Class 3 — Symmetric Encryption

Section 1: Opening & Hook

“Let’s rewind time for a moment. Imagine you are a merchant in the ancient Silk Road. You’re sending a letter to your business partner in another city, telling him how much gold to pay for silk. If a bandit intercepts your messenger, your deal is ruined — your prices are exposed.

Now imagine you’re Julius Caesar, the Roman general. You send military orders across the empire. If the enemy reads them, Rome could fall.

Fast forward to World War II: the Germans built the Enigma machine, believing it would keep their communications unbreakable. The Allies cracked it, and the war changed.

Fast forward again: 2016. Hackers infiltrated the **Bangladesh Bank** through the SWIFT system and sent fraudulent transfer requests. $81 million vanished overnight.

Different centuries, same problem: how do we protect secrets when they travel?

That, ladies and gentlemen, is where **symmetric encryption** begins — the oldest, fastest, and still most widely used form of cryptography.”

Section 2: Symmetric Cipher Model

“Let’s formalize this.

In symmetric encryption, the process is simple in theory:

1. We start with **plaintext** — the original message.
2. We feed it into an **encryption algorithm**.
3. With the help of a secret key, the algorithm produces **ciphertext**.
4. The receiver, who also has the same key, applies the **decryption algorithm**, getting back the plaintext.

So the key is the *single shared secret*. Both sender and receiver must know it, and no one else must.

Analogy: Think of it like a house key. If you and your roommate share the same key, you can both open the door. But if anyone else gets that key, your home is no longer safe.

This is why the security of symmetric systems rests entirely on the **key distribution problem**. How do we share the key safely before we even start talking? That’s a puzzle we’ll revisit.”

Reference: Stallings, *Cryptography & Network Security*, 8th Ed., Chapter 2 (p. 29–31).

Section 3: Classical Encryption Techniques

Part A: Substitution Ciphers

Speaking Script:

“Substitution is the simplest form of encryption: replace each letter with another.

The most famous is the **Caesar Cipher**, where each letter is shifted by 3.

Example:

- Plaintext: ATTACK AT DAWN
- Ciphertext (shift 3): DWWDFN DW GDZQ

Formally, if we assign numbers A=0, B=1, …, *Z*=25, then:

C=(P+k)mod 26C = (P + k) \mod 26

C=(P+k)mod26

where *P* is the plaintext letter number, *k* is the shift, *C* is the ciphertext letter.

Decryption is the reverse:

P=(C−k)mod 26P = (C - k) \mod 26

P=(C−k)mod26

Math Example (from Stallings):

Plaintext: HELLO

- H=7 → (7+3)=10 → K
- E=4 → (4+3)=7 → H
- L=11 → (11+3)=14 → O
- O=14 → (14+3)=17 → R

Ciphertext = KHOOR

→ This is why Caesar's own messages were safe only as long as enemies didn't know the shift. But once you guess the key, the system collapses."

Story: Al-Kindi, a 9th-century Arab mathematician, invented **frequency analysis**. He realized some letters occur more often in language (like 'E' in English). By analyzing ciphertext, you can spot patterns and break substitution ciphers.

Analogy: Think of musical chairs — the players (letters) keep moving around, but the number of chairs stays the same. Frequency betrays the music.

Reference: Stallings, Ch. 2.2–2.4.

Part B: Monoalphabetic vs Polyalphabetic

- Monoalphabetic = one fixed mapping for the whole text. Easy to break.
- Polyalphabetic = mapping changes (Vigenère cipher).

Monoalphabetic Cipher

A **monoalphabetic cipher** is a **substitution cipher** where **each letter in the plaintext is replaced with exactly one corresponding letter** in the ciphertext alphabet.

- The substitution pattern remains **constant throughout the message**.
- It uses a **fixed key** that defines how letters are mapped.

Example

Plain Text:
ABCDEFGHIJKLMNOPQRSTUVWXYZ

Encrypted:
QWERTYUIOPASDFGHJKLZXCVBNM

Then:

- A → Q
- B → W
- C → E
- D → R
- ... and so on.

Plaintext: HELLO
→ Ciphertext: ITSSG

Encryption:

1. Choose or generate a **random substitution key** (mapping of alphabets).
2. Replace each plaintext character using the mapping.

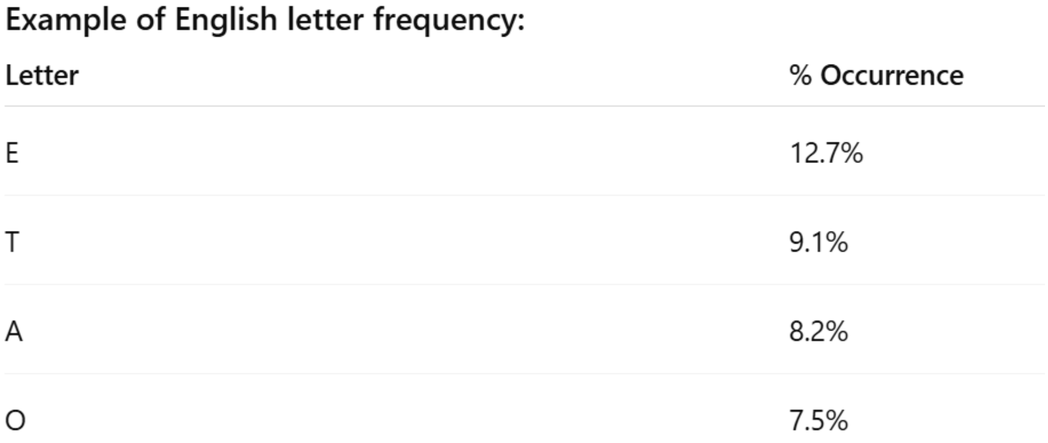
Decryption:

1. Use the **inverse key mapping**.
2. Replace ciphertext letters with corresponding plaintext letters.

Weakness – Frequency Analysis

Even though the key space is huge, the cipher is **vulnerable to frequency analysis** because:

- Each plaintext letter always maps to the same ciphertext letter.
- Common letters like **E, T, A, O** appear often.
- By comparing frequency of ciphertext letters with standard English frequencies, attackers can guess mappings.



Example of English letter frequency:

Letter	% Occurrence
E	12.7%
T	9.1%
A	8.2%
O	7.5%

So, if a ciphertext has a symbol occurring ~12%, it probably represents **E**

If, instead, the “cipher” line can be any permutation of the 26 alphabetic characters,

then there are 26! or greater than 4 * 10^{26} possible keys.

Math (Vigenère):

Ci=(Pi+Ki)mod 26C_i = (P_i + K_i) \mod 26

Ci=(Pi+Ki)mod26

where the key *K* is repeated.

Example (Key = LEMON, Plaintext = ATTACK):

A (0) + L (11) → L

T (19) + E (4) → X

T (19) + M (12) → F

A (0) + O (14) → O

C (2) + N (13) → P

K (10) + L (11) → V

Ciphertext = LXFOPV

This resisted frequency analysis longer, but even Vigenère eventually fell to advanced statistical methods.

Reference: Stallings, Ch. 2.5.

Part C: Transposition Ciphers

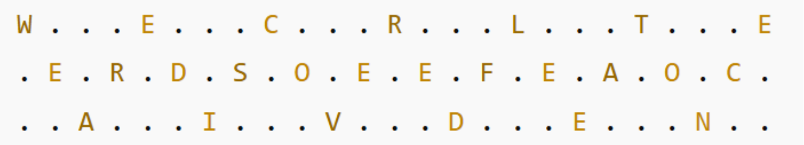
“Now, instead of replacing letters, what if we just shuffle their order? That’s a transposition cipher.

Rail Fence Cipher

Encryption Steps:

1. Choose number of rails (say 3).
2. Write letters in zigzag fashion.
3. Read row by row.

Example:
Plaintext = WEAREDISCOVEREDFLEEATONCE
Rails = 3



```
W . . . E . . . C . . . R . . . L . . . T . . . E
. E . R . D . S . O . E . E . F . E . A . O . C .
. . A . . . I . . . V . . . D . . . E . . . N . .
```

Read row by row →

Ciphertext: WECRLTEERDSOEEFEAOCAIVDEN

Nothing changed, except the order.

Columnar Transposition Cipher

Encryption Steps:

1. Choose a **keyword** (e.g., CIPHER).
2. Write plaintext in rows under keyword.
3. Number columns based on alphabetical order of keyword letters.
4. Read columns in sorted order.

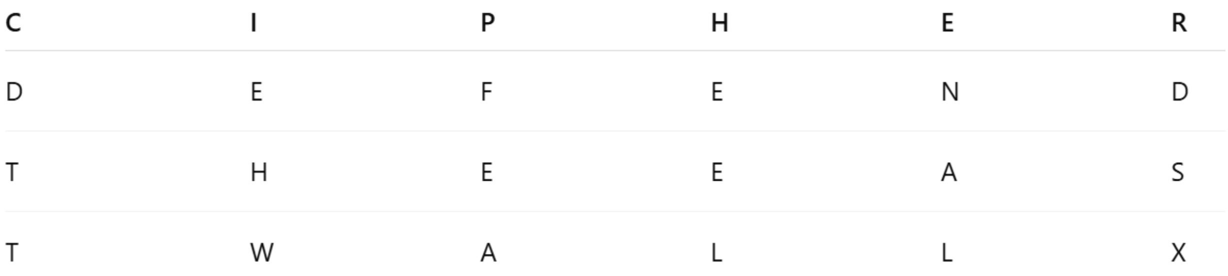
Example:
Plaintext: DEFEND THE EAST WALL
Keyword: CIPHER

Total letters = 17

Keyword length = 6 (C I P H E R)

We’ll need to fill 3 rows (since 6 × 3 = 18, one extra cell will be padded).

Write plaintext in a table under the keyword

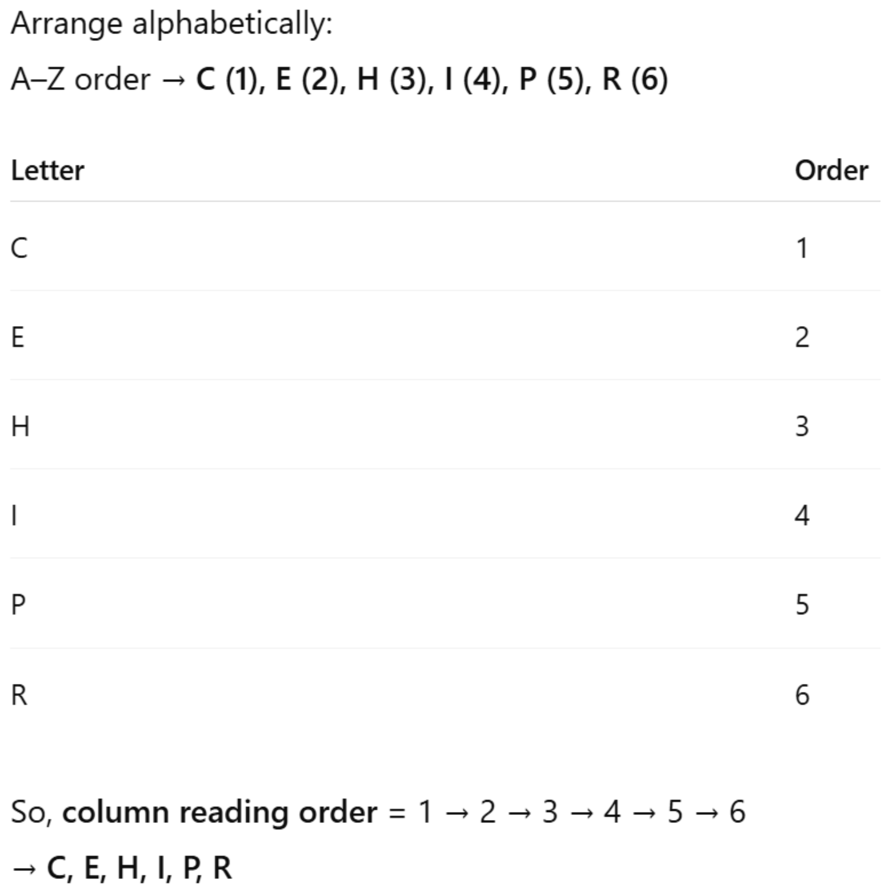


C	I	P	H	E	R
D	E	F	E	N	D
T	H	E	E	A	S
T	W	A	L	L	X

Determine column order

Keyword: **C I P H E R**

Arrange alphabetically:
A–Z order → **C (1), E (2), H (3), I (4), P (5), R (6)**



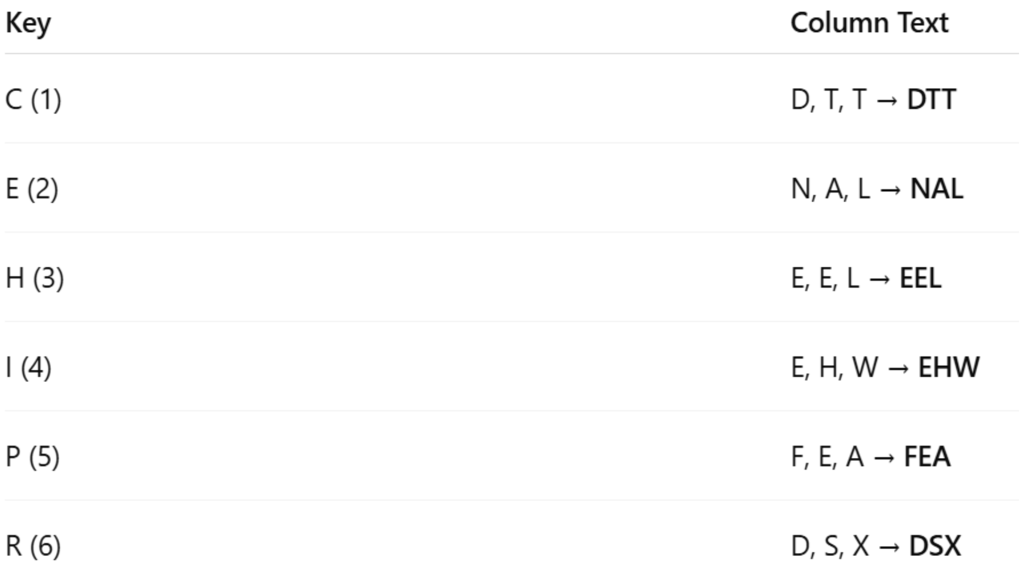
Arrange alphabetically:

A–Z order → **C (1), E (2), H (3), I (4), P (5), R (6)**

Letter	**Order**
C	1
E	2
H	3
I	4
P	5
R	6

So, **column reading order** = 1 → 2 → 3 → 4 → 5 → 6

→ **C, E, H, I, P, R**



Key	Column Text
C (1)	D, T, T → **DTT**
E (2)	N, A, L → **NAL**
H (3)	E, E, L → **EEL**
I (4)	E, H, W → **EHW**
P (5)	F, E, A → **FEA**
R (6)	D, S, X → **DSX**

Ciphertext = DTT + NAL + EEL + EHW + FEA + DSX

→

Decryption (reverse)

To decrypt:

1. Write ciphertext in columns in alphabetical order of keyword.
2. Reorder columns by original keyword sequence.
3. Read row-wise.

Columnar transposition: write in rows, read in columns."

Analogy: Like shuffling cards — same deck, different order.

Reference: Stallings, Ch. 2.7.

Section 4: Substitution + Transposition = Stronger

“Individually, substitution and transposition are weak. But combined, they give us powerful ciphers. This is called a **product cipher**.

Example: Substitution hides meaning, transposition spreads it out. Together, they make frequency analysis harder.

The **Playfair Cipher** (used by British Army in WWI) used digraph substitution, hiding letter frequencies better than Caesar.

This idea — mix substitution and transposition — is the ancestor of **modern block ciphers** like DES and AES.”

Reference: Stallings, Ch. 2.8.

Section 5: Block Cipher Design Principles

“Claude Shannon, the father of information theory, gave us two principles: **confusion** and **diffusion.**

- Confusion = obscure relationship between key and ciphertext (done via substitution).
- Diffusion = spread plaintext across ciphertext so patterns disappear (done via transposition).

Analogy: Imagine cooking soup. Confusion is hiding the recipe (ingredients unknown). Diffusion is stirring it so no ingredient stays clumped.

This is why block ciphers use multiple **rounds** of substitution and permutation. Each round adds layers of confusion and diffusion until the ciphertext looks completely random.”

Case Study:

- **DES** → based on 16 rounds of substitution + permutation. Safe in the 1970s, broken today because of short key length (56-bit).
- **AES** → uses substitution boxes (S-boxes) and mixing steps. Stronger because of longer keys (128/192/256-bit).

Reference: Stallings, Ch. 3.2–3.4.

Section 6: DES, 2DES, 3DES

DES (Data Encryption Standard)

- Developed by IBM, adopted in 1977 (FIPS 46).

- 64-bit block, 56-bit key.
- 16 rounds of substitution–permutation.

Math Note (Stallings Ch. 3):

Initial permutation → 16 Feistel rounds → final permutation.

Round function: Expansion + XOR key + S-box + P-box.

Weakness:

- 56-bit key space = 2^56 ≈ 7.2 × 10^16 keys.
- Brute-forced in 1998 by EFF’s Deep Crack in 56 hours.

Double DES (2DES)

- Encrypt with K1, then K2.
- Vulnerable to **meet-in-the-middle attack**: requires ~2^57 operations instead of 2^112.

Triple DES (3DES)

- Encrypt–Decrypt–Encrypt with K1, K2, (and optionally K3).
- Effective key length ~112 bits.
- Still used in banking (ATM networks, SWIFT) but being phased out.

Story:

- SWIFT secure messages once relied on 3DES. Transition to AES was mandated because 3DES is too slow and less secure.

Section 7: AES – Modern Standard

Advanced Encryption Standard (AES)

- Adopted in 2001 (Rijndael algorithm).
- Block size = 128 bits. Key sizes = 128, 192, 256 bits.
- Operates in rounds (10, 12, 14 depending on key size).

Round Functions:

1. SubBytes (non-linear substitution).

2. ShiftRows (permutation).
3. MixColumns (diffusion via linear algebra over GF(2^8)).
4. AddRoundKey (XOR with round key).

Math Detail (Stallings Ch. 5, Paar Ch. 4):

- AES works in GF(2^8). Each byte = polynomial of degree < 8.
- Example: 0x57 ⊗ 0x83 in GF(2^8) with irreducible polynomial m(x) = x^8 + x^4 + x^3 + x + 1.

Why secure?

- Key size large (2^128 operations infeasible).
- Resistant to known cryptanalysis.
- Extremely fast in hardware/software.

Real-world:

- WhatsApp, Signal, TLS, VPNs, 5G networks all rely on AES.
- The gold standard of symmetric crypto.

Section 8: Attacks on Symmetric Encryption

1. **Brute force** (try all keys).
    - DES broken, AES safe.
2. **Cryptanalysis**
    - Linear cryptanalysis.
    - Differential cryptanalysis (studied by Biham and Shamir).
3. **Side-channel attacks**
    - Attack implementation, not algorithm.
    - Example: Measuring power consumption of smart cards to deduce AES keys.

Story:

- 2010s: Researchers extracted AES keys from smartphones using electromagnetic side-channel analysis. No math broken — just physics exploited.

Closing Thought:

“Symmetric encryption is like the heartbeat of cybersecurity. Fast, efficient, proven. But it lives and dies by its keys. In the next class, we’ll see how asymmetric encryption solves the key distribution problem — but also why it can’t replace symmetric ciphers in practice.”

References (Class 3)

- **Stallings, Cryptography & Network Security, 8th Ed.**
  - Chapter 2: Symmetric Cipher Model.
  - Chapter 3: Classical Encryption Techniques.
  - Chapter 3–5: DES, AES, Block Cipher Principles.
  - Chapter 6: Block Cipher Modes & Applications.
- **Paar & Pelzl, Understanding Cryptography**
  - Chapter 2: Historical Ciphers.
  - Chapter 3: DES.
  - Chapter 4: AES.
- **Case Studies:**
  - EFF’s Deep Crack (1998) — DES broken.
  - SWIFT’s migration from 3DES to AES.
  - Side-channel AES key recovery (smartcards, smartphones).