VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

VL2020210102684

Information Security Analysis and Audit

Review 3

Encryption/Decryption using ECC

Under the guidance of Prof. Chandra Mohan B

Course Code: CSE3501

Slot: G2

Team members:

1. Ritika Kayal, 18BCE2518
2. Srinivas N, 18BCE0048
3. Amritanshi Saxena, 18BCE2524

1. Abstract

With the rise of the internet, it has become more and more common for important, critical documents to be shared through electronic means. This means that it has become essential that documents and important details be kept confidential through the means of encryption. ECC has risen in popularity and is dubbed “The Successor to RSA” as it is capable of achieving the same security of a 1024 bit RSA key with just 208 bits. Thus, it is the most optimal method for securing data against breaches and unauthorized access.

Documents have also grown in size over time as more detail can be stored due to larger and faster storage availability. Excessive amounts of time is wasted on reading filler and unnecessary content in documents to understand them and this becomes an issue as it limits the productivity of an individual . Skimming through large documents may also lead to users missing important details. Hence a smart Natural Language Processing system that can parse through documents / text files / URLs can help save precious time while also conveying all the important facts/details needed.

2. Introduction

Elliptic Curve Cryptography (ECC)

ECC is one of the modern families of public-key cryptosystems, built on the algebraic structures of the elliptic curves over finite fields and also on the difficulty of the Elliptic Curve Discrete Logarithm Problem (ECDLP). It showcases all major abilities of asymmetric cryptosystems which are: encryption, key exchange and signatures .It is known as a normal present day replacement of the RSA cryptosystem, since ECC utilizes more small signatures and keys than contrasted with RSA for same degree of security giving a fast key generation just as key arrangement and key signatures. ECC isn't an independent algorithm, it is utilized to produce keys for both Symmetric and Asymmetric algorithms. Hence we shall be using the Elliptic Curve Integrated Encryption Scheme (ECIES) to encrypt our data. This algorithm is based on the use of symmetric keys (i.e. the sender and the receiver have the same keys for encryption/decryption). The Elliptic curve we shall be using is one that is approved by the National Institute of Standards and Technology (NIST) and field tested by the NSA. These equations are generally of the form: **x^2 = a*x^3 + b*x + c , where a,b,c are real numbers**

Natural Language Processing

The web is overwhelmed with a huge measure of information with text information, for example, online news, sites, stories, and other data storage facilities. Text Summarization helps in speaking to textual data in a conservative structure without bargaining the semantic significance is an ultimate objective of text summarization models.

We will implement the NLP techniques of extractive text summarisation that analyses the content to determine important sentences to be displayed in the summary. We will be doing this with an algorithm that will implement the spacy library of python to decrease the text bodies but ensuring its original and real meaning or giving a great insight into the actual/original text.

3. Project Learning Experience and Tools Utilised

A variety of tools were utilised to develop our project application. We implemented the code using Python and a variety of open source libraries to perform Natural language processing and web scraping.

The tools are listed as follows:

- Randomizer to generate polynomial values for ECC
- pandas for Data Manipulation
- pickleshare for the Data Structures
- numpy for Array Data Calculations
- tkinter for the User Interface
- SpaCy for Natural Language Processing
- BeautifulSoup for web scraping

4. Input and Output

Input:

- Text in the form of Plain Text or Text Document or Website URL. (This is the text to be summarised).

Output:

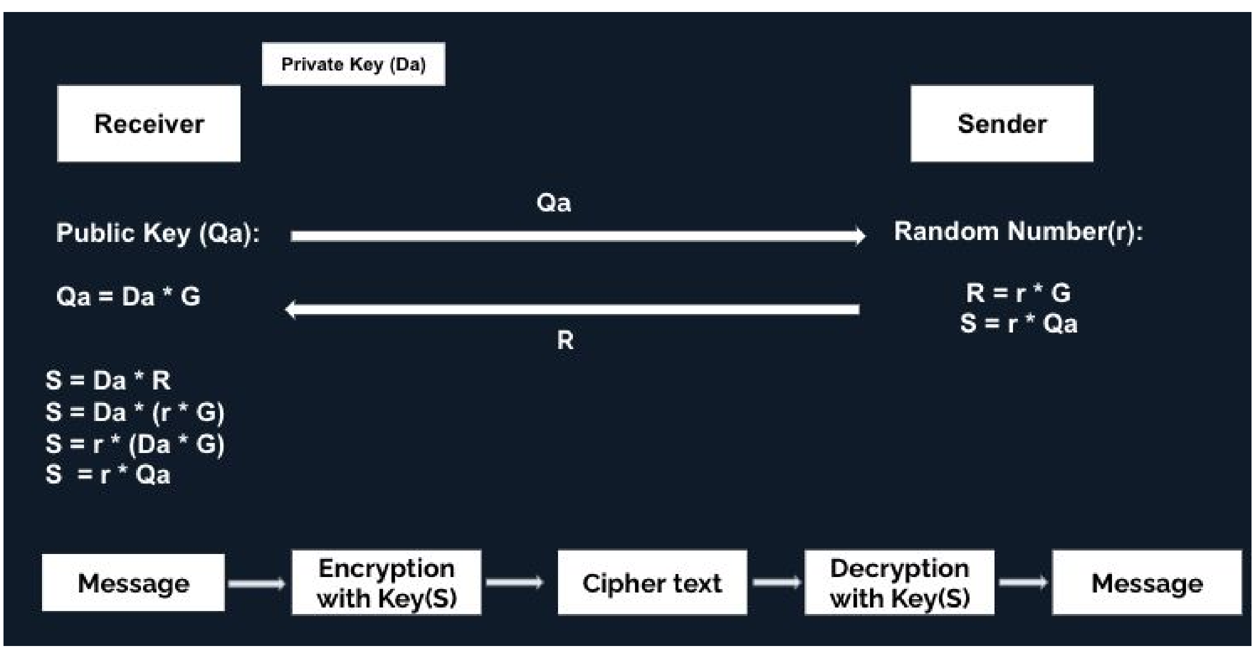
- Summarised document
- Encrypted text
- Public Key

5. Process Methodology

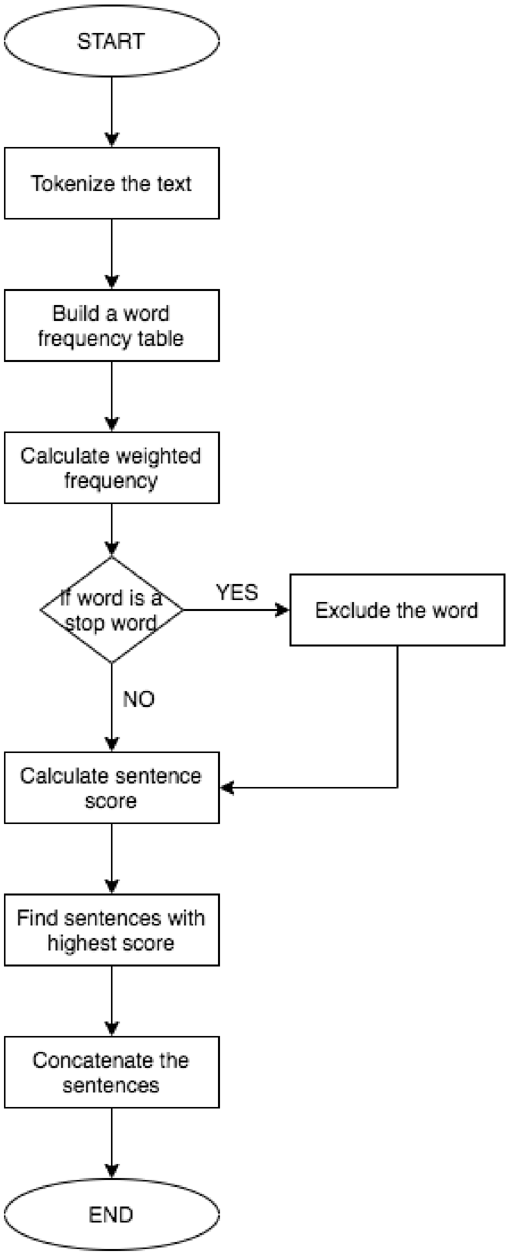
Description of the ECIES algorithm, SpaCy text summarisation algorithm and web scraping algorithm.

Encryption / Decryption using the Elliptic Curve Integrated Encryption Scheme:

1. Select Elliptic Curve for Encryption from verified sources, in our case the National Institute of Standards and Technology (NIST).
2. Use a randomizer to select the start of the line segment on the curve and the number of symmetric operations to be applied.
3. Encrypt the data given as input with the help of the key generated.
4. Send the encrypted data along with the public key.
5. Generate private key with the help of public key and secret value known only to the receiver.
6. Decrypt the data using the private key.

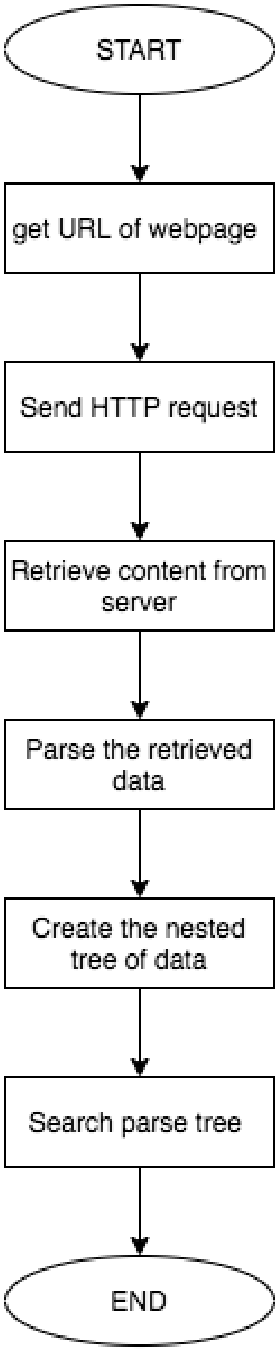


Spacy Text Summarisation



1. Tokenize the text to be summarised.
2. Build a word frequency table which is a dictionary of words and their counts.
3. Calculate the weighted frequency by dividing each frequency by the maximum frequency.
4. Calculate sentence scores based on the number of words in each sentence excluding the stopwords with the help of the weighted frequencies calculated.
5. Find n sentences with the highest score.
6. Concatenate them to produce the summary with the desired number of sentences.

Web Scraping algorithm



1. Use a HTTP library to send an HTTP request to the URL of the desired webpage.

2. Retrieve the HTML content of the desired webpage from the server.

3. Parsing the retrieved data using a parser which creates a nested/tree structure of the HTML data. This can be done using an HTML parser library such as html5lib.

4. Navigate and search the parse tree created using another third-party python library, Beautiful Soup.

6. Task List

i. File handling

ii. Web scraping Model

iii. NLTK, Spacy Corpus building

iv. Model Hyper-Parameters Optimisation and Sentence Score Calculation

v. Summarisation of Document

vi. Tkinter Setup

vii. UI for URL tab

viii. UI for File upload/Text Document tab

ix. Elliptic Curve Line Segment Calculations.

x. ECIES Key generation

xi. Encryption

xii. Decryption

7. Implementation

ECIES Code

We define the Elliptic Curve by declaring its different attributes.

- p*p -> The size of the finite field the curve is in
- n-> The order of the curve.
- a, b -> Constants in the equation y^2 = x^3 + a*x + b(mod p)
- g-> The curve generator point
- h -> The cofactor

The curve we have chosen for our implementation is: y^2 = x^3 + x + 7

```
EllipticCurve = collections.namedtuple('EllipticCurve', 'name p a b g n h')

curve = EllipticCurve(
    'secp256k1',
    # Field characteristic.
    p=0xfffffffffffffffffffffffffffffffffffffffffffffffffffffffefffffc2f,
    # Curve coefficients.
    a=0,
    b=7,
    # Base point.
    g=(0x79be667ef9dcbbac55a06295ce870b07029bfcdb2dce28d959f2815b16f81798,
       0x483ada7726a3c4655da4fbfc0e1108a8fd17b448a68554199c47d08ffb10d4b8),
    # Subgroup order.
    n=0xfffffffffffffffffffffffffffffffebaaedce6af48a03bbfd25e8cd0364141,
    # Subgroup cofactor.
    h=1,
)
```

We encode the text given to the encryption /decryption module into byte format and pad any empty spaces with 0’s. We selected a block size of 128 bits (16 bytes) so the entire message is split into multiple blocks of this length each of which are encrypted and concatenated together. We perform encryption / decryption using a 256 bit key for maximum security.

```
def enc_long(n):            #Encoding large number to a sequence of bytes
    s = ""
    while n > 0:
        s = chr(n & 0xFF) + s
        n >>= 8             #Shifting 8bits i.e. 1 byte
    return s


############################################################# Padding for Data

BLOCK_SIZE = 16  # Bytes                      (128 bits per block)
pad = lambda s: s + (BLOCK_SIZE - len(s) % BLOCK_SIZE) * \
                chr(BLOCK_SIZE - len(s) % BLOCK_SIZE)
unpad = lambda s: s[:-ord(s[len(s) - 1:])]

##############################################################################


def encrypt(plaintext,key, mode):
    encobj = AES.new(key,mode)             #256-bit
    return(encobj.encrypt(plaintext))

def decrypt(ciphertext,key, mode):
    encobj = AES.new(key,mode)
    return(encobj.decrypt(ciphertext))
```

Throughout the process of key generation, we use arithmetic operations to calculate the position and geometric values of the multiple points on the elliptic curve. This includes operations such as:

Point Negation: Used to find the mirror image of a point on the opposite side of the x-axis that also lies on the elliptic curve.

Point Addition: The two points must be connected and their intersection yields the conjugate of the result of the elliptic field addition. We need to flip this intersection point to find the additional result.

Scalar Multiplication: This forms one of the fundamental steps in the ECC key generation process. The inverse of this process (i.e. finding the value of k given the product) is known as the discrete log problem and is extremely difficult to solve (NP incomplete).

```python
def point_neg(point):                                   #Mirror image allong x axis
    """Returns -point."""
    assert is_on_curve(point)                           #Should be a point on the curve

    if point is None:
        # -0 = 0
        return None

    x, y = point
    result = (x, -y % curve.p)

    assert is_on_curve(result)                          #Result should lie on the curve

    return result


def point_add(point_a, point_b):
    """Returns the result of point_a + point_b according to the group law."""
    assert is_on_curve(point_a)
    assert is_on_curve(point_b)

    if point_a is None:                             # 0 + point_b = point_b
        return point_b
    if point_b is None:                             # point_a + 0 = point_a
        return point_a

    x1, y1 = point_a
    x2, y2 = point_b

    if x1 == x2 and y1 != y2:                   # point_a + (-point_a) = 0
        return None

    if x1 == x2:                                #  point_a == point_b.
        m = (3 * x1 * x1 + curve.a) * inverse_mod(2 * y1, curve.p)
    else:
        # This is the case point_a != point_b.
        m = (y1 - y2) * inverse_mod(x1 - x2, curve.p)

    x3 = m * m - x1 - x2
    y3 = y1 + m * (x3 - x1)
    result = (x3 % curve.p,
              -y3 % curve.p)
```

<u>Key Generation</u>: We generate a random private key of length n (order of the elliptic polynomial curve) using which we create a public key. This is done by applying this randomly generated value to the generator point (g) of the elliptic curve to create a new point on the curve which serves as a public key. It is nearly impossible to trace back the private key due to the sheer size and number of possible combinations.

```python
################################################## # Keypair generation and ECIES

def make_keypair():
    """Generates a random private-public key pair."""
    private_key = random.randrange(1, curve.n)          #Generate Random Key
    public_key = scalar_mult(private_key, curve.g)      #Make Public key

    return private_key, public_key
```

```python
def scalar_mult(k, point):
    """Returns k * point computed using the double and point_add algorithm."""
    assert is_on_curve(point)

    if k % curve.n == 0 or point is None:
        return None

    if k < 0:
        # k * point = -k * (-point)
        return scalar_mult(-k, point_neg(point))

    result = None
    addend = point

    while k:
        if k & 1:
            # Add.
            result = point_add(result, addend)

        # Double.
        addend = point_add(addend, addend)

        k >>= 1

    assert is_on_curve(result)

    return result
```

Natural Language Processing Code

We make use of the SpaCy python library to perform Natural Language Processing. We pass the raw text through the NLP pipeline to perform tokenization, sentence segmentation, lemmatization, etc. For each word that does not belong to the list of stopwords, we append its frequency to a table.

```python
# NLP Pkgs
import spacy
nlp = spacy.load('en')
# Pkgs for Normalizing Text
from spacy.lang.en.stop_words import STOP_WORDS
from string import punctuation
# Import Heapq for Finding the Top N Sentences
from heapq import nlargest


def text_summarizer(raw_docx):
    raw_text = raw_docx
    docx = nlp(raw_text)
    stopwords = list(STOP_WORDS)
    # Build Word Frequency # word.text is tokenization in spacy
    word_frequencies = {}
    for word in docx:
        if word.text not in stopwords:
            if word.text not in word_frequencies.keys():
                word_frequencies[word.text] = 1
            else:
                word_frequencies[word.text] += 1
```

We then divide each frequency by the max frequency and calculate the sentence score by summing up the frequencies of each word in the sentence. To keep it fair, we only evaluate sentences with word length less than 30. Finally we concatenate the top 7 sentences with the highest sentence scores to the summary to obtain the output.

```python
    maximum_frequncy = max(word_frequencies.values())

    for word in word_frequencies.keys():
        word_frequencies[word] = (word_frequencies[word]/maximum_frequncy)
    # Sentence Tokens
    sentence_list = [ sentence for sentence in docx.sents ]

    # Sentence Scores
    sentence_scores = {}
    for sent in sentence_list:
        for word in sent:
            if word.text.lower() in word_frequencies.keys():
                if len(sent.text.split(' ')) < 30:
                    if sent not in sentence_scores.keys():
                        sentence_scores[sent] = word_frequencies[word.text.lower()]
                    else:
                        sentence_scores[sent] += word_frequencies[word.text.lower()]


    summarized_sentences = nlargest(7, sentence_scores, key=sentence_scores.get)
    final_sentences = [ w.text for w in summarized_sentences ]
    summary = ' '.join(final_sentences)
    return summary
```

User Interface Code

The UI for our project is implemented using the python library 'tkinter'. It offers great flexibility in UI creation but is also fairly lightweight and does not degrade performance. Here we create a primary window to open on launch and set its default size, title and background colour. We then create a series of tabs aligned to the top left of the window for the various different functionalities we have implemented.

```python
# ADD TABS TO NOTEBOOK
tab_control.add(tab1, text=f'{"Home":^40s}')
tab_control.add(tab3, text=f'{"Plain Text":^39s}')
tab_control.add(tab4, text=f'{"File Upload":^37s}')
tab_control.add(tab5, text=f'{"URL analysis":^36s}')

label1 = Label(tab1, text= 'About',font='Helvetica 18 bold',padx=5, pady=5)
label1.grid(column=1, row=0, pady=5)

label3 = Label(tab3, text= 'Plain Text Summarisation and Encryption/Decryption',font='Helvetica 16 bold',padx=5, pady=5)
label3.grid(column=1, row=0, pady=5)

label4 = Label(tab4, text= 'File Summarisation and Encryption/Decryption',font='Helvetica 16 bold',padx=5, pady=5)
label4.grid(column=1, row=0, pady=5)

label5 = Label(tab5, text= 'URL Summarisation and Encryption/Decryption',font='Helvetica 16 bold',padx=5, pady=5)
label5.grid(column=1, row=0, pady=5)

tab_control.pack(expand=1, fill='both')
```

```
# Structure and Layout
window = Tk()                          #Window for the GUI
window.title("Encryption/Decryption using ECIES")
window.geometry("1400x1200")                             #Window
window.config(background='blue')

style = ttk.Style(window)                                  #Posit
style.configure('lefttab.TNotebook', tabposition='wn') #tabs p


##################################################################
tab_control = ttk.Notebook(window,style='lefttab.TNotebook')

tab1 = ttk.Frame(tab_control)           #Home Tab
tab2 = ttk.Frame(tab_control)
tab3 = ttk.Frame(tab_control)           #Plaint Text Tab
tab4 = ttk.Frame(tab_control)           #File Upload Tab
tab5 = ttk.Frame(tab_control)           #URL Summarisation tab
tab6 = ttk.Frame(tab_control)

# ADD TABS TO NOTEBOOK
```

After initialising different tabs to separate the functionalities of our project, we must define their names and font. We also have to set their relative position and order

After the creation of separate tabs, we must define the contents of each tab. We do this by defining a Main title for the tab followed by the required components. In our case this includes,

- File Explorer Button
- Summarization Button
- Encryption / Decryption Button
- Text Box to display selected text
- Text Box to display encrypted / decrypted and summarised content along with the keys used

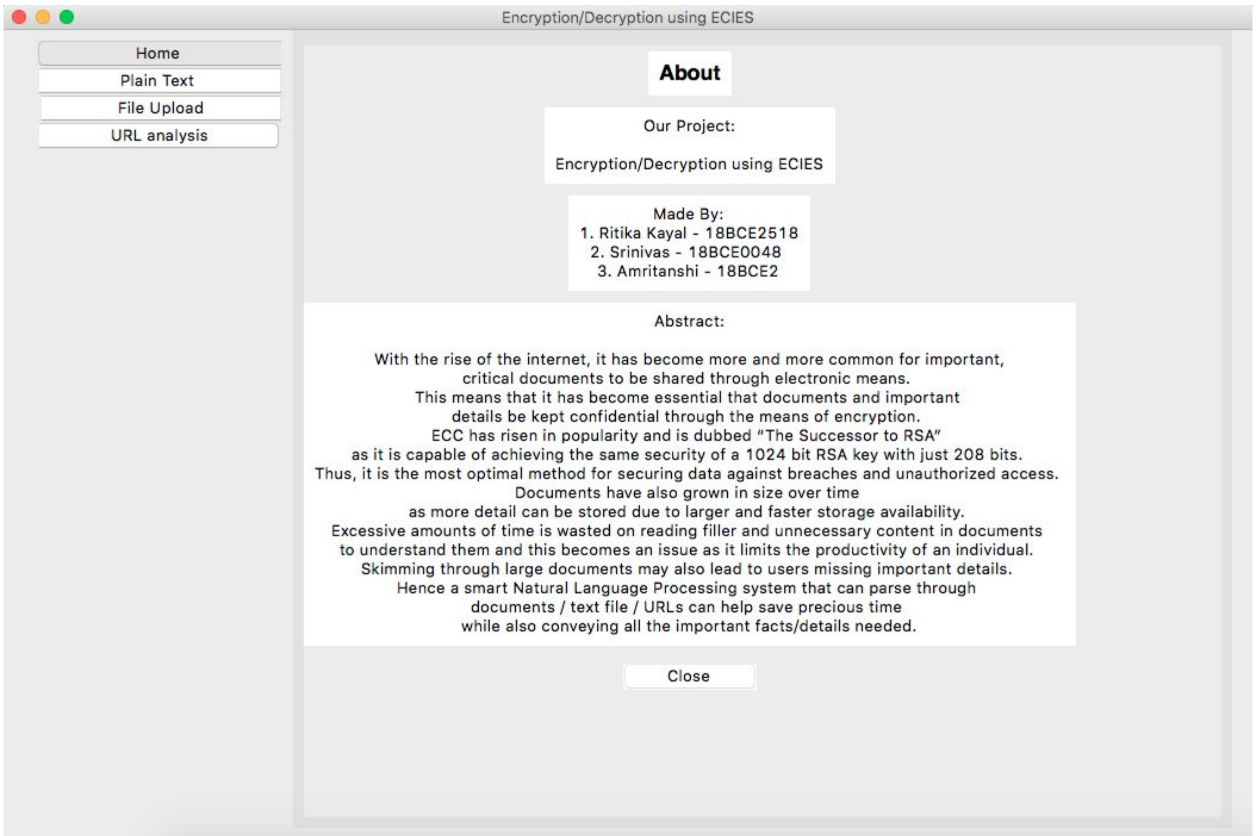
This process is repeated for all 3 tabs present in the project along with a Home tab.

```python
##################################################### File Upload Summar
l1=Label(tab4,text="Open File To Summarize", font='Helvetica 14 bold')
l1.grid(row=1,column=1)
displayed_file = ScrolledText(tab4,height=8)
displayed_file.grid(row=2,column=0, columnspan=3,padx=5,pady=5)


b0=Button(tab4,text="Open File", width=12, command=openfiles)
b0.grid(row=3,column=0,padx=10,pady=10)

b2=Button(tab4,text="Summarize", width=12,command=get_file_summary)
b2.grid(row=3,column=2,padx=10,pady=10)

button5=Button(tab4,text="Encrypt", command=encrypt_file, width=12)
button5.grid(row=4,column=0,padx=10,pady=10)

button6=Button(tab4,text="Decrypt", command=decrypt_file, width=12)
button6.grid(row=4,column=2,padx=10,pady=10)

l1=Label(tab4,text="Private Key", font='Helvetica 14 bold')
l1.grid(row=5,column=1)
tab4_display1 = ScrolledText(tab4, height=1)            #Decryption Key
tab4_display1.grid(row=6,column=0, columnspan=3,padx=5,pady=5)

l1=Label(tab4,text="Output", font='Helvetica 14 bold')
l1.grid(row=7,column=1)
# Display Screen
tab4_display_text = ScrolledText(tab4,height=8)
tab4_display_text.grid(row=8,column=0, columnspan=3,padx=5,pady=5)

# Allows you to edit
tab4_display_text.config(state=NORMAL)

b1=Button(tab4,text="Reset", width=12,command=clear_text_file)
b1.grid(row=9,column=0,padx=10,pady=10)

b5=Button(tab4,text="Save", command=save_summary1, width=12)
b5.grid(row=9,column=1,padx=10,pady=10)

b3=Button(tab4,text="Clear Result", width=12,command=clear_text_result)
b3.grid(row=9,column=2,padx=10,pady=10)
########################################################################
```

8. Results And Discussion

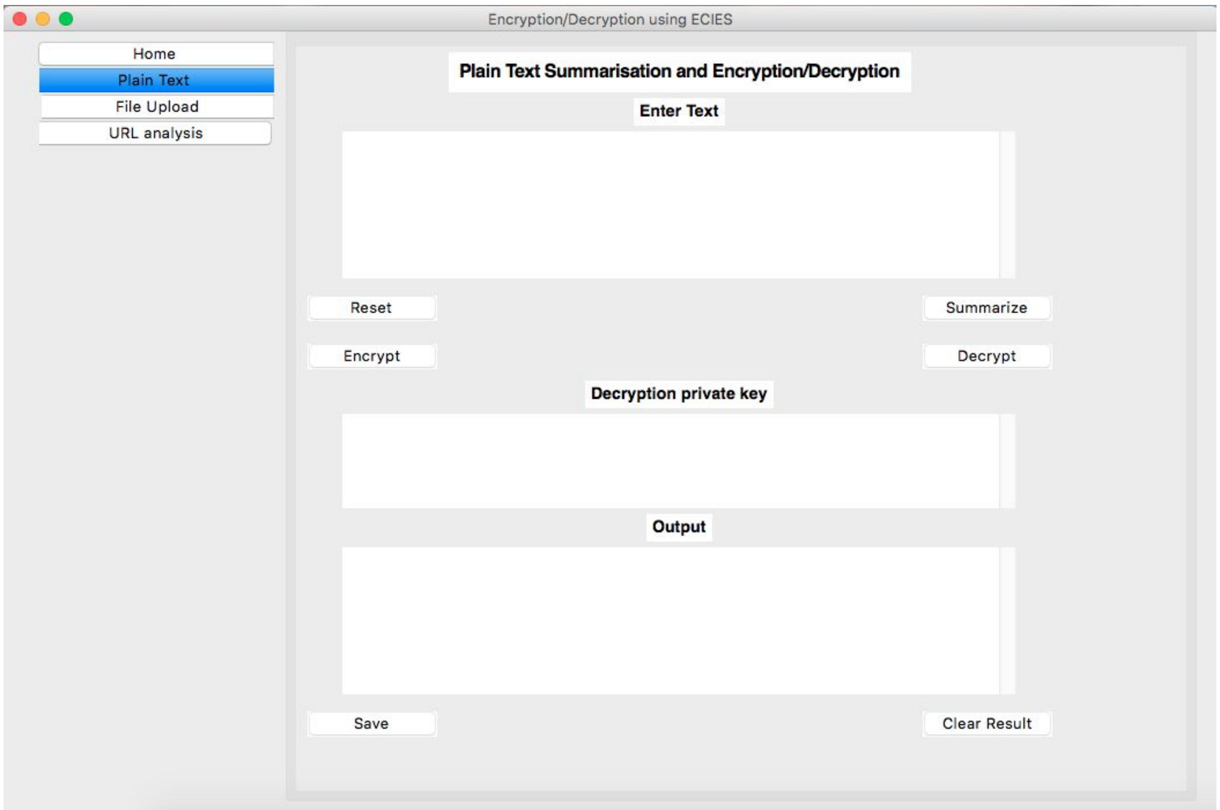
Home Tab:

This tab provides a description of the abstract of our project. It also contains details of the project members. It has a close button that allows you to exit the application when clicked.



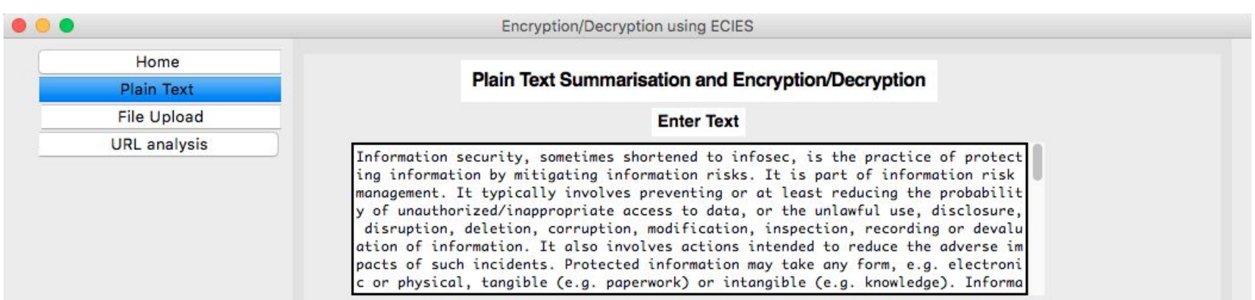
Plain Text Tab:

This tab allows the user to enter plain text to perform encryption/decryption and text summarisation. The Reset and Clear result buttons clears the input and output respectively. The Save button allows us to save the output in a .txt file.



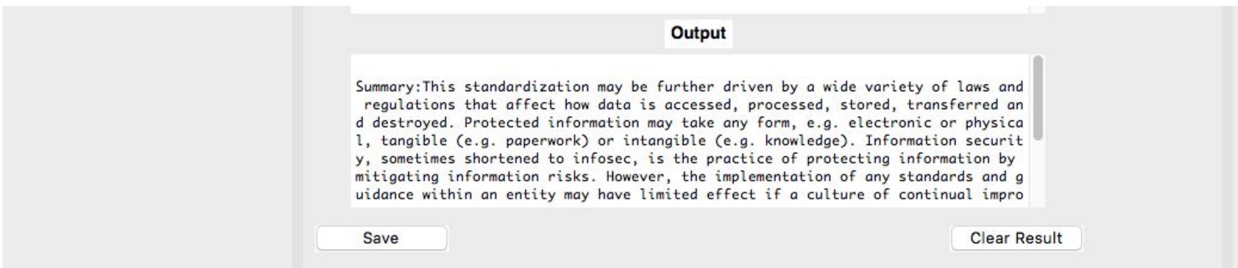
Inputting on the Plain Text Tab

Plain text can simply be copied and pasted into the Enter Text textbox. Here text about information security has been pasted into the text box.



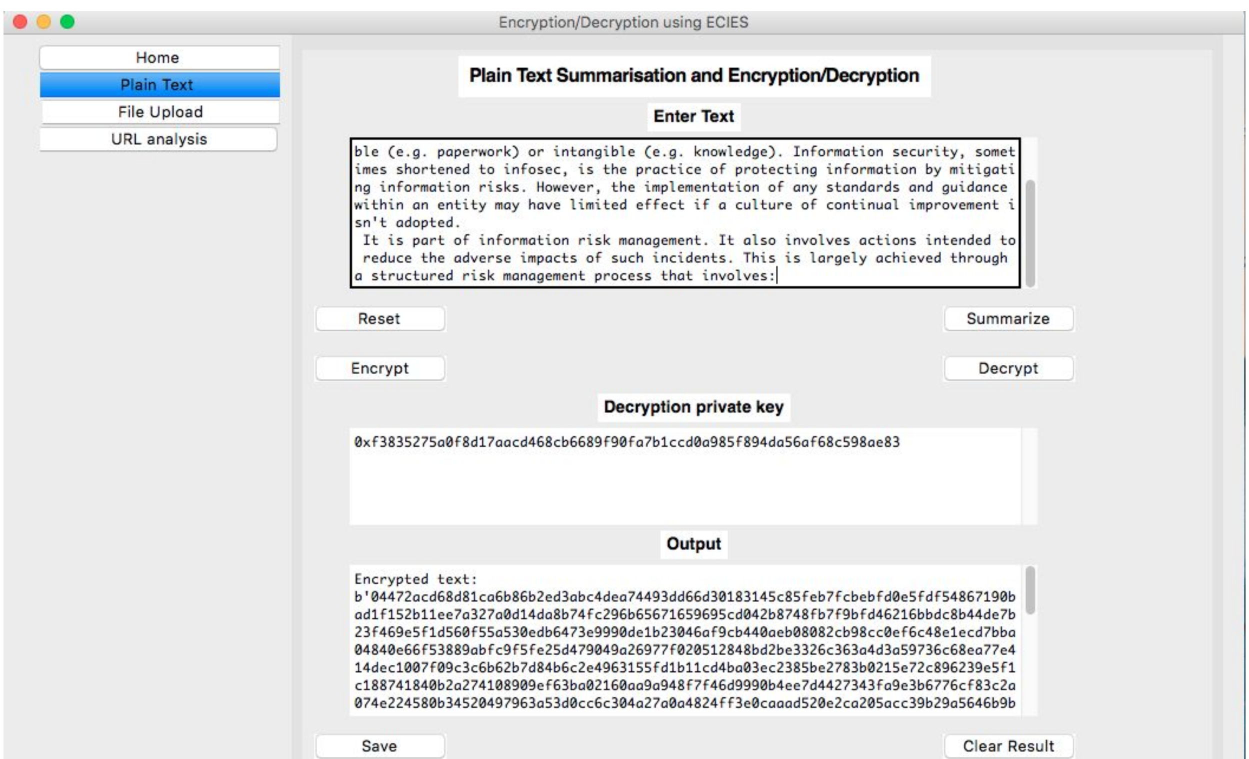
Summarisation on the Plain Text Tab

When the Summarize button is clicked the summarised text gets displayed in the output text box. This can further be encrypted by copying it into the input box and clicking on the Encrypt button.



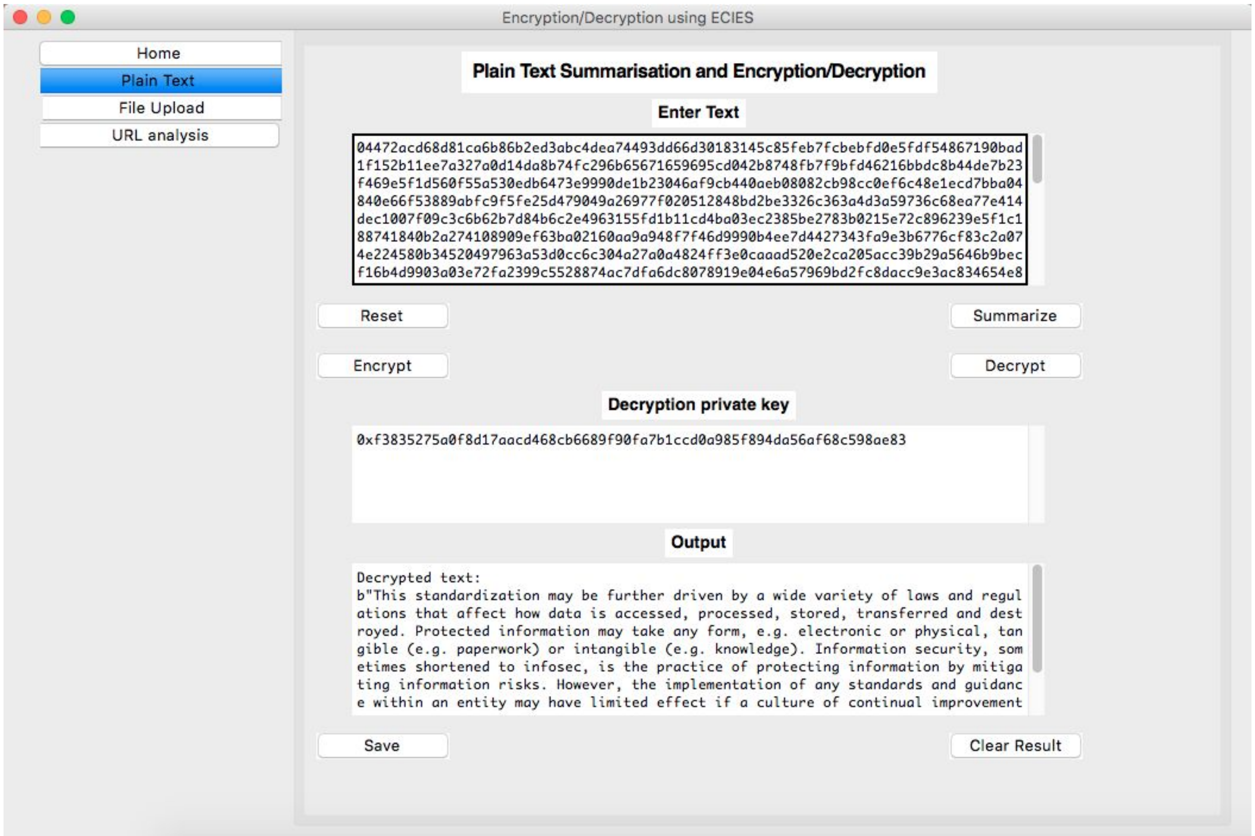
Encryption on the Plain Text Tab

Upon clicking the Encrypt button, the encrypted text gets displayed in the Output text box. The Decryption private key gets displayed in the corresponding textbox which needs to be noted down and inputted when performing decryption.



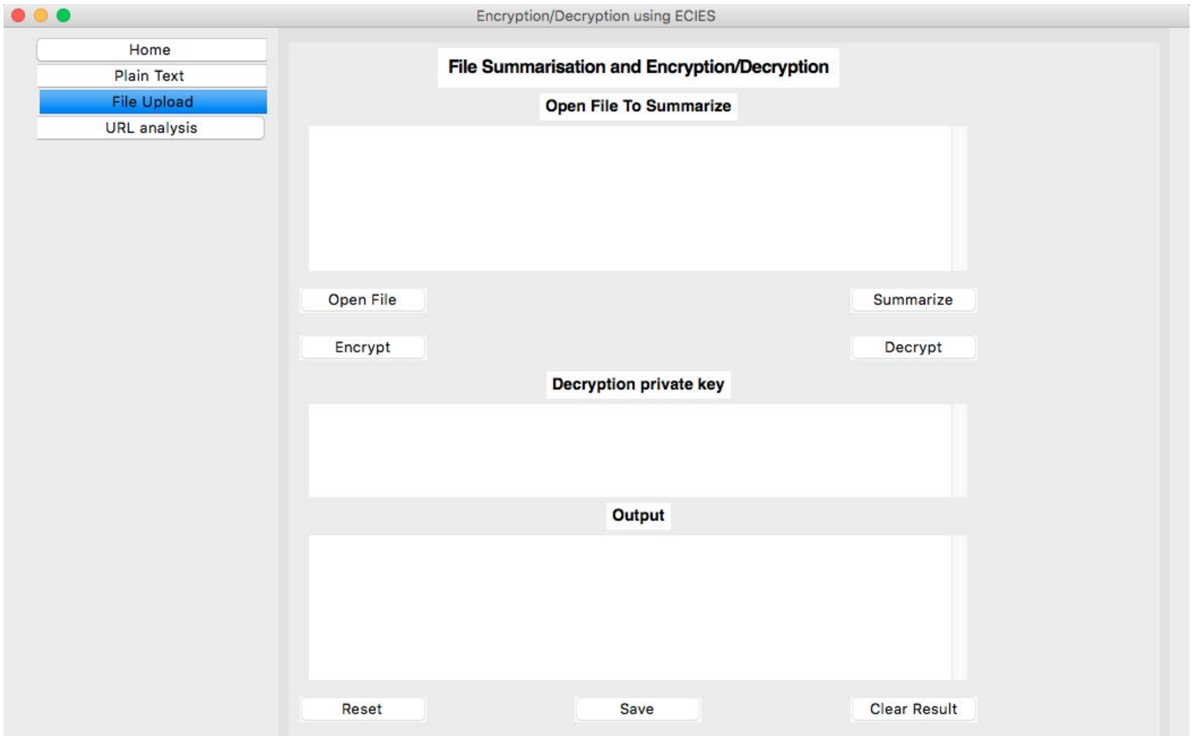
Decryption on the Plain Text Tab

The encrypted text is inputted in the text box and the decryption private key that was noted down earlier is inputted as well. The Decrypt button performs decryption and the Output is displayed in the corresponding textbox.



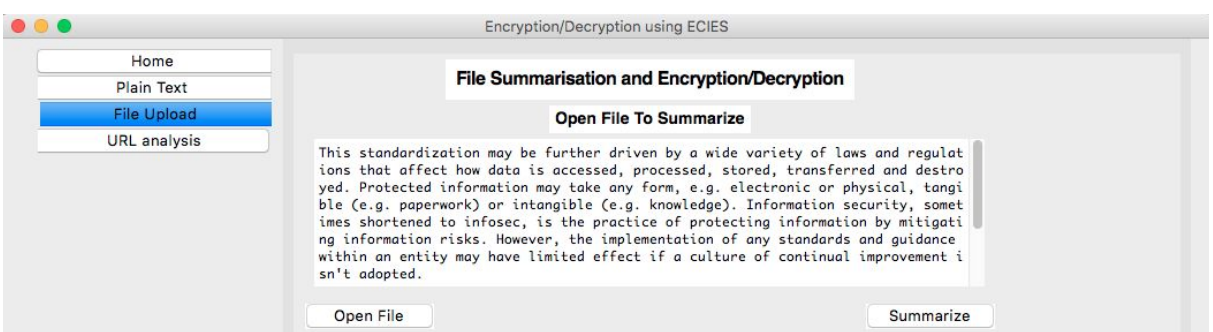
File Upload Tab:

This tab allows the user to read a file to perform encryption/decryption and text summarisation. The Reset and Clear result buttons clears the input and output respectively. The Save button allows us to save the output in a .txt file.



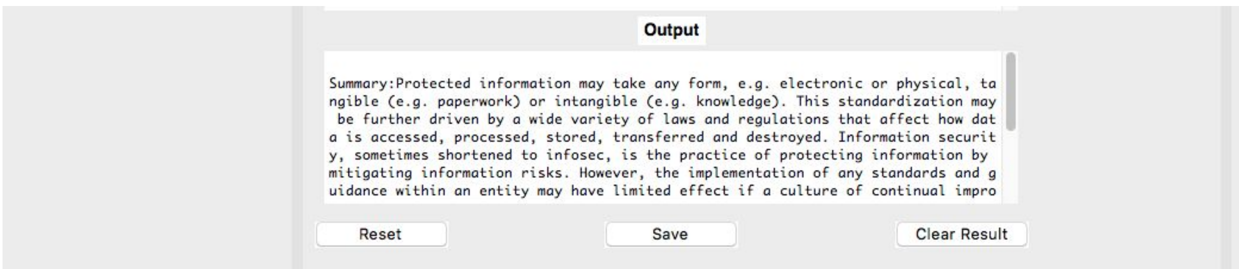
Reading a file on the File Upload Tab

The Open File button allows the user to open a .txt file and display its contents in the input text box. This can later be summarised, encrypted or decrypted according to the user's needs.



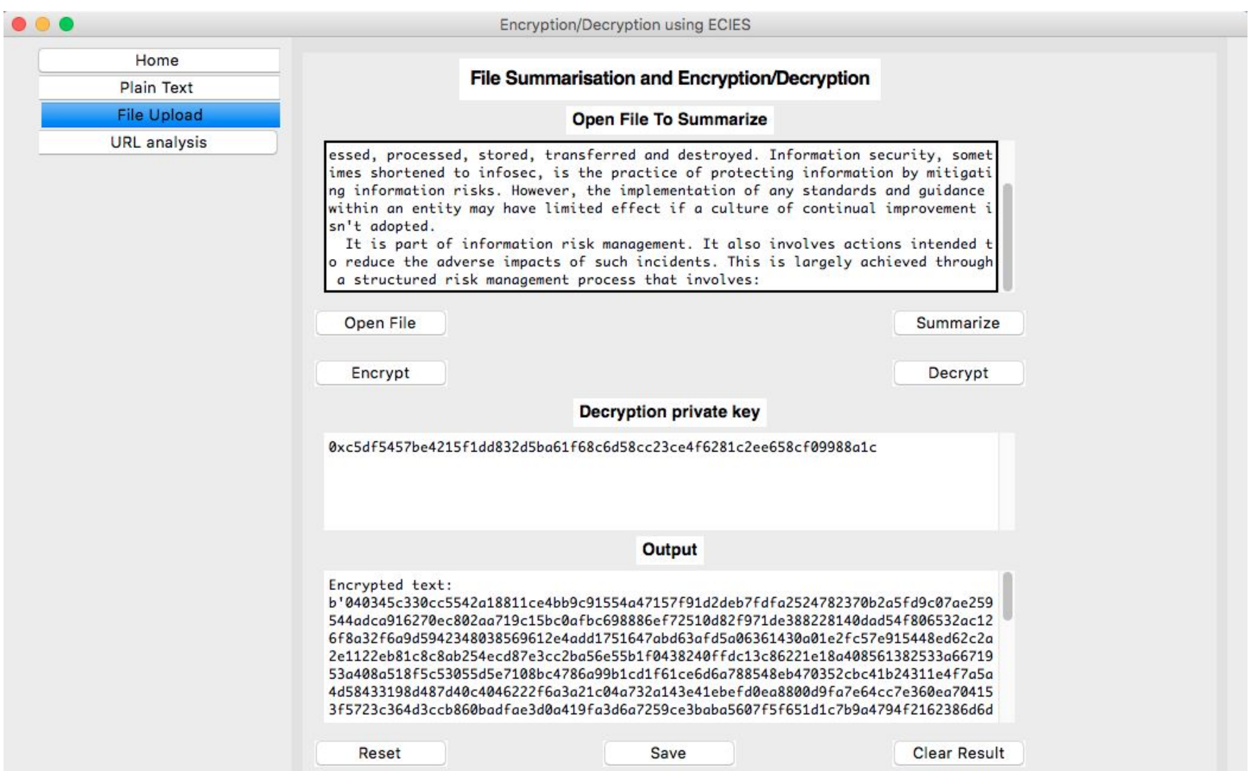
Summarisation on the File Upload Tab

When the Summarize button is clicked the summarised text gets displayed in the output text box. This can further be encrypted by copying it into the input box and clicking on the Encrypt button.



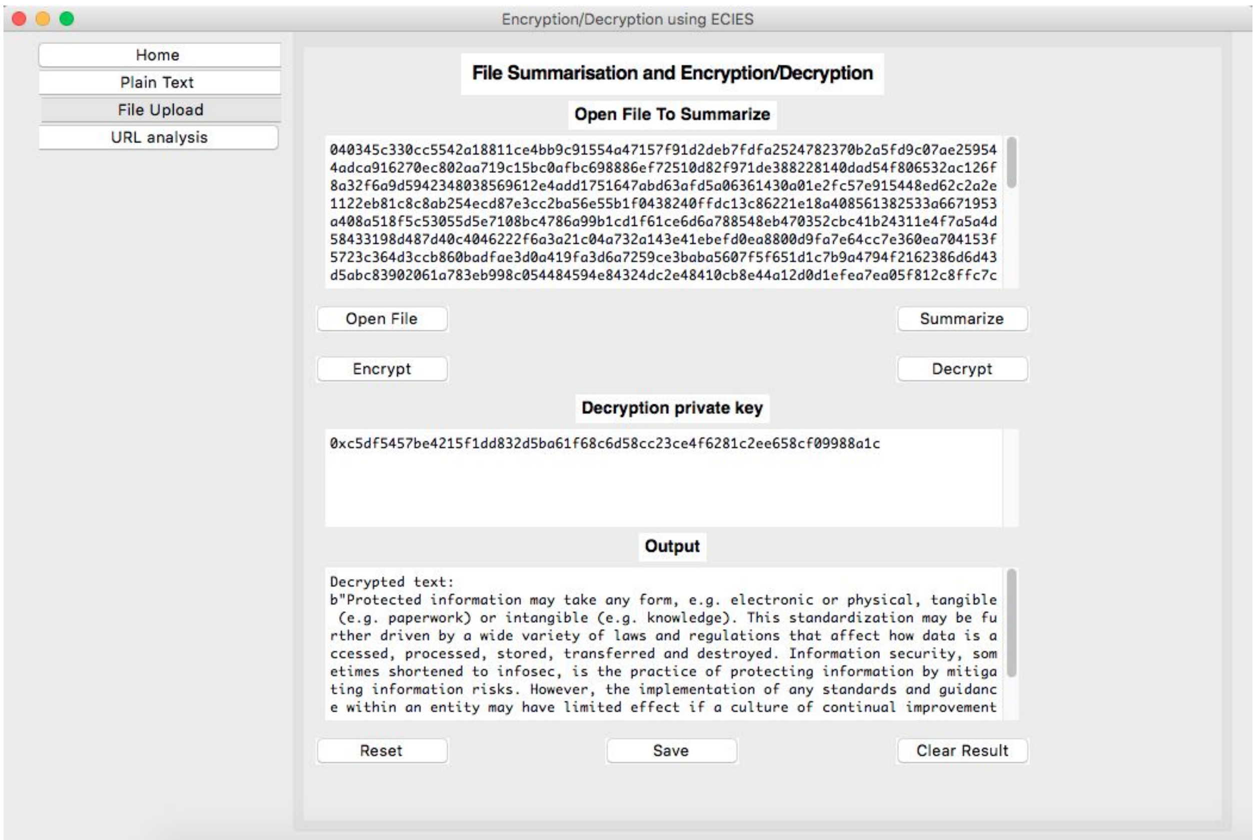
Encryption on the File Upload Tab

Upon clicking the Encrypt button, the encrypted text gets displayed in the Output text box. The Decryption private key gets displayed in the corresponding textbox which needs to be noted down and inputted when performing decryption.



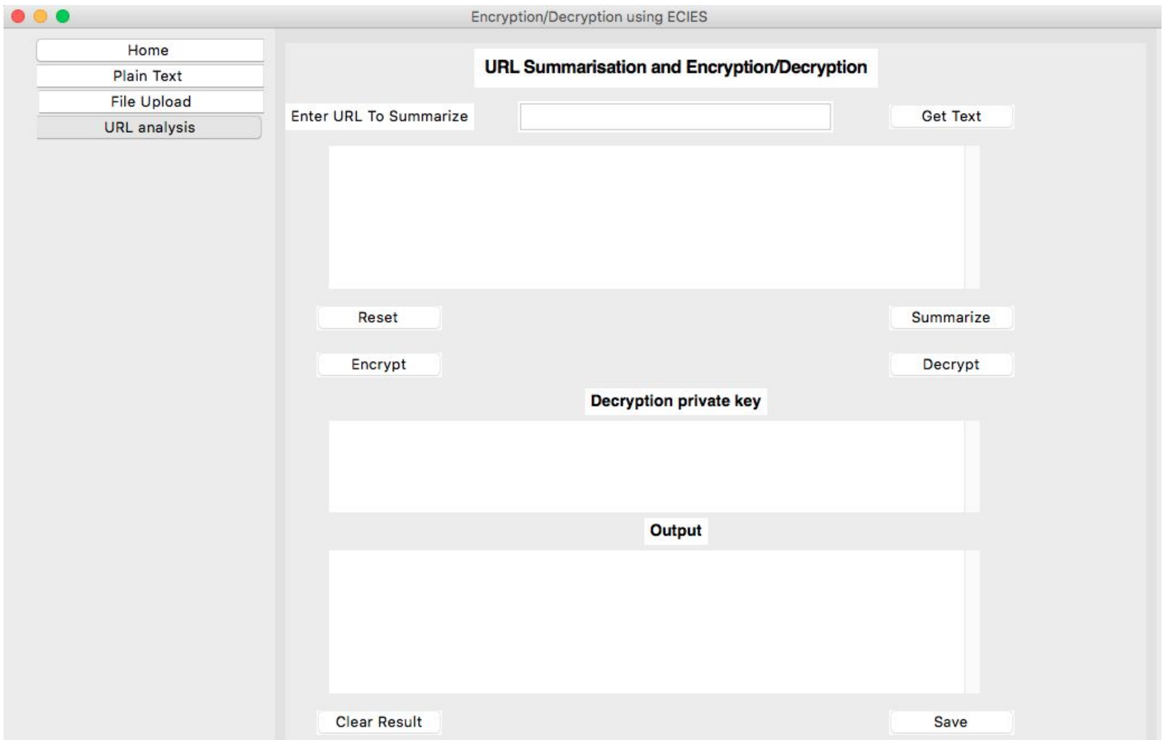
Decryption on the File Upload Tab

A file containing encrypted text is inputted and the decryption private key that was noted down earlier is inputted as well. The Decrypt button performs decryption and the Output is displayed in the corresponding textbox.



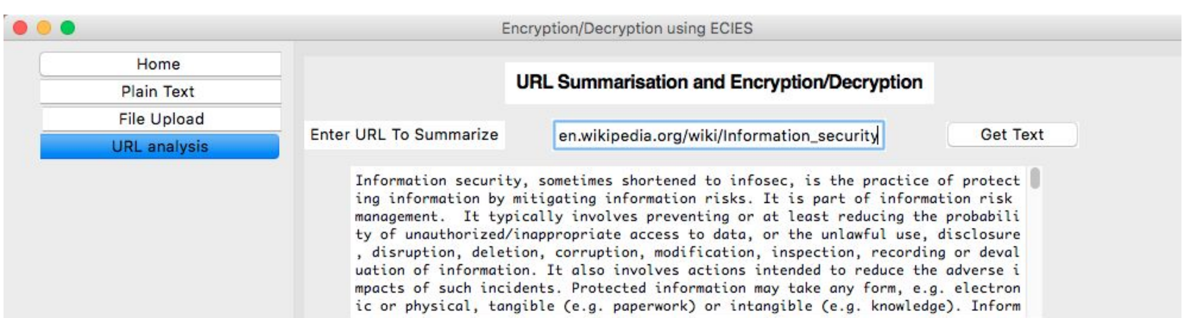
URL Analysis Tab:

This tab performs web scraping to obtain text given a specific URL. The Reset and Clear result buttons clears the input and output respectively. The Save button allows us to save the output in a .txt file.



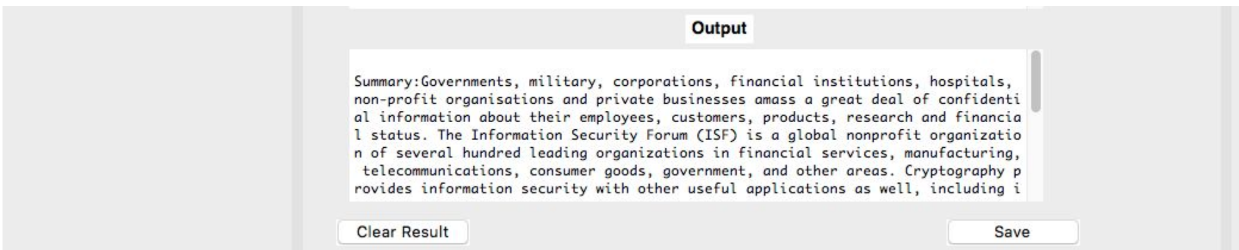
Web scraping on the URL analysis Tab

When a valid URL is inputted, the Get Text button performs web scraping to read the text from the website and display it in the input text box. This can later be summarised, encrypted or decrypted according to the user's needs.



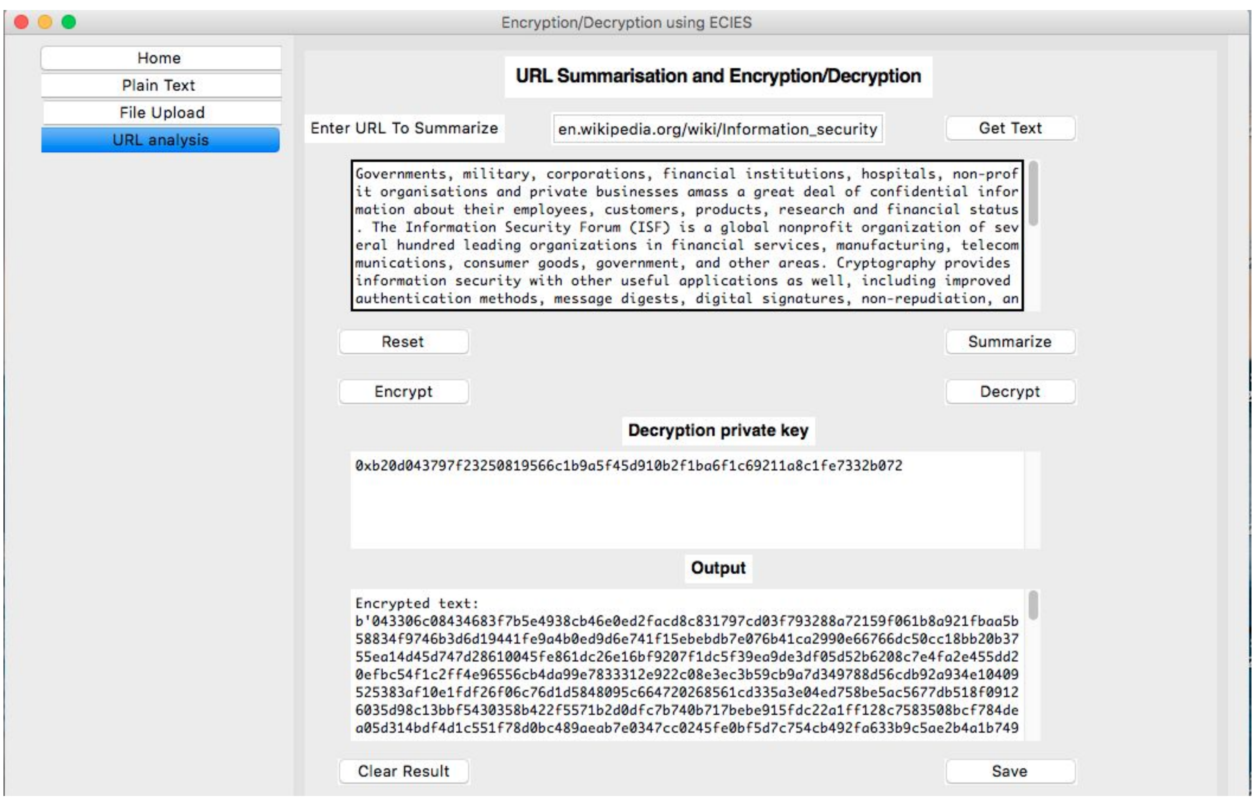
Summarisation on the URL analysis Tab

When the Summarize button is clicked the summarised text gets displayed in the output text box. This can further be encrypted by copying it into the input box and clicking on the Encrypt button.



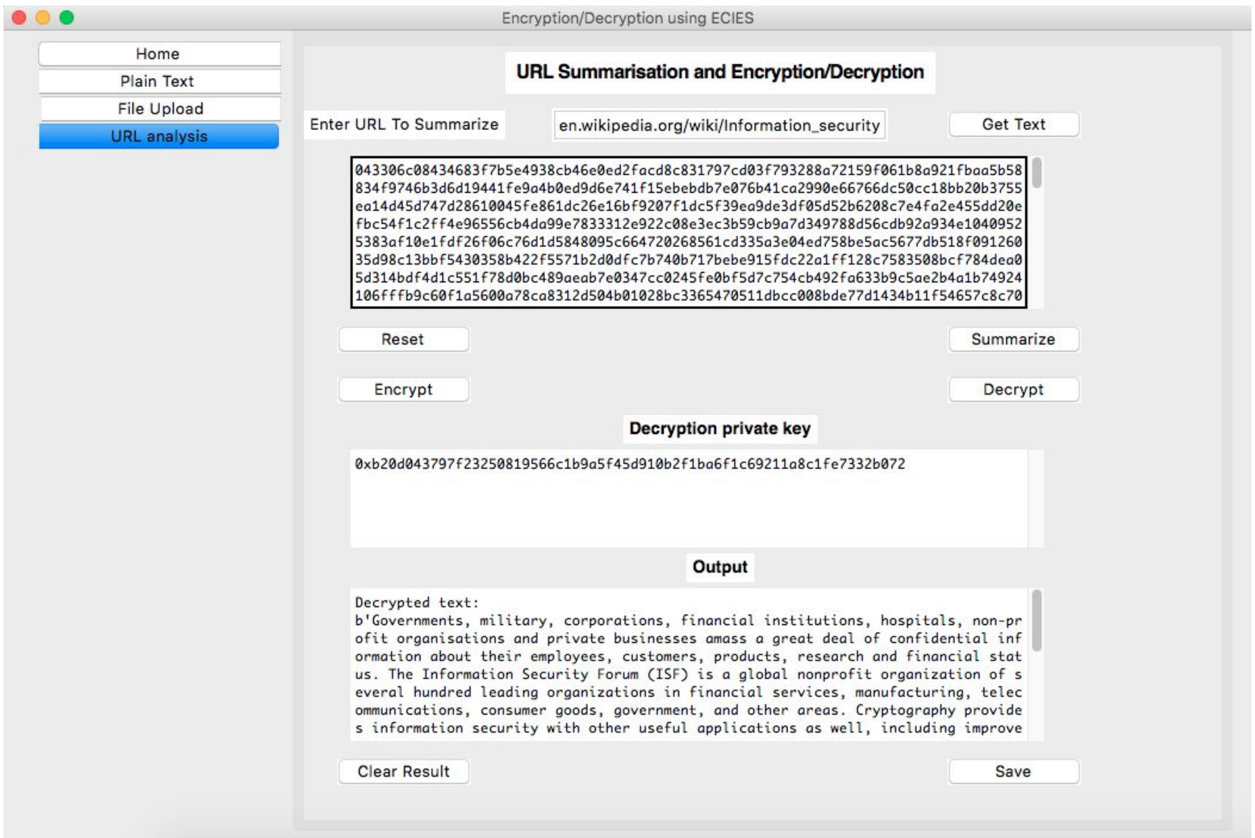
Encryption on the URL analysis Tab

Upon clicking the Encrypt button, the encrypted text gets displayed in the Output text box. The Decryption private key gets displayed in the corresponding textbox which needs to be noted down and inputted when performing decryption.



Decryption on the URL analysis Tab

A URL containing encrypted text is inputted and the decryption private key that was noted down earlier is inputted as well. The Decrypt button performs decryption and the Output is displayed in the corresponding textbox.



9. Conclusion

Documents which would have taken large amounts of time to read are summarised to include keywords / details to convey the original message in a shorter, more readable form. This saves time spent reading unnecessary details and memory as well.

This document is then encrypted using highly secure ECC algorithm to ensure that the CIA triad principles are upheld, i.e. The contents of the document are confidential, cannot be tampered with and is available only to authorized users when they need it.

10. Plagiarism Report

10/29/2020 Amritanshi Saxena 18BCE2524 - Plag check

Originality report

COURSE NAME
check

STUDENT NAME
Amritanshi Saxena 18BCE2524

FILE NAME
Amritanshi Saxena 18BCE2524 - Plag check

REPORT CREATED
Oct 29, 2020

Summary

Flagged passages	0	0%
Cited/quoted passages	0	0%