The Chinese University of Hong Kong

Department of Information Engineering

Summer Workshop – Fun with Information Engineering and Security

Lab 3 – Classic Ciphers

Introduction

Encryption is the process of transforming data into an unreadable form, to ensure confidentiality and security. Encryption algorithms use mathematical operations and cryptographic techniques to convert plaintext into ciphertext, making it difficult for unauthorized parties to decipher the original information.

In this lab, we will touch upon classical ciphers, examining traditional encryption methods that have shaped the field of cryptography. The session will cover key concepts related to encryption, and try to break them with some hands-on exercises.

Section 1: Basic concepts of cipher

Section 1.1: Random Number

Random numbers are essential in cryptography for various purposes, including key generation, initialization vectors (IVs), nonces, and random challenges. Randomness is crucial to ensure the security and unpredictability of cryptographic operations, preventing attackers from exploiting patterns or biases.

In computer systems, generating truly random numbers is challenging. Instead, pseudo-random number generators (PRNGs) are commonly used. PRNGs produce a sequence of numbers that appear random but are generated using a deterministic algorithm with an initial value called a seed.

Try it!

Copy the following code snippet to https://iesummerworkshop.github.io/py-lab.html coding section.

```
import random

print(random.randint(0,9))
```

By using `random.randint()` function, the program generates random numbers in the range 0 to 9.

1. Try to compile for few more times. Are the results the same?
2. How do we generate numbers in the range 1 to 100?

A cryptographically secure pseudorandom number generator is a type of random number generator that generates numbers that are suitable for use in cryptographic algorithms. Unlike regular PRNGs that are designed for speed and efficiency, cryptographically secure PRNGs prioritize security and unpredictability.

In cryptography, random numbers are crucial for various purposes such as generating

encryption keys, initialization vectors, and nonces. If these numbers are not truly random or predictable, it can lead to vulnerabilities and compromise the security of cryptographic systems

Try it!

Copy the following code snippet to https://iesummerworkshop.github.io/py-lab.html coding section.

```
import os

size = 5
result = os.urandom(size)

print(result)
```

Now we are using another random function `os.urandom()`. This is a Python module to generate a string of size random bytes. The return value of this function is suitable for cryptographic use since it is unpredictable enough.

The following is a sample output of the program:

```
b'\xd6\xa8\x04\x8a\x8c'
```

The letter 'b' represents a byte string, it is a sequence of bytes:

- \xd6 corresponds to the hexadecimal value D6
- \xa8 corresponds to the hexadecimal value A8
- \x04 corresponds to the hexadecimal value 04
- \x8a corresponds to the hexadecimal value 8A
- \x8c corresponds to the hexadecimal value 8C

Try to compile for few more times. Are the results the same?

Section 1.2: Padding

Padding refers to adding extra bits or bytes to the plaintext or ciphertext to ensure that it meets the required block size or alignment for a specific cryptographic algorithm. The need for padding arises when the length of the plaintext or ciphertext is not a multiple of the block size. For example, in base64 encoding, when we divide bits into 6-bit chunks if the last chunk is less than 6 bits, we will pad zeros at the end.

Try it!

Copy the following code snippet to https://iesummerworkshop.github.io/py-lab.html coding section.

```
str = "Pad" + "ding"

print(str)
print(str.rjust(20, "X"))
print(str.ljust(20, "O"))
```

`rjust()` and `ljust()` are method of the variable `str`. `rjust()` and `ljust()` correspond to right padding and left padding respectively.

1. What is the output? How many X and O are padded?
2. How to pad the string if the number of characters needs to be divisible by 5?

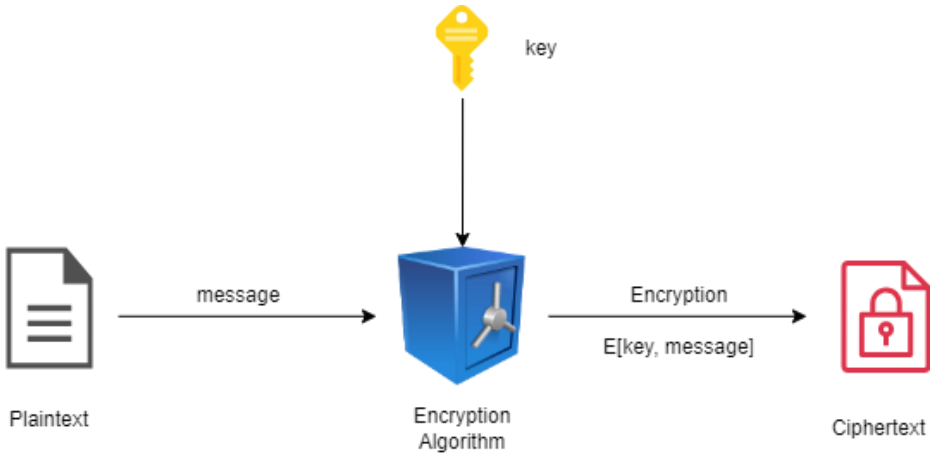
Section 2: Classic Cipher

Classic ciphers are encryption techniques that have been used historically to secure messages. These ciphers operate on a small set of characters or symbols and often involve simple substitution or transposition techniques. Although they may not provide strong security in modern contexts, classic ciphers offer an excellent introduction to the fundamental concepts of cryptography.

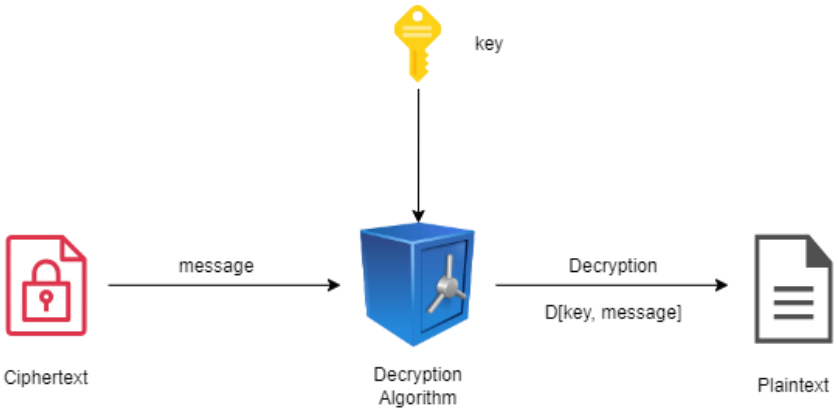
The following are some terminologies we use in cryptography:

- Plaintext (明文): the input of the encryption process. It is an original and readable message.
- Ciphertext (密文): the output of the encryption process. It is difficult for unauthorized individuals to understand or decrypt.
- Encryption (加密): the operation to transform plaintext to ciphertext
- Decryption (解密): the operation to transform ciphertext to plaintext
- Algorithm (演算法): a set of mathematical rules or procedures used to encrypt or decrypt data. It defines the steps and operations of encryption and decryption
- Key (金鑰): a piece of information used by an algorithm to control the transformation

Process of Encryption



Process of Decryption



Section 2.1: Encoding vs Encryption

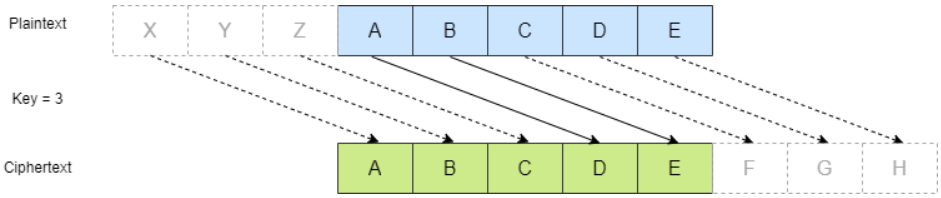
Although encoding and encryption describe a process of transforming data into another format, they are different in terms of purpose and security level. The following table compares the differences between encoding and encryption:

	Encoding	Encryption
Purpose	Represent data in a specific format	Protect data from unauthorized access
Security	No intentional security measures	Provides confidentiality
Reversibility	Reversible with decoding	Reversible with decryption
Data Transformation	Changes data representation	Transforms data into unreadable form
Key/Algorithm	No key required	Requires encryption key and algorithm
Data Protection	No protection against unauthorized access	Protects data from unauthorized access
Examples	Character encoding (ASCII, Unicode), Base64	AES, RSA, DES

Section 2.2: Caesar (shift) Cipher

The Caesar cipher is one of the simplest and most well-known substitution ciphers. It involves shifting the letters of the alphabet by a fixed number of positions, each letter is substituted with another letter from the alphabet.

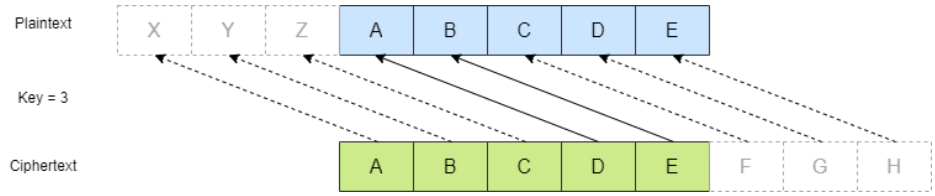
Encryption



With key = 3, the letter shifts to the **right** by 3 positions. Letter A becomes D, B becomes E … Y becomes B, and Z becomes C.

e.g. HELLO WORLD → KHOOR ZRUOG

Decryption

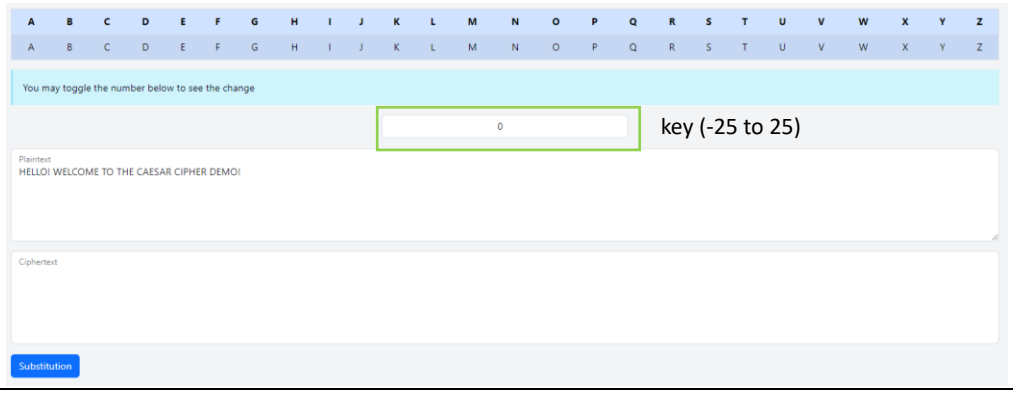


With key = 3, the letter shifts to the **left** by 3 positions. Letter D becomes A, E becomes B … B becomes Y, A becomes X.

e.g. HQJLQHHULQJ → ENGINEERING

Try it!

You may go to https://iesummerworkshop.github.io/caesar-cipher.html.



Plaintext:

```
The study of Information Engineering (IE) encompasses elements of Computer Science, Electronic Engineering, Data Science, Information Security, Information Theory, Telecommunications Networking, and Artificial Intelligence, and their integration. The knowledge and skills the students gain will find wide applications in emerging areas such as Artificial Intelligence of Things, Big Data Analytics, B5G/6G, Cloud/Edge Computing, Computer Vision, Cryptocurrency, Industry 4.0, the Metaverse and Smart Cities. The multi-disciplinary nature of our programme is what makes it unique and rewarding.
```

Key: 13

What is the ciphertext?

Algorithm of Caesar Cipher:

Encryption: $(plaintext + key)\ mod\ 26$

Decryption: $(plaintext - key)\ mod\ 26$

Here we assume 'A' represents 0, 'B' repsents 1, 'C' represents 2, …, 'Z' represents 25.

Need to be careful (adjust the mapping index) if you're dealing with ASCII.

[Write Python (decrypt) together: https://iesummerworkshop.github.io/py-lab.html]

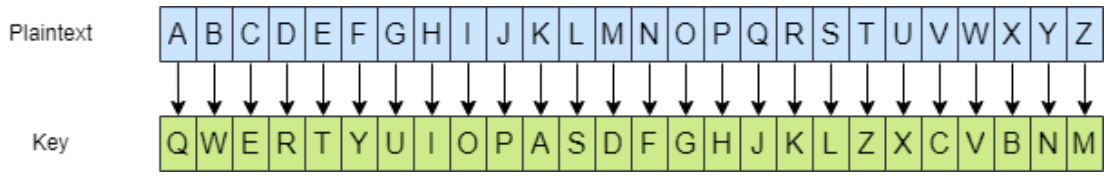
Question:

1. Why mod 26 is needed?
2. How many possibilities is the key in Caesar Cipher?

Section 2.3: Monoalphabetic Cipher

In a monoalphabetic substitution cipher, each letter of the plaintext is replaced by a single corresponding letter or symbol. This means that each occurrence of a particular letter in the plaintext will be replaced by the same letter or symbol in the ciphertext.

Encryption

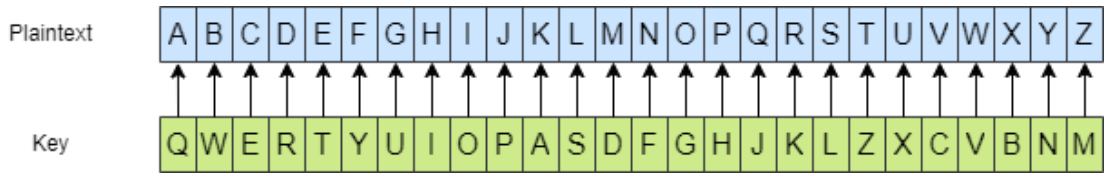


Key = QWERTYUIOPASDFGHJKLZXCVBNM

Letter A becomes Q, B becomes W … Y becomes N, Z becomes M.

e.g. HELLO WORLD → ITSSG VGKSR

Decryption

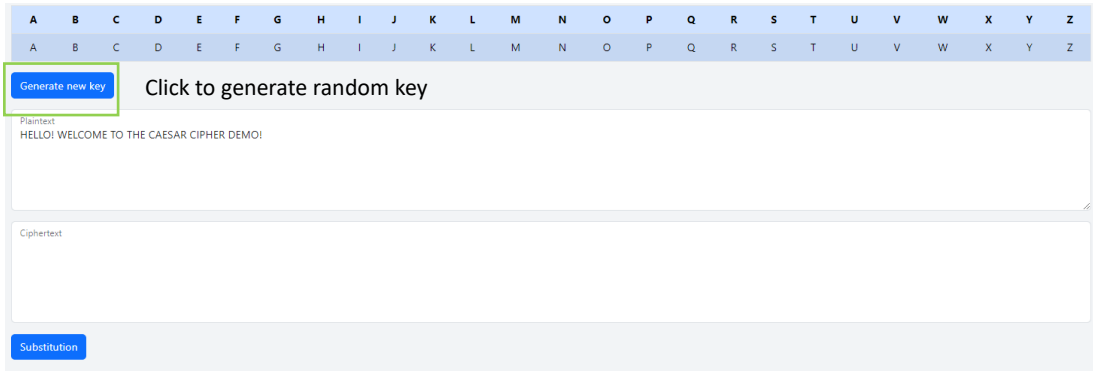


Key = QWERTYUIOPASDFGHJKLZXCVBNM

Letter Q becomes A, W becomes B … N becomes Y, M becomes Z

Try it! [TBC]

You may go to https://iesummerworkshop.github.io/monoalphabetic-cipher.html



Plaintext:

The IE department was established in 1989 as the first IE department and remains one of a kind in Hong Kong. The department is devoted to nurturing and educating engineering leaders for the information world of today and tomorrow. The professors are dedicated educators and world-class researchers, many of whom had extensive experience with leading research institutions worldwide before joining the department. The department has a strong presence in top-tier venues in the global scientific community, as

```
well as strong connections in the local IT industry in Hong Kong.
```
What is the ciphertext?

Be careful (remember to adjust the mapping index) if you're dealing with ASCII.

[Write Python together: https://iesummerworkshop.github.io/py-lab.html]

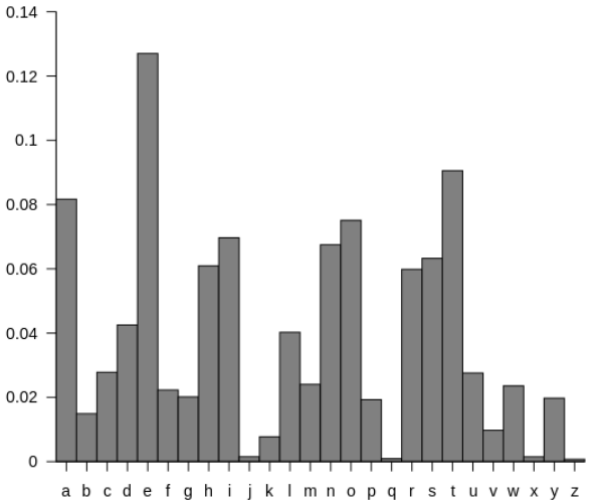
Question

1. How many possibilities is the key in monoalphabetic cipher?

Section 2.4: Breaking classic cipher with frequency analysis

Frequency analysis is a technique used in cryptography to decipher substitution ciphers (e.g. Caesar cipher and monoalphabetic cipher). It is based on the observation that certain letters or symbols tend to occur more frequently in a given language. By analyzing the frequency distribution of letters or symbols in a ciphertext, it becomes possible to make guesses about the substitutions made and ultimately decrypt the message.

The underlying principle of frequency analysis is that different letters or symbols have different probabilities of occurrence in a language. By comparing the observed frequencies of letters in the ciphertext with the expected frequencies in the target language (e.g., English), it is possible to identify patterns and make inferences about the substitutions made. The following diagram shows the relative frequency in the English language:



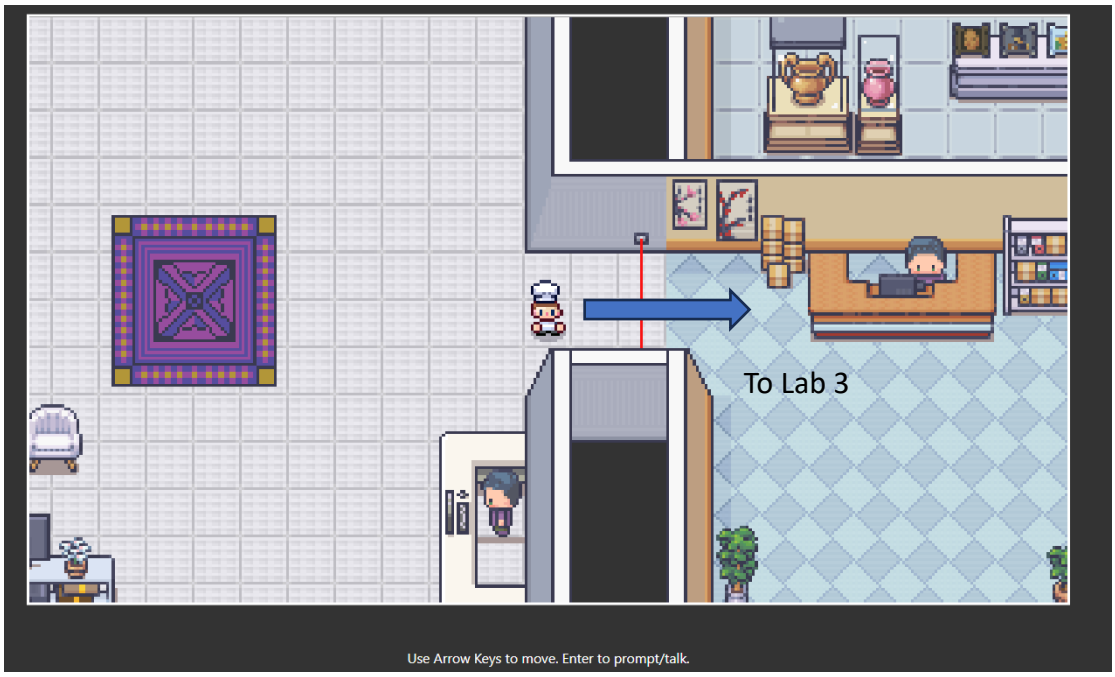
Try it!

- What are the top 5 most frequent characters, based on the above graph?
1.
2.
3.
4.
5.

Note: If necessary, the idea of frequency analysis can be extended to look at words instead of individual alphabets. For example, the word “the” is quite common in English. And even if the underlying plaintext is not in English, the frequency feature idea can still apply. For instance, El/La/Los/Las are relatively common in Spanish (similar meaning to “the” in English).

Section 3: Challenge time

Now, please go to https://iesummerworkshop.github.io/rpg/page.html to enjoy your challenge! Walk in front of the computer and press enter to interact with it. Mark down the key you get!

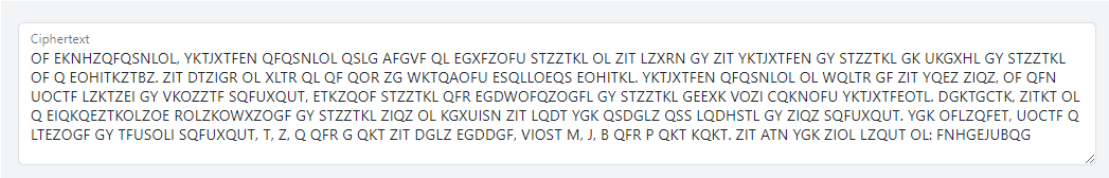


1. [☆ ☆] Frequency Analysis on monoalphabetic cipher

a. Go to https://iesummerworkshop.github.io/frequency-analysis.html

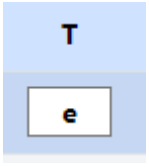
b. paste the following paragraph to the ciphertext section:

```
OF EKNHZQFQSNLOL, YKTJXTFEN QFQSNLOL QSLG AFGVF QL EGXFZOFU STZZTKL OL ZIT LZXRN GY ZIT YKTJXTFEN GY STZZTKL GK UKGXHL GY STZZTKL OF Q EOHITKZTBZ. ZIT DTZIGR OL XLTR QL QF QOR ZG WKTQAOFU ESQLLOEQS EOHITKL. YKTJXTFEN QFQSNLOL OL WQLTR GF ZIT YQEZ ZIQZ, OF QFN UOCTF LZKTZEI GY VKOZZTF SQFUXQUT, ETKZQOF STZZTKL QFR EGDWOFQZOGFL GY STZZTKL GEEXK VOZI CQKNOFU YKTJXTFEOTL. DGKTGCTK, ZITKT OL Q EIQKQEZTKOLZOE ROLZKOWXZOGF GY STZZTKL ZIQZ OL KGXUISN ZIT LQDT YGK QSDGLZ QSS LQDHSTL GY ZIQZ SQFUXQUT. YGK OFLZQFET, UOCTF Q LTEZOGF GY TFUSOLI SQFUXQUT, T, Z, Q QFR G QKT ZIT DGLZ EGDDGF, VIOST M, J, B QFR P QKT KQKT. ZIT ATN YGK ZIOL LZQUT OL: FNHGEJUBQG
```

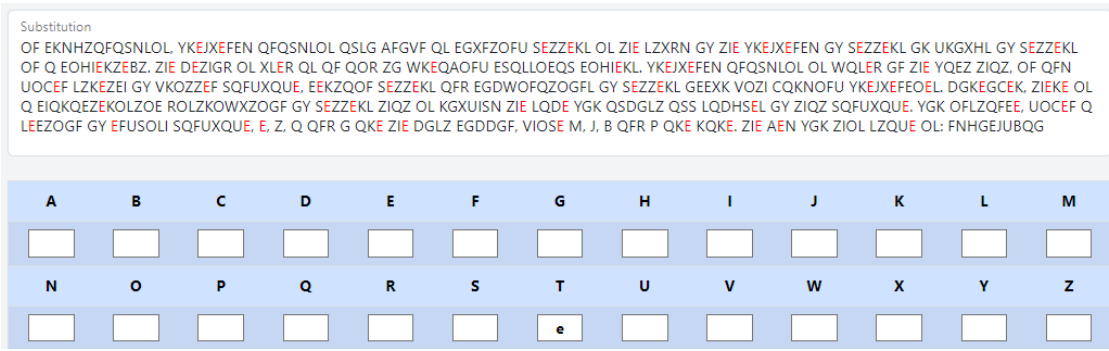


c. Click on Analysis button, a diagram should be generated

d. You may input the corresponding substituted characters below each character, e.g. the most frequent character in this paragraph is T, it is very likely that it corresponds to E



e. Click on Substitution to see what is the result after substituting the characters in the ciphertext



f. Continue the guessing until all the boxes are filled

2. [☆ ☆] Caesar cipher encrypted

Given a key (integer) and a plaintext message, return the encrypted message (ciphertext).

Interact with this counter to submit the key for lab 3 and lab 4