TP 1: Anciens Schémas de Chiffrement : Sécurité et Performance

1 Déscription du TP

La cryptographie est un art qui pratique l'étude de la dissimulation d'informations dans un format illisible par les humains. En fait, la cryptographie moderne croise les disciplines des mathématiques, de l'informatique et du génie électrique.

Avant l'ère moderne, la cryptologie était presque synonyme de chiffrement, où l'information était convertie d'un état lisible à un état apparemment dépourvu de sens.

La capacité de protéger et de sécuriser l'information est vitale pour la croissance du commerce électronique et pour la croissance d'Internet lui-même. Beaucoup de gens ont besoin ou souhaitent utiliser la sécurité des communications et des données dans différents domaines. Le chiffrement des données joue un rôle majeur dans la sécurité. Par exemple, les banques utilisent des méthodes de chiffrement dans le monde entier pour traiter les transactions financières. Cela implique le transfert de sommes importantes d'une banque à une autre. Les banques utilisent également des méthodes de chiffrement pour protéger les numéros d'identification de leurs clients aux distributeurs automatiques de billets de la banque.

De nombreuses entreprises (en ligne) et même des centres commerciaux vendent n'importe quel article, des fleurs aux bouteilles de vin, sur Internet et ces transactions sont effectuées à l'aide de cartes de crédit et de navigateurs Internet sécurisés, y compris des techniques de chiffrement (et des clés numériques). Les clients utilisant Internet souhaitent savoir si la connexion est sécurisée ou non, notamment lorsqu'ils envoient leurs informations de carte de crédit et d'autres détails financiers les concernant dans un environnement multinational. Cela ne fonctionnera qu'avec l'utilisation de méthodes de chiffrement solides et inaltérables. Votre directeur informatique vous donnera des instructions pour chiffrer les données à l'aide de divers algorithmes de chiffrement afin de sécuriser les informations de l'organisation.

Selon la table de correspondance suivante, chacune des 26 lettres de l'alphabet est associée à l'un des entiers de 0 à 25.

A	B	C	D	E	F	G	H	I	J	K	L	M
↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕
0	1	2	3	4	5	6	7	8	9	10	11	12

N	O	P	Q	R	S	T	U	V	W	X	Y	Z
↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕
13	14	15	16	17	18	19	20	21	22	23	24	25

1.1 Environnement du TP

Pour réaliser ce laboratoire, vous devez écrire le code pour différentes techniques de chiffrement classiques et les analyser. De plus, assurez-vous d'ajouter des titres, des étiquettes d'axe et des légendes (le cas échéant) aux graphiques.

1.2 Objectifs du TP

Ce laboratoire vous montrera comment chiffrer des données et comment les utiliser. Il vous apprendra à :

1. Utiliser différentes techniques de chiffrement classiques.
2. Utiliser des techniques de cryptanalyse telles que la force brute et les attaques par fréquence.

1.3 Durée du TP

4 heures (deux heures sous surveillance en laboratoire et deux heures supplémentaires non supervisées).

2 Tâches du TP

Dans cet TP, nous appliquerons plusieurs chiffrements classiques (fonctions de chiffrement et de déchiffrement) tels que le chiffre de César, et analyserons leur niveau de sécurité. Dans ce qui suit, la fonction `numericChar` qui prend comme paramètre une lettre de l'alphabet et renvoie son index dans la liste $chain$ (on suppose que $chain$ est défini comme une chaîne de caractères constante) est fournie.

```
alphabet="abcdefghijklmnopqrstuvwxyz"

def numericChar(chain,car):
  car=car.lower()
  return chain.index(car)
```

Exercice 1: Chiffre de César

Le but de cette Exercice est d'implémenter des fonctions Python pour le chiffrement/déchiffrement avec le chiffre de César, ainsi que pour les attaques. Le chiffre de César est une variante du chiffre de décalage (avec une clé secrète k =3) et il était utilisé par Jules César pour envoyer des communications d'importance militaire pour son empire. Aujourd'hui, c'est une technique étudiée en cryptographie sous la classification des chiffres de substitution, un sujet qui relève également de la cryptographie classique. Il est également connu sous le nom de chiffre de décalage, code de César, ou décalage de César, qui peut être réalisé selon l'équation suivante :

$$c = (p + k)\ \%26 \tag{1}$$

où p et c représentent la valeur numérique de la lettre de l'alphabet originale et de la lettre chiffrée, respectivement. De plus, $\%$ est la fonction modulo qui renvoie le reste de la division. En utilisant le reste de la division à la fin, nous garantirons que le caractère chiffré fera toujours partie de l'alphabet (supérieur ou égal à 0 et inférieur à 26).

Pour récupérer le texte en clair, l'algorithme de déchiffrement du chiffre de César (décalage) doit être appliqué, ce qui peut être fait selon l'équation suivante :

$$p = (c - k)\ \%26 \tag{2}$$

où p représente la valeur numérique d'une lettre de l'alphabet déchiffrée.

Dans ce qui suit, la fonction de chiffrement qui prend une clé (sous forme d'un entier de 0 à 25) et une chaîne de caractères est fournie. Ces fonctions ne devraient fonctionner que sur les caractères 'a', 'b', . . ., 'z' (en minuscules et en majuscules), et devraient laisser inchangés tous les autres caractères. Elle devrait renvoyer les lettres chiffrées (le texte chiffré). Les entrées de cette fonction sont les suivantes :

- Message (texte en clair/texte chiffré).
- Clé secrète k (3 comme clé par défaut). La clé que vous entrez ici représente le décalage de l'alphabet.

```
def encryptShift(origMessage,key='d'):
  encryptedMessage=""

  if key.isalpha():
    key=numericChar(alphabet,key)


  for car in origMessage:
      if car.lower() in alphabet:
        encryptedMessage+=alphabet[(numericChar(alphabet,car)+key)%26]
      else:
        encryptedMessage+=car
  return encryptedMessage
```

1. Écrivez la fonction de déchiffrement correspondante.
2. Appliquez d'abord la fonction de chiffrement/déchiffrement à un message aléatoire et vérifiez que vous récupérez le même texte en clair d'origine après le déchiffrement.
3. Implémentez une fonction qui effectue une attaque par force brute sur un texte chiffré. Elle devrait afficher une liste des clés et des déchiffrements associés. Elle devrait également prendre un paramètre optionnel qui permet de spécifier une sous-chaîne et n'affiche que les textes en clair potentiels (choisis ou connus) contenant ce déchiffrement.
4. Affichez la sortie de votre fonction de chiffrement pour les paires (clé, texte en clair) suivantes:
   - k = 6, plaintext (texte en clair) = "Get me a vanilla ice cream, make it a double."
   - k = 15, plaintext = "I don't much care for Leonard Cohen."
   - k = 16, plaintext = "I like root beer floats."
5. Affichez la sortie de votre fonction de déchiffrement pour les paires (clé, texte chiffré) suivantes :
   - k = 12, ciphertext = "nduzs ftq buzq oazqe."
   - k = 3, ciphertext = "fdhvdu qhhgv wr orvh zhljkw"
   - k = 20, ciphertext = "ufgihxm uly numnys."

6. Affichez la sortie de votre fonction d'attaque (partie 2) sur les textes chiffrés suivants. Si un mot-clé optionnel est spécifié, transmettez-le à votre fonction d'attaque

   - ciphertext = "gryy gurz gb tb gb nzoebfr puncry.", keyword = 'chapel'
   - ciphertext = "wziv kyv jyfk nyve kyv tpdsrcj tirjy.", keyword = 'cymbal'
   - ciphertext ="baeeq klwosjl osk s esf ozg cfwo lgg emuz.", no keyword

Exercice 2 : Chiffre affine

Le but de cet exercice est d'implémenter des fonctions Python pour le chiffrement/déchiffrement avec le chiffre affine, ainsi que pour les attaques. Une amélioration du chiffrement de César est le chiffrement affine, définit dans la suite:

$$c = (a \times p) + b \; mod \; n$$

Où p est le caractère en clair et c est le caractère chiffré après le chiffrement, et a et b sont les coefficients (clé secrète) du chiffre affine. a devrait avoir un inverse a^{-1}, qui est l'inverse de $a \; mod \; n$.

L'algorithme de déchiffrement affine est donné par :

$$p = a^{-1} \times (c - b) \; mod \; n$$

a est inversible si le $pgcd(a, \; n) = 1$. De plus, vous devez utiliser cette règle pour trouver a^{-1}, qui est $a \times a^{-1} \; mod \; n = 1$, comme présenté dans la fonction suivante.

```
def inverse(a,n):
  for it in range(1,n):
    if (a*it%n==1):
     return it
```

1. Implémentez des fonctions Python qui effectuent le chiffrement/déchiffrement avec le chiffre Affine, en utilisant une clé constituée d'une paire d'entiers (a, b), tous deux dans l'intervalle $1, \; 2, \; \ldots, \; 25$ avec a devant avoir un inverse. Les fonctions devraient fonctionner sur des chaînes de caractères et laisser inchangés tous les caractères non alphabétiques. Montrez le fonctionnement de vos fonctions sur un exemple.

2. Chiffrez "YES" en utilisant la fonction de chiffrement Affine avec $a = 3$ et $b = 7$.

3. Le chiffre "QXFM" a été obtenu en appliquant la fonction de chiffrement Affine avec les mêmes coefficients. Déchiffrez-le en utilisant la fonction de déchiffrement correspondante.

Exercice 3 : Chiffre de Vigenère

Le chiffre de Vigenère est une méthode de chiffrement de textes alphabétiques. Un chiffre polyalphabétique est un chiffre basé sur une opération de substitution qui utilise plusieurs alphabets de substitution (tables). Le chiffrement du texte original est réalisé en utilisant le carré de Vigenère ou la table de Vigenère.

La table se compose des alphabets écrits 26 fois dans différentes lignes, chaque alphabet étant décalé cycliquement vers la gauche par rapport à l'alphabet précédent, ce qui correspond aux 26 chiffres de César possibles. À différents moments du processus de chiffrement, le chiffre utilise un alphabet différent tiré de l'une des lignes. L'alphabet utilisé à chaque étape dépend d'un mot-clé répétitif.

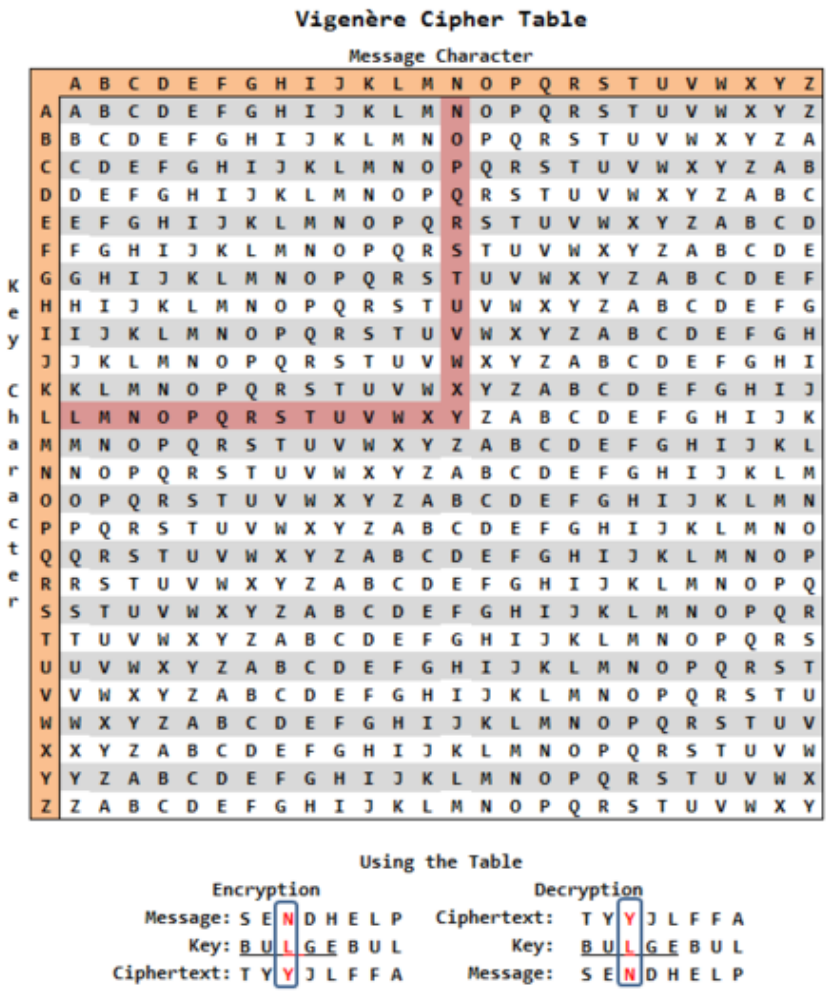

Figure 1: The Vigenère square or Vigenère table.

Une implémentation plus simple du chiffre de Vigenère peut être réalisée de manière algébrique en convertissant ['a', 'b', . . ., 'z'] en nombres [0, 1, . . ., 25], comme décrit dans le chiffre de César.

Le $i^{ème}$ caractère du texte en clair (p_i) et de la clé (k_i) sont additionnés modulo 26.

$$c_i = (p_i + k_i); \tag{3}$$

et le déchiffrement peut être effectué selon l'équation suivante :

$$d_i = (c_i - k_i); \tag{4}$$

Où d_i représente le $i^{ème}$ caractère du texte en clair déchiffré.

Écrivez des fonctions Python pour effectuer le chiffrement/déchiffrement avec le chiffre de Vigenère, puis montrez le fonctionnement de vos fonctions sur un texte en clair de votre choix.

Exercice 4 : Fonctions d'Analyse de Sécurité

Dans cet exercice, écrivez un message que vous souhaitez protéger dans un nouveau fichier. En effet, le message sélectionné est de votre choix personnel, cependant, vous devez vous assurer qu'il a une longueur

supérieure à 64 caractères pour une analyse ultérieure. En général, plus la longueur de votre message est grande, plus l'analyse ultérieure sera précise.

1. Écrivez une fonction `frequenceLettres` qui prend en paramètre le texte chiffré et renvoie un vecteur de taille 26 dont la cellule d'indice i contient la probabilité d'occurrence de la lettre i. Ensuite, la distribution sera tracée. Vous devriez utiliser cette fonction pour représenter graphiquement la distribution du texte en clair et du texte chiffré pour chaque chiffre (plt.hist(....)).

2. Écrivez une fonction qui génère un graphique de dispersion illustrant la relation entre le texte en clair (plaintext) et le texte chiffré (ciphertext). Autrement dit, montrez comment chaque caractère dans le texte en clair, désigné sous le nom de $x(t)$, varie par rapport à son caractère correspondant dans le texte chiffré, $y(t)$. Créez des graphiques de dispersion pour chaque algorithme de chiffrement (plt.scatter(x,y)).

3. Écrivez une fonction qui calcule le nombre (ou le pourcentage) d'éléments différents entre deux vecteurs d'entrée (qui ont la même longueur). Nous devrions utiliser cette fonction pour détecter le pourcentage d'éléments (lettres) différents entre le texte en clair et le texte chiffré pour chaque algorithme de chiffrement.

4. Que remarquez-vous à propos des résultats des histogrammes lorsque le texte est chiffré avec le chiffre de Vigenère par rapport aux résultats du chiffre de César ?

5. Lequel des chiffres discutés (César, Affine et Vigenère) est le chiffre le plus sécurisé, et pourquoi ? Vous devriez principalement considérer à la fois les attaques statistiques et par force brute.

Exercice 5 : Test de Performance

Veuillez écrire un script pour mesurer les performances des chiffres précédemment implémentés. Dans ce test, vous devriez mesurer le temps d'exécution en fonction de la longueur du message. Un exemple pour calculer le temps d'exécution de n'importe quelle opération est inclus ci-dessous :

```
from datetime import datetime

start_time = datetime.now()
# ajoutez votre code ici
end_time = datetime.now()
print('Durée : {}'.format(end_time - start_time))
```

Fournissez le graphique montrant vos résultats mesurés pour tous les chiffres. Résumez brièvement vos observations.