TP 1: Anciens Schémas de Chiffrement : Sécurité et Performance

1 Description du TP

La cryptographie est l'art de dissimuler des informations sous une forme illisible. La cryptographie moderne est un domaine qui croise les mathématiques, l'informatique et le 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 Internet. Le chiffrement des données joue un rôle majeur dans la sécurité. Par exemple, les banques utilisent des méthodes de chiffrement pour protéger les transactions financières et les numéros d'identification de leurs clients.

De nombreuses entreprises en ligne utilisent des techniques de chiffrement pour sécuriser les transactions effectuées via des cartes de crédit. Les clients souhaitent savoir si leurs informations financières sont en sécurité. Votre directeur informatique vous donnera des instructions pour chiffrer les données à l'aide de divers algorithmes afin de sécuriser les informations de l'organisation.

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

Figure 1: Correspondance entre les lettres de l'alphabet et leurs indices.

1.1 Environnement du TP

Pour ce laboratoire, vous devez écrire du code pour différentes techniques de chiffrement classiques et les analyser. Assurez-vous d'ajouter des titres, des étiquettes d'axe et des légendes aux graphiques.

1.2 Objectifs du TP

Ce laboratoire 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 (2 heures sous surveillance en laboratoire et 2 heures supplémentaires non supervisées).

2 Tâches du TP

Dans ce TP, nous appliquerons plusieurs chiffrements classiques tels que le chiffre de César, et analyserons leur niveau de sécurité.

Exercice 1: Chiffre de César

Le but de cet 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 $k = 3$) utilisé par Jules César.

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

Pour récupérer le texte en clair :

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

où p et c représentent respectivement la valeur numérique des lettres de l'alphabet original et chiffré. De plus, $\%$ est la fonction modulo qui renvoie le reste de la division. En outre, la fonction d'index fournit la représentation décimale des lettres de l'alphabet. L'utilisation du reste de la division à la fin nous garantira que le caractère chiffré/dechiffré fera toujours partie de l'alphabet (supérieur ou égal à 0 et inférieur à 26).

```
alphabet="abcdefghijklmnopqrstuvwxyz"

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

```
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 la fonction de chiffrement/déchiffrement à un message aléatoire et vérifiez que vous récupérez le même texte en clair.

3. Implémentez une fonction qui effectue une attaque par force brute sur un texte chiffré.

4. Affichez la sortie de votre fonction de chiffrement pour les paires suivantes:

   - k = 6, texte en clair = "Get me a vanilla ice cream, make it a double."
   - k = 15, texte en clair = "I don't much care for Leonard Cohen."
   - k = 16, texte en clair = "I like root beer floats."

Exercice 2 : Chiffrement Affine

Le but de cette question est d'implémenter des fonctions Python pour le chiffrement/déchiffrement avec le chiffre affine, ainsi que des attaques. Une amélioration du chiffre de César est le chiffre affine donné par :

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

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 doit 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 $gcd(a,\ n) = 1$. De plus, vous devez utiliser cette règle pour trouver a^{-1}, tel que $a \times a^{-1} \; mod \; n = 1$, 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 affine, étant donné une clé composée d'une paire d'entiers (a, b), tous deux dans $1,\ 2,\ \ldots,\ 25$ avec a qui doit avoir un inverse. Les fonctions doivent fonctionner sur des chaînes de caractères et laisser tout caractère non alphabétique inchangé. 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 après avoir appliqué la fonction de chiffrement affine avec les mêmes coefficients. Déchiffrez-le en utilisant la fonction de déchiffrement correspondante.

Exercice 3: Chiffre de Vigenère

Le but de cet exercice est d'implémenter des fonctions Python pour le chiffrement/déchiffrement avec le chiffre de Vigenère, ainsi que pour les attaques. Le chiffre de Vigenère est une méthode de chiffrement des 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 se fait à l'aide du carré de Vigenère ou de la table de Vigenère.

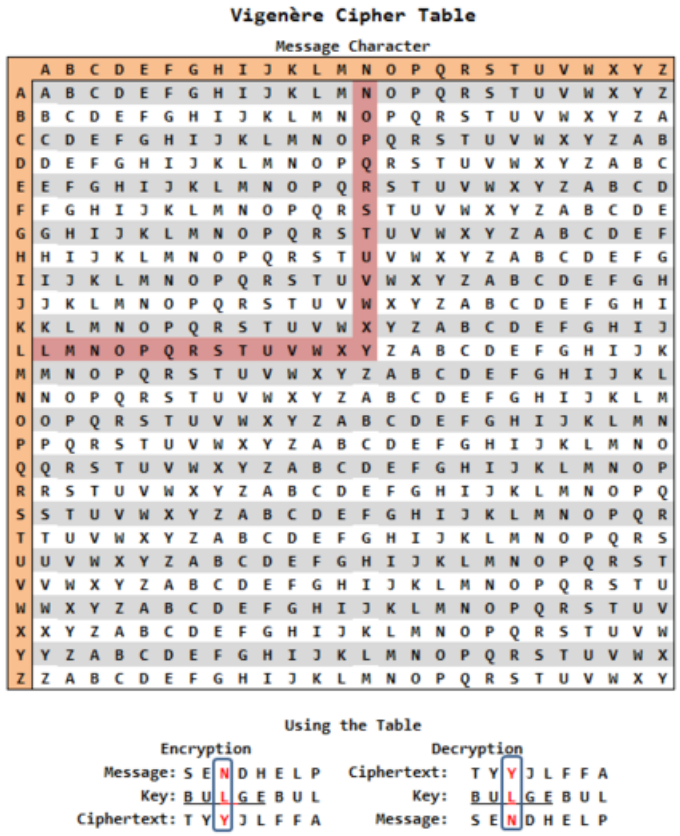

Figure 2: Le carré de Vigenère ou table de Vigenère.

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

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

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

$$c_i = (p_i + k_i) \ \%26 \tag{3}$$

Le déchiffrement peut être effectué selon l'équation suivante :

$$d_i = (c_i - k_i) \ \%26 \tag{4}$$

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

1. Implémentez une fonction pour le chiffrement et une autre pour le déchiffrement de Vigenère.
2. Appliquez-la à un message aléatoire et chiffrez/déchiffrez le message.

Exercice 4 : Fonctions d'Analyse de Sécurité

Dans cet exercice, vous écrirez un message que vous souhaitez protéger dans un nouveau fichier. Le message doit être choisi librement, mais il est impératif qu'il ait une longueur supérieure à 64 caractères pour permettre une analyse plus approfondie. En général, plus le message est long, plus l'analyse qui suivra sera précise.

1. Écrivez une fonction `frequenceLettres` qui prend en paramètre le texte chiffré et renvoie un vecteur de taille 26, où chaque cellule d'indice i contient la probabilité d'occurrence de la lettre associée. Utilisez ensuite cette fonction pour tracer la distribution de fréquence des lettres du texte en clair et du texte chiffré pour chaque méthode de chiffrement.
2. Écrivez une fonction qui génère un graphique de dispersion illustrant la relation entre le texte en clair ($x(t)$) et le texte chiffré ($y(t)$). Ce graphique doit montrer comment chaque caractère du texte en clair varie par rapport à son caractère correspondant dans le texte chiffré. Créez des graphiques de dispersion pour chaque algorithme de chiffrement en utilisant la fonction `plt.scatter(x, y)`.
3. Écrivez une fonction qui calcule le nombre (ou le pourcentage) d'éléments différents entre deux vecteurs d'entrée de même longueur. Cette fonction sera utilisée pour déterminer le pourcentage de lettres différentes entre le texte en clair et le texte chiffré pour chaque méthode de chiffrement.
4. Que remarquez-vous à propos des résultats des histogrammes lorsque le texte est chiffré avec l'algorithme de Vigenère par rapport à ceux obtenus avec l'algorithme de César ?
5. Parmi les algorithmes de chiffrement discutés (César, Affine, et Vigenère), lequel vous semble le plus sécurisé, et pourquoi ? Vous devez principalement prendre en compte les attaques statistiques et par force brute.

Exercice 5 : Test de Performance

Écrivez un script pour mesurer les performances des algorithmes de chiffrement précédemment implémentés. Dans ce test, vous devrez mesurer le temps d'exécution en fonction de la longueur du message. Un exemple de code pour calculer le temps d'exécution de n'importe quelle opération est fourni 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))
```

Générez un graphique illustrant vos résultats pour tous les algorithmes de chiffrement testés. Résumez brièvement vos observations.

Exercice 6 : Attaques Statistiques sur les Chiffres de Substitution

Analyse de Fréquence

- Écrivez une fonction qui effectue une analyse de fréquence sur un texte chiffré. La fonction doit afficher les fréquences des lettres dans le texte chiffré et les comparer avec les fréquences des lettres

dans la langue française. En utilisant cette fonction, essayez de déchiffrer un message chiffré avec le chiffrement de César sans connaître la clé.

Chiffrement Affine

- Implémentez une attaque par force brute sur un texte chiffré par le chiffre affine. Considérez toutes les combinaisons possibles de paramètres multiplicatifs et additifs.
- Discutez des difficultés potentielles rencontrées lors de cette attaque par rapport au chiffre de César.

Cryptanalyse Différentielle

- Écrivez une fonction qui compare deux textes chiffrés dérivés de textes en clair légèrement différents. La fonction doit calculer et afficher le pourcentage de différence entre les deux textes chiffrés. Utilisez cette analyse pour évaluer la sensibilité de chaque algorithme de chiffrement aux petites modifications du texte en clair.