ECRYPT STREAM CIPHER

ANURAG PAUL
(RA1611008020082)
IT-'B' IIIrd YEAR
SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
RAMAPURAM, CHENNAI-600 089

SREERAM YASHASVI
(RA1611008020106)
IT-'B' IIIrd YEAR
SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
RAMAPURAM, CHENNAI-600 089

PRATIK NIKHADE
(RA1611008020111)

MR. SIVAMOHAN
MENTOR

IT-'B' IIIrd YEAR
SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
RAMAPURAM, CHENNAI-600 089

ABSTRACT

The project is built on a network security model which consists of Cryptographic techniques which involve encryption and decryption of data to provide security & confidentiality of data. Cryptography is the implementation and analysis of techniques for secure communication in the presence of participants called adversaries. This essentially involves techniques like Caesar cipher, Vigenere cipher, Diffie-Hellman key exchange, Rail Fence ,Columnar cipher. Generally cryptography is about creating and examining protocols that prevent external participants or the public from accessing private messages.

Users can pick and choose between the multiple algorithms and the encoded data will be displayed on the screen. This can be shared and the same key can be used to decrypt at other end.

INTRODUCTION

Cryptography involves producing written or generated codes that allow information to be kept undisclosed. Cryptography converts data into a format that is indecipherable for an unauthorized user, allowing it to be transmitted without unauthorized entities deciphering it back into a readable format, thus compromising the data. It is a method of storing and transmitting data in a particular form so that only those for whom it is intended can read and process it. Cryptography not only protects data from theft or alteration, but can also be used for user authentication. Cryptography involves encryption and decryption of messages. Various algorithms are used for encrypting and decrypting of the data. The information cannot be read without a key to decrypt it.

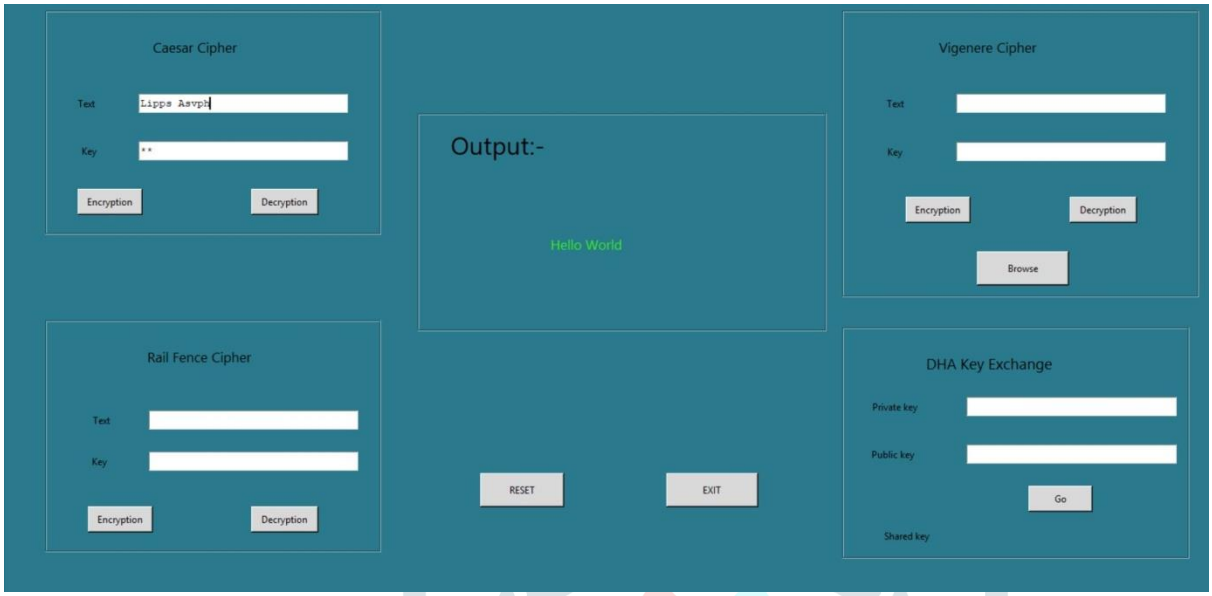
Methodology and Design

The Graphical User Interface(GUI) is designed using Pycharm IDE.

The GUI was designed to make it feel intuitive. Each algorithm has a textbox for input and an individual button to encrypt it. Output has been designated a separate space in the center.

This prototype can have far-reaching advantages. The primary aim of this project is to provide the community users with an additional layer of security for all their documents. They will have complete control over the security of the document.

GRAPHICAL USER INTERFACE



ALGORITHMS

```
def encryptc(self):
    text = self.Entry2.get()
    s = self.Entry4.get()
    result = ""
    # traverse text
    for i in range(len(text)):
        char = text[i]
        if(char==" "):
            result +=' '
        else:
            # Encrypt uppercase characters
            if (char.isupper()):
                result += chr((ord(char) + int(s)+4 - 65) % 26 + 65)

                # Encrypt lowercase characters
            else:
                result += chr((ord(char) + int(s)+4 - 97) % 26 + 97)
```

Fig 1.1 - Shift algorithm encryption

```
def decryptc(self):

    key = 'abcdefghijklmnopqrstuvwxyz'
    KEY = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
    result = ''
    text = self.Entry2.get()
    s = self.Entry4.get()
    for l in text:
        try:
            if (l==' '):
                result += ' '

            else:
                if(l.islower()):
                    i = (key.index(l) - int(s) - 4) % 26
                    result += key[i]
                else:
                    i = (KEY.index(l) - int(s) - 4) % 26
                    result += KEY[i]
        except ValueError:
            result += l
    self.Label6["text"] = result
```

Fig 1.2 – Shift algorithm decryption

The Shift cipher constitutes a simple substitution where a key number specifies how many alphabets are skipped over to replace every letter. This intuitive method was initially very popular but slowly lost its relevance due to other flexible algorithms.

```
def deffi(self):

    p = 353
    a = self.Entry7.get()
    B = self.Entry8.get()
    result = (int(B) ** int(a)) % p
    result = result % 26
    self.Label12["text"] = result
```

Fig 2.1 – Diffie-Helman algorithm

The Diffie-Helman algorithm provides an interesting case. In this process the information is not being shared but a key is being generated by two users. This is useful when a key is generated by two people and information is communicated. Even if the data is stored and analysed, the key cannot be discerned. This is because it was never saved, never transmitted and never made visible anywhere. It uses the principles of public key cryptography but is not asymmetric cryptograph because nothing is ever encrypted or decrypted.

```
def encryptr(self):
    text = self.Entry3.get()
    key = self.Entry6.get()
    if(int(key)==1):
        self.Label6["text"] =text

    else:
        rail = [['\n' for i in range(len(text))]
                for j in range(int(key))]

        dir_down = False
        row, col = 0, 0

        for i in range(len(text)):

            if (row == 0) or (row == int(key) - 1):
                dir_down = not dir_down


            rail[row][col] = text[i]
            col += 1


            if dir_down:
                row += 1
            else:
                row -= 1

        result = []
        for i in range(int(key)):
            for j in range(len(text)):
                if rail[i][j] != '\n':
                    result.append(rail[i][j])
```

Fig 3.1 – Railfence algorithm encryption

```
def decryptr(self):
    cipher = self.Entry3.get()
    key = self.Entry6.get()
    if(int(key)==1):
        self.Label6["text"] = cipher
    else:
        rail = [['\n' for i in range(len(cipher))]
                for j in range(int(key))]
        dir_down = None
        row, col = 0, 0
        for i in range(len(cipher)):
            if row == 0:
                dir_down = True
            if row == int(key) - 1:
                dir_down = False
            rail[row][col] = '*'
            col += 1
            if dir_down:
                row += 1
            else:
                row -= 1
        index = 0
        for i in range(int(key)):
            for j in range(len(cipher)):
                if ((rail[i][j] == '*') and
                        (index < len(cipher))):
                    rail[i][j] = cipher[index]
                    index += 1
        result = []
        row, col = 0, 0
        for i in range(len(cipher)):
            if row == 0:
                dir_down = True
            if row == int(key) - 1:
                dir_down = False

            if (rail[row][col] != '*'):
                result.append(rail[row][col])
                col += 1
            if dir_down:
                row += 1
            else:
                row -= 1
```

Fig 3.2 – Railfence algorithm decryption

The Railfence(sometimes called the zigzag cipher) is a transposition cipher that scrambles the order of the letters of a message using a simple algorithm. This algorithm works by writing the message alternately and then reading off each line.

```
def decryptv(self):
    alphabets = "abcdefghijklmnopqrstuvwxyz"
    p = ""
    c = self.Entry1.get()
    k = self.Entry9.get()
    kpos = []
    for x in k:
        kpos.append(alphabets.find(x))
    i = 0
    for x in c:
        if i == len(kpos):
            i = 0
        pos = alphabets.find(x.lower()) - kpos[i]
        if pos < 0:
            pos = pos + 26
        p += alphabets[pos].lower()
        i += 1
    self.Label6["text"] = p
```

Fig 4.1 – Vigenere cipher decryption

```
def encryptv(self):
    alphabets = "abcdefghijklmnopqrstuvwxyz"
    p = self.Entry1.get()
    k = self.Entry9.get()
    c = ""
    kpos = []
    for x in k:

        kpos.append(alphabets.find(x))
    i = 0
    for x in p:
        if i == len(kpos):
            i = 0
        pos = alphabets.find(x) + kpos[i]
        print(pos)
        if pos > 25:
            pos = pos - 26
        c += alphabets[pos].capitalize()
        i += 1
```

Fig 4.2 – Vigenere cipher encryption

Vigenere cipher is based on polyalphabetic substitution using multiple substitution alphabets. The encryption is achieved by creating a vigenere table which consists of all the alphabets written 26 times. Each alphabet uses a different alphabet from a different row. This selection is based on a repeating keyword.

Conclusion

This project provided a platform to understand and implement the latest technologies. This enabled us to gain a deeper understanding of the underlying technologies, also providing insight into how these different techniques mesh together in real-life applications. Efficient Cryptography techniques have become quite essential for any secure application. We are confident that our progress in this regard will hold us in good stead in our future endeavours.