Assignment 1: Advanced Cryptography and Cryptanalysis (COSC5196)

Mihirkumar Mistry (Student ID: 249419480)
Divkumar Patel (Student ID: 249417620)
Group Number: 11

October 2024

Introduction

This report outlines the development of three task: one for calculating the Greatest Common Divisor (GCD), another for decrypting a Caesar cipher, and a third for executing a Caesar cipher brute force attack. Each program serves to illustrate fundamental concepts in mathematics and cryptography. The GCD program explores efficient methods for determining the largest divisor common to two numbers, while the Caesar cipher programs delve into the basics of encryption and decryption, revealing how simple ciphers can be compromised. Through this assignment, we aim to enhance our understanding of algorithmic design.

Task-1: GCD (Greatest Common Divisor)

An integer that divides two numbers without leaving a remainder is known as their greatest common divisor (GCD). In other words, given two integers, ***a*** and ***b***, the GCD is the largest integer that divides both ***a*** and ***b*** exactly. This is typically expressed as ***GCD(a, b)***.[1]

Description / Steps

- First we will take the two integer input from user (number1, number2).
- Then, as per the Euclidean algorithm, program will check the number2 is greater then the number1. If yes, then program will perform the swap operation.
- After that, the recursive function get_gcd will executed to find the value of the GCD. This function will call recursively till the remained is 0.
- At the end, a print function will print the output in console.

GCD Program (Python)

```python
"""
Created on Sun Oct  6 23:55:27 2024
@author: Divkumar Patel
Student ID: 249417620
Group Number: 11
"""

# Take integer number1 from user
number1 = int(input('Enter integer number1:'))
# Take integer number2 from user
number2 = int(input('Enter integer number2:'))

# Step 1: Check if number2 > number1
if number2 > number1:
    # Swap number1 and number2 value
    temp = number1
    number1 = number2
    number2= temp

# Calcualte GCD function
def get_gcd(number1, number2):
    return number2 == 0 and number1 or get_gcd(number2, number1 %
    number2)

# Call function to calculate GCD
gcd = get_gcd(number1, number2)

# Print gcd value
print(f"GCD of {number1} and {number2} is: {gcd}.")
```

Output

```
In [1]: runfile('C:/Users/divpa/OneDrive/Documents/GCD.py', wdir='C:/Users/divpa/OneDrive/Documents')
Enter integer number1:710
Enter integer number2:310
GCD of 710 and 310 is: 10.

In [2]: runfile('C:/Users/divpa/OneDrive/Documents/GCD.py', wdir='C:/Users/divpa/OneDrive/Documents')
Enter integer number1:23210
Enter integer number2:22385
GCD of 23210 and 22385 is: 55.

In [3]:
```

Figure 1: Output of GCD

- The output of the GCD(710,310) is 10.
- The output of the GCD(23210,22385) is 55.

Task-2: Caesar Cipher Decryption Algorithm

A Caesar cipher is a replacement encryption technique. Which is basic encryption technique that generates coded text by shifting all the letters in a message by a certain amount of positions in the alphabet.[2][3]

Description / Steps

- First, we will take the a cipher text and a key inputs from the user.
- To decode the cipher text, program will convert the alphabet to it's appropriate ASCII value.
- Then with the help of a character shifting technique, program will shift the character as per the key value.
- The final result or the plain text will be printed in the console.

Caesar Cipher Decryption Program (Python)

```
"""
Created on Mon Oct  7 00:28:03 2024
@author: Mihirkumar Mistry
Student ID: 249419480
Group Number: 11
"""
# Take cipher text from the user
cipher_text = input('Enter cipher text:')
# Take key value from the user
key = int(input('Enter key:'))

# Caesar cipher decryption function
def caesar_cipher_decryption(cipher_text, key):
    plain_text = ""

    for char in cipher_text:
        # Check if the char is a alphabet or not
        if char.isalpha():
            # Get the ASCII value of the base char, based on the case
            start = ord('A') if char.isupper() else ord('a')
            # Finding the plaintext using character shifting algorithm
            decrypted_char = chr((ord(char) - start - key) % 26 + start)
            # Add the resulting char
            plain_text += decrypted_char
        else:
            # Keep the Non-alphabet character as it is
            plain_text += char

    return plain_text

# Call caesar_cipher_decryption with user input
```

```
print('Plaintext:',caesar_cipher_decryption(cipher_text, key))
```

Output

```
In [1]: runfile('D:/Masters/ACAC/Assignment 1/caeser-cipher.py', wdir='D:/Masters/ACAC/Assignment 1')
Enter cipher text:ez oz zc yze ez oz esle td esp bfpdetzy
Enter key:11
Plaintext: to do or not to do that is the question

In [2]:
```

Figure 2: Output Caesar Cipher Decryption

Overall, for the ***cipher text:*** *"ez oz zc yze ez oz esle td esp bfpdetzy"* and ***key:*** *11*, program gives ***plain text:*** *"to do or not to do that is the question"*.

Task-3: Caesar Cipher Brute Force Attack

In a Caesar cipher brute force attack, the aim is to decrypt the original message by trying every possible letter shift. Since the English alphabet has 26 letters, there are typically 25 shifts to test. The Caesar cipher is simple, making it vulnerable to brute force and frequency analysis, where you compare the most common letters in the encrypted text to those frequently used in the language. Because of this simplicity, brute-force attacks are often very effective at cracking the code.[2][3]

Description / Steps

- First, we will take the a cipher text input from the user.
- To decode the cipher text, program will use all possible key (from 1 to 25).
- The program will use the same Caesar cipher decryption logic. It will print the decrypted text for each iteration it perform.
- We can easily identify the correct plain text by looking at output.

Caesar Cipher Brute Force Attack Program (Python)

```
"""
Created on Mon Oct  7 00:55:03 2024
@author1: Mihirkumar Mistry (Student ID: 249419480)
@author2: Divkumar Patel (Student ID: 249417620)
Group Number: 11
```

```
"""
# Take cipher text from the user
cipher_text = input('Enter cipher text:')

# Caesar cipher attack function
def caesar_cipher_brute_force_attack(cipher_text):
    # Loop through 1 to 26 keys
    for key in range(1, 26):
        plain_text = ""

        for char in cipher_text:
            # Check if the char is a alphabet or not
            if char.isalpha():
                # Get the ascii value of the base char, based on the case
                start = ord('A') if char.isupper() else ord('a')
                # Finding the plaintext using character shifting algorithm
                decrypted_char = chr((ord(char) - start - key) % 26 + start)
                # Add the resulting char
                plain_text += decrypted_char
            else:
                # Keep the Non-alphabet character as it is
                plain_text += char

        print(f"Key: {key}, Plaintext: {plain_text}")


# Call caesar_cipher_decryption with user input
caesar_cipher_brute_force_attack(cipher_text)
```

Output

```
In [1]: runfile('D:/Masters/ACAC/Assignment 1/caesar-cipher-attack.py', wdir='D:/Masters/ACAC/Assignment 1')
Enter cipher text:jryy qbar v nccergvngr vg
Key: 1, Plaintext: iqxx pazq u mbbdqfumfq uf
Key: 2, Plaintext: hpww ozyp t laacpetlep te
Key: 3, Plaintext: govv nyxo s kzzbodskdo sd
Key: 4, Plaintext: fnuu mxwn r jyyancrjcn rc
Key: 5, Plaintext: emtt lwvm q ixxzmbqibm qb
Key: 6, Plaintext: dlss kvul p hwwylaphal pa
Key: 7, Plaintext: ckrr jutk o gvvxkzogzk oz
Key: 8, Plaintext: bjqq itsj n fuuwjynfyj ny
Key: 9, Plaintext: aipp hsri m ettvixmexi mx
Key: 10, Plaintext: zhoo grqh l dssuhwldwh lw
Key: 11, Plaintext: ygnn fqpg k crrtgvkcvg kv
Key: 12, Plaintext: xfmm epof j bqqsfujbuf ju
Key: 13, Plaintext: well done i appretiate it
Key: 14, Plaintext: vdkk cnmd h zooqdshzsd hs
Key: 15, Plaintext: ucjj bmlc g ynnpcrgyrc gr
Key: 16, Plaintext: tbii alkb f xmmobqfxqb fq
Key: 17, Plaintext: sahh zkja e wllnapewpa ep
Key: 18, Plaintext: rzgg yjiz d vkkmzodvoz do
Key: 19, Plaintext: qyff xihy c ujjlyncuny cn
Key: 20, Plaintext: pxee whgx b tiikxmbtmx bm
Key: 21, Plaintext: owdd vgfw a shhjwlaslw al
Key: 22, Plaintext: nvcc ufev z rggivkzrkv zk
Key: 23, Plaintext: mubb tedu y qffhujyqju yj
Key: 24, Plaintext: ltaa sdct x peegtixpit xi
Key: 25, Plaintext: kszz rcbs w oddfshwohs wh

In [2]:
```

Figure 3: Output Of Caesar Cipher Brute Force Attack

Overall, for the ***cipher text:*** *"jryy qbar v nccergvngr vg"*, program gives ***plain text:*** *"well done i appretiate it"* at ***key value:*** *13*. So, we can say that 13 is a correct key for this encrypted message.

Acknowledgement

- **Task-1: GCD (Greatest Common Divisor):**
  - Divkumar Patel (Student Id: 249417620)
- **Task-2: Caesar Cipher Decryption Algorithm:**
  - Mihirkumar Mistry (Student Id: 249419480)
- **Task-3: Caesar Cipher Attack:**
  - Mihirkumar Mistry (Student Id: 249419480)
  - Divkumar Patel (Student Id: 249417620)
- **Assignment Report:**
  - Mihirkumar Mistry (Student Id: 249419480)
  - Divkumar Patel (Student Id: 249417620)

Conclusion

In this assignment, we created three programs: one to find the Greatest Common Divisor (GCD), another to decrypt a Caesar cipher, and a third to perform a Caesar cipher brute force attack.

The GCD program used the Euclidean algorithm, demonstrating how efficiently we can find the GCD of two numbers. The decryption program showed how to reverse the Caesar cipher, highlighting the connection between encryption and decryption. Lastly, the Caesar cipher attack program revealed the weaknesses of simple ciphers by brute force attack.

Overall, this assignment helped us understand important concepts in algorithms and cryptography, emphasizing the need for stronger security in today's digital world.

References

[1] William Stallings. *Cryptography and network security: principles and practice, 7th Edition*. Pearson, 2017.

[2] given=GeeksforGeeks given i=G. Caesar cipher in cryptography.

[3] Tutorialspoint. Cryptography with python - caesar cipher.