });
    }

    async function encryptfile() {
        btnEncrypt.disabled=true;
```

```
		var plaintextbytes=await readfile(objFile)
		.catch(function(err){
			console.error(err);
		});
		var plaintextbytes=new Uint8Array(plaintextbytes);

		var pbkdf2iterations=10000;
		var passphrasebytes=new TextEncoder("utf-
8").encode(txtEncpassphrase.value);
		var pbkdf2salt=window.crypto.getRandomValues(new
Uint8Array(8));

		var passphrasekey=await window.crypto.subtle.importKey('raw',
passphrasebytes, {name: 'PBKDF2'}, false, ['deriveBits'])
		.catch(function(err){
			console.error(err);
		});
		console.log('passphrasekey imported');

		var pbkdf2bytes=await window.crypto.subtle.deriveBits({"name":
'PBKDF2', "salt": pbkdf2salt, "iterations": pbkdf2iterations, "hash": 'SHA-
256'}, passphrasekey, 384)
		.catch(function(err){
			console.error(err);
		});
		console.log('pbkdf2bytes derived');
		pbkdf2bytes=new Uint8Array(pbkdf2bytes);

		keybytes=pbkdf2bytes.slice(0,32);
		ivbytes=pbkdf2bytes.slice(32);

		var key=await window.crypto.subtle.importKey('raw', keybytes,
{name: 'AES-CBC', length: 256}, false, ['encrypt'])
		.catch(function(err){
			console.error(err);
		});
		console.log('key imported');
```

```
            var cipherbytes=await window.crypto.subtle.encrypt({name:
"AES-CBC", iv: ivbytes}, key, plaintextbytes)
            .catch(function(err){
                  console.error(err);
            });

            if(!cipherbytes) {
                  spnEncstatus.classList.add("redspan");
                  spnEncstatus.innerHTML='<p>Error encrypting file.  See
console log.</p>';
                  return;
            }

            console.log('plaintext encrypted');
            cipherbytes=new Uint8Array(cipherbytes);

            var resultbytes=new Uint8Array(cipherbytes.length+16)
            resultbytes.set(new TextEncoder("utf-8").encode('Salted__'));
            resultbytes.set(pbkdf2salt, 8);
            resultbytes.set(cipherbytes, 16);

            var blob=new Blob([resultbytes], {type: 'application/download'});
            var blobUrl=URL.createObjectURL(blob);
            aEncsavefile.href=blobUrl;
            aEncsavefile.download=objFile.name + '.enc';

            spnEncstatus.classList.add("greenspan");
            spnEncstatus.innerHTML='<p>File encrypted.</p>';
            aEncsavefile.hidden=false;
      }

      async function decryptfile() {
            btnDecrypt.disabled=true;

            var cipherbytes=await readfile(objFile)
            .catch(function(err){
                  console.error(err);
```

```
        });
        var cipherbytes=new Uint8Array(cipherbytes);

        var pbkdf2iterations=10000;
        var passphrasebytes=new TextEncoder("utf-
8").encode(txtDecpassphrase.value);
        var pbkdf2salt=cipherbytes.slice(8,16);


        var passphrasekey=await window.crypto.subtle.importKey('raw',
passphrasebytes, {name: 'PBKDF2'}, false, ['deriveBits'])
        .catch(function(err){
            console.error(err);

        });
        console.log('passphrasekey imported');

        var pbkdf2bytes=await window.crypto.subtle.deriveBits({"name":
'PBKDF2', "salt": pbkdf2salt, "iterations": pbkdf2iterations, "hash": 'SHA-
256'}, passphrasekey, 384)
        .catch(function(err){
            console.error(err);
        });
        console.log('pbkdf2bytes derived');
        pbkdf2bytes=new Uint8Array(pbkdf2bytes);

        keybytes=pbkdf2bytes.slice(0,32);
        ivbytes=pbkdf2bytes.slice(32);
        cipherbytes=cipherbytes.slice(16);

        var key=await window.crypto.subtle.importKey('raw', keybytes,
{name: 'AES-CBC', length: 256}, false, ['decrypt'])
        .catch(function(err){
            console.error(err);
        });
        console.log('key imported');

        var plaintextbytes=await window.crypto.subtle.decrypt({name:
```

```
"AES-CBC", iv: ivbytes}, key, cipherbytes)
            .catch(function(err){
                console.error(err);
            });

            if(!plaintextbytes) {
                spnDecstatus.classList.add("redspan");
                spnDecstatus.innerHTML='<p>Error decrypting file.
Password may be incorrect.</p>';
                return;
            }

            console.log('ciphertext decrypted');
            plaintextbytes=new Uint8Array(plaintextbytes);

            var blob=new Blob([plaintextbytes], {type:
'application/download'});
            var blobUrl=URL.createObjectURL(blob);
            aDecsavefile.href=blobUrl;
            aDecsavefile.download=objFile.name + '.dec';

            spnDecstatus.classList.add("greenspan");
            spnDecstatus.innerHTML='<p>File decrypted.</p>';
            aDecsavefile.hidden=false;
        }

</script>
```

CSS CODE :

```
@import
url('https://fonts.googleapis.com/css?family=Poppins:400,500,600,700&displ
ay=swap');

*{

  font-family: 'Poppins', sans-serif;
}

.content1{
  font-size: 2.5rem;
  font-weight: 600;
  color: red;
}
.content1 .p{
  font-size: 2.1875rem;
  font-weight: 600;
  color: red;
}
footer{
  position: static ;
  bottom: 0px;
  width: 100%;
  background: #1111;
}
.main-content{
  display: flex;
}
.main-content .box{
  flex-basis: 50%;
  padding: 10px 20px;
}
.box h2{
  font-size: 1.125rem;
  font-weight: 600;
```

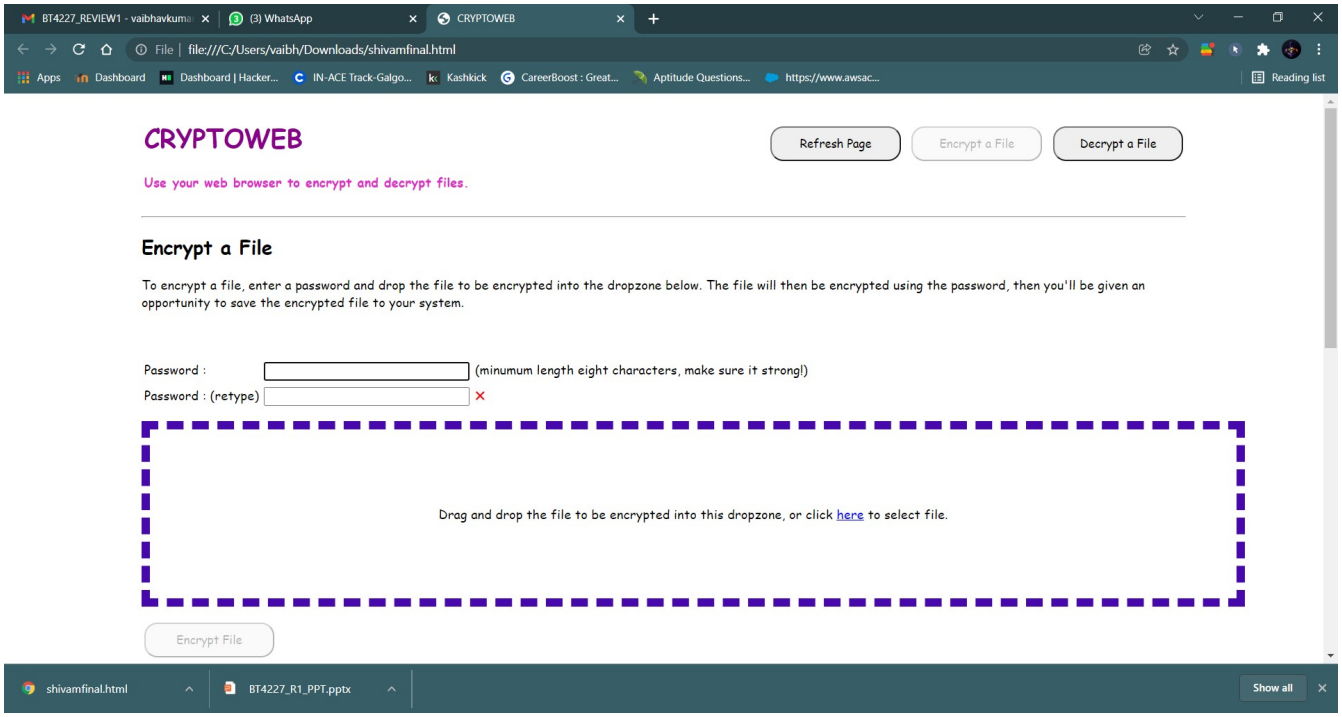
```
  text-transform: uppercase;
}
.box .content{
  margin: 20px 0 0 0;
  position: relative;
}
.box .content:before{
  position: absolute;
  content: '';
  top: -10px;
  height: 2px;
  width: 100%;
  background: blue;
}
.box .content:after{
  position: absolute;
  content: '';
  height: 2px;
  width: 15%;
  background: #f12020;
  top: -10px;
}
.left .content p{
  text-align: justify;
}
.left .content .social{
  margin: 20px 0 0 0;
}
.left .content .social a{
  padding: 0 2px;
}
.left .content .social a span{
  height: 40px;
  width: 40px;
  background: #1a1a;
  line-height: 40px;
  text-align: center;
  font-size: 18px;
```

```
  border-radius: 5px;
  transition: 0.3s;
}
.left .content .social a span:hover{
  background: #f1200;
}
.center .content .fas{
  font-size: 1.4375rem;
  background: #1a1a;
  height: 45px;
  width: 45px;
  line-height: 45px;
  text-align: center;
  border-radius: 50%;
  transition: 0.3s;
  cursor: pointer;
}
.center .content .fas:hover{
  background: #f12020;
}
.center .content .text{
  font-size: 1.0625rem;
  font-weight: 500;
  padding-left: 10px;
}
.center .content .phone{
  margin: 15px 0;
}
.right form .text{
  font-size: 1.0625rem;
  margin-bottom: 2px;
  color: #656565;
}
.right form .msg{
  margin-top: 10px;
}
.right form input, .right form textarea{
  width: 100%;
```

```
  font-size: 1.0625rem;
  background: white;
  padding-left: 10px;
  border: 1px solid green;
}
.right form input:focus,
.right form textarea:focus{
  outline-color: #3498db;
}
.right form input{
  height: 35px;
}
.right form .btn{
  margin-top: 10px;
}
.right form .btn button{
  height: 40px;
  width: 100%;
  border: none;
  outline: none;
  background: #f12020;
  font-size: 1.0625rem;
  font-weight: 500;
  cursor: pointer;
  transition: .3s;
}
.right form .btn button:hover{
  background: #000;
}
.bottom center{
  padding: 5px;
  font-size: 0.9375rem;
  background: #151515;
}
.bottom center span{
  color: #25d9bb;
}
.bottom center a{
```

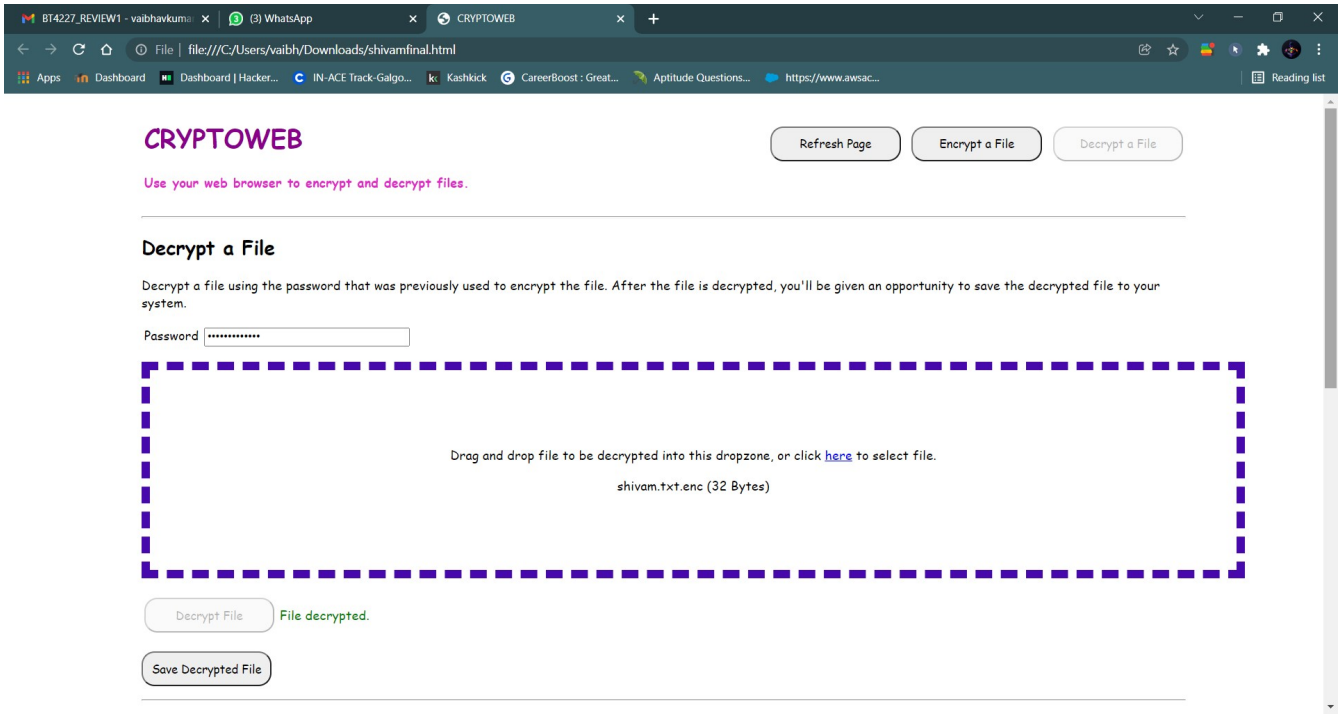
```
  color: #f12020;
  text-decoration: none;
}
.bottom center a:hover{
  text-decoration: underline;
}
@media screen and (max-width: 900px) {
  footer{
    position: relative;
    bottom: 0px;
  }
  .main-content{
    flex-wrap: wrap;
    flex-direction: column;
  }
  .main-content .box{
    margin: 5px 0;
  }
```

IMPLEMENTATION



As the above picture shows , the interface and UI of the web application is simple and easy to interact . The buttons are easy to read and interact with and the users can easily navigate across the application .

Firstly we need to upload file on the website either by clicking the “here” button or by simply drag and dropping . Then we need to encrypt the file using password . Then we need to encrypt the file by clicking the “Encrypt file” button , this will simply encrypt the file and the file can be downloaded using the download button .



This above picture shows how the website works and how it can be downloaded .

Decrypting a file as simple as the encrypting was . We just have to upload the file and enter the file . This will decrypt the file and and we can download the file .

6. REFERENCES

[1] H.Abdulzahra, R. AHMAD, and N. M. NOOR, “Security enhancement; Combining cryptograhy and steganography for data hiding in images,” ACACOS, Applied Computational Science,pp.978-960,2014.

[2] P. R. Ekatpure and R. N.Benkar, “A comparative study of steganography & cryptography,”2013.

[3] M. H. Rajyaguru, “Cryptography-combination of cryptography and steganography with rapidly changing keys ,”Interntional Journal of Emerging Technology and Advanced Engineering,ISSN ,pp.2250-2459,2012.

[4] D. Seth. L. Ramanathan, and A.Pandey, “Security enhancement; Combining cryptography and steganography,” International Journal of Computer Applications(0975-8887)Volume,2010.

[5] J. V. Karthik and B. V. Reddy, “Authentication of secret information in image steganography,” International Journal of Computer Science and Network Security(IJCSNS), vol. 14, no. 6. P. 58. 2014.