Hacettepe University Computer Engineering Department Information Security Lab.

Homework 1

Subject : Implementing and Breaking Basic Ciphers
Due Date : 23.10.2024
Language : Python
T.A.s : Ali Baran TAŞDEMİR, Sibel KAPAN

Introduction:

In this assignment, you will explore the classic ciphers by both implementing and breaking simple encryption schemes. Through this hands-on project, you will gain an understanding of basic cryptographic techniques like substitution and transposition. You will design functions to apply both encryption and decryption functions for various classic ciphers. And you will apply basic cryptanalysis techniques to break encrypted messages.

Your assignment is divided into two main parts;

- **Part 1: Cipher Implementation.** You will implement three classic ciphers—Caesar, Affine, and Monoalphabetic—and build functions that can both encrypt and decrypt messages using these ciphers.
- **Part 2: Cryptanalysis.** In this section, you will attempt to break encrypted messages using brute force and frequency analysis.

Part 1: Cipher Implementation (20)

In this part, you will write functions to implement encryption/decryption logic of given basic ciphers.

Caesar Cipher (5)

The first task is one of the simplest and widely known encryption methods, "Caesar's Cipher". The technique involves shifting the letters in plaintext by a fixed number of positions. The shifting can be represented by aligning two alphabets; the plain and the cipher alphabet.

Plain	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Cipher	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F

Table 1: Caesar cipher using left rotation 20 places (right rotation 6)

The encryption operation is performed by using a plain to cipher mapping. For example, to encrypt the word "Encryption", the letters of the word starting with "E" are replaced by the corresponding letter from the cipher list. If you use 20 as the shifting factor (Table 1) the resulting encrypted text will be; "Ktixevzout".

- The letter "E" which is the 5th letter in the plain alphabet will be replaced with the 5th letter in the cipher alphabet which is "K".
- The second letter "N", the 14th letter in the plain alphabet will be replaced with the 14th letter in the cipher alphabet, "T".
- ...

In this part, you will design two functions;

- `encrypt_caesar(plaintext, shift)`
- `decrypt_caesar(ciphertext, shift)`

Affine Cipher (5)

The affine cipher is a form of monoalphabetic substitution cipher in which each letter of the alphabet is first converted to its numerical equivalent, then encrypted using a mathematical formula, and finally converted back to a letter. This formula ensures that each letter corresponds to only one other letter, and vice versa, making it a basic substitution cipher governed by a specific rule for letter mapping. As a result, it shares the vulnerabilities of other substitution ciphers. The encryption is performed using the function $(ax + b) mod 26$, where b represents the shift magnitude.

The general formula of an affine cipher is,

$$E(x) = (ax + b) mod m$$

where m is the size of the alphabet, a and b are the keys. The value a must be chosen such that a and m are coprime (relatively prime). And b represents the shifting factor. *Since we are working with the English alphabet, the m will be chosen as 26.*

Plain	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Cipher	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Table 2: The English alphabet is represented as numerical values.

The encryption is performed by using the given formula. For this example, we pick $a = 3$ and $b = 5$. With the plaintext "ENCRYPTION" we will use the formula step-by-step;

- For the letter E, we use the numeric value (Table 2) 4. $E(4) = (3 * 4 + 5) mod 26$ results in $E(4) = 17$. And 17 is the numeric value of the letter R.
- For the letter N, we use the numeric value (Table 2) 13. $E(13) = (3 * 13 + 5) mod 26$ results in $E(13) = 18$. And 18 is the numeric value of the letter S.
- ...

After we calculate all letters, the ciphertext for "ENCRYPTION" will be "RSLEZYKDVS".

In this part, you will design two functions;

- `encrypt_affine(plaintext, a, b)`
- `decrypt_affine(ciphertext, a, b)`

Mono-alphabetic Substitution Cipher (10)

A mono-alphabetic substitution cipher is a type of cipher where each letter in the plaintext is replaced by a corresponding letter from a fixed substitution alphabet. The mapping between the letters remains consistent throughout the message, meaning each letter always substitutes for the same letter. While this method is simple and easy to implement, it is also vulnerable to frequency analysis, as the structure of the original text can be revealed by analyzing the frequency of letters in the ciphertext.

Plain	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Cipher	Q	W	E	R	T	Y	U	I	O	P	A	S	D	F	G	H	J	K	L	Z	X	C	V	B	N	M

Table 3: A substitution alphabet to encrypt plaintext with monoalphabetic substitution

The encryption is performed by using the map between plain and cipher alphabets. For example, when encrypting the plaintext ”Encryption”;

- For the letter E, we use the mapping and replace the letter E with T.
- For the letter N, we use the mapping and replace the letter N with F.
- ...

After applying mappings to all letters, the cipher text will be ”Tfeknhz”.

In this part, you will design two functions;

- `encrypt_mono(plaintext, key)`
- `decrypt_mono(ciphertext, key)`

Part 2: Cryptanalysis (70)

In this part of the assignment, you will use cryptanalysis techniques to break the ciphers that you implemented in the first part. We will use ciphertexts encrypted by Caesar, Affine and Mono-alphabetic substitution ciphers. We will use an English dictionary to help the decrypting process. You will use this dictionary to help you break the ciphers by comparing decrypted outputs to the words in the dictionary. The plaintexts used to generate ciphertext is created by the words in the dictionary.

You can download the dictionary from this link[1].

Note: You must validate your decrypted outputs from the dictionary and print it to an output file named `break_[mode].txt`. The file only include the decrypted text.

- **Caesar Cipher (15).** Write a function that brute-forces all possible shifts (1-25) and compares the outputs with words from the dictionary. `break_caesar(ciphertext)`
- **Affine Cipher (20).** Implement a brute-force method to try different values of a and b, and use the dictionary to check the decrypted message. `break_affine(ciphertext)`

[1] `https://drive.google.com/file/d/1F0AzVeRfD4QMvf4F7baZf6QMJ6Gd0R8f/view?usp=sharing`

- **Mono-alphabetic Cipher (35).** Write a function that uses the frequency of letters in the ciphertext and compares it to the frequency of letters in the English language. Use the dictionary to further validate the output. `break_mono(ciphertext)`

Implementation Details

Arguments

You will create two seperate Python scripts for each part. For the first part, you will create `ciphers.py`. The program must be executed by command line arguments. The arguments are listed below;

```
$python cipher.py [-s SHIFT] [-a A] [-b B] [-k KEY] cipher file mode
```

- *cipher* denotes the name of the cipher techniques. We implement three different ciphers in the first part. You will use "caesar" for the Caesar cipher, "affine" for the Affine cipher, and "mono" for the Mono-alphabetic cipher.
- $file$ denotes the name/path of the input file.
- *mode* denotes encryption and decryption. To encrypt input use *e*, to decrypt the input use *d*.
- s $SHIFT$ denotes the shifting amount for Caesar Cipher. (optional)
- $-a$ A denotes the a value for Affine Cipher. (optional)
- $-b$ B denotes the b value for Affine Cipher. (optional)
- k KEY denotes the key alphabet for Mono-alphabetic Cipher. (optional)

For the second part, you will create a Python script named `break.py`. The program must be executed by command line arguments. The arguments are listed below;

```
$python break.py cipher file
```

- *cipher* denotes the name of the cipher techniques. We implement three different ciphers in the first part. You will use "caesar" for the Caesar cipher, "affine" for the Affine cipher, and "mono" for the Mono-alphabetic cipher.
- $file$ denotes the name/path of the input file.

Examples:

Caesar Cipher encryption with shift value 13 and the input file "plain.txt".

```
$python cipher.py caesar plain.txt e -s 13
```

Mono-alphabet encryption with the key alphabet "QWERTYUIOPASDFGHJKLZXCVBNM" and the input file "plain.txt".

```
$python cipher.py mono plain.txt e -k QWERTYUIOPASDFGHJKLZXCVBNM
```

Affine encryption with $a = 3$ and $b = 5$ and the input file "plain.txt".

```
$python cipher.py affine plain.txt e -a 3 -b 5
```

Break the encrypted text in ”coded.txt” by using the Caesar Cipher attack.

```
$python break.py caesar coded.txt
```

Tips and Notes:

1. Do not use any library outside of Python Standard Library. If you think you have to, please ask before using it.
2. Your code must use the command line arguments described in the corresponding section. You can use Python's `argparse` library.
3. For simplicity, there will be no punctuation or numbers in plaintexts.
4. Your code efficiency will be included in the grading.
5. You must submit the homework in groups of two.
6. You should prepare a report that describes your approach to the problem with the details of your implementation. You must write down all group members' names and IDs. Reports will be graded too.
7. You can ask your questions about the homework via Piazza. (`https://piazza.com/hacettepe.edu.tr/fall2024/bbm465`)
8. T.A.s as himself has the right to partially change this document. However, the modifications will be announced in the Piazza system. In this case, it is your obligation to check the Piazza course page periodically.
9. You will submit your work via the submission system.
   (`www.submit.cs.hacettepe.edu.tr`)
   The submission format is given below:

```
→ <group id.zip>
    → src /ciphers.py
    → src /break.py
    → report.pdf
```

Policy

All work on assignments must be done with your own group unless stated otherwise. You are encouraged to discuss with your classmates about the given assignments, but these discussions should be carried out in an abstract way. That is, discussions related to a particular solution to a specific problem (either in actual code or in the pseudocode) will not be tolerated. In short, turning in someone else's work(from the internet), in whole or in part, as your own will be considered as a violation of academic integrity. Please note that the former condition also holds for the material found on the web as everything on the web has been written by someone else.