

# **"Cryptography, Steganography and their Application Overview"**

## **MINI PROJECT REPORT of 18CSE381T – Cryptography**

*Submitted by*

**Anish Bharath (RA2111030010092)  
Anamika Jain (RA2111030010098)  
Pawan Siddh (RA2111030010109)  
Pulkit Khanna (RA2111030010113)  
Kartik Jindal (RA2111030010124)**

*Under the Guidance of*

**Dr. A. PRABHU CHAKKARAVARTHY**  
Assistant Professor, NWC

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

**BACHELOR OF TECHNOLOGY  
in  
COMPUTER SCIENCE ENGINEERING  
with specialization in Cyber Security**



**DEPARTMENT OF NETWORKING AND COMMUNICATIONS  
COLLEGE OF ENGINEERING AND TECHNOLOGY  
SRM INSTITUTE OF SCIENCE AND TECHNOLOGY  
KATTANKULATHUR- 603 203**

**NOVEMBER 2023**



**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**  
**KATTANKULATHUR – 603 203**

**BONAFIDE**

This is to certify that **18CSE381T – Cryptography**, Mini Project titled "**Cryptography, Steganography and their Application Overview**" is the bonafide work of **Anish Bharath (RA2111030010092)**, **Anamika Jain (RA2111030010098)**, **Pawan Siddh (RA2111030010109)**, **Pulkit Khanna(RA2111030010113)**, **Kartik Jindal(RA2111030010124)** who undertook the task of completing the project within the allotted time.

**SIGNATURE**

**Dr. A. PRABHU**  
**CHAKKARAVARTHY**  
Assistant Professor  
Department of Networking and  
Communications  
SRM INSTITUTE OF SCIENCE AND  
TECHNOLOGY

**SIGNATURE**

**Dr. ANNAPURANI. K**  
Professor and Head  
Department of Networking and  
Communications  
SRM INSTITUTE OF SCIENCE AND  
TECHNOLOGY

## **TABLE OF CONTENTS**

<b>Chapter No.</b>	<b>Title</b>	<b>Page No.</b>
1.	Abstract	1
2.	Literature Review	2
3.	Algorithm Used	5
4.	Architecture Diagram	8
5.	Result and Discussion	22
6.	Conclusion	26
7.	References	27

## Abstract:

Online transactions are now a common occurrence in our lives. Given how frequently we all use online transactions today, it is crucial to address relevant security issues including authorization, data protection, and confidentiality. In this study, we'll examine steganography and cryptography and how they're applied to protecting cloud-based web transactions. Steganography conceals the existence of a message by encoding it in another digital medium, such as an image or an audio file, while cryptography transforms the data into cipher text that is often unreadable to the average user. This report aims to provide a general review of steganography, cryptography, its applications, and methods for conducting online transactions in a secure manner.

# Literature Review

## Introduction

Cryptography is the protecting technique of data from the unauthorized party by converting into the non-readable form. The main purpose of cryptography is maintaining the security of the data from third party.

Steganography is a Greek word which means concealed writing. The word "steganos" means "covered" and "graphical" means "writing". Thus, steganography is not only the art of hiding data but also hiding the fact of transmission of secret data. Steganography hides the secret data in another file in such a way that only the recipient knows the existence of the message.

With the growing advancement of technology in the current world and need of data security growing similarly, the application area of cryptography and steganography is vast. In this report, we have tried to apply cryptographic and steganographic techniques to secure transaction that happens over the internet.

## Literature survey:

In Choudhari Amar, Shaikh Sultanhusen, Ghadge Vashista, Prof. Sonawane V. D, Prof. Naved Raza, A Review - Transaction Security Using Steganography and Visual Cryptography, Online shopping is the retrieval of product information via the Internet and issue of purchase order through electronic purchase request, billing of credit or debit card information and shipping of product by mail order or home delivery by courier. Identity theft and phishing are the common dangers of online shopping. Identity theft is the stealing of someone's identity in the form of personal information and misusing that information for making purchase and opening of bank accounts or arranging credit cards.

In Shailendra M. Pardeshi, a Study on Combine use of Steganography and Cryptography for Data Hiding, The simplest approach to hiding data within an image file is called least significant bit (LSB) insertion. In this method, we can take the binary representation of the hidden data and overwrite the LSB of each byte within the cover image. If we are using 24-bit color, the amount of change will be minimal and indiscernible to the human eye.

## Problem Statement:

Cloud service companies' storage of customer data is vulnerable to many threats. One of them is single point of failure, which will affect the availability of data and make it more challenging for the customer to get his stored data from the server in the event that a cloud service provider server fails or crashes. Data availability is a significant issue that might be compromised if the cloud service provider (CSP) goes out of business.

The data integrity risk is the second one. Integrity is the level of assurance that the data stored in the cloud is accurate and secure against unauthorized inadvertent or malicious alterations. A cloud service customer cannot totally rely on a cloud service provider to assure the storage of his important data because such concerns are no longer helpful issues.

Additionally, typical transaction systems do not have a phishing requirement, which can be damaging and encourage the use of social engineering and technical deception.

## Algorithm Used:

In this demonstration, we have used Playfair cipher to encrypt the data. But that can be changed as per the requirements. A more complex algorithm can be used to ensure even better security.

The **Playfair cipher** or **Playfair square** or **Wheatstone–Playfair cipher** is a manual symmetric encryption technique and was the first literal digram substitution cipher. The scheme was invented in 1854 Charles Wheatstone, but bears the name of Lord Playfair for promoting its use.

The technique encrypts pairs of letters (*bigrams* or *digrams*), instead of single letters as in the simple substitution cipher and rather more complex Vignere cipher systems then in use. The Playfair is thus significantly harder to break since the frequency analysis used for simple substitution ciphers does not work with it. The frequency analysis of bigrams is possible, but considerably more difficult. With 600 possible bigrams rather than the 26 possible monograms (single symbols, usually letters in this context), a considerably larger cipher text is required in order to be useful.

Using "playfair example" as the key (assuming that I and J are interchangeable), the table becomes (omitted letters in red):

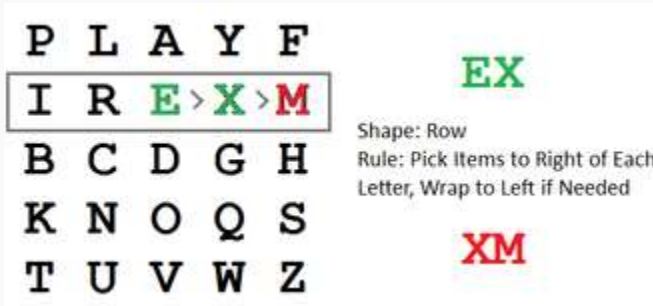
P	L	A	Y	F <sup>A</sup>
I	R	E	X <sup>A</sup>	M <sup>PLE A</sup>
B	C	D <sup>EF</sup>	G	H <sup>I=J</sup>
K <sup>LM</sup>	N	O <sup>P</sup>	Q <sup>R</sup>	S
T	U	V	W <sup>XY</sup>	Z

The first step of encrypting the message "hide the gold in the tree stump" is to convert it to the pairs of letters "HI DE TH EG OL DI NT HE TR EX ES TU MP" (with the null "X" used to separate the repeated "E"s). Then:

1. The pair HI forms a rectangle, replace it with BM	<table><tbody><tr><td>P</td><td>L</td><td>A</td><td>Y</td><td>F</td></tr><tr><td>I</td><td>R</td><td>E</td><td>X</td><td>M</td></tr><tr><td>B</td><td>C</td><td>D</td><td>G</td><td>H</td></tr><tr><td>K</td><td>N</td><td>O</td><td>Q</td><td>S</td></tr><tr><td>T</td><td>U</td><td>V</td><td>W</td><td>Z</td></tr></tbody></table> <div>Shape: Rectangle Rule: Pick Same Rows, Opposite Corners</div> <div>HI</div> <div>BM</div>	P	L	A	Y	F	I	R	E	X	M	B	C	D	G	H	K	N	O	Q	S	T	U	V	W	Z
P	L	A	Y	F																						
I	R	E	X	M																						
B	C	D	G	H																						
K	N	O	Q	S																						
T	U	V	W	Z																						



<p>2. The pair DE is in a column, replace it with OD</p>	<div data-bbox="824 247 1122 541"> <p>P L A Y F</p> <p>I R E X M</p> <p>B C D G H</p> <p>K N O Q S</p> <p>T U V W Z</p> </div> <div data-bbox="1256 268 1321 310">DE</div> <div data-bbox="1175 352 1446 436"> <p>Shape: Column</p> <p>Rule: Pick Items Below Each Letter, Wrap to Top if Needed</p> </div> <div data-bbox="1256 468 1321 510">OD</div>
<p>3. The pair TH forms a rectangle, replace it with ZB</p>	<div data-bbox="824 636 1122 930"> <p>P L A Y F</p> <p>I R E X M</p> <p><del>B C D G H</del></p> <p>K N O Q S</p> <p><del>T U V W Z</del></p> </div> <div data-bbox="1256 667 1321 709">TH</div> <div data-bbox="1198 741 1404 825"> <p>Shape: Rectangle</p> <p>Rule: Pick Same Rows, Opposite Corners</p> </div> <div data-bbox="1256 856 1321 898">ZB</div>
<p>4. The pair EG forms a rectangle, replace it with XD</p>	<div data-bbox="824 1018 1122 1312"> <p>P L A Y F</p> <p>I R <del>E X</del> M</p> <p>B C <del>D G</del> H</p> <p>K N O Q S</p> <p>T U V W Z</p> </div> <div data-bbox="1256 1045 1321 1087">EG</div> <div data-bbox="1198 1119 1404 1203"> <p>Shape: Rectangle</p> <p>Rule: Pick Same Rows, Opposite Corners</p> </div> <div data-bbox="1256 1234 1321 1276">XD</div>
<p>5. The pair OL forms a rectangle, replace it with NA</p>	<div data-bbox="824 1400 1122 1694"> <p>P <del>L A</del> Y F</p> <p>I R E X M</p> <p>B C D G H</p> <p>K <del>N O</del> Q S</p> <p>T U V W Z</p> </div> <div data-bbox="1250 1434 1312 1476">OL</div> <div data-bbox="1198 1507 1404 1591"> <p>Shape: Rectangle</p> <p>Rule: Pick Same Rows, Opposite Corners</p> </div> <div data-bbox="1250 1623 1312 1665">NA</div>
<p>6. The pair DI forms a rectangle, replace it with BE</p>	

7. The pair NT forms a rectangle, replace it with KU	
8. The pair HE forms a rectangle, replace it with DM	
9. The pair TR forms a rectangle, replace it with UI	
10. The pair EX (X inserted to split EE) is in a row, replace it with XM	
11. The pair ES forms a rectangle, replace it with MO	
12. The pair TU is in a row, replace it with UV	
13. The pair MP forms a rectangle, replace it with IF	

Thus the message "hide the gold in the tree stump" becomes "BM OD ZB XD NA BE KU DM UI XM MO UV IF", which may be restructured as "BMODZ BXDNA BEKUD MUIXM MOUVI F" for ease of reading the cipher text.

## Architecture Diagram/ Framework

### Symmetric Key Cryptography:

Symmetric-key cryptography refers to encryption methods in which both the sender and receiver share the same key (or, less commonly, in which their keys are different, but related in an easily computable way). This was the only kind of encryption publicly known until June 1976. Symmetric key ciphers are implemented as either block ciphers or stream ciphers. Symmetric key cryptography schemes are usually categorized such as stream ciphers or block ciphers. Stream ciphers work on a single bit (byte or computer word) at a time and execute some form of feedback structure so that the key is repeatedly changing. Figure 5.1 below shows the block diagram for symmetric encryption.

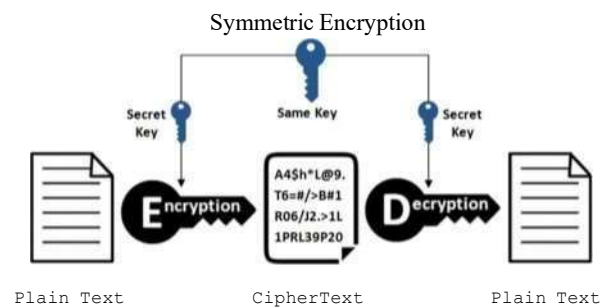


Figure 5.1 : Symmetric Encryption Block diagram

## Asymmetric Key Cryptography:

Asymmetric cryptography, also known as public-key cryptography, is a process that uses a pair of related keys one public key and one private key — to encrypt and decrypt a message and protect it from unauthorized access or use. When someone wants to send an encrypted message, they can pull the intended recipient's public key from a public directory and use it to encrypt the message before sending it. The recipient of the message can then decrypt the message using their related private key. If the sender encrypts the message using their private key, the message can be decrypted only using that sender's public key, thus authenticating the sender. These encryption and decryption processes happen automatically; users do not need to physically lock and unlock the message. Many protocols rely on asymmetric cryptography, including the transport layer security (TLS) and secure sockets layer (SSL) protocols, which make HTTPS possible. Figure 5.2 below shows the block diagram of asymmetric key cryptography.

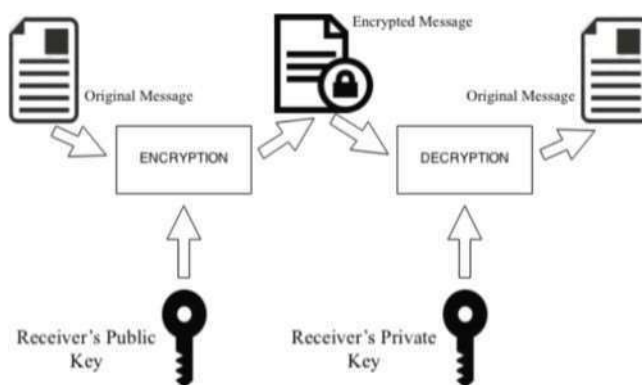


Figure 5.2 : Asymmetric Encryption Block diagram

## Implementation

Either of the above discussed frameworks. i.e Asymmetric and Symmetric key cryptography can be used to secure the transaction over the internet. The main concern here is not the cryptography that is used but instead how that can happen in secure and effective manner.

The amount of information a client submits to an online retailer is limited by asking for only the information necessary to verify that the customer used its bank account to make the payment. By establishing a central Certified Authority (CA) and combining the use of steganography and cryptography, this is accomplished.

Account numbers linked to the shopping card may be among the pieces of information that the retailer receives. The data will only verify that genuine customers made the payment. The proposed solution uses text-based steganography to disguise the customer's unique authentication password in contact with the bank. In its original format, the customer authentication information (account no.) in relation to the merchant is positioned above the cover text. Now a snapshot of two texts is taken. From the snapshot image, two shares are generated using visual cryptography.

After this, Users can post their own merchant's account information and shares on the website. The CA now shares the original image after retrieving it and combining it with the buyers' shares. The bank now receives the merchant account information from CA along with the cover text, which contains the client authentication password. identify of the client. After the bank confirms the client is a real one and the CA has provided the merchant with the password for customer authentication, the bank transfers funds from the customer account to the provided merchant account. Utilizing client authentication information, the merchant's payment system verifies receipt of payment after receiving the funds.

To demonstrate this, we have used AWS cloud service to create two sockets, one to represent the customer and other to represent the CA.

The CA generates a unique key and sends it to the customer, which they have to use before making any transaction. Once the request is made, two snapshots are generated. One at the user side and other at server side. The encrypted text is sent to CA using text steganography to the CA. The CA then compares if the cipher text generated is same at both ends or not. If both of them are same then, it is made by authenticated user and hence is allowed. If they are not same, then there has been a breach or unauthorized user is trying to make request. And hence is rejected.

## CA.py

```
import
socket
import
datetime
import
random
s=socket.
socket()

for
s.bind(('127.0.0.1',3600))
key=''
a=['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o',
'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z']
for i in range(random.randint(7,10)):
    key+=random.
choice (a) def
toLowerCase(text) :
return text.
lower() def
removespaces(text ):
newText = "for i in
text:
    if ' ':
        continue
    else:
        newText = newText +
i return newText def
Diagram(text ):Diagram = []
group = 0 for i in range(2,
len(text), 2) :
    Diagram . append (text [group : i] )
    group+= 1
    Diagram . append (text[group
: ] )
return Diagram
def ):
FillerLetter(text
k
= len(text) if k
% 2 == 0 for i in
range(0, k, 2)
```

```

-- --
if text[i] text[i+1] :
    new word = text[e:i+1] + str( new
word
=
- FillerLetter(new_word) break else:
    new word = text

for i

else:
    in

== text[i+
text[i+
d = text[0:i+1] + str('x')
d = FillerLetter(new_word)
range(e, k-1, 2) if text[i]
new word = + text[i+1: ]
    new
word
break
else:
    new word =

= ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'k', 'l', 'm',
'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z']
generateKeyTable(word, list1):
y_letters = []
text return new word list1

```

```

key_letters
for i in
word:
    if i not in key_letters:
        key_letters . append( i)
compElements = []
for i in
key_letters:
    if i not in compElements:
        compElements . append
(i) for i in list1:
    if i not in compElements:
        compElements . append
(i)
matrxx = while compElements
.ements != []:
    ppend(compElements[

```



```

matrix . : 5] ) compElements =
compElements[5: ]
return matrix
def search(mat,
element): for i
in range(S) :
for j in
range(S) :

+
Text[i+1]:
]

    if(mat[i][j] ==
        return i, j element) :
def encrypt_RowRule(matr, elr, elc,
e2r, e2c) : char1 = if elc == 4 :
char1 = matr[e1r][0] else:
char1 =
=
matr[e1r]
[e1c+1]

char2 =
' if e2c
==4:
char2 = matr[e2r][0]
else:
char2= matr[e2r][e2c+1]

return char1, cha4r2 def
encrypt_ColumnRule(matr, elr, elc, e2r,
e2c) : char1 = , if elr == 4: char1 =
matr[0][elc] else:
char1
= matr[e1r+1][e1c]

char2 = , if e2r
== 4: char2 = matr[0]
[e2c] else:
char2 = matr[e2r+1][e2c]

```

```

        return char1, char2
    def encrypt_RectangleRule(matr, elr, elc, e2r, e2c):
        char1 = matr[elr][e2c]
        char2 = matr[e2r][elc]
        return char1, char2

def encryptByPlayfairCipher(Matrix, plainList):
    CipherText = []
    for i in range(len(plainList)):
        c1 = e
        c2 = 0
        ele1_x, ele1_y = search(Matrix, plainList[i][1])
        ele2_x, ele2_y = search(Matrix, plainList[i][1])
        if ele1_x == ele2_x:
            c1, c2 = encrypt_RowRule(Matrix, ele1_x, ele1_y, ele2_x, ele2_y)
            # Get 2 letter cipher Text
        elif ele1_y == ele2_y:
            c1, c2 = encrypt_ColumnRule(Matrix, ele1_x, ele1_y, ele2_x, ele2_y)
        else:
            c1, c2 = encrypt_RectangleRule(Matrix, ele1_x, ele1_y, ele2_x, ele2_y)

        cipher = c1 + c2
        CipherText.append(cipher)
    return CipherText

```

```

text Plain = "hello"
text
PlainTextLi

if

key = toLowerCase(key)
Matrix = generateKeyTable(key, list1)
CipherList =
encryptByPlayfairCipher(Matrix,
PlainTextList) CipherText = for i in
CipherList:
    CipherText i
print(key)
s. listen(S)
print("Server is up and
running ...")while True:

    c, addr=s . accept()
    print("Connected to " ,
    addr)
    c. send( key .
    encode()) msg1=c .
    recv(1024) .decode(
    ) print(msg1)
    msg=msg1.
    split( ' , ')[1]#
    print() if
    (msg==CipherText) :
        print("Acc
        ep ted ")
    else : print(
    "Rejected" )
s. close()
User.py
import
socket
c=socket.
socket()
nect(('127.e.e.1

```

```

' , 36ee) ) key1=c .
recv(2048) .
decode( ) print("The
key is key1) key=
input( "Enter the key
")def
toLowerCase(text) :
return text. lower()
def removeSpaces(text
):newText = for i in
text:
    if i " ":
        continue
    else:
        newText = newText +
i return newText def
Diagraph(text) Diagraph = []
group for i in range(2,
len(text), 2) :Diagraph.
append (text [group : i] )
        group = i

Diagraph. append
(text[group : ] ) return
Diagraph def
FillerLetter(text k =
len(text) if k % 2 e :
for i in range(e, k, 2):if
text[i] text[i+1] :

    new word =
text[e:i+1] + str(
        ): +
text[i+1: ] new :
word -Diagraph. append
(text[group : ] ) return
Diagraph def
FillerLetter(text k =
len(text) if k % 2 e
for i in range(e, k, if
text[i] text[i+1] :

new word = text[e:i+1] + str( + text[i+1: ]
new word -

```

```

        FillerLetter(new_word)
    breakelse:

        new word -
text else :
    for i in
        range(e, k-1
            2) .if text[i]
            text[i+1] :

                new word - text[e:i+1] +
                str( 'x' ) + text[i+1: ] new
                word
                FillerLetter(new_word) break
            else :

                new word - text

return new word list1
= ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'k', 'l', 'm',
  'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z']

def generateKeyTable(word,
list1) :key_letters = []
for i in word:
    if i not in key_letters:
        key_letters.append( i)
    compElements = []
    for i in
        key_letters:
            if i not in compElements:
                compElements.
                append( i) for i in list1:
                    if i not in compElements:

                        compElements.append( i)
matrix =while compElements !
- matrix. : 5] )
compElements =
compElements[5: ]

return
matrix def
search(mat,
element for

```

```

i in range(S)
:
    for j in range(S) :
        if(mat[i][j] == element) :
            return i, j
def encrypt_RowRule(matr,
elr, elc, e2r, e2c) :
    char1 = ''
    if elc
= 4: char1 = matr[e1r][0]
else:
    char1
        =
    matr[elr]
    [elc+1]
char2 =
if e2c = 4:
    char2 = matr[e2r][0]
else :
    char2 = matr[e2r][e2c+1]

return char1, char2
def
encrypt_ColumnRule(matr, elr, elc, e2r,
e2c) :
    char1 = ''
    if elr ==4 :
        char1 = matr[e]
        [elc]
    else:
        char1
            =
        matr[elr+1]
        [elc]

char2 = ''
if e2r
==4 char2 =
matr[e] [e2c]
else:
    char2= matr[e2r+1][e2c]

```

```

return
char1, char2
def elr, elc, e2r, e2c) :char1 =
char1= matr[e1r][e2c]

char2 = ' ' char2
= matr[e2r][e1c]

return char1, char2
def
encryptByPlayfairCipher(Matrx, plainList
ix, plainList
CipherText for
iin range(e, len(plainList)
) :
    cl = e
    e1_x, ele1_y =
    plainList[i][0]e2_x, search(Matrx, ele2_y
    =
    search(Matrx, plainList[i][1])
    i f ele1 cl, c 2 = encrypt_RowRule(Matrx,
    ele1_x, ele1_y, ele2_x, ele2_y)
    # Get 2 letter cipherText
    elif ele1_y e le2_y :
    cl, c2 = encrypt_ColumnRule(Matrx, ele
    _x, ele1_y, ele2_x, ele2_y) else:
    cl, enc rypt_RectangleRule (
    Matrx, ele1_x, ele1_y, ele2_x, ele2
    cipher =
    CipherText c1 + . append(cipher)
return CipherText text Plain
input( "Enter the text ") text Plain
-

1]) != 2:
    = PlainTextList[-1]+'z'
removeSpaces(toLowerCase(text_Plain)
) PlainTextList =
Diagraph(FillerLetter(text_Plain))

```

```
if len(PlainTextList[-1] )  
    PlainTextList[-1]
```



## Results and Discussion

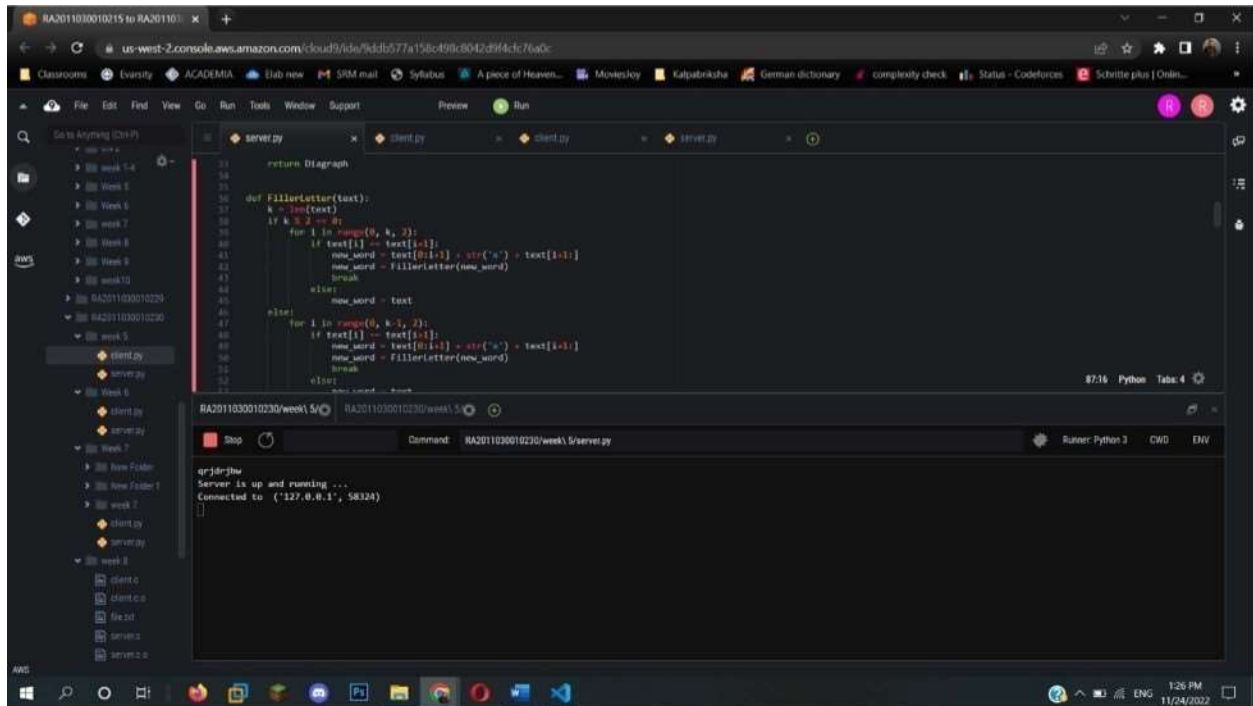


Figure 7. 1

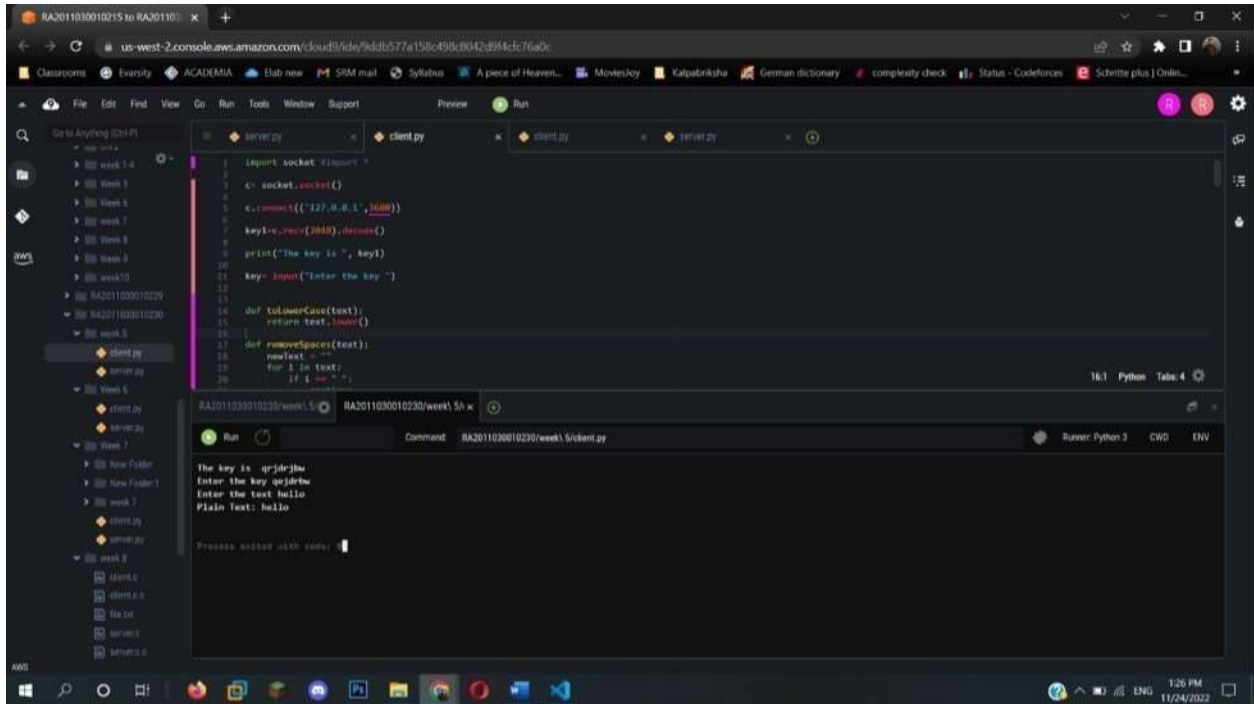


Figure 7. 2

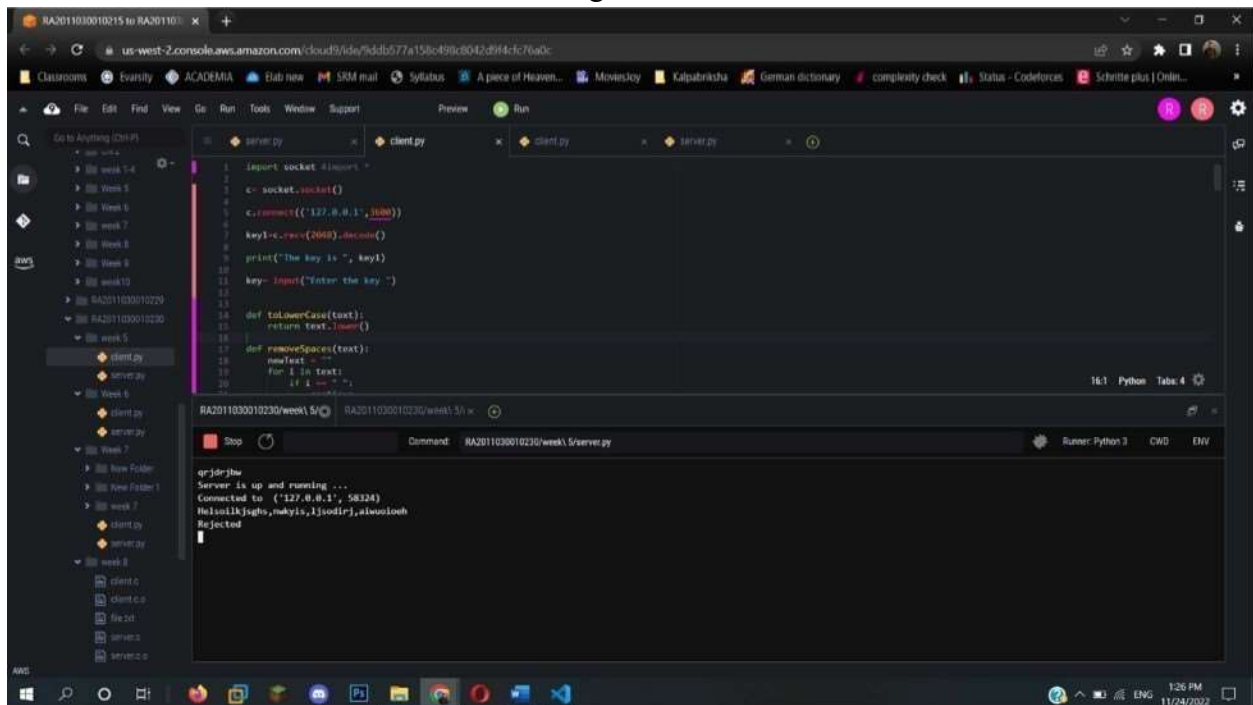


Figure 7. 3

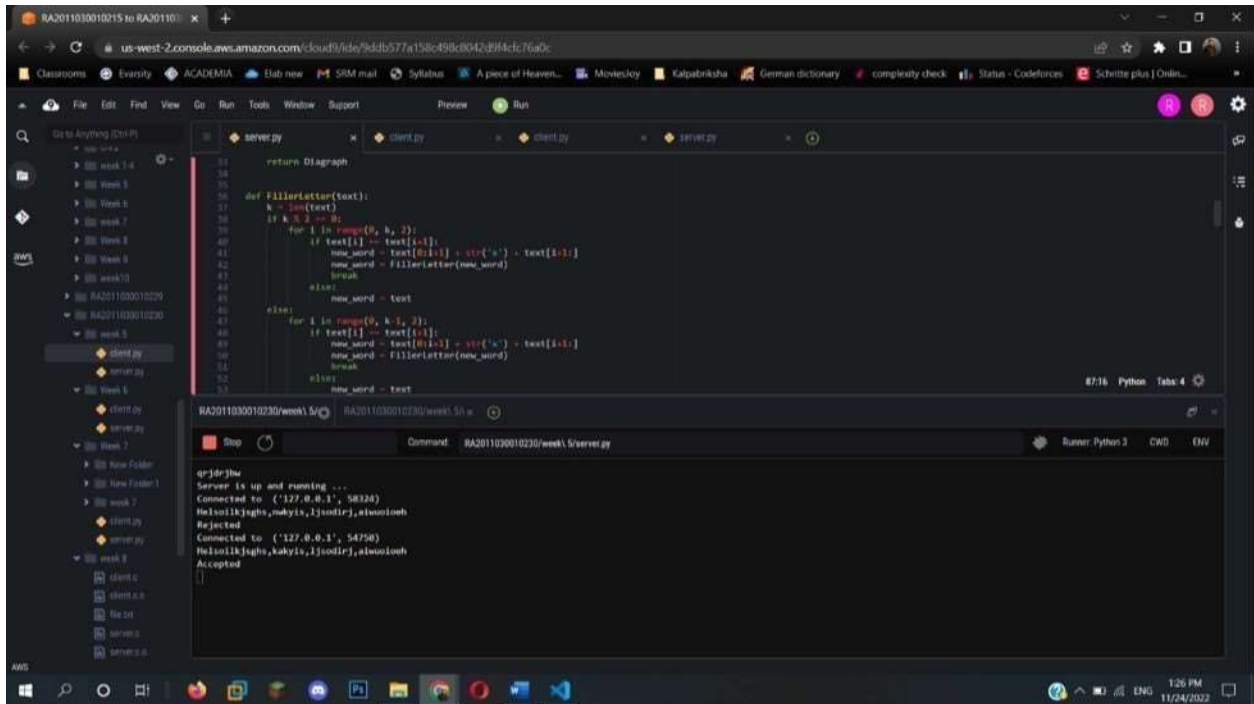


Figure 7. 4

Figure 7.1 demonstrates the setup of a CA which is just now connected to a client that is wishing to make a transaction request. Upon connecting with the client, it generates a secret key and transmits it over a secure channel to the client.

The client now must use the same key to transmit the information. In figure 7.2 we have used some other key to see what happens. We can see in figure 7.3 that the request has been rejected.

This is because there are two snapshots, one generated on client side and one on CA. As a result, when a different key is used, two of the generated values do not match and hence the input is rejected.

When the same key that was sent was CA is used in client side, then both snapshot matches and hence the request is accepted which is demonstrated in figure 7.4

Likewise, in all the figures, steganography based on text is observable. The cipher text generated is not sent directly to the CA, but it is hidden somewhere in a random string that is generated in the client side so that

even if the transmission channel is breached during transmission, the attacker will not have access to the actual cipher text as it is hidden making the cryptanalysis process extremely difficult.

## Conclusion

Hence, in this project, we had a brief overview on cryptography and steganography, and we used the same to secure the transactions happening online by establishing a Central Authority to validate information being passed to it. It helps to provide customer data privacy and prevents misuse of data at merchant's side.

Although we have used simple playfair encryption algorithm in this report for the demonstration, it can be converted into some advanced cryptographic algorithms for enhanced security and multiple encryptions can be performed as well for added layer of security.

## References

- [1] Shailendra M. Pardeshi, A Study on Combine use of Steganography and Cryptography for Data Hiding, 2015, IJSRSET, Volume 1, Issue 2, Print ISSN:2395-1990 | Online ISSN: 2394-4099
- [2] Z. V. Patel, S. A. Gadhiya, A Survey Paper on Steganography and Cryptography, International Multidisciplinary Research Journal (RHIMRJ), May2015, Volume-2, Issue-5, Online ISSN: 2349-7637
- [3] Choudhari Amar, Shaikh Sultanhusen, Ghadge Vashista, Prof. Sonawane V. D, Prof. Naved Raza, A Review - Transaction Security Using Steganography and Visual Cryptography, 2016, IJSRSET, Volume 2, Issue 1, Print ISSN: 2395-1990, Online ISSN: 2394-4099
- [4] Joseph Selvanayagam, Akash Singh, Joans Michael, Jaya Jeswani, SECURE FILE STORAGE ON CLOUD USING CRYPTOGRAPHY, Mar-2018, IRJET, Volume 5, Issue 3, Print ISSN: 2395-0072, Online ISSN: 2395-0056







