



المعهد التحضيري للدراسات الهندسية بنابل
Institut Préparatoire aux Etudes d'Ingénieurs de Nabeul



Université de Carthage
University of Carthage
جامعة قرطاج

Module : Informatique

Classes: 2^{ème} Années MP – PC – T

PARTIE III: Simulation numérique

Thème 2: Cryptographie

Enseignant : Dr. Khemaies GHALI (ghali_khemaies@yahoo.fr)

Page Facebook: <https://www.facebook.com/groups/ipein.info/>

Année universitaire 2018/2019



Plan

- ☐ Introduction à la cryptographie
- ☐ Code de César
- ☐ Chiffre affine
- ☐ Code de Vigenère
- ☐ Méthode Hill
- ☐ Méthode RSA
- ☐ Codage de Hamming

Introduction à la cryptographie (1/3)



Fig.1: Protocole de chiffrement

- ❑ **Cryptologie** : Il s'agit d'une science mathématique comportant deux branches : la cryptographie et la cryptanalyse
- ❑ **Cryptographie** : La cryptographie est l'étude des méthodes donnant la possibilité d'envoyer des données de manière confidentielle sur un support donné.

Introduction à la cryptographie (2/3)

- ❑ **Confidentialité:** Transmettre un message sans qu'un intermédiaire ne puisse le comprendre:
 - codage au départ puis décodage à l'arrivée.
- ❑ **Cryptage:** Transformation d'un message clair en un message crypté.
- ❑ **Décryptage:** transformation inverse du cryptage qui permet de retrouver à partir d'un message crypté, le message clair correspondant.
- ❑ **Clé k:** Il s'agit du paramètre impliqué et autorisant des opérations de chiffrement et/ou déchiffrement.
 - Dans le cas d'un **algorithme symétrique**, la clef est identique lors des deux opérations.
 - Dans le cas **d'algorithmes asymétriques**, elle diffère pour les deux opérations.

Introduction à la cryptographie (3/3)

- ❑ **Cryptographie asymétrique:** ssl, ssh, https, clés publiques, clés privées.
- ❑ **Cryptanalyse:** Opposée à la cryptographie, "casser" un code, i.e. découvrir les méthodes de codage ou de décodage en observant des messages codes (Turing).
- ❑ **Diverses méthodes:** César, Vigenère, Hill, Enigma et Turing, RSA
- ❑ **Protocole:** description de l'ensemble des données nécessaires pour mettre en place le mécanisme de cryptographie (ensembles des messages claires, les messages cryptés, les clés possibles,...)

Le code César

- ❑ Il s'agit d'un des plus simples et des chiffres classiques les plus populaires. Son principe est un décalage des lettres de l'alphabet.
- ❑ Dans les formules ci-dessous, « p » est l'indice de la lettre de l'alphabet, « k » est le décalage.
 - Pour le chiffrement, on aura la formule:
$$C = E(p) = (p + k) \bmod 26$$
 - Pour le déchiffrement, il viendra:
$$p = D(C) = (C - k) \bmod 26$$
- ❑ Si on connaît l'algorithme utilisé (ici César), la cryptanalyse par force brute est très facile.
 - En effet, dans le cas du chiffre de César, seules 25 (!) clés sont possibles $k \in [0, 25]$.

César : En Python

```
#cesar.py : codage et décodage en code César
def cesar_char(c,d):
    alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
    n = len(alphabet)
    for i in range(len(alphabet)):
        if c == alphabet[i]: return(alphabet[(i+d)%n])
    return(c)
#codeur sésar
def cesar(msg,d):
    msg_code = ""
    for c in msg: msg_code = msg_code + cesar_char(c,d)
    return(msg_code)
#décodeur sésar
def decesar(msg,d):
    return(cesar(msg,-d))
```

```
>>> msg = "CE N'EST PAS FACILE DE LIRE LES MESSAGES SECRETS!"
>>> mc = cesar(msg,10)
>>> mc
"MO X'OCD ZKC PKMSVO NO VSBO VOC WOCKQOC COMBODC!"
>>> decesar(mc,10)
"CE N'EST PAS FACILE DE LIRE LES MESSAGES SECRETS!"
....
```

Cryptanalyse du code de César (1/3)

Analyse fréquentielle:

- Lorsque la langue de départ et la technique de chiffrement sont connus, on peut **exploiter les régularités du langage** par le principe d'analyse de la fréquence d'une lettre.
- Cette technique ne fonctionne bien que si le **message chiffré est suffisamment long pour avoir des moyennes significatives.**

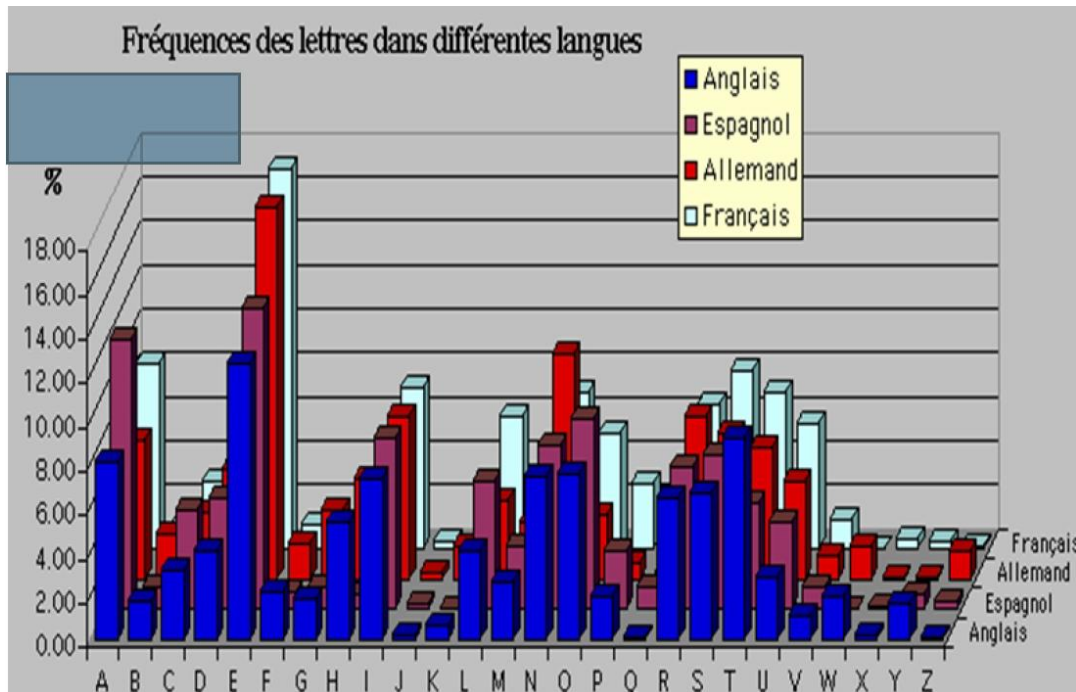


Fig.2: Fréquences des lettres dans différentes langues

Cryptanalyse du code de César (2/3)

❑ Alors la lettre "E" est la plus fréquente en français:

➤ on cherche la lettre la plus fréquente dans le message codé!

```
1. def rang(c):
2.     return(ord(c)-65)
3. def frequences(m):
4.     Lf = []
5.     for c in m:
6.         if rang(c) >= 0 and rang(c) <= 25:
7.             f = len([x for x in m if x == c])
8.             if not ((c,f) in Lf):
9.                 Lf = Lf + [(c,f)]
10.    #L = sorted(Lf, key=lambda e: - e[1])
11.    L = sorted(Lf, key=lambda e: e[1],
12.              reverse=True)
13.    return L
```

Cryptanalyse du code de César (3/3)

```
1. def cassecesar(m) :  
2.     Lf = frequences(m)  
3.     Em = Lf[0][0]  
4.     d = (rang(Em) - rang("E")) % 26  
5.     return(decesar(m,d), d)
```

```
1. >>> MC="GHFUBSWHU XQ PHVVDJH, F'HVW SDUIRLV XWLOH!\n2. Q'HVW-FH-SDV?"  
3. >>> cassecesar(MC)  
4. ("DECRYPTER UN MESSAGE, C'EST PARFOIS UTILE!N'EST-CE-  
   PAS?", 3)  
5. >>>
```

Chiffre affine (1/2)

❑ On dit qu'une fonction est affine lorsqu'elle est de la forme $x \rightarrow ax + b$, c'est-à-dire un polynôme de degré 1.

❑ L'idée est d'utiliser comme fonction de chiffrement une fonction affine du type:

$$y = (ax + b) \bmod 26,$$

➤ où a et b sont des constantes, et où x et y sont des nombres correspondant aux lettres de l'alphabet ($A=0, B=1, \dots$).

➤ On peut remarquer que si $a = 1$, alors on retrouve le chiffre de César où b est le décalage (le k du chiffre de César).

❑ **Propriété de neutralité** : si $b = 0$, alors "A" est toujours chiffré "A" car il ne subit aucun décalage!

Chiffre affine (2/2)

- ❑ Pour le chiffre affine, la clé est constituée de (k_1, k_2) :
 - où $k_1, k_2 \in [0, 25]$ et
 - telle que $\text{PGCD}(k_1, 26) = 1$ (c.-à-d. k_1 et 26 premiers entre eux).
- ❑ Le chiffrement en lui-même est donné par:
$$c_i = f(m_i) = k_1 * m_i + k_2 \text{ mod } 26$$
- ❑ Pour le déchiffrement:
$$m_i = f^{-1}(c_i) = k_1^{-1} (c_i - k_2) \text{ mod } 26$$
- ❑ Par le chiffre affine, on obtient 312 clés possibles. En effet,
 - pour obéir à la propriété de k_1 , il n'y a que 12 choix possibles.
 - Et puisque k_2 peut prendre n'importe quelle valeur dans $[0, 25]$,
 - il vient $12 * 26 = 312$.

Chiffre affine : Exemple

❑ Soient la clé = $(k_1, k_2) = (3, 11)$

❑ Transformation de chiffrement :

$$c_i = f(m_i) = 3 * m_i + 11 \bmod 26$$

❑ Transformation de déchiffrement :

$$m_i = f^{-1}(c_i) = k_1^{-1} (c_i - k_2) \bmod 26$$

$$\text{or par définition: } k_1 * k_1^{-1} \bmod 26 = 1$$

$$\Rightarrow 3 * 9 \bmod 26 = 1$$

$$\text{d'où } k_1^{-1} = 3^{-1} \bmod 26 = 9$$

$$m_i = f^{-1}(c_i) = 9 * (c_i - 11) \bmod 26$$

❑ Ainsi, pour une suite de lettres telle que:

$$\text{'NSA'} \rightarrow 13 \ 18 \ 0 \rightarrow 24 \ 13 \ 11 \rightarrow \text{'YNL'}$$

Cryptanalyse du chiffre affine (1/2)

1. Il faut tout d'abord établir la fréquence relative de chaque lettre du texte chiffré, par analyse de fréquence.

"HGAHY RAEFT GAGRH DGAGM OEHIY RAAOT ZGAGJ
GKFDG AZGSB INNTG KGRHE NNIRG"

- On dénombre 12 fois la lettre "G" et 8 fois la lettre "A".

Supposons que le langage original du texte est le français.

2. Sur base de l'analyse de fréquences (Fig.2), il faut ensuite dériver les équations correspondantes. E, A, S, I,

- En émettant l'hypothèse que le bigramme « ES » est plus fréquent que « EA », on en déduit les équations suivantes:

$$\left[\begin{array}{l} E \rightarrow G \text{ d'où } f(E) = G \\ S \rightarrow A \text{ d'où } f(S) = A \end{array} \right] \longrightarrow \left[\begin{array}{l} f(4) = 6 \\ f(18) = 0 \end{array} \right]$$

Cryptanalyse du chiffre affine (2/2)

3. On peut maintenant résoudre les équations pour retrouver k_1 et k_2 :

$$\begin{cases} f(4) = 6 \\ f(18) = 0 \end{cases} \longrightarrow \begin{cases} 4 * k_1 + k_2 = 6 \bmod 26 & (1) \\ 18 * k_1 + k_2 = 0 \bmod 26 & (2) \end{cases}$$

$$(1) - (2) \rightarrow -14k_1 = 6 \bmod 26 \rightarrow 12k_1 = 6 \bmod 26 \text{ d'où } k_1 = 7$$

$$(2) \rightarrow 18*7+k_2 = 0 \bmod 26 \rightarrow 22+k_2 = 0 \bmod 26$$

$$\rightarrow k_2 = -22 \bmod 26 \rightarrow k_2 = 4$$

4. La fonction de déchiffrement est donc la suivante :

$$\begin{cases} m_i = k_1^{-1} (c_i - k_2) \bmod 26 \\ k_1^{-1} * k_1 = 1 \bmod 26 \end{cases} \longrightarrow m_i = 15 * (c_i - 4) \bmod 26$$

Et donc: "HGAHYRAEFTGAGRHDGAGMOEHIYRAAOTZGAGJGKFDGAZGSB

INNTGKGRHENNIRG" devient:

"TESTONSAPRESENTLESEQUATIONSSURDESEXEMPLESDECHIFFREMENTAFFINE"

Code de Vigenère (1/2)

- ❑ Le **chiffre de Vigenère** (1568) est un système de chiffrement **polyalphabétique** (c'est-à-dire qu'on (dé)chiffre par paquets et pas par des lettres les unes après les autres):
 - c'est un chiffrement par **substitution**,
 - mais une même lettre du message clair peut être remplacée par des lettres différentes, contrairement à un système de chiffrement **mono-alphabétique** comme le chiffre de César,
- ❑ Cette méthode **résiste ainsi à l'analyse de fréquences classique**, ce qui est un avantage décisif sur les chiffrements mono-alphabétiques.
- ❑ Cependant le **chiffre de Vigenère a été percé** par *Friedrich Kasiski* et dont la méthode a été publié en 1863.

Code de Vigenère (2/2)

- ❑ **Principe:** code de César a décalage variable, donne par une clé secrète.
- ❑ **Exemple:** avec la clé "SALUT", "IPEI NABEUL" se code "APPC GSBPOE"
- ❑ **Décodage:** avec la même clé, et le décodage de César.
- ❑ **Cryptanalyse:**
 - analyse fréquentielle:
 - ❖ recherche de répétitions de motifs,
 - ❖ longueur de la clé déduite des périodes des motifs par pgcd,
 - puis cryptanalyse de César.

Vigenère : Chiffrement (1/2)

- ❑ La Table (ou carré) de Vigenère est donnée sur la diapo suivante:
 1. Pour chaque lettre en clair, on sélectionne la colonne correspondante,
 2. et pour une lettre de la clé on sélectionne la ligne adéquate,
 3. puis au croisement de la ligne et de la colonne on trouve la lettre chiffrée.
- ❑ La lettre de la clé est à prendre dans l'ordre dans laquelle elle se présente,
 - et on répète la clé en boucle autant que nécessaire.

Vigenère : Chiffrement (2/2)

Exemple:

❑ **Clé** : musique

❑ **Texte** : J'adore écouter la radio toute la journée

Texte en clair : j'adore ecouter la radio toute la journee

Clé répétée : M USIQU EMUSIQU EM USIQU EMUSI QU EMUSIQU

^ ^^^

| ||Colonne O, ligne I : on obtient la lettre W.

| |Colonne D, ligne S : on obtient la lettre V.

| Colonne A, ligne U : on obtient la lettre U.

Colonne J, ligne M : on obtient la lettre V.

❑ **Le texte chiffré est alors :**

V'UVWHY IOIMBUL PM LSLYI XAOLM BU NAOJVUY

Table de Vigenère

		Lettre en clair																									
		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
C l é u t i l i s é e	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

Vigenère : Déchiffrement

❑ Si on veut déchiffrer ce texte:

1. On regarde pour chaque lettre de la clé répétée la ligne correspondante,
2. Et on y cherche la lettre chiffrée sur la même ligne,
3. La première lettre de la colonne que l'on trouve ainsi est la lettre déchiffrée.

Texte chiffré : V'UVWHY IOIMBUL PM LSLYI XAOLM BU NAOJVUY

Clé répétée : M USIQU EMUSIQU EM USIQU EMUSI QU EMUSIQU

^ ^^^

| ||Ligne I, on cherche W: on trouve la colonne O

| |Ligne S, on cherche V: on trouve la colonne D

| Ligne U, on cherche U: on trouve la colonne A

Ligne M, on cherche V: on trouve la colonne J

Vigenère : Principe mathématique

❑ Mathématiquement, on identifie les lettres de l'alphabet aux nombres de 0 à 25 (A=0, B=1...).

❑ Les opérations de chiffrement et de déchiffrement sont, pour chaque lettre, celles du **chiffre de César**.

❑ En désignant la i^e lettre du texte clair par Texte[i], la i^e du chiffré par Chiffré[i], et la i^e lettre de la clé, répétée suffisamment de fois, par Clés[i], elle se formalise par :

➤ $\text{Chiffré}[i] = (\text{Texte}[i] + \text{Clés}[i]) \text{ modulo } 26$

➤ $\text{Texte}[i] = (\text{Chiffré}[i] - \text{Clés}[i]) \text{ modulo } 26$

❑ Pour le chiffrement il suffit d'effectuer l'addition des deux lettres puis de **soustraire 26 si le résultat dépasse 26**.

❑ Pour le déchiffrement il suffit d'effectuer la soustraction et **d'ajouter 26 si le résultat est négatif**.

Chiffrement de Hill (1/2)

- ❑ Cette méthode de codage a été publiée par *Lester S. Hill* en 1929 [2].
- ❑ C'est un chiffre *polygraphique*, c'est-à-dire qu'on (dé)chiffre les lettres *par paquets*.
- ❑ Un chiffrement *bigraphique*, c'est-à-dire que nous grouperons les lettres deux par deux.
- ❑ Les lettres sont d'abord remplacées par leur *rang dans l'alphabet de 0 à n-1 avec $n = 26$* (c'est nbr des caractères):

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
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

Chiffrement de Hill (2/2)

□ **Matrice clé de cryptage:** Une matrice A ($p \times p$) agit sur les vecteurs X_i de longueur « p » de la manière suivante,

- Les caractères sont alors regroupés par blocs de p caractères formant un certain nombre de vecteurs $X_i(x_{i0}, x_{i1}, \dots, x_{ip})$, Les nombres « x_{ij} » étant compris entre 0 et $n - 1$,
- Pour un chiffrement **bigraphique** $p = 2$:

$$Y_i = A \cdot X_i \Rightarrow \begin{pmatrix} y_{i0} \\ y_{i1} \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \cdot \begin{pmatrix} x_{i0} \\ x_{i1} \end{pmatrix} \Rightarrow \begin{cases} y_{i0} = ax_{i0} + bx_{i1} \\ y_{i1} = cx_{i0} + dx_{i1} \end{cases}$$

Avec : A : Matrice de cryptage

X_i : Vecteur de deux lettres de texte en claire

Y_i : Vecteur de deux lettres de texte crypté

□ Pour le décryptage on utilise la matrice inverse A^{-1} : $X_i = A^{-1} \cdot Y_i$

Hill : Définition mathématique

- ❑ On utilise φ un isomorphisme de l'anneau $\left(\frac{\mathbb{Z}}{p\mathbb{Z}}\right)^n$ pour coder, et φ^{-1} pour décoder,
- ❑ **Exp:** $p = 26$, on code une lettre « L » par son rang $r(L)$ dans l'alphabet: $r(L) \in \left(\frac{\mathbb{Z}}{p\mathbb{Z}}\right)$
- ❑ Un message $M = X_1, \dots, X_{nm} \in \left(\frac{\mathbb{Z}}{p\mathbb{Z}}\right)$ est coupé en « m » mots $X_i \in \left(\frac{\mathbb{Z}}{p\mathbb{Z}}\right)^n$ de longueur « n » lettres: $M = X_1, \dots, X_m$
- ❑ Soit $A \in GL_n\left(\frac{\mathbb{Z}}{p\mathbb{Z}}\right)$ avec GL_n c'est l'espace des matrices inversibles: on code chaque X_i avec AX_i et M par $MC = AX_1, \dots, AX_m$
- ❑ Pour le décodage on code avec A^{-1} : $M = A^{-1}MC$

Hill : Exemple $p = 26$; $n = 2$

□ Soit $A = \begin{pmatrix} 1 & 12 \\ 1 & 17 \end{pmatrix}$, $\det(A) = 5$ $\begin{pmatrix} a & b \\ c & d \end{pmatrix}^{-1} = \frac{1}{ad - bc} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$

□ $\text{pgcd}(5, 26) = 1$ donc $\det(A)$ est inversible dans $\frac{\mathbb{Z}}{26\mathbb{Z}}$ d'inverse

$$1/5 = 5^{-1} = -5 \bmod 26 = 21 \bmod 26$$

□ D'où $A \in GL_2\left(\frac{\mathbb{Z}}{26\mathbb{Z}}\right)$

➤ Il faut inverser la matrice A . Il suffit de prendre la transposée de

sa comatrice: $\text{com}A^t = \begin{pmatrix} d & -b \\ -c & a \end{pmatrix} = \begin{pmatrix} 17 & -12 \\ -1 & 1 \end{pmatrix}$

➤ et la multiplier (modulo 26) par l'inverse du $\det(A) = 5^{-1}$ c'est-à-dire par 21 :

$$A^{-1} = 21 \begin{pmatrix} 17 & -12 \\ -1 & 1 \end{pmatrix} \bmod 26 = \begin{pmatrix} 357 & -252 \\ -21 & 21 \end{pmatrix} \bmod 26 = \begin{pmatrix} 19 & 8 \\ 5 & 21 \end{pmatrix}$$

Hill : en Python (1/2)

```
def rang(c):
    return(ord(c)-65)

def lettre(r):
    return(chr(65+r))

def mult2(A,x):
    return([(x[0]*A[0][0]+x[1]*A[0][1]) % p,
            (x[0]*A[1][0]+x[1]*A[1][1]) % p])

def code_hill(M,A):
    n = A.shape[0]
    m = len(M) // n
    M1 = [rang(x) for x in M]
    MC1 = []
    for k in range(m):
        MC1 = MC1 + mult2(A,M1[2*k:2*k+2])
    return(''.join([lettre(x) for x in MC1]))
```

Hill : en Python (2/2)

```
>>>
>>> p=26
>>> import numpy as np
>>> A = np.array([[1,12],[1,17]])
>>> M = 'INSTITUTPREPARATOIREAUXETUDES DINGENIEURSDEN
>>> ABEUL'
>>> MC = code_hill(M,A)
>>> MC
'IVMDCTOFLSCZWDULGUNHGCTNZVZTCRIVCWFTKGZLZTNNXRWZ'
>>>
>>> Ainv = np.array([[19,8],[5,21]])
>>> code_hill(MC,Ainv)
'INSTITUTPREPARATOIREAUXETUDES DINGENIEURSDENABEUL'
>>> |
```

RSA : Rivest - Shamir – Adleman (1/2)

- ❑ La cryptographie asymétrique la plus utilisée actuellement (ssl, ssh, https, etc)
 - Fondé sur le fait qu'il est difficile de factoriser de longs entiers (2000 bits et +)
 - C'est un chiffrement basé sur le calcul exponentiel.
- ❑ Sa sécurité repose sur la fonction unidirectionnelle suivante :
 - Le calcul du produit de 2 nombres premiers est aisé.
 - Mais la factorisation d'un nombre en ses deux facteurs premiers est beaucoup plus complexe.
- ❑ Il s'agit du système le plus connu et le plus largement répandu, basé sur l'élévation à une puissance dans un champ fini sur des nombres entiers modulo un nombre premier.

RSA : Rivest - Shamir – Adleman (2/2)

- ❑ Ce cryptosystème utilise deux clés « d » et « e »:
 - Le chiffrement se fait selon $C = M^e \bmod n$
 - Et le déchiffrement par $M = C^d \bmod n$
- ❑ La sécurité repose sur le coût nécessaire pour factoriser de grands nombres.
 - Le nombre de factorisation prend environ $O(e^{\log n \log(\log n)})$ opérations ce qui demande un temps de calcul trop important pour les machines actuelles.
 - On l'utilise pour la confidentialité, l'authentification, ou encore une combinaison des 2.

RSA : Principe

- ❑ On possède 2 clés, l'une **publique** (e,n) et une **privée** (d,n) .
- 1. La première étape revient à **choisir « n » de très grande valeur**.
 - **$n = p*q$** , avec p et q 2 très **grands nombres premiers**.
 - En pratique, si p et q ont 100 chiffres décimaux, n = 200 chiffres.
 - Selon le niveau de sécurité souhaité, la taille de « n » peut varier : 512 bits, 768, 1024, 2048 ou 4096 .
- 2. On choisit un **très grand entier « e », premier à $(p-1)*(q-1)$** .
 - La clé publique sera formée par (e,n) .
- 3. On choisira ensuite un « d » tel que: **$e * d = 1 \bmod (\phi(n))$**
 - La **clé privée** sera donnée par (d,n) .
- 4. En fin : **on jette « p » et « q », les détruire pour éviter les fuites**.
 - Le cryptanalyste devant retrouver ces valeurs!

RSA : Justification mathématique

□ D'après les théorèmes d'Euler et de Fermat, on sait que:

$$a^{\phi(n)} \equiv 1 \pmod{n} \text{ et } a^{\phi(n)} \pmod{n} = 1 \text{ où } (a,n) = 1.$$

□ Dans le RSA, on a $n = p * q$. Et $\phi(n)$ donne le nombre d'entiers positifs plus petits que n et relativement premiers à n :

➤ si p et q sont premiers, $\phi(p) = p - 1$ et $\phi(q) = q - 1$

➤ Si $n = p * q$ alors $\phi(n) = \phi(p) * \phi(q) = (p - 1) (q - 1)$

□ Le choix de e et $d \Rightarrow e * d \equiv 1 \pmod{\phi(n)} = k \phi(n) + 1, k \in \mathbb{N}$

□ Par définition et propriétés des opérations modulo n , on a :

➤ $D_k(E_k(M)) = ((M)^e \pmod{n})^d \pmod{n} = (M^e)^d \pmod{n} = M^{e*d} \pmod{n}$

Et donc :

➤ $M^{e*d} = M^{k \phi(n)+1} = M^{k \phi(n)} \cdot M \pmod{n} = 1 \cdot M \pmod{n} = M \pmod{n}$

$$M^{e*d} = M \pmod{n}$$

RSA : Résumé

1. Génération de 2 nombres premiers p et q
2. Calcul de $n = p * q$
3. Choisir “ e ” tel que $3 < e < \phi(n)$ et $(e, \phi(n)) = 1$
4. Calculer “ d ” tel que $e * d = 1 \bmod \phi(n)$
5. Clé publique : (e, n)
6. Clé privée : (d, n)
7. p et q doivent rester secrets, voire supprimés
8. $C = M^e \bmod n$ et $M = C^d \bmod n$

RSA : Exemple (1/2)

- ❑ Soient $p = 47$, $q = 59$: c'est-à-dire $n=2773$ et $\phi(n) = (p-1) * (q-1) = 2668$ (nombre d'éléments relativement premiers à n et $< n$).
- ❑ Soit $e = 19$ (par exemple, et on a bien $(e, \phi(n))=1$).
 - On détermine que $d = 983$ (inverse modulaire de e sur $\mathbb{Z}/\phi(n)\mathbb{Z}$).
 - La clé publique est donc $(19, 2773)$ et la clé privée est $(983, 2773)$.
- ❑ Soit le codage par la position dans l'alphabet du mot «IPEINABEUL»: 08 15 04 08 13 00 01 04 20 11
- ❑ On procède selon deux conditions :
 1. Découpage en morceaux de même longueur, ce qui empêche la simple substitution : 081 504 081 300 010 420 11_
 - On ajoute un padding initial si nécessaire.

008 150 408 130 001 042 011

RSA : Exemple (2/2)

2. Découpage en morceaux de valeur inférieure à n , car opération mod n .

$008^{19} \bmod 2773$	2095
$150^{19} \bmod 2773$	0747
$408^{19} \bmod 2773$	2640
$130^{19} \bmod 2773$	2305
$001^{19} \bmod 2773$	0001
$042^{19} \bmod 2773$	1588
$011^{19} \bmod 2773$	0750

Le chiffrement



$2095^{983} \bmod 2773$	008
$0747^{983} \bmod 2773$	150
$2640^{983} \bmod 2773$	408
$2305^{983} \bmod 2773$	130
$0001^{983} \bmod 2773$	001
$1588^{983} \bmod 2773$	042
$0750^{983} \bmod 2773$	011

Le déchiffrement

- Lors du déchiffrement, sachant qu'il faut obtenir des blocs de 2 éléments (grâce au codage particulier de l'exemple), on a bien:

I	P	E	I	N	A	B	E	U	L
08	15	04	08	13	00	01	04	20	11

Fonction totient d'Euler

□ Soit la fonction d'Euler définie par :

$$\phi(m) = \#\{n \leq m \mid \text{pgcd}(n, m) = 1\}$$

➤ $\phi(n)$ est l'indicateur d'Euler, c'est-à-dire le cardinal des entiers inversibles de n .

□ Si « n » est premier: $\phi(n) = n-1$

➤ La fonction $\phi(n)$ donne le nombre d'entiers positifs plus petits ou égaux à n relativement premiers à n .

➤ D'où la réécriture suivante: $\phi = \sum_{k=1}^m 1$ avec $(k, m) = 1$

➤ « m » est premier $\Rightarrow \phi(m) = \text{nombre d'éléments inférieurs à } m$

$$\diamond \text{Exp.: } n = 11 \Rightarrow \phi(n) = n-1 = 10$$

□ Soient p et q deux nombres premiers et $n = pq$. Alors:

$$\phi(n) = \phi(pq) = \phi(p) \cdot \phi(q) = (p-1)(q-1)$$

Théorème d'Euler

□ Soit a et m deux entiers premiers entre eux $\text{pgcd}(a,m) = 1$,

Alors: $a^{\phi(m)} = 1 \bmod (m)$

□ C'est une nouvelle technique pour le calcul de l'inverse de « a » modulo « m » : $a^{-1} * a^{\phi(m)-1} = 1 \bmod (m) \Rightarrow$

$$x = a^{-1} = a^{\phi(m)-1} \bmod m$$

□ **Exemple :** Quel est l'inverse de 5 mod 7 ?

□ Comme 7 est premier, sa fonction totient est $\phi(7) = 7-1 = 6$ (c'est le nbr, des entiers premiers avec 7 sont: 1,2,3,4,5 et 6).

➤ On a bien $\text{pgcd}(5, 7) = 1$ alors $5^6 = 1 \bmod 7$

➤ d'où $5^{-1} = 5^5 \bmod 7 = 3$ c'est l'inverse de 5 mod 7

Petit théorème de Fermat

Théorème: Soit p un nombre premier, $(a,p)=1$. Alors :

$$a^{p-1} = 1 \mod p$$

Application pour le test de primalité:

- Si $n \in \mathbb{N}$ est donné, soit « a » tel que $(a,n) = 1$.
 - On calcule alors a^{n-1} .
 - Si $a^{n-1} \neq 1 \mod n$ alors « n » n'est pas premier.
 - Il faut cependant remarquer que cette condition n'est pas nécessaire et suffisante.
- Exp.: $a=2$ et $n=7 \Rightarrow (a,n)=1$ et $a^{n-1}=2^6=64 \mod 7 = 1$
 - d'où $n=7$ est premier

Annexe 1: Nombres premiers entre eux (1/3)

- ❑ En mathématiques, on dit que deux entiers a et b sont **premiers entre eux**, que a est **premier avec b** ou **premier à b** ou encore que a et b sont **co-premiers** (ou encore **étrangers**):
 - Si leur plus grand commun diviseur est égal à 1 ; en d'autres termes, s'ils n'ont aucun diviseur autre que 1 et -1 en commun.
 - De manière équivalente, ils sont premiers entre eux s'ils n'ont aucun facteur premier en commun.
- ❑ Par exemple, 6 et 35 sont premiers entre eux, mais 6 et 27 ne le sont pas parce qu'ils sont tous les deux divisibles par 3.
- ❑ 1 est premier avec tout entier ;
- ❑ 0 est uniquement premier avec 1 et -1 .

Annexe 1: Nombres premiers entre eux (2/3)

□ Théorème de Bachet-Bézout (ou Identité de Bézout6):

- Soient a et b deux entiers relatifs.
- Si d est le PGCD de a et b , alors il existe deux entiers relatifs x et y tels que $ax + by = d$.

□ Théorème de Bézout:

- Les entiers relatifs « a » et « b » sont premiers entre eux si et seulement s'il existe des entiers relatifs « x » et « y » tels que $ax + by = 1$.
- Cette condition équivaut à : « b » a un inverse pour la multiplication modulo a , c'est-à-dire : il existe un nombre entier « y » tel que $by \equiv 1 \pmod{a}$.

□ Application: Le théorème de Bachet-Bézout intervient dans de nombreux domaines de la théorie des nombres.

- Il intervient dans la recherche d'inverse modulo n : $b^{-1} \bmod n$
- il est donc utile dans le décodage de message en cryptographie,

Annexe 1: Nombres premiers entre eux (3/3)

Exemple:

- ❑ Considérons les deux nombres $a=120$ et $b=23$ avec un $\text{PGCD}(a,b) = 1$, donc a et b sont premiers entre eux,
- ❑ D'après le théorème de Bézout on a : $ax + by = 1 \Rightarrow 120x + 23y = 1$
- ❑ On appliquons l'algorithme d'Euclide étendu pour calculer $x = -9$ et $y = 47$.
 - D'où $120*(-9) + 23*47 = 1$:
 - Ceci signifie que -9 est l'inverse pour la multiplication de 120 modulo 23 , parce que $(-9) \times 120 \pmod{23} = 1 \Rightarrow 120^{-1} \pmod{23} = -9 \pmod{23} = 14$
 - De même 47 est l'inverse, pour la multiplication modulo 120 , de 23 , parce que $47 \times 23 \pmod{120} = 1 \Rightarrow 23^{-1} \pmod{120} = 47$
- ❑ Il y a un calculateur de $b^{-1} \pmod{n}$ en ligne sur le lien suivant:

<https://www.apprendre-en-ligne.net/crypto/rabin/euclide.html>

Annexe 2 : Calcul de l'inverse de $x \bmod n$

Théorème : Soit un entier a premier à n . Alors a est inversible dans $\mathbb{Z}/n\mathbb{Z}$, c'est-à-dire qu'il existe b tel que

$$ab \equiv 1 \bmod n.$$

```
1. def pgcd(a,b):
2.     """pgcd(a,b): calcul du 'Plus Grand Commun Diviseur'
3.     entre les 2 nombres entiers a et b"""
4.     while b != 0:
5.         r = a%b
6.         a,b = b,r
7.     return a
8. def inv_mod_n(a,n):
9.     """inv_mod_n(a,n): fonction pour calculer "b" l'inverse
10.    de a mod n => b * a = 1 mod n"""
11.    if pgcd(a,n) != 1:
12.        print("{} et {} pas premiers entre eux".format(a,n))
13.        return
14.    for b in range(n):
15.        if b*a % n == 1:
16.            print("inverse de {} mod {} est: {}".format(a,n,b))
17.            return b
```

Références bibliographiques

- [1] Loïc Pottier, « Cryptographie », WPAP 2016, Sfax, Tunisie, 2-3 juin 2016
- [2] Hill Lester S., « Cryptography in an Algebraic Alphabet », *American Mathematical Monthly*, 36, 1929, pp. 306-312
- [3] G.Michaud-Brière, Y.Pearson, S.Perreault, Louis-Olivier Roof, « La cryptographie », <http://nomis80.org/cryptographie/cryptographie.html>
- [4] Algorithme d'Euclide,
<http://www.labri.fr/perso/betrema/deug/poly/euclide.html#bezout>
- [5] Nombres premiers,
<http://www.labri.fr/perso/betrema/deug/poly/premiers.html>