[CRYPTOGRAPHIE] CHIFFREMENT AFFINE

Notion réinvestie : coefficients de Bézout

Le chiffrement affine est une méthode de cryptographie basée sur un chiffrement par substitution mono-alphabétique, c'est-à-dire que la lettre d'origine n'est remplacée que par une unique autre lettre.
On choisit deux entiers naturels a et b comme clés.
Chaque lettre du texte à chiffrer est d'abord remplacée par son équivalent numérique x, puis chiffrée par le calcul du reste de la division euclidienne par 26 de l'expression affine ax+b, soit l'entier z tel que :

$$z \equiv a\,x+b\,[26]\,.$$

> Si le coefficient *a* vaut 1, alors le codage affine correspond au *chiffre de César.*
> Selon Suétone, Jules César l'utilisait avec l'alphabet grec (inintelligible pour la plupart des Gaulois mais langue maîtrisée par les élites dirigeantes romaines) et un décalage de trois sur la droite pour certaines de ses correspondances secrètes, notamment militaires.

A. Chiffrer

Prenons dans cette partie et la suivante les clés $a=11$ et $b=8$.

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

Chiffrer le mot « amour ».

B. Déchiffrer

On souhaite décoder le mot « ehgegzij ».

1. Un exemple

Commençons par la lettre e.
On cherche donc x tel que $4 \equiv 11x + 8$ [26], c'est-à-dire tel que $11x + 4 \equiv 0$ [26].
Déterminer x.

2. Cas général

Notons y l'équivalent numérique d'une lettre chiffrée.
On cherche donc x tel que $11x + 8 \equiv y$ [26], autrement dit tel que $11x \equiv y - 8$ [26].

a) Déterminer deux entiers u_0 et v_0 tels que $11u_0 - 26v_0 = 1$.
On a donc : $11u_0 \equiv 1\,[26]$.

b) En déduire que $x \equiv (y-8)u_0\,[26]$.

c) Les fonctions suivantes (langage : Python) permettent de coder un message.
Expliquer les algorithmes et compléter la fonction codage.

```
def num(caractere): #pour convertir les caractères en nombres
    return ord(caractere)-65-32 #les majuscules commencent à 65, les minuscules à 32

def lettre(n):
    return chr(n+65+32)

def supprime_accent(ligne): #supprime les accents du texte source
    accent = ['é', 'è', 'ê', 'à', 'ù', 'û', 'ç', 'ô', 'î', 'ï', 'â']
    sans_accent = ['e', 'e', 'e', 'a', 'u', 'u', 'c', 'o', 'i', 'i', 'a']
    for i in range(len(accent)):
        ligne = ligne.replace(accent[i], sans_accent[i])
    return ligne
```

```
def codage(message, a, b):
    mescode = ""
    for c in message:
        if c==" " or c="." or c="'" or c="," or c="[" or c="]" or c="!" or c="?":
            mescode=mescode + " "
        else:
            y= ……
            mescode=mescode + ……
    return mescode
```

d) Créer alors une fonction pour déchiffrer le mot « ehgegzij » puis le texte suivant :

« uv xggzgwsyja cus av ilnscua i gtyanfa pa rnay zay wgnszzay y ajgvva pa z uvslgnksja pa zaun fsa aj pa zaun wnivp paygaufnakavj pay haunay aj pay haunay yivy nsav lisna szy va egvvisyyavj pgve riy z avvus eajja cuayjsgv ayj tsav p uv hgkka p uv ysvwa geeura zgsv pa lusn zi kgvgjgvsa zay ivskiub zi naehanehavj aj ea cu szy napgujavj za rzuy e ayj pa zi fgsn eayyan ein azza va eayya cua rgun ajna nakrzieaa rin zi raun eiuya pa jguj illisnakavj z sviejsgv ayj psfsva e ayj rgunjivj egvjna azza cua z hgkka y ayj svyunwa zus yauz pivy zi vijuna ayj sveiritza pa yurrgnjan zi kgvgjgvsa zus yauz fauj i jguj rnsb cua cuazcua ehgya innsfa v skrgnja cugs rin zi sz ya kgvjna svpswva pa ygv iveajnav za taygsv pa vgufaiuja ayj za lisj p uv wgnszza lgunfgma aksz esgniv »

C. Clés possibles

1. Démontrer que si a et 26 sont premiers entre eux, alors le décodage est toujours possible.

2. Démontrer que si a et 26 ne sont pas premiers entre eux, alors le décodage n'est pas possible.

3. Combien de clés de chiffrement existe-t-il ?
Comment casser un message crypté par chiffrement affine ?
Écrire un algorithme qui fait cela.