MPSI 2
Année scolaire 2014/15

INFORMATIQUE
Introduction à la cryptologie

1 Introduction

On s'intéresse à la situation suivante : une première personne (Alice) souhaite transmettre un message m à une deuxième personne (Bob) sans qu'une troisième personne (Eve) ne soit en mesure de comprendre ce message. Pour cela, elle va transformer le message m en un message chiffré c à l'aide d'une clé k qu'elle partage avec Bob. On introduit le vocabulaire suivant :

- Chiffrer : transformer m en c à l'aide de la clé. C'est ce que fait Alice.
- Déchiffrer : retrouver m à partir de c et de la clé. C'est ce que fait Bob.
- Décrypter : retrouver m à partir de c sans connaitre la clé. C'est ce qu'essaie de faire Eve.

Nous travaillerons dans le cadre suivant : les messages seront des chaînes de caractères rédigées en français et n'utilisant que les caractères "a", "b", ..., "z" (pas de majuscules, pas de caractères accentués, pas d'espaces ni de signes de ponctuation, ...).

Pour effectuer des opérations sur les 26 caractères, on leur associe un entier de 0 à 25 ($a = 0$, $b = 1$, ... $z = 25$), les opérations algébriques se font alors modulo 26. On a par exemple :

- $c + d = 2 + 3 = 5 = f$
- $p + w = 15 + 22 = 1 \times 26 + 11 = 11 = l$
- $p - w = 15 - 22 = -1 \times 26 + 19 = 19 = t$

Pour réaliser ceci, on utilisera la fonction `ord(x)` qui retourne le code ASCII (un entier) du caractère x ainsi que la fonction `chr(x)` qui retourne le caractère dont le code ASCII est l'entier x.

Exercice 1 ***Opérations sur les caractères.***

1. *Donner les codes ASCII des caractères, "a", "b", "c" , "d", ... Que remarque-ton ?*
2. *Écrire une fonction* `c2m(x)` *qui retourne l'entier entre* 0 *et* 25 *correspondant au caractère* x.
3. *Écrire une fonction* `m2c(x)` *qui retourne le caractère correspondant à l'entier* x, $0 \leq x \leq 25$.
4. *Écrire une fonction* `addc(x,y)` *qui retourne le caractère correspondant à l'addition des deux caractères* x *et* y.

2 Le code de César

Le code de César fonctionne ainsi :

1. La clé k est un caractère choisi entre "a" et "z".
2. Alice chiffre chaque caractère de son message en lui ajoutant la clé, ainsi le message $m =$ "*bonjour*" chiffré avec la clé "b" donne le message chiffré $c =$ "*cpokpvs*".
3. Bob déchiffre chaque caractère du message chiffré en lui ajoutant la clé de déchiffrement $k_d = a - k$.

Exercice 2 ***Chiffrement du code de César.***

1. *Écrire une fonction* `cesar(m,k)` *qui retourne le chiffrement de la chaîne de caractères* m *avec la clé* k.
2. *Chiffrer le message "lacleestsouslepaillasson" avec la clé "u".*

Exercice 3 ***Déchiffrement du code de César.***

1. *Écrire une fonction* `cleD(k)` *qui retourne la clé de déchiffrement associée à la clé* k.
2. *Écrire une fonction* `cesarD(c,k)` *qui déchiffre la chaîne de caractère* c *avec la clé* k.
3. *Déchiffrer le message "eczaelcoultopulnlddpfypgtecp" qui a été chiffrée avec la clé "l".*

Le code de César n'est pas très robuste : puisqu'il n'y a que 26 clés possibles, il suffit de toutes les tester pour décrypter un message (on parle alors d'attaque exhaustive).

Exercice 4 ***Attaque exhaustive du code de César.***
Écrire un programme qui affiche les 26 *messages en clair possibles correspondant au message chiffré* `testC1`.

Il existe une méthode d'attaque statistique plus efficace qui nous permettra de plus d'attaquer des codes plus complexes où une attaque exhaustive n'est pas envisageable.
Cette méthode se base sur le principe d'Al-Kindi (IX) : dans les langues naturelles, la loi d'apparition des caractères n'est pas la loi uniforme. Pour le français par exemple, on observe les fréquences d'apparitions suivantes :

FRÉQUENCE DES LETTRES DANS LA LANGUE FRANÇAISE
source : *Germinal*, Émile Zola. (786 770 lettres)

E	17,18%		P	2,37 %
A	9,22%		V	1,64 %
S	7,82%		F	1,10 %
I	7,50%		H	1,07 %
T	7,37%		G	1,06 %
N	7,03%		B	1,04 %
R	6,61%		Q	1,02 %
U	6,36%		X	0,41 %
L	6,13%		J	0,38 %
O	4,91%		Y	0,23 %
D	3,75%		Z	0,12 %
C	3,02%		K	0,01 %
M	2,65%		W	0,20 %

Face à un texte chiffré, il suffit donc de détecter le caractère le plus représenté et de supposer qu'il s'agit du "e". On en déduit alors la clé de déchiffrement et on peut donc déchiffrer le message.

Remarque : pour que cette approche statistique fonctionne, il faut donc que le message soit suffisamment grand pour que les statistiques aient un sens, et qu'il se s'agisse pas d'un texte de Georges Perec.

Exercice 5 ***Attaque statistique du code de César.***

1. *Écrire une fonction* `frequence(c)` *qui retourne la liste* L *des occurrences des caractères de la chaîne* c *(*$L[0]$ *: nombre de "a" dans* c, $L[1]$ *: nombre de "b" dans* c,...*).*
2. *Écrire une fonction* `cleA(c)` *qui retourne la clé de déchiffrement la plus probable du message chiffré* c.
3. *Écrire une fonction* `cesarA(c)` *qui décrypte le message chiffré* c.
4. *Décrypter le message* `testC2`.

3 Le code de Vigenère

Le code de Vigenère est une amélioration très sensible du code de César :

1. La clé k est une chaîne de caractères.
2. Alice chiffre le i^{eme} caractère de son message en lui ajoutant le j^{eme} caractère de la clé ou j est i modulo la longueur de la clé. Par exemple le message $m =$ "*bonjour*" chiffré avec la clé "*bec*" donne le message chiffré $c =$ "*cspksws*" ($b + b = c$, $o + e = s$, $n + c = p$, $j + b = k$, $o + e = s$, $u + c = w$, $r + b = s$).
3. Bob déchiffre le message chiffré en lui appliquant le même algorithme avec la clé de déchiffrement.

Exercice 6 ***Chiffrement du code de Vigenère.***

1. *Écrire une fonction* `vigenere(m,k)` *qui retourne le chiffrement de la chaîne de caractères* m *avec la clé* k.
2. *Chiffrer le message "cechiffrementsembleplusrobuste" avec la clé "grjm".*

Exercice 7 ***Déchiffrement du code de Vigenère.***

1. *Écrire une fonction* `cleDV(k)` *qui retourne la clé de déchiffrement associée à la clé* k.
2. *Écrire une fonction* `vigenereD(c,k)` *qui déchiffre la chaîne de caractère* c *avec la clé* k.
3. *Déchiffrer le message "wamgibcjbsuwwhxchawqtssmzpbhpyitz" qui a été chiffré avec la clé "opi".*

Même en se limitant à une taille maximale donnée, il n'est pas raisonnable d'espérer faire une attaque exhaustive du code de Vigenère : le nombre de clés potentielles est trop important. Il est toutefois possible d'attaquer ce code par des approches statistiques basées sur des raffinements du principe d'Al-Kindi. L'attaque s'effectue en deux temps : premièrement trouver $|k|$ (la taille de la clé k), deuxièmement trouver la clé de déchiffrement.

3.1 Première étape : trouver $|k|$

On peut obtenir la taille de la clé grâce à **l'indice de coïncidence**[1].
L'indice de coïncidence de deux chaînes de caractères est égal au nombre de caractères communs (même valeur et même position) divisé par la taille de la chaîne la plus petite. Par exemple l'indice de coïncidence de "martingale" et de "casino" est 1/6 :

c		*a*		s	i	n	o
m		*a*		r	t	i	n

La probabilité que deux caractères de deux chaînes rédigées en français soient égaux est environ de 0.0778, tandis qu'il n'est que de $1/26 \approx 0.0385$ pour des chaînes aléatoires. D'après la loi des grands nombres, l'indice de coïncidence de deux chaînes de caractères rédigées en français tend donc vers 0.0778 quand la longueur des chaînes tend vers $+\infty$, alors que celui de deux chaînes de caractères aléatoires tend vers 0.0385. Encore une fois, il faut que les chaînes de caractères soient "suffisamment" grandes pour espérer que l'indice de coïncidence s'approche de ces valeurs.

1. En cherchant sur internet, vous trouverez sans doute d'autres définitions de l'indice de coïncidence. Ces définitions (en tout cas leur utilisation) sont équivalentes.

Pour déterminer la longueur de la clé avec laquelle c a été chiffrée, on va calculer pour tout entier $1 \leq i \leq |c|//10$ l'indice de coïncidence entre la chaîne de caractères c et la chaîne de caractères c décalée à gauche de i position(s) :

c	e	x	e	m	p	l	e
c_1	x	e	m	p	l	e	
c_2	e	m	p	l	e		

Lorsque i n'est pas un multiple de $|k|$, les deux chaînes doivent avoir un indice de coïncidence de l'ordre de 0.0385, tandis que lorsque i est un multiple de $|k|$, l'indice de coïncidence doit être proche de 0.0778. Ainsi, on peut repérer les multiples de $|k|$ grâce aux pics de l'indice de coïncidence.

Remarque : Il s'agit encore et toujours de propriétés statistiques, il est donc tout à fait possible que l'indice de coïncidence correspondant à un multiple de $|k|$ soit petit, voire (mais c'est plus rare) que celui d'un entier qui n'est pas un multiple de $|k|$ soit grand.

Exercice 8 ***Attaque statistique du code de Vigenère : première étape.***

1. *Écrire une fonction* `coincidence(c1,c2)` *qui retourne l'indice de coïncidence des deux chaînes de caractères $c1$ et $c2$.*
2. *Écrire une fonction* `coincidenceV(c)` *qui retourne la liste L des décalages $1 \leq i \leq len(c)//10$ donnant un indice de coïncidence strictement supérieur à 0.05.*
3. *Appliquer cette fonction à la chaîne de caractères* `testV` *et en déduire la longueur probable de la clé de chiffrement.*

3.2 Deuxième étape : trouver la clé de déchiffrement

Si c est une chaîne de caractère écrite en français, les probabilités d'apparition des caractères dans les sous-chaînes extraites "régulièrement" de c sont les même que dans c. Par exemple, les chaînes c_0, c_1 et c_2 obtenues à partir de c en ne conservant que les caractères d'indice congru à 0, 1, 2 modulo 3 ont en moyenne elles aussi 17% de "e".
Ceci permet donc d'attaquer le code de Vigenère quand on connaît la taille $|k|$ de la clé : il suffit de découper la chaîne c en $|k|$ sous-chaîne c_i, $i \in [\![0, |k|[\![$. Tous les caractères d'une même sous-chaîne ayant été codés avec le même caractère de la clé, on peut décrypter chaque sous-chaîne comme si elle avait été codée par un chiffrement de César et donc obtenir la clé de déchiffrement de chaque sous-chaîne. Il suffit alors de concaténer ces clés pour obtenir la clé de déchiffrement de c.

Remarque : Pour que l'attaque statistique sur les sous-chaînes fonctionne, il faut encore une fois que ces sous-chaînes soient grandes. Cela justifie la limitation aux décalages de taille inférieure à $len(c)//10$ dans la section précédente (c'est même optimiste...).

Exercice 9 ***Attaque statistique du code de Vigenère : deuxième étape.***

1. *Écrire une fonction* `souschaines(c,l)` *qui retourne la liste des l sous-chaînes extraites régulièrement de c.*
2. *Écrire une fonction* `cleAV(c,l)` *qui retourne la clé de déchiffrement du message chiffré c, c ayant été chiffré avec une clé de longueur l.*
3. *Écrire une fonction* `vigenereA(c,l)` *qui décrypte le message chiffré c avec une clé de longueur l.*
4. *Décrypter le message* `testV`*.*