1

Lab

**REPORT OF PRATICE LAB 1**

**Lab 1**

**Subject: Algorithms and Processes in comp-security**

**Class: CS4243.O11.CTTT**

|  |  |
| --- | --- |
| **Giảng viên hướng dẫn** | Lecturer: Nguyễn Xuân Hà |
| **Sinh viên thực hiện** | Phạm Thùy Dung (20521214) |
| Lương Gia Hân (19521468) |
| **Mức độ hoàn thành** | Hoàn thành |
| **Thời gian thực hiện** | From 27-09-2023 to 12-10-2023 |
| **Tự chấm điểm** | 9/10 |

* 1. **CÁC BƯỚC THỰC HÀNH**

**Gợi ý:** *Ghi rõ từng bước thực hành, chụp hình ảnh screenshot để báo cáo thêm trực quan*

* 1. **TRẢ LỜI CÁC CÂU HỎI**

**Gợi ý:** *Trả lời câu hỏi đúng, đầy đủ, cần giải thích lý do tại sao có được đáp án, có các hình ảnh, bằng chứng để chứng minh tính đúng đắn.*

**Ví dụ:**

*Câu 1. Địa chỉ IP máy tính của bạn là gì?*

***Trả lời:*** *192.168.1.106*

Để xem địa chỉ IP của máy tính trên Windows, mở **Control Panel** và chọn **View network status and tasks.** Chọn mạng tương ứng đang sử dụng để kết nối Internet, chọn **Details** trong cửa sổ trạng thái. Xem địa chỉ IP trong Ipv4 Address



**C. BÀI TẬP LAB 1**

*Câu 1. Kickoff the code:*

*A blue background with black text

Description automatically generated*

*A black and white picture of a padlock

Description automatically generated*

***Answer:*** *the combination: 6 9 1, which is a valid solution*

The nature of the clues provided in this puzzle allows for multiple possible interpretations. If we don't approach them systematically, different answers can emerge.

Let's re-examine the clues:

1. 2 9 1: One number is correct and in the right place.

2. 2 4 5: One number is correct but in the wrong place.

3. 4 6 3: Two numbers are correct but in the wrong places.

4. 5 7 8: Nothing is correct.

5. 5 6 9: One number is correct but in the wrong places.

Starting with the most definitive information:

* Clue 4 clearly indicates that the numbers 5, 7, and 8 are not in the code, so we can immediately exclude them.
* From Clue 1, we can derive three potential interpretations:
* The first number is 2.
* The second number is 9.
* The third number is 1.
* Clue 2 suggests that one of the numbers 2, 4, or 5 is correct but not in its current position. Since we know from Clue 4 that 5 isn't in the code, the possible numbers are 2 or 4, but they are not in their current positions.
* Clue 3 implies that two of the numbers — 4, 6, or 3 — are correct but not in their current positions.
* From Clue 5, after excluding 5, we deduce that either 6 or 9 is correct but not in their shown positions.

Now, if we make an assumption based on Clue 1 that the second number is 9, we can deduce:

* The first number can't be 5, 7, 8, 9, 2, 4, or 3. This leaves us with 6 as the first number.
* The second number is 9 (from our assumption).
* For the third number, considering it can't be 5, 7, 8, 9, or 6, and given that 4 isn't in the third position based on Clue 3, it must be 1.

=> This gives us the combination: 6 9 1, which is a valid solution.

A number with stars and text

Description automatically generated with medium confidence

A grid with colorful circles and stars

Description automatically generated

Let replace simple in a set of star to ( .Then, transform the table 1 by replacing the star to

Here is my example:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

After that, we have equation:

For the rows, we get:

* =

For the columns, we get:



Because there isn't any specific number in the equation, there's a high likelihood that there are multiple answers to this exercise. Therefore, I will assume some results for the equations without question marks to gradually figure it out.

First, I will highlight the equations that don't contain question marks. Then, I will assume some results to derive answers for the remaining unknowns. It's like a trial. If the trial is incorrect, I will assume a different hypothesis.

In three highlighted equations, there are 7 variables. They includes: . So, I assume, , , , and .

Let’s replace them into equations:

12 + = 36

*Exercise 2. Caeser cipher*

A screenshot of a computer program

Description automatically generated

***Answer:*** *There are two kinds of people in this world: those who are looking for a reason and those who are finding success. Those who are looking for a reason always seeking the reasons why the work is not finished. And people who find success are always looking for reasons why the work can be completed.*

I will use Python to solve this problem:

Here is my code:

Base on the tips give me above, we will have:

1. **encrypt(plain\_text, key)**:
   * **Purpose**: Encrypts a given plain text using the Caesar cipher with a provided key.
   * **Parameters**:
     + **plain\_text**: The text that needs to be encrypted.
     + **key**: An integer between 0 and 25, representing the number of positions each letter in the plain text should be shifted to get the encrypted text.
   * **How it works**: The function goes through each character in the **plain\_text**. If the character is a letter (either lowercase or uppercase), it's shifted by the given **key**. If it's any other character (like punctuation or whitespace), it's added to the encrypted text as-is.

A screen shot of a computer program

Description automatically generated

1. **decrypt(cipher\_text, key)**:
   * **Purpose**: Decrypts a given cipher text encrypted using the Caesar cipher with a provided key.
   * **Parameters**:
     + **cipher\_text**: The encrypted text that needs to be decrypted.
     + **key**: The Caesar cipher key used for encryption.
   * **How it works**: This function works similarly to **encrypt**, but it shifts the characters backwards (or subtracts the key) to revert the encryption.

A screen shot of a computer program

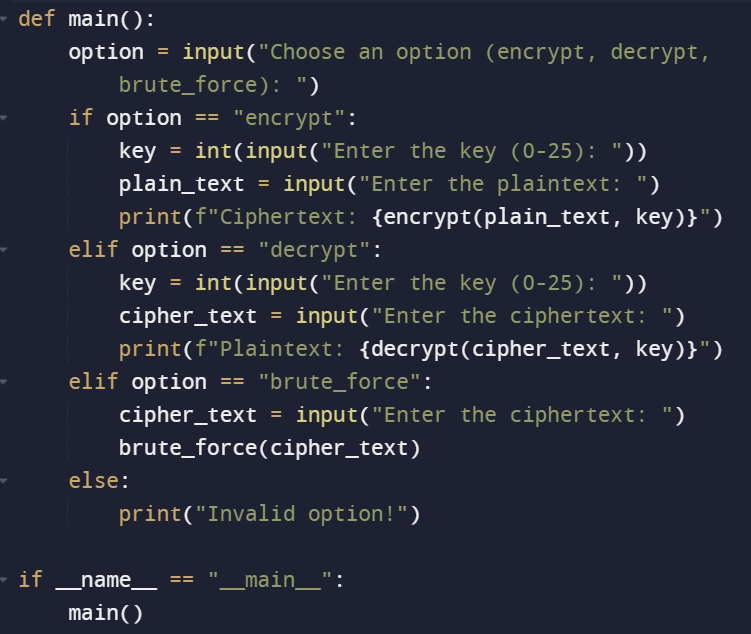
Description automatically generated

1. **brute\_force(cipher\_text)**:
   * **Purpose**: Attempts to decrypt the given cipher text using all possible Caesar cipher keys (from 1 to 25).
   * **Parameter**:
     + **cipher\_text**: The encrypted text that we are trying to brute force decrypt.
   * **How it works**: The function loops over each possible key and prints out the decrypted text using that key. This is helpful if you don't know the encryption key and want to see potential plaintexts.

A computer code with white text

Description automatically generated

1. **main()**:
   * **Purpose**: The main driver function that provides an interface for the user to interact with the encryption, decryption, and brute force functionalities.
   * **How it works**: The function prompts the user to choose an option (encrypt, decrypt, or brute\_force). Depending on the option chosen, the function will prompt for further inputs (like key, plaintext or ciphertext) and then call the corresponding function to produce and display the result.
2. **if \_\_name\_\_ == "\_\_main\_\_": main()**:
   * **Purpose**: Ensures that the **main()** function is executed when the script is run as a standalone file, but not if the script is imported as a module in another file.



Output:

Using brute\_force because:

* + The function loops over each possible key and prints out the decrypted text using that key. This is helpful if you don't know the encryption key and want to see potential plaintexts.

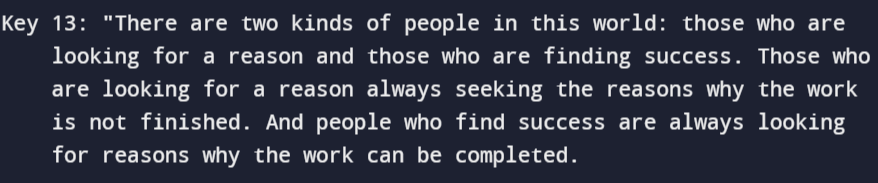


Then, enter the ciphetext:

A screenshot of a computer

Description automatically generated

Then, we have result in key 13



*Exercise 3.* *Mono-alphabetic substitution cipher and frequency analysis*

*A close-up of a text

Description automatically generated*

Answer:

**Step 1**: Understand English letter frequencies. In typical English text:

* The most common letters are **E, T, A, O, I, N, S, H, R, D, L, U**.
* The most common bigrams (two-letter combinations) are **th, he, in, er, an, re, es, on, at, en, nd, ti, ea, or, te, of, ed, is, it, al, ar, st, to, wa, ha, ng, la, le, li, ve, co, me, de, hi, ri, ro, ic, ne, nt, ea, ra, ce, lo, ch, ma, el, ly, cl, day**.
* The most common trigrams (three-letter combinations) are **the, and, tha, ent, ion, tio, for, nde, has, nce, edt, tis, oft, sth, men**.

**Step 2**: Check the frequency of letters in the given ciphertext. I will not calculate each frequency here due to the length, but for instance, by skimming through the text, **y**, **n**, **x**, **v**, **q**, **m**, **u**, **h**, **z**, **t**, and **l** seem to be pretty frequent.

**Step 3**: Start with the most frequent letters and see if they can be matched with English frequencies.

For instance, if **y** appears most frequently in the ciphertext, there's a chance **y** could be **e** in plaintext.

**Step 4**: Look for common patterns. For instance, the word **ytn** appears often. It might stand for **the**. If that's the case, then **y=e, t=t, n=h**. You can use this guess to decrypt parts of the text and see if it makes sense.

**Step 5**: Continue to decrypt the text using the guesses made in the previous steps. If a substitution doesn't make sense, backtrack and try a different guess.

Given the constraints, let's try to solve this using a few initial assumptions based on common patterns:

Using the assumption that **ytn = the**, we get: **y=e, t=t, n=h**.

Let's replace these letters and see what we get. But , it takes to long to find the result , so I will use file to get the answer:

Here is step-by step:

1. Go to this website: <https://www.cryptool.org/en/cto/n-gram-analysis>
2. Enter the ciphertext to it:

A screenshot of a computer

Description automatically generated

Here is result:

A screenshot of a graph

Description automatically generated

A screenshot of a graph

Description automatically generated

With Rank 1, that means the frequency appear the most.

Look ad Rand with 3-gram, ytm appear the moste , and then compare to website: <http://norvig.com/mayzner.html>

A screenshot of a graph

Description automatically generated

Therefore I can have the decision

*Exercise 4. Playfair cipher*

*Exercise 5. Playfair cipher*

*A close-up of a message

Description automatically generated*

*Here is may code:*

*A screenshot of a computer code

Description automatically generated*

*A computer screen shot of a code

Description automatically generated*

*A screenshot of a computer program

Description automatically generated*

*Here is my output:*

*A screen shot of a computer

Description automatically generated*

*A screen shot of a computer screen

Description automatically generated*

**Here is my link of GitHub:**

**https://github.com/amenosakura/Sercirity1**