Week 1: Cryptography Fundamentals

CRYPTOGRAPHY BASICS

Plaintext : The **original message** that needs to be protected.
Ciphertext : The **scrambled (encrypted)** version of the plaintext that is not readable without a secret key.
Key : A **secret piece of data** (like a password or code) used in the process of encrypting or decrypting a message.
Encryption : The process of **converting plaintext into ciphertext** using a key, to keep the information secret.
Decryption : The process of **converting ciphertext back into plaintext** using a key, to make it readable again.

Symmetric Key Cryptography :

1. The **same key** is used for both encryption and decryption.
2. Fast but needs safe key sharing.
3. Example: **AES (Advanced Encryption Standard)**

Example : Secret Box has only one lock so Receiver and sender has same key.

Asymmetric Key Cryptography :

Uses **two keys**:

- A **public key** (for encryption)
- A **private key** (for decryption)

1. More secure for communication.
2. Example: **RSA (Rivest–Shamir–Adleman)**

Example : Secret Box has two locks so Receiver and sender can have two different keys.

SOME BASIC CIPHERS

CAESAR CIPHER

Shift each letter in the plaintext by a fixed number of positions in the alphabet.

Example:

If the shift = **3**:
A → D   B → E   C → F   ...   Z → C (wraps around)
Plaintext: HELLO
Ciphertext: KHOOR

Decryption:

Just shift in the opposite direction.

VIGENERE CIPHER

The Vigenère cipher is a **polyalphabetic substitution cipher** that uses a **repeating keyword** to shift letters of the plaintext.

The following table can be used to encode a message:
Examples :

Alphabet Indexing

We convert letters to numbers (A = 0, B = 1, ..., Z = 25):

Letter	Value
A	0
B	1
...	...
Z	25

Encryption Formula

Let:

- **P** = Plaintext letter
- **K** = Key letter
- **C** = Ciphertext letter

Each character is encrypted as:
$C_i = (P_i + K_i) \bmod 26$
Where:

- P_i is the index of the i^{th} plaintext letter
- K_i is the index of the corresponding key letter
- C_i is the resulting ciphertext index

Decryption Formula

$P_i = (C_i - K_i + 26) \bmod 26$

PLAYFAIR CIPHER

The Playfair cipher encrypts **pairs of letters** (called **digraphs**) using a **5×5 grid** of letters created from a keyword.
The keyword is used to build a 5×5 matrix. Repeated letters are removed, and **I and J are considered the same**.

Example Keyword: `MONARCHY`
Remove duplicates
Final Grid:

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

Plaintext

- Break into pairs (digraphs).
- If the same letter appears twice in a pair, insert an 'X' between them.
- If there's an odd letter at the end, add an 'X'.

Let's encrypt: **HELLO**
Pairs → `HE`, `LX`, `LO`

Apply encryption rules using the 5×5 grid

1. Same Row
→ Replace each letter with the **one to its right** (wrap around if needed).

2. Same Column
→ Replace each letter with the **one below** it (wrap around if needed).

3. Rectangle Rule
→ Replace each letter with the one in its **row** but in the **other letter's column**.
Encrypt `HE`:

- H → (2nd row, 2nd col)
- E → (3rd row, 1st col) → Rectangle → `C` and `F`

`HE → CF`
Encrypt `LX`:

- L → (4th row, 1st col)
- X → (5th row, 4th col) → Rectangle → `U` and `S`

`LX → US`
Encrypt `LO`:

- L → (4th row, 1st col)
- O → (1st row, 2nd col) → Rectangle → `P` and `M`

`LO` → `PM`

Final Ciphertext: `CFUSPM`

Decryption Rules :

Use the **same 5×5 grid** as encryption (made from the same keyword).

Step 1: Split ciphertext into digraphs (pairs)

Just break the encrypted message into 2-letter chunks: e.g. `CFUSPM` → `CF`, `US`, `PM`

Step 2: Apply decryption rules:

1. Same Row

→ Replace each letter with the one **to its left** (wrap around to the end if needed)

2. Same Column

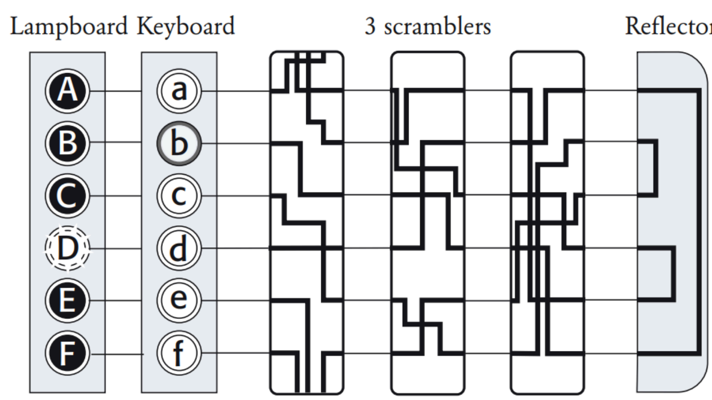
→ Replace each letter with the one **above** it (wrap around to bottom if needed)

3. Rectangle Rule

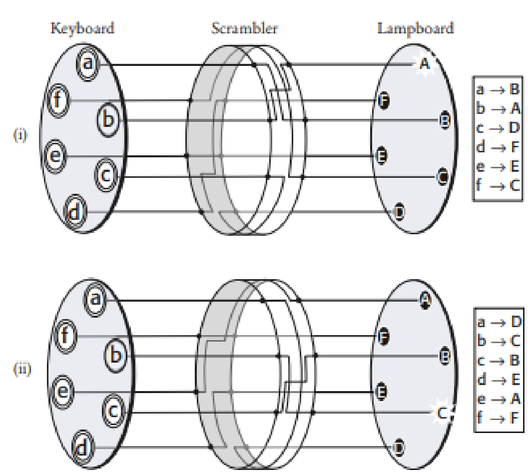
→ Replace each letter with the one in the **same row**, but the **column of the other letter**

ENIGMA

What is the Enigma Machine?



The **Enigma** was an **electromechanical cipher machine** used by **Nazi Germany during World War II** to send secret military messages. It looked like a typewriter but had complex internal wiring that made its encryption extremely difficult to break.



How It Worked

The Enigma used **several key components**:

1. Keyboard

- You press a key (say, `A`) to encrypt a letter.

2. Rotors (Wheels)

- The heart of Enigma. Each rotor has 26 positions (A–Z), and it scrambles input letters by internal wiring.
- There are usually **3 to 5 rotors**, and their order and starting positions form part of the **key**.

3. Reflector

- The signal goes through the rotors to a **reflector**, which bounces it back through the rotors in reverse.
- Ensures that encryption is **symmetrical**: same machine setting encrypts and decrypts.

4. Plugboard (Steckerbrett)

- Swaps pairs of letters **before and after** the rotors.
- Adds another layer of complexity.

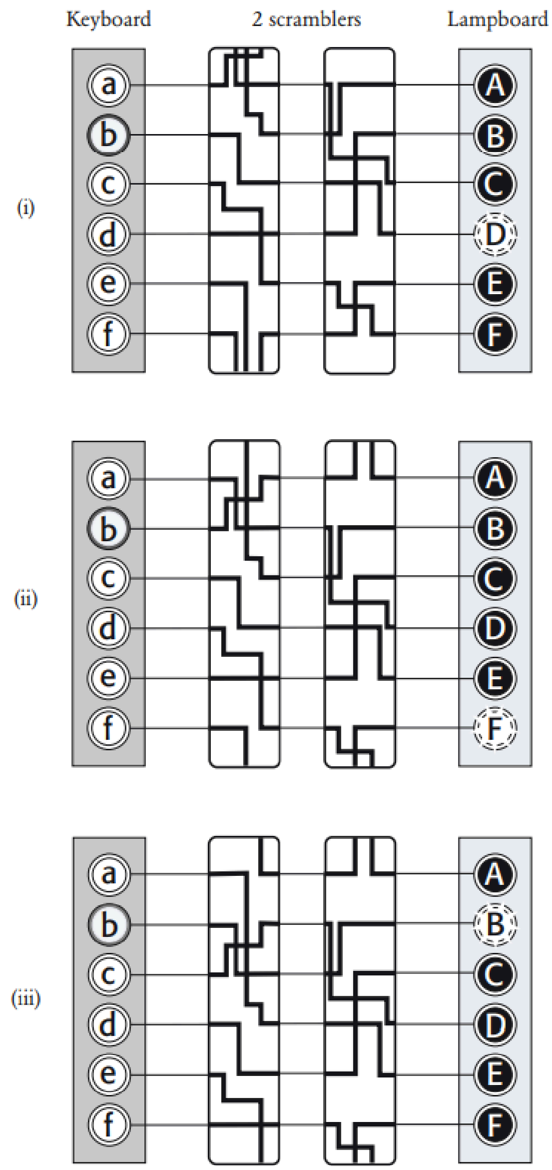
5. Lamp board

- When you press a key, a **different letter lights up** — that's the encrypted character

Say you press A → The signal passes through: *Plugboard → Rotors (forward) → Reflector → Rotors (backward) → Plugboard*

And maybe A becomes G.

Now, because **rotors rotate** after each key press (like an odometer), pressing A again might become Z next time. This **changing encryption** is what made Enigma so hard to crack.

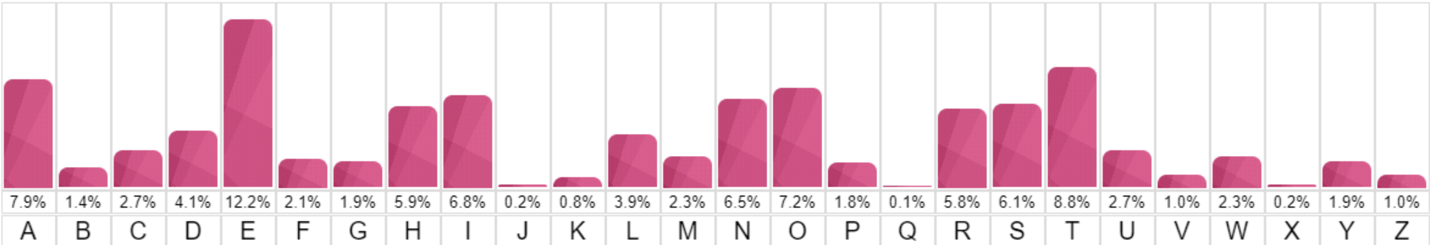


FREQUENCY ANALYSIS

In cryptography, frequency analysis is the study of the **frequency of letters** or groups of letters in a ciphertext. The method is used as an aid to breaking **substitution ciphers.**

Frequency analysis consists of **counting the occurrence of each letter** in a text. Frequency analysis is based on the fact that, in any given piece of text, certain letters and combinations of letters occur with varying frequencies. For instance, given a section of English language, letters **E, T, A and O** are the most common, while letters Z, Q and X are not as frequently used.

The following chart shows the frequency of each letter of the alphabet for the English language:



We can assume that most samples of text written in English would have a similar distribution of letters. However this is only true if the sample of text is long enough. A very short text may lead to a significantly different distribution.

When trying to decrypt a cipher text based on a substitution cipher, we can use a frequency analysis to help identify the most recurring letters in a cipher text and hence make **hypothesis** of what these letters have been encoded as (e.g. E, T, A, O, etc). This will help us decrypt some of the letters in the text. We can then recognise **patterns/words** in the partly decoded text to identify more substitutions.

```
import matplotlib.pyplot as plt
```

Week 1 : Resources Used - THE CODE BOOK | 101computing.net