Activity based

Project Report on

Digital Forensics

C2P2 Project Module - II

Submitted to Vishwakarma University, Pune

Under the Initiative of

Contemporary Curriculum, Pedagogy, and Practice (C2P2)

By

Ammar Arsiwala

SRN No : 202101224

Roll No : 58

Div : C

Last Year Engineering

Faculty Incharge:- Seema Vanjire

Date Of Project 2:- 17/10/24

Department of Computer Engineering

Faculty of Science and Technology

Academic Year

Digital forensics: Project II

Project Name : Encrypted Communication Analysis Tool

Problem Statement:

With the rise in digital communication, ensuring data security and privacy has become crucial. The tool aims to assist users in encrypting and analyzing encrypted text using classical encryption methods such as Caesar and Vigenère ciphers. By providing a user-friendly interface for both encryption and decryption, this tool helps users understand how these ciphers work and how to analyze encrypted communications, enhancing data security awareness

Introduction:

In today's digital era, protecting sensitive information during communication is more important than ever. Encryption plays a vital role in ensuring data privacy and security by converting readable information into an unreadable format that can only be deciphered by authorized users. This project introduces an encrypted communication analysis tool that allows users to encrypt and decrypt messages using classical encryption techniques such as the Caesar and Vigenère ciphers. By offering an intuitive interface, the tool aims to make the process of encryption and decryption easy to understand, helping users strengthen their knowledge of cryptography and secure communication practices.

Related Study:

1. Tattersall, James J. *Elementary Number Theory in Nine Chapters*. Cambridge University Press, 1999.
2. Al-Kadi, Ibrahim A. “The Origins of Cryptology: The Arab Contributions.” *Cryptologia* 16, no. 2 (1992): 97-126.
3. Trappe, W., and C. Washington. *Introduction to Cryptography with Coding Theory*. New Jersey: Prentice Hall, 2002.
4. Kahn, David. *The Codebreakers: The Story of Secret Writing*. Macmillan, 1967.

5. Sahinaslan, Ender, and Onder Sahinaslan. “Cryptographic Methods and Development Stages Used Throughout History.” *AIP Conference Proceedings* 2086, no. 1 (2019): 030033.

Algorithm:

1. **Input Encrypted Text**: Ask the user to input an encrypted text string.

2. **Frequency Analysis**:

    - Count the frequency of each letter in the encrypted text.
    - Calculate the percentage of occurrences for each letter and display the analysis.

3. **Caesar Cipher Decryption**:

    - If the user opts for Caesar decryption, apply a brute-force approach by shifting the text for all possible 26 shift values and display the results.
    - Allow the user to choose the correct shift value and decrypt the text.

4. **Vigenère Cipher Decryption**:

    - If the user selects Vigenère decryption, ask for a key.
    - Decrypt the text using the provided key by shifting each character based on the corresponding character in the key.

5. **Display Results**: Output the decrypted text after applying either Caesar or Vigenère decryption based on user choice.

Mathematical model

1. Caesar Cipher (Decryption) Model:

For each character ccc in the encrypted text: D(c)=(E(c)−shift)mod 26D(c) = (E(c) - \text{shift}) \mod 26D(c)=(E(c)−shift)mod26 Where:

- E(c)E(c)E(c) is the character’s ASCII code (adjusted by 65 for uppercase, or 97 for lowercase letters),
- shift\text{shift}shift is the Caesar shift value (0 to 25),

- D(c)D(c)D(c) is the decrypted character.

2. Vigenère Cipher (Decryption) Model:

For each character cic_ici in the encrypted text: D(ci)=(E(ci)−E(ki))mod 26D(c_i) = (E(c_i) - E(k_i)) \mod 26D(ci)=(E(ci)−E(ki))mod26 Where:

- E(ci)E(c_i)E(ci) is the ASCII code of the encrypted character cic_ici,
- E(ki)E(k_i)E(ki) is the ASCII code of the corresponding key character kik_iki,
- D(ci)D(c_i)D(ci) is the decrypted character.

Code

```
from collections import Counter


def caesar_decrypt(text, shift):
    decrypted_text = ""
    for char in text:
        if char.isalpha():
            shift_amount = 65 if char.isupper() else 97
            decrypted_text += chr((ord(char) - shift_amount - shift) % 26 +
shift_amount)
        else:
            decrypted_text += char
    return decrypted_text


def frequency_analysis(text):
    counter = Counter(char for char in text if char.isalpha())
    total_chars = sum(counter.values())
    freq_percentages = {char: (count / total_chars) * 100 for char, count in
counter.items()}

    print("\nFrequency analysis of encrypted text:")
    for char, freq in sorted(freq_percentages.items(), key=lambda x: -x[1]):
        print(f"{char}: {freq:.2f}%")


def brute_force_caesar(text):
    print("\nBrute-forcing all possible Caesar shifts:")
    for shift in range(26):
        print(f"\nShift by {shift}:")
        print(caesar_decrypt(text, shift))
```

```
def vigenere_decrypt(text, key):
    decrypted_text = []
    key_length = len(key)
    key_as_int = [ord(i) for i in key]
    text_as_int = [ord(i) for i in text]

    for i in range(len(text_as_int)):
        if chr(text_as_int[i]).isalpha():
            shift_amount = 65 if chr(text_as_int[i]).isupper() else 97
            value = (text_as_int[i] - shift_amount - (key_as_int[i % key_length] -
shift_amount)) % 26
            decrypted_text.append(chr(value + shift_amount))
        else:
            decrypted_text.append(chr(text_as_int[i]))

    return ''.join(decrypted_text)


def main():

    encrypted_text = input("Enter the encrypted text: ")


    frequency_analysis(encrypted_text)


    print("\nDo you want to try Caesar cipher decryption?")
    try_caesar = input("Enter 'yes' to proceed, or 'no' to skip: ").lower()

    if try_caesar == 'yes':

        brute_force_caesar(encrypted_text)
        shift_value = int(input("\nEnter the shift value you think is correct: "))
        decrypted_caesar_text = caesar_decrypt(encrypted_text, shift_value)
        print(f"\nDecrypted text using Caesar cipher (shift by {shift_value}):")
        print(decrypted_caesar_text)
    else:
        print("Skipping Caesar cipher decryption.")


    print("\nDo you want to try Vigenère cipher decryption?")
    try_vigenere = input("Enter 'yes' to proceed, or 'no' to skip: ").lower()

    if try_vigenere == 'yes':
        key = input("Enter the Vigenère cipher key: ")
        decrypted_vigenere_text = vigenere_decrypt(encrypted_text, key)
        print(f"\nDecrypted text using Vigenère cipher (key: {key}):")
        print(decrypted_vigenere_text)
```

```
    else:
        print("Skipping Vigenère cipher decryption.")


if __name__ == "__main__":
    main()
```

Execution Steps:

1. **User Input**:
   - The program starts by asking the user to enter an encrypted text string.
2. **Frequency Analysis**:
   - The program performs frequency analysis of the encrypted text, counting the occurrences of each letter and calculating their frequency percentages.
3. **Caesar Cipher Decryption**:
   - The user is asked whether they want to try Caesar cipher decryption.
   - If the user opts for it:
     - The program brute-forces all possible Caesar cipher shifts (0 to 25), displaying each potential decrypted text.
     - The user then chooses a shift value, and the program decrypts the text using that shift.
4. **Vigenère Cipher Decryption**:
   - The user is asked whether they want to try Vigenère cipher decryption.
   - If the user opts for it:
     - They provide a key, and the program decrypts the text using the Vigenère cipher algorithm.

5. **Display Results**:

   - The program displays the decrypted text for both Caesar and Vigenère ciphers (if applicable).

6. **End**:

   - The program ends after performing the selected decryption operations.

Result Analysis :

- **Frequency Analysis**:

  ❖ The frequency distribution of letters in the encrypted text is calculated and displayed. This analysis helps in recognizing common patterns and might hint toward the decryption key.

- **Caesar Cipher Decryption**:

  ❖ Brute-forcing all Caesar cipher shifts provides 26 possible decrypted texts. The user selects a shift based on readable output, and the correct decryption is produced if the right shift is chosen.

- **Vigenère Cipher Decryption**:

  ❖ Using the provided key, the program attempts to decrypt the text. The accuracy depends on the correct input key.

- **Outcome**:

  ❖ If the text becomes readable after decryption attempts, it indicates the cipher was correctly cracked. The comparison of decrypted text with known patterns can confirm accuracy.

Screen Images Of Project:

Simple Caesar Cipher Text:

```
Enter the encrypted text: Uifsf jt b tfdsfu dpef!

Frequency analysis of encrypted text:
f: 27.78%
s: 11.11%
t: 11.11%
d: 11.11%
U: 5.56%
i: 5.56%
j: 5.56%
b: 5.56%
u: 5.56%
p: 5.56%
e: 5.56%

Do you want to try Caesar cipher decryption?
Enter 'yes' to proceed, or 'no' to skip: yes

Brute-forcing all possible Caesar shifts:

Shift by 0:
Uifsf jt b tfdsfu dpef!

Shift by 1:
There is a secret code!
```

Text with Non-Alpha Characters:

```
Enter the encrypted text: Khoor Zruog 123!

Frequency analysis of encrypted text:
o: 30.00%
r: 20.00%
K: 10.00%
h: 10.00%
Z: 10.00%
u: 10.00%
g: 10.00%

Do you want to try Caesar cipher decryption?
Enter 'yes' to proceed, or 'no' to skip: yes

Brute-forcing all possible Caesar shifts:

Shift by 0:
Khoor Zruog 123!

Shift by 1:
Jgnnq Yqtnf 123!

Shift by 2:
Ifmmp Xpsme 123!

Shift by 3:
Hello World 123!

Shift by 4:
Gdkkn Vnqkc 123!

Shift by 5:
Fcjjm Umpjb 123!
```

Vigenère Cipher Text:

```
PS D:\LY SEM 1\DF C2P2\analysis> python -u "d:\LY SEM 1\DF C2P2\analysis\hello_vr5.py"
Enter the encrypted text: Bipuls X!

Frequency analysis of encrypted text:
B: 14.29%
i: 14.29%
p: 14.29%
u: 14.29%
l: 14.29%
s: 14.29%
X: 14.29%

Do you want to try Caesar cipher decryption?
Enter 'yes' to proceed, or 'no' to skip: no
Skipping Caesar cipher decryption.

Do you want to try Vigenère cipher encryption and decryption?
Enter 'yes' to proceed, or 'no' to skip: yes
Enter the Vigenère cipher key: KEY

Encrypted text using Vigenère cipher (key: KEY):
Lghyjk H!

Decrypted text using Vigenère cipher (key: KEY):
Bipuls X!
```

Text with No Shift for Caesar:

```
Enter the encrypted text: Hello World!

Frequency analysis of encrypted text:
l: 30.00%
o: 20.00%
H: 10.00%
e: 10.00%
W: 10.00%
r: 10.00%
d: 10.00%

Do you want to try Caesar cipher decryption?
Enter 'yes' to proceed, or 'no' to skip: yes

Brute-forcing all possible Caesar shifts:

Shift by 0:
Hello World!
```

Brute Force Without Correct Shift:

```
Enter the encrypted text: Tqxxa Iadxp!

Frequency analysis of encrypted text:
x: 30.00%
a: 20.00%
T: 10.00%
q: 10.00%
I: 10.00%
d: 10.00%
p: 10.00%

Do you want to try Caesar cipher decryption?
Enter 'yes' to proceed, or 'no' to skip: yes

Brute-forcing all possible Caesar shifts:

Shift by 0:
Tqxxa Iadxp!

Shift by 1:
Spwwz Hzcwo!

Shift by 2:
Rovvy Gybvn!

Shift by 3:
Qnuux Fxaum!

Shift by 4:
Pmttw Ewztl!

Shift by 5:
Olssv Dvysk!
```

```
Shift by 6:
Nkrru Cuxrj!

Shift by 7:
Mjqqt Btwqi!

Shift by 8:
Lipps Asvph!

Shift by 9:
Khoor Zruog!

Shift by 10:
Jgnnq Yqtnf!

Shift by 11:
Ifmmp Xpsme!

Shift by 12:
Hello World!

Shift by 13:
Gdkkn Vnqkc!

Shift by 14:
Fcjjm Umpjb!

Shift by 15:
Ebiil Tloia!

Shift by 16:
Dahhk Sknhz!
```

```
Shift by 17:
Czggj Rjmgy!

Shift by 18:
Byffi Qilfx!

Shift by 19:
Axeeh Phkew!

Shift by 20:
Zwddg Ogjdv!

Shift by 21:
Yvccf Nficu!

Shift by 22:
Xubbe Mehbt!

Shift by 23:
Wtaad Ldgas!

Shift by 24:
Vszzc Kcfzr!

Shift by 25:
Uryyb Jbeyq!
```

Conclusion

In conclusion, the encrypted communication analysis tool effectively combines frequency analysis, Caesar cipher brute-forcing, and Vigenère cipher decryption techniques to uncover hidden messages within encrypted texts. The process provides flexibility in decryption by allowing users to test various cipher methods, enhancing the chances of identifying the correct key. By analyzing letter frequencies and offering brute-force decryption, the tool simplifies complex cryptographic challenges, making it a valuable asset for understanding and breaking simple encryption methods in real-world scenarios.