MẬT MÃ CỔ ĐIỂN

- Hệ thống này đều dựa trên sơ đồ mã hóa đối xứng.
- Dịch vụ bảo mật duy nhất mà các hệ thống này cung cấp là tính bảo mật thông tin.
- Không giống như các hệ thống hiện đại (coi dữ liệu là hệ nhị phân (binary)), **các hệ thống cổ điển hoạt động trên bảng chữ cái như một thành phần cơ bản.**
- Có 2 loại mật mã cổ điển:
    - **Mật mã thay thế** (Substitution Cipher)
    - **Mật mã chuyển vị** (Transposition Cipher)

1. Phân loại:

a/ Mật mã thay thế (Substitution Cipher):

- Bất kỳ ký tự nào của bản rõ từ tập hợp các ký tự cố định đã cho sẽ được thay thế bằng một số ký tự khác từ cùng một tập hợp (tùy thuộc vào khóa để có sự thay thế khác nhau).
  Ví dụ: với độ dịch chuyển là 2 thì A sẽ được thay thế bằng C, B trở thành D, .v.v.
- **Mật mã thay thế** được chia thành:
    - ***Mật mã đơn chữ cái:*** **mỗi** ký tự trong bản rõ được ánh xạ tới **một** ký tự trong bản mã.
        - Cho dù một ký hiệu xuất hiện bao nhiêu lần trong bản rõ thì nó sẽ tương ứng với cùng một ký hiệu trong bản mã
        - Nghĩa là **độ dài của bản rõ và bản mã là như nhau.**
        - **Các loại mật mã:** Mật mã Shift (hay còn gọi là mật mã Caesar, mật mã cộng), Mật mã nhân, Mật mã Affine
    - ***Mật mã đa chữ cái:*** **Một** ký tự trong bản rõ có thể được thay thế bằng **nhiều** ký tự khác nhau trong bản mã, tùy vào vị trí xuất hiện.
        - Mỗi lần xuất hiện khác nhau của một ký tự có ánh xạ khác nhau tới bản mã.
        - **Các loại mật mã:** Mật mã Auto-key, Mật mã Vigenere, Mật mã Playfair, Mật mã Hill, One Time Pad, Mật mã Rotor

b/ Mật mã chuyển vị:

- **không xử lý việc thay thế ký tự này bằng một ký tự khác**.
- Nó tập trung vào việc **thay đổi vị trí** của các ký tự trong **bản rõ.**
- Một ký hiệu ở vị trí đầu tiên trong bản rõ có thể xuất hiện ở vị trí thứ 7 trong bản mã.
- **Các loại mật mã:** Mật mã chuyển vị cột, Mật mã Rail-Fence.

2. Mật mã thay thế - Mật mã Caesar

Mỗi chữ cái được thay thế bằng một chữ cái nằm bên phải k ký tự trong bảng chữ cái theo modulo bằng số chữ cái trong bảng chữ cái:

$$C_k(j) = (j + k) \pmod n$$

Trong đó:

- *j* – vị trí ban đầu của chữ cái trong bảng chữ cái.
- *k* là khóa dịch chuyển - số vị trí dịch (shift).
- $C_k(j)$ – Số thứ tự của chữ cái thay thế
- *n* - sức mạnh của bảng chữ cái ban đầu (số lượng chữ cái trong bảng chữ cái)

Công thức giải mã:

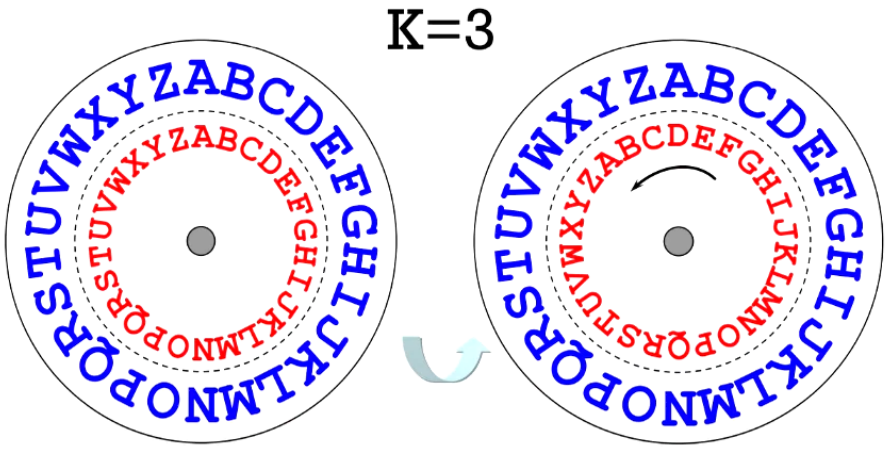
- **Mục tiêu ta đi tìm j – vị trí ban đầu của ký tự trong bảng chữ cái.**
- **Đặt** $i = C_k(j) = (j + k) \bmod n$. Xem như i là số dư của phép chia (j+k) mod n. Vậy ta có thể nói i và (j+k) đồng dư theo phép modulo n. Ta có thể viết: $j + k \equiv i \pmod n$
- Trừ k ở 2 vế để tìm j: $j \equiv i - k \pmod n$ (1)
- Vì i−k có thể âm (nếu i < k), ta cần đảm bảo kết quả nằm trong khoảng [0, n−1]. Trong số học modulo, nếu i−k<0, ta có thể **cộng thêm một bội số của n** để đưa về giá trị dương: $i - k + n \equiv i - k \pmod n$
- Vậy có thể viết lại (1) như sau: $j \equiv i - k + n \pmod n \Rightarrow j = i - k + n \pmod n$

  $$\Rightarrow j = [C_k(j) - k + n] \bmod n$$

  Có thể viết lại: $C_k^{-1}(i) = C_{n-k} = (i + n - k) \pmod n$

Ý nghĩa của công thức

- Kết quả được tính **theo modulo n** (tức là nếu vượt quá n, sẽ quay vòng lại từ đầu bảng chữ cái).
- Nếu **j–k nhỏ hơn 0**, ta cộng thêm n để kết quả luôn hợp lệ trong modulo n.
- **Mã hóa** dịch chữ cái về bên phải k vị trí.
- **Giải mã** dịch chữ cái về bên trái k vị trí.



Ví dụ 1: Cho bản rõ: TOIYEUVIETNAM Khóa k = 4. Ta có thể viết lại bảng chữ cái mới như sau:

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

Bản rõ	**T**	**O**	**I**	**Y**	**E**	**U**	**V**	**I**	**E**	**T**	**N**	**A**	**M**
i	19	14	8	24	4	20	21	8	4	19	13	0	12
(i + k) mod 26	23	18	12	1	8	24	25	12	8	23	17	4	16
Bản mã	X	S	M	B	I	Y	Z	M	I	X	R	E	Q

Vậy bản mã là: XSMBIYZMIXREQ

- Trong loại mật mã Caesar này, **khóa được cho bởi số k (0 <= k <= n - 1)** và một từ khóa hoặc câu ngắn.
- **Trong trường hợp có từ khóa** để tránh tình huống tấn công vét cạn
  - Bảng chữ cái được viết ra, và bên dưới nó, **bắt đầu từ vị trí k là từ khoá.**
  - Các chữ cái còn lại được viết theo thứ tự bảng chữ cái **sau từ khóa**. Kết quả là, chúng ta nhận được tra cứu cho từng chữ cái.
  - Không bắt buộc tất cả các chữ cái của từ khóa phải khác nhau, tuy nhiên phải viết ra từ khóa vào bảng chữ cái mới nhưng **không lặp lại các chữ cái**.

Ví dụ 2:

Từ khoá: "yes", k =2	
Bảng gốc	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
Bảng mới	x, z, **y, e, s**, a, b, c, d, f, g, h, i, j, k, l, m, n, o, p, q, r, t, u, v, w
Văn bản nguồn	data
Văn bản mã hoá	expx

- *Số lượng chìa khóa trong hệ thống của Caesar **với từ khóa** là **n!**.*
- *Để giải mã, cần sử dụng khóa mã hóa đã biết để xác định sự tương ứng của bản gốc, bản thay thế bảng chữ cái và thực hiện thay thế ngược lại.*

Nhận xét:

- Nếu cần thiết, bảng chữ cái có thể được mở rộng với dấu chấm câu, chữ in hoa, số, để mật mã có thể xử lý tất cả các ký tự của văn bản nguồn.
- Mật mã Caesar dễ bị **tấn công brute-force**, vì chỉ có **n khóa có thể có** (ví dụ, với tiếng Anh có 26 khóa).
- **3 đặc điểm chính để áp dụng tấn công theo kiểu brute-force trong thuật toán này**
  - Giải thuật mã hóa và giải mã được biết trước
  - Số khóa để thử rất ít
  - Ngôn ngữ của bản rõ được biết trước và dễ dàng nhận ra
- **Giải quyết vấn đề (tổng quát)**
  - Sử dụng nhiều khóa
  - Bản rõ có thể được nén lại (Huffman, ZIP) để cho người đọc khó nhận ra ngôn ngữ sử dụng.

3. Mật mã thay thế - Hệ thống mật mã Affine

- Vị trí của mỗi chữ cái trong bảng chữ cái được ánh xạ tới vị trí tương đương mới của bảng chữ cái đó. Vị trí mới được tính thông qua một hàm toán học tuyến tính.
- các chữ cái của bảng chữ cái có kích thước m được ánh xạ tới các số nguyên từ 0 đến m-1.
- Key cho mật mã Affine bao gồm 2 số a và b. Điều kiện là **a phải nguyên tố cùng nhau với m.**
- Công thức về mặt toán học có dạng: $E(x) = (ax + b) \bmod m$
  - *m* là độ dài của bảng chữ cái.
  - *a* và *b*: khoá của mật mã.
  - gcd(a, m) = 1 → a là phần tử khả nghịch.
  - *x:* vị trí của ký tự trong bảng chữ cái gốc.
- Nếu a = 1, ta quay về bài toán mật mã Caesar.
- Khi giải mã, chúng ta thực hiện các chức năng ngược lại (hoặc nghịch đảo) trên bản mã để lấy ra bản rõ.

$$D(y) = a^{-1}(y - b) \bmod m$$

  - *y* là **số thứ tự** của chữ cái trong bảng chữ cái **mã hóa**.
  - **a^{-1}** là nghịch đảo modulo của a theo modulo m, nghĩa là $1 = a^{-1}a \bmod m$.
  - *D(y)* là số thứ tự của chữ cái trong bảng chữ cái **gốc**.

Ví dụ: Lấy k = (5,6)

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
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

Bản rõ:

"hentoithubay"

	h	e	n	t	o	i	t	h	u	b	a	y
x	7	4	13	19	14	8	19	7	20	1	0	24

y = 5x + 6 mod 26

y	15	0	19	23	24	20	23	15	2	11	6	22
	p	a	t	x	y	u	x	p	c	l	g	w

Bản mã:

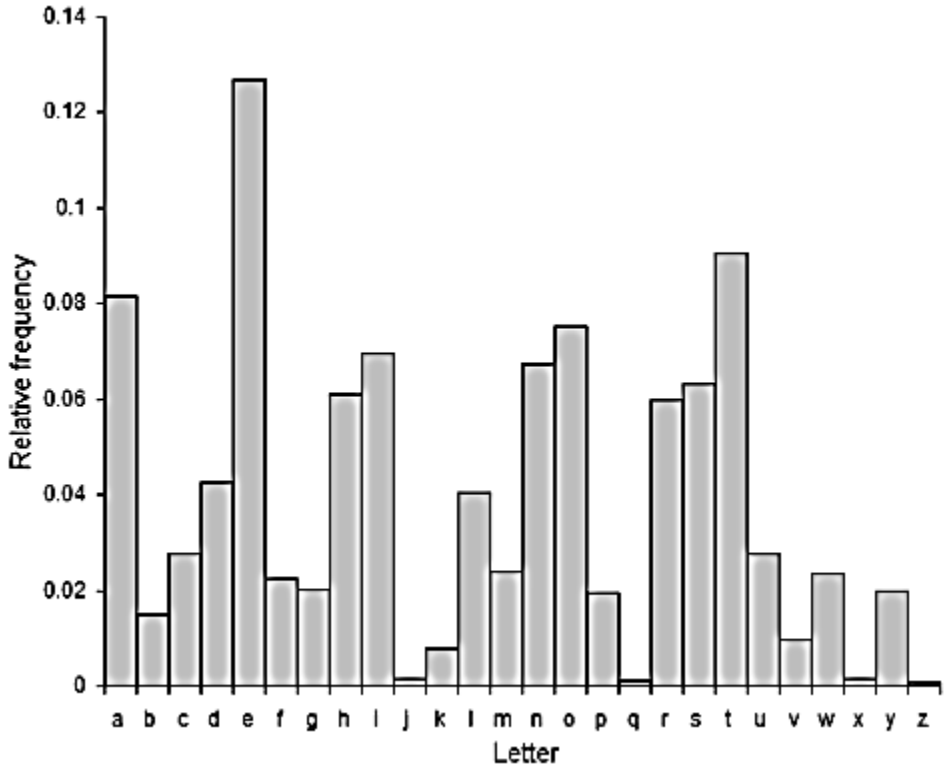
"patxyuxpclgw"

Thuật toán giải mã có dạng: $D(y) = 5^{-1}(y - 6) \bmod 26 = 21(y - 6) \bmod 26$

Cả mật mã Affine và mật mã Caesar đều dễ bị phá bằng phương pháp **phân tích tần số** (frequency analysis), vì chúng là mật mã thay thế đơn chữ cái.

Nguyên lý của phân tích tần số:

- Trong một ngôn ngữ (ví dụ: tiếng Anh), tần suất xuất hiện của các chữ cái không đồng đều:
  - Chữ cái xuất hiện nhiều nhất: E (khoảng 12.7%), tiếp theo là T (9.1%), A (8.2%), v.v.
  - Chữ cái ít xuất hiện: Z (0.074%), Q (0.095%), v.v.



- Trong mật mã thay thế đơn chữ cái, mỗi chữ cái trong văn bản gốc luôn được thay thế bởi cùng một chữ cái trong văn bản mã hóa. Do đó, tần suất xuất hiện của các chữ cái trong văn bản mã hóa sẽ phản ánh tần suất của các chữ cái trong văn bản gốc.
  - **Ví dụ:**
    - Nếu chữ D xuất hiện nhiều nhất trong văn bản mã hóa, ta có thể giả định D tương ứng với E (chữ cái phổ biến nhất trong tiếng Anh).
    - Giả sử H (vị trí 7) và Q (vị trí 16) là hai chữ cái xuất hiện nhiều nhất, ta giả định chúng tương ứng với E (vị trí 4) và T (vị trí 19) trong tiếng Anh.
  - Sau đó áp dụng các công thức toán học để tìm khóa.

4. Mật mã thay thế - Mật mã Vigenère

- Khóa được đưa ra bởi một cụm từ của các chữ cái d.
- **Chữ cái của bản mã được tìm thấy tại giao điểm của:**
  - **cột** được xác định **bằng chữ cái của bản rõ**, và
  - **dòng** được xác định **bằng chữ cái của khóa.**
- Công thức mã hóa của mật mã Vigenère là: $Vig_d(m_i) = (m_i + k_{i \bmod d}) (\bmod n)$
  - m_i: Số thứ tự của chữ cái thứ i trong văn bản gốc (plaintext).
  - $k_{i \bmod d}$: Số thứ tự của chữ cái thứ i trong khóa.
  - $Vig_d(m_i)$: Số thứ tự của chữ cái thứ iii trong văn bản mã hóa (ciphertext).
  - *n:* Kích thước bảng chữ cái (ví dụ: n=26).
- Công thức giải mã (nghịch đảo) là: $Vig_d^{-1}(c_i) = (c_i - k_{i \bmod d} + n) (\bmod n)$
  - c_i: Số thứ tự của chữ cái thứ i trong văn bản mã hóa.
  - $k_{i \bmod d}$: Số thứ tự của chữ cái thứ i trong khóa.
  - n: Kích thước bảng chữ cái.

- **Ví dụ:** Nếu khóa không đủ độ dài so với độ dài của bản rõ
  - **TH1: Lặp khóa:** Viết **lặp lại các ký tự trong khóa** từ phía sau của chuỗi khóa, đến khi nó bằng chiều dài của bản rõ thì thôi.
    - Văn bản gốc: "HELLO".
    - Khóa: "KEY" (K = 10, E = 4, Y = 24).
    - Độ dài khóa: d = 3.
    - Bảng chữ cái: n = 26.

    Lặp lại khóa để khớp với độ dài văn bản gốc (5 chữ cái): "KEYKE". Ta tính $k_{i \bmod d}$:

    - $i = 0$: $0 \bmod 3 = 0$, $k_0 = 10$ (chữ K).
    - $i = 1$: $1 \bmod 3 = 1$, $k_1 = 4$ (chữ E).
    - $i = 2$: $2 \bmod 3 = 2$, $k_2 = 24$ (chữ Y).
    - $i = 3$: $3 \bmod 3 = 0$, $k_0 = 10$ (chữ K).
    - $i = 4$: $4 \bmod 3 = 1$, $k_1 = 4$ (chữ E).

    Áp dụng công thức mã hóa $Vig_d(m_i) = (m_i + k_{i \bmod d}) \bmod 26$:

    - H (vị trí 7), $k_{0 \bmod 3} = 10$: $(7 + 10) \bmod 26 = 17 \rightarrow R$,
    - E (vị trí 4), $k_{1 \bmod 3} = 4$: $(4 + 4) \bmod 26 = 8 \rightarrow I$,
    - L (vị trí 11), $k_{2 \bmod 3} = 24$: $(11 + 24) \bmod 26 = 35 \bmod 26 = 9 \rightarrow J$,
    - L (vị trí 11), $k_{3 \bmod 3} = 10$: $(11 + 10) \bmod 26 = 21 \rightarrow V$,
    - O (vị trí 14), $k_{4 \bmod 3} = 4$: $(14 + 4) \bmod 26 = 18 \rightarrow S$.

    Văn bản mã hóa: "RIJVS".

Giải mã:

Dùng công thức $Vig_d^{-1}(c_i) = (c_i - k_{i \mod d} + 26) \mod 26$:

- R (vị trí 17), $k_{0 \mod 3} = 10$: $(17 - 10 + 26) \mod 26 = 7 \rightarrow H$,
- I (vị trí 8), $k_{1 \mod 3} = 4$: $(8 - 4 + 26) \mod 26 = 4 \rightarrow E$,
- J (vị trí 9), $k_{2 \mod 3} = 24$: $(9 - 24 + 26) \mod 26 = 11 \rightarrow L$,
- V (vị trí 21), $k_{3 \mod 3} = 10$: $(21 - 10 + 26) \mod 26 = 11 \rightarrow L$,
- O (vị trí 14), $k_{4 \mod 3} = 4$: $(14 - 4 + 26) \mod 26 = 14 \rightarrow O$.

Kết quả: "HELLO", đúng với văn bản gốc.

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

- **TH2: Khóa tự động: Viết các ký tự trong bản rõ** từ phía sau của chuỗi khóa, đến khi nó bằng chiều dài của bản rõ thì thôi.
  - M = ALLWORKANDNOPLAYMA
  - K = WHENINRO
  - AutoKey = K + M

M = A L L W O R K A N D N O P L A Y M A

Key = W H E N I N R O A L L W O R K A N D

C = W S P J W E B O N O Y K D C K Y Z D

- **Sự khác biệt giữa mật mã Vigenère và mật mã đơn chữ cái**
  - Mật mã Vigenère thuộc lớp **mật mã thay thế đa chữ cái** (polyalphabetic substitution cipher), khác với mật mã Caesar và mật mã Affine (mật mã thay thế đơn chữ cái). Sự khác biệt chính là:
    - **Mật mã đơn chữ cái (Caesar, Affine):**
      - Mỗi chữ cái trong văn bản gốc luôn được thay thế bởi cùng một chữ cái trong văn bản mã hóa.
      - Ví dụ: Trong mật mã Caesar với k = 3, A luôn thành D, B luôn thành E, v.v.
      - Điều này làm cho mật mã dễ bị phá bằng **phân tích tần số**, vì tần suất chữ cái trong văn bản mã hóa phản ánh trực tiếp tần suất trong văn bản gốc.
    - **Mật mã đa chữ cái (Vigenère):**
      - Một chữ cái trong văn bản gốc có thể được thay thế bởi các chữ cái khác nhau trong văn bản mã hóa, tùy thuộc vào vị trí của nó và chữ cái tương ứng trong khóa.
      - Ví dụ: Trong ví dụ trên, L ở vị trí 3 được mã hóa thành J (vì khóa là Y = 24), nhưng L ở vị trí 4 được mã hóa thành V (vì khóa là K = 10).
      - **Điều này làm cho tần suất chữ cái trong văn bản mã hóa không còn phản ánh trực tiếp tần suất trong văn bản gốc**, khiến phân tích tần số thông thường trở nên vô nghĩa.

- **Phá mật mã Vigenère bằng phương pháp Kasiski**
  - Mật mã Vigenère khó phá hơn mật mã Caesar hay Affine, nhưng vẫn có thể bị phá bằng **phương pháp Kasiski**, do nhà toán học người Đức Friedrich Kasiski đề xuất vào thế kỷ XIX.
  - **Nguyên lý của phương pháp Kasiski:**
    - **Tìm các đoạn lặp lại trong văn bản mã hóa:**
      - Nếu hai đoạn văn bản mã hóa giống hệt nhau, có khả năng chúng được mã hóa từ hai đoạn văn bản gốc giống nhau, và chúng được mã hóa bằng cùng một đoạn của khóa.
      - Khoảng cách giữa hai đoạn lặp lại sẽ là bội số của độ dài khóa.
    - **Xác định độ dài khóa (l):**
      - **Tìm tất cả các đoạn lặp lại** trong văn bản mã hóa và **tính khoảng cách giữa chúng.**
      - **Ước chung lớn nhất** (hoặc các ước số chung) **của các khoảng cách này thường là độ dài của khóa.**
    - **Phân tích tần số từng nhóm:**
      - Khi đã biết độ dài khóa l, chia văn bản mã hóa thành l nhóm, mỗi nhóm chứa các chữ cái được mã hóa bằng cùng một chữ cái của khóa.
      - Mỗi nhóm giờ đây tương đương với một mật mã Caesar (vì cùng một chữ cái khóa được sử dụng), và ta có thể áp dụng phân tích tần số để tìm từng chữ cái của khóa.

- Để tăng độ dài của cụm mật khẩu và độ phức tạp của phân tích mật mã, bạn có thể sử dụng **thành phần mật mã Vigenere,** là **mã hóa nhiều Vigenere với các cụm mật khẩu khác nhau.**
    - Thay vì chỉ sử dụng một cụm mật khẩu, ta sử dụng nhiều cụm mật khẩu (ví dụ: k, l, ..., s).
    - Mỗi cụm mật khẩu có độ dài riêng (ví dụ: d_k, d_l,..., d_s).
    - Các cụm mật khẩu được áp dụng đồng thời lên văn bản gốc, tạo ra một chu kỳ lặp lại phức tạp hơn.
    - **Công thức mã hóa:**

      $$Vig^*(m_i) = (m_i + k_{i \mod m_k} + l_{i \mod m_l} + \cdots + s_{i \mod m_s}) \mod n$$

      Trong đó:

      - m_i: Số thứ tự của chữ cái thứ i trong văn bản gốc (plaintext).
      - $k_{i \mod m_k}$: Số thứ tự của chữ cái trong cụm mật khẩu k tại vị trí $i \mod m_k$.
      - $l_{i \mod m_l}$: Số thứ tự của chữ cái trong cụm mật khẩu l tại vị trí $i \mod m_l$.
      - $s_{i \mod m_s}$: Số thứ tự của chữ cái trong cụm mật khẩu s tại vị trí $i \mod m_s$.
      - $m_k, m_l, ..., m_s$: Độ dài của các cụm mật khẩu $k, l, ..., s$.
      - n: Kích thước bảng chữ cái (ví dụ: $n = 26$ với bảng chữ cái tiếng Anh).
      - $Vig^*(m_i)$: Số thứ tự của chữ cái thứ i trong văn bản mã hóa (ciphertext).

      Công thức giải mã (nghịch đảo):

      Để giải mã, ta thực hiện phép nghịch đảo của hàm mã hóa:

      $$Vig^{*-1}(c_i) = (c_i - (k_{i \mod m_k} + l_{i \mod m_l} + \cdots + s_{i \mod m_s}) + n) \mod n$$

      Trong đó:

      - c_i: Số thứ tự của chữ cái thứ i trong văn bản mã hóa.
      - Các ký hiệu khác tương tự như trên.
    - **Tại sao sử dụng nhiều cụm mật khẩu làm tăng độ phức tạp?**

Chu kỳ lặp lại phức tạp hơn:

- Trong mật mã Vigenère thông thường, nếu độ dài khóa là d, thì chu kỳ lặp lại của khóa là d. Điều này có nghĩa là cứ sau d ký tự, khóa sẽ lặp lại, và các đoạn lặp lại trong văn bản mã hóa có thể được dùng để xác định d (bằng phương pháp Kasiski).
- Với nhiều cụm mật khẩu ($k, l, ..., s$) có độ dài $m_k, m_l, ..., m_s$, chu kỳ lặp lại tổng thể của hệ thống sẽ là **bội số chung nhỏ nhất (LCM)** của các độ dài $m_k, m_l, ..., m_s$:
    - Ví dụ: Nếu $m_k = 3$, $m_l = 4$, chu kỳ lặp lại là $\text{LCM}(3, 4) = 12$.
    - Nếu các độ dài $m_k, m_l, ..., m_s$ là các số nguyên tố cùng nhau (coprime), thì chu kỳ lặp lại sẽ bằng tích của chúng: $m_k \cdot m_l \cdot \ldots \cdot m_s$.

Tăng độ dài chu kỳ lặp lại:

- Chu kỳ lặp lại dài hơn làm cho phương pháp Kasiski trở nên khó khăn hơn:
  - Phương pháp Kasiski dựa vào việc tìm các đoạn lặp lại trong văn bản mã hóa để suy ra độ dài khóa.
  - Nếu chu kỳ lặp lại là $m_k \cdot m_l \cdot \ldots \cdot m_s$, số lượng ký tự cần thiết để tìm các đoạn lặp lại sẽ lớn hơn nhiều, đặc biệt nếu các độ dài $m_k, m_l, \ldots, m_s$ lớn và nguyên tố cùng nhau.

Tăng độ phức tạp của phân tích tần số:

- Trong mật mã Vigenère thông thường, sau khi xác định độ dài khóa d, ta có thể chia văn bản mã hóa thành d nhóm và áp dụng phân tích tần số cho từng nhóm (mỗi nhóm tương ứng với một mật mã Caesar).
- Với nhiều cụm mật khẩu, mỗi chữ cái trong văn bản mã hóa là kết quả của nhiều phép dịch (từ nhiều cụm mật khẩu). Điều này làm cho tần suất chữ cái trong văn bản mã hóa trở nên phức tạp hơn, không còn phản ánh trực tiếp tần suất trong văn bản gốc, ngay cả khi đã chia thành các nhóm.

Khó xác định từng cụm mật khẩu:

- Để phá mã, kẻ tấn công cần xác định từng độ dài $m_k, m_l, \ldots, m_s$, sau đó tìm từng cụm mật khẩu $k, l, \ldots, s$. Điều này đòi hỏi phải giải nhiều hệ phương trình phức tạp hơn so với mật mã Vigenère thông thường.
- Nếu các độ dài $m_k, m_l, \ldots, m_s$ là các số nguyên tố cùng nhau, việc tìm chu kỳ lặp lại và tách biệt các cụm mật khẩu sẽ càng khó khăn.

5. Hình vuông Polybus

- Bảng chữ cái được sắp xếp vào một ma trận (thường là 5×5 hoặc 6×6, tùy thuộc vào kích thước bảng chữ cái).
- Mỗi chữ cái được ánh xạ thành một cặp số (hàng, cột) tương ứng với vị trí của nó trong ma trận.
- Để mã hóa, mỗi chữ cái được thay thế bằng cặp số tọa độ của nó.
- Để giải mã, từ cặp số tọa độ, ta tìm lại chữ cái tương ứng trong ma trận.
- Ma trận được giữ bí mật và tạo thành khóa mã hóa.

	1	2	3	4	5
1	A	B	C	D	E
2	F	G	H	I, J	K
3	L	M	N	O	P
4	Q	R	S	T	U
5	V	W	X	Y	Z

- **Ví dụ:**
  - Câu mẫu: TRUONG GIAO THONG VAN TAI TPHCM

$$\begin{aligned} T &\to (4,4) \to 44, \\ R &\to (4,2) \to 42, \\ U &\to (4,5) \to 45, \\ O &\to (3,4) \to 34, \\ N &\to (3,3) \to 33, \\ G &\to (2,2) \to 22, \end{aligned}$$

$$\begin{aligned} G &\to (2,2) \to 22, \\ I &\to (2,4) \to 24, \\ A &\to (1,1) \to 11, \\ O &\to (3,4) \to 34, \end{aligned}$$

$$\begin{aligned} T &\to (4,4) \to 44, \\ H &\to (2,3) \to 23, \\ O &\to (3,4) \to 34, \\ N &\to (3,3) \to 33, \\ G &\to (2,2) \to 22, \end{aligned}$$

$$\begin{aligned} V &\to (5,1) \to 51, \\ A &\to (1,1) \to 11, \\ N &\to (3,3) \to 33, \\ T &\to (4,4) \to 44, \\ A &\to (1,1) \to 11, \\ I &\to (2,4) \to 24, \end{aligned}$$

$$\begin{aligned} T &\to (4,4) \to 44, \\ P &\to (3,5) \to 35, \\ H &\to (2,3) \to 23, \\ C &\to (1,3) \to 13, \\ M &\to (3,2) \to 32. \end{aligned}$$

  - Mật mã: 444245343322 22241134 4423343322 511133 441124 4435231332
  - **Giải mã:** Chú ý **24 → I/J,** nên 24 có thể là I hoặc J. Tuy nhiên, dựa vào ngữ nghĩa của văn bản gốc ta chọn I cho các vị trí này (vị trí 8 và 22), vì "GIAO" và "TAI" có nghĩa trong tiếng Việt.

- Hình vuông Polybius là một dạng mật mã thay thế đơn chữ cái (monoalphabetic substitution cipher), vì mỗi chữ cái trong văn bản gốc luôn được ánh xạ thành một cặp số cố định.
- Tuy nhiên, thay vì thay thế bằng một chữ cái khác, nó thay thế bằng một cặp số (bigram). Do đó, ta có thể áp dụng **phân tích tần số** trên các bigram (cặp số) trong văn bản mã hóa:
  - Các bigram xuất hiện nhiều nhất trong văn bản mã hóa sẽ tương ứng với các chữ cái phổ biến nhất trong ngôn ngữ (ví dụ: trong tiếng Anh, E, T, A; trong tiếng Việt, có thể là A, N, H).
  - Sau khi xác định một số bigram phổ biến, ta thay thế chúng và dùng ngữ nghĩa của văn bản để suy ra các chữ cái còn lại.
- **Tổng số khóa (các cách sắp xếp bảng chữ cái)**
  - Ma trận 5×5 có 25 ô, chứa 25 ký tự (gộp I và J).
  - Tổng số cách sắp xếp 25 ký tự vào 25 ô là $25!$ (giai thừa của 25):

$$25! = 25 \times 24 \times 23 \times \cdots \times 1$$

    Đây là một số rất lớn, khoảng 1.55×10^{25}, cho thấy số lượng khóa có thể rất lớn, làm tăng độ an toàn của mật mã.
  - Nếu bảng chữ cái có kích thước khác (ví dụ: $6 \times 6 = 36$ ô), số khóa sẽ là $36!$, v.v.

  → Tổng số khóa (các tùy chọn khác nhau để đặt bảng chữ cái trong ma trận) là n!.

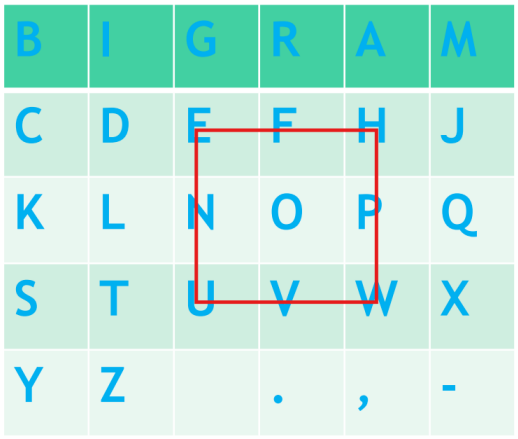
6. Mã hoá BIGRAM (Mã hóa PLAYFAIR)

- các cặp chữ cái (bigram) trong văn bản gốc (plaintext) được thay thế bằng các cặp chữ cái khác trong văn bản mã hóa (ciphertext) dựa trên một bảng thay thế (thường là ma trận).
- Phương pháp này **thuộc nhóm mật mã thay thế** (substitution cipher), nhưng thay vì thay thế từng chữ cái (như trong mật mã Caesar hay Affine), nó thay thế từng cặp chữ cái (bigram).

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

(ở đây từ khóa được chọn là BIGRAM, nếu chọn từ khóa khác, các ký tự trong ma trận sẽ có sự bố trí khác).

- **Nguyên tắc:**
  - Nếu cả hai chữ cái bigram của văn bản gốc **thuộc cùng một cột** của bảng, thì các chữ cái của mật mã được coi là các **chữ cái nằm bên dưới chúng**. Do đó, ký tự RF cung cấp ký tự mật mã FO. (R → F, F → O)
    - Nếu **chữ cái của bản rõ ở hàng dưới cùng**, thì chữ cái **tương ứng từ hàng trên cùng được lấy làm mật mã**, ví dụ bigram SY cho ký tự mật mã YB. (S → Y, Y → B)
    - Biểu đồ của một chữ cái hoặc một cặp chữ cái giống hệt nhau cũng tuân theo quy tắc này và văn bản **GG** cung cấp cho mật mã **EE**.
  - Nếu cả hai chữ cái bigram của văn bản gốc **thuộc cùng một hàng** của bảng, thì các **chữ cái ở bên phải chúng** được coi là chữ cái của mật mã. Do đó, biểu đồ GR cung cấp văn bản mật mã RA. (G → R, R → A)
    - Nếu **ký tự** của bản rõ nằm **ở cột ngoài cùng bên phải**, thì một ký tự từ **cột ngoài cùng bên trái của cùng một hàng** được lấy cho mật mã và biểu đồ AM cung cấp cho mật mã MB. (A → M, M → B)
  - Nếu cả hai chữ cái bigram của văn bản gốc **nằm ở các hàng và cột khác nhau**, thì hai chữ cái như vậy được lấy thay thế, để cả bốn chữ cái đó đại diện cho **một hình chữ nhật**. Ví dụ: bigram EW được mã hóa là HU.



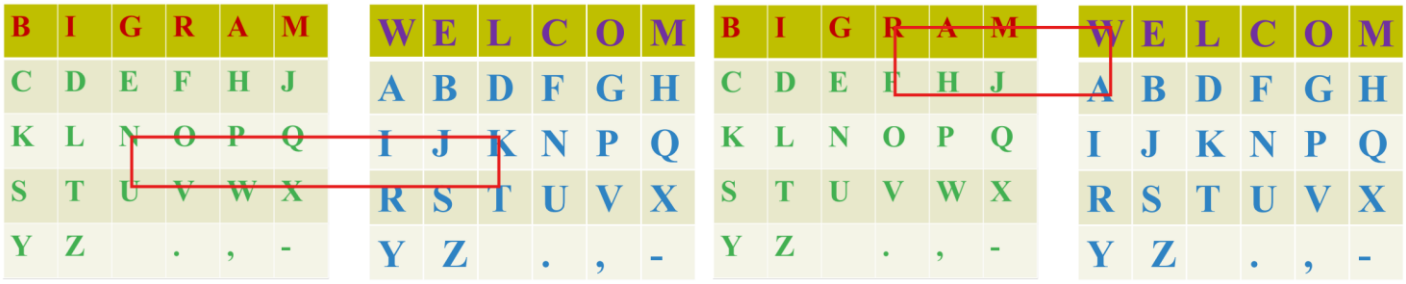
- Việc điền vào ô vuông với bảng chữ cái **có thể là ngẫu nhiên**, hoặc nó có thể được xác định bởi **một cụm từ khóa** nhất định. Tất cả các ký hiệu trong cụm từ khóa đó (nhưng không lặp lại) **được viết ở đầu ma trận** và sau đó là các chữ cái khác của bảng chữ cái được viết ra theo thứ tự.
- **Ví dụ:**
  - Key: "BIGRAM".
  - Văn bản gốc (plaintext): "HELLO-".
  - Bản mã: mã hóa từng cặp ký tự.
    - "HE": cùng hàng nên H → J, E → F
    - "LL": 2 ký tự giống nhau nên L → T.
    - "O-": 2 ký tự thuộc 2 nơi khác nhau → Tạo thành hình chữ nhật, do đó O → Q, - → .
    - Vậy bản mã là: JFTTQ.

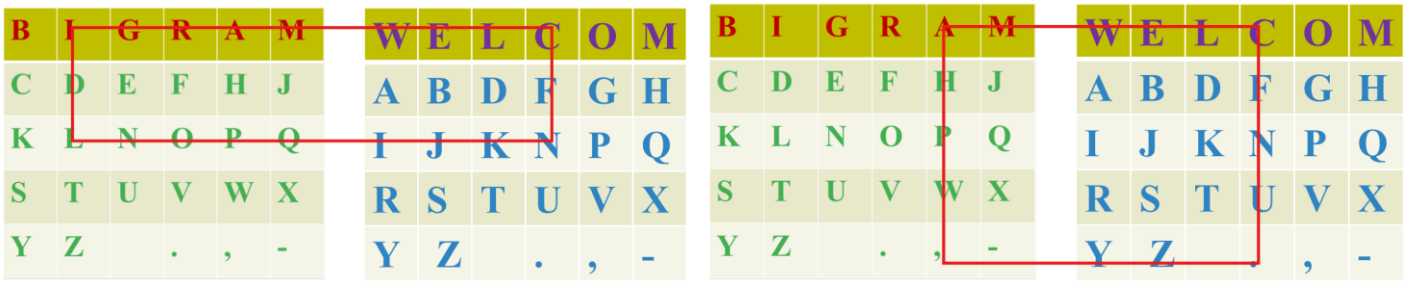
- **MÃ HOÁ BIGRAM VỚI HAI BẢNG VUÔNG**
  - Mỗi cặp ký tự (**bigram**) từ bản rõ sẽ được mã hóa bằng cách sử dụng hai bảng khác nhau.
  - **Nếu hai ký tự không nằm trên cùng một hàng**, chúng sẽ tạo thành một hình chữ nhật trong hai bảng. Hai ký tự được mã hóa sẽ là **hai góc còn lại của hình chữ nhật**.
  - **Nếu hai ký tự nằm trên cùng một hàng**, chúng sẽ được thay thế bằng ký tự **cùng hàng nhưng ở cột tiếp theo**. Nếu ký tự ở cột cuối cùng, nó sẽ quay lại cột đầu tiên.
  - **Ví dụ:**

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

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

    - Key: **BIGRAM** và **WELCOM**
    - Bản rõ: **UKRAINA.** → Bản mã: **TNWFCLC,**





Độ bền mật mã:

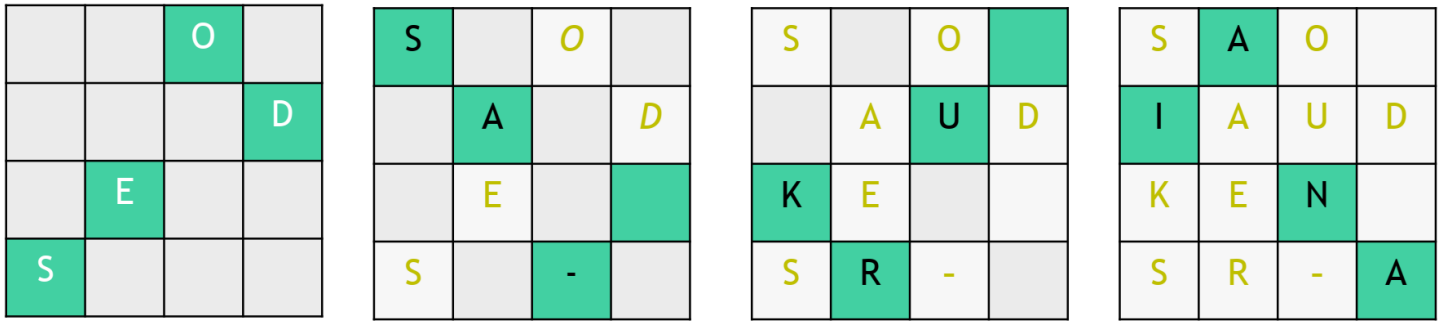
- Playfair mạnh hơn các phương pháp thay thế đơn chữ cái (như Caesar hay Affine) vì:
  - Nó hoạt động trên bigram, làm tăng số lượng kết hợp có thể ($26 \times 26 = 676$ bigram so với 26 chữ cái).
  - Cùng một chữ cái có thể được mã hóa thành các chữ cái khác nhau tùy thuộc vào chữ cái đi cùng trong bigram.
- Tuy nhiên, Playfair vẫn có thể bị phá bằng phân tích tần số bigram:
  - Trong một ngôn ngữ (như tiếng Anh), một số bigram xuất hiện thường xuyên hơn (ví dụ: "TH", "HE", "IN").
  - Phân tích tần suất bigram trong văn bản mã hóa có thể giúp suy ra các ánh xạ.

Số lượng khóa:

- Số cách sắp xếp bảng Playfair là $25!$ (vì ma trận 5×5 có 25 ô, gộp I và J).
- Tuy nhiên, trong thực tế, từ khóa thường ngắn hơn, nên số lượng khóa thực tế nhỏ hơn $25!$, nhưng vẫn rất lớn, làm cho việc thử tất cả các tùy chọn (brute force) trở nên khó khăn.

7. Hình vuông cardano

- Đây là một kỹ thuật mã hóa dựa trên việc sử dụng một lưới (grid) có các ô được đục lỗ (hoặc đánh dấu) để che giấu thông điệp.
- Để mã hóa, nhà toán học và triết học người Ý J. Cardano đề xuất sử dụng một hình vuông với một số lỗ hở trên đó.
- Các lỗ được cắt theo cách sao cho khi xoay hình vuông 90, 180 và 270 độ, tất cả các vị trí của hình vuông ban đầu lần lượt xuất hiện trong các khe.
- Sau đó viết các ký tự của thông điệp vào các ô được đục lỗ.
- **Mã hóa**:
  - **Đặt lưới lên một tờ giấy N×N.**
  - Viết các ký tự của thông điệp vào các ô đục lỗ.
  - Xoay lưới 90 độ (theo chiều kim đồng hồ hoặc ngược lại), đặt lại lên tờ giấy, và tiếp tục viết các ký tự tiếp theo của thông điệp.
  - **Lặp lại bước xoay 4 lần (0°, 90°, 180°, 270°),** mỗi lần điền một phần thông điệp.
  - Sau 4 lần xoay, *các ô còn lại (không thuộc ô đục lỗ) được điền bằng các ký tự ngẫu nhiên để che giấu thông điệp (hoặc có thể để trắng tùy TH).*
- **Giải mã**:
  - Để giải mã thông điệp, bạn cần có một bản sao chính xác của hình vuông được sử dụng để mã hóa **(vị trí của các khe trên hình vuông là khóa)**
  - Sử dụng cùng một lưới Cardano.
  - Đặt lưới lên tờ giấy đã mã hóa, đọc các ký tự trong các ô đục lỗ.
  - Xoay lưới 90 độ và lặp lại 4 lần để đọc toàn bộ thông điệp.
- **Ví dụ:** Thông tin cần mã hóa: **ODESSA – UKRAINA**



0 độ **90 độ** **180 độ** **270 độ**

**** Các ô xanh là các ô đục lỗ**

Sau đó ta thu được bản mã bằng cách viết từng dòng của ma trận. Bản mã: **SAO IAUDKEN SR-A**

- Số lượng các tùy chọn khác nhau cho vị trí của các khe trong hình vuông NxN là $4^{\left(\frac{N^2}{4}\right)}$, đối với hình vuông 6x6 cho 262.144 tùy chọn (tương đương với khóa 18 bit).
- Tuy nhiên, mật mã này (cũng như tất cả các hoán vị) bị suy yếu trong thực tế.
- Trong quá trình phân tích mật mã, có thể sử dụng các đặc điểm của ngữ âm của ngôn ngữ quốc gia bản địa (các tổ hợp ký hiệu phổ biến nhất hoặc không được chấp nhận cho ngôn ngữ này, độ dài trung bình của từ, v.v.).

8. Mật mã hoán vị với một từ khoá

- Các chữ cái của từ khóa **không lặp lại** được viết ở hàng đầu tiên của bảng, từ đó **xác định số cột của bảng** đó.
- Các chữ cái của tin nhắn được ghi lại trong bảng theo **từng dòng**, **từ trái sang phải, từ trên xuống dưới**, bao gồm khoảng trắng nếu có.
- Nếu thông điệp không đủ để điền đầy bảng, thêm ký tự đệm (thường là X).
- Bảng được hình thành theo cách này được sắp xếp theo cột, tiêu chí sắp xếp là thứ tự truyền ký tự của dòng đầu tiên trong bảng (ở ví dụ dưới, ta lật ngược lại bảng theo cột).
- Sau khi phân loại, văn bản được mã hóa được viết lại theo từng **cột**.
- **Để giải mã** ta viết chuỗi đã mã hóa lên bảng theo từng cột, sau đó lật ngược bảng, và viết lại bản rõ theo từng dòng.
- **Ví dụ:**

Key: **CALL**

Thông tin: **ODESSA – UKRAINA**

C	A	L	L
O	D	E	S
S	A		-
	U	K	R
A	I	N	A

L	L	A	C
E	S	D	O
	-	A	S
K	R	U	
N	A	I	A

Mã hoá: **E KNS-RADAUIOS A**