République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Abderrahmane Mira de Béjaia

Faculté des Sciences Exactes
Département de Recherche Opérationnelle

Mémoire de fin d'études

Mathématiques Appliquées

Spécialité : Modélisation mathématique et évaluation des performances des réseaux

Exploration de quelques méthodes de cryptanalyse et leur application sur des exemples concrets

Présenté par :

MECHANI Sarah

Soutenu devant le jury composé de :

AOUDIA Zohra	Université de Béjaia	Présidente
BOULFEKHAR Samra	Université de Béjaia	Examinatrice
ASLI Larbi	Université de Béjaia	Examinateur
DJABRI Rabah	Université de Béjaia	Promoteur

Promotion : 2022-2023

Dédicaces

J'ai l'honneur de dédier ce travail à :

Je dédie ce travail en premier lieu à moi-même, après une année pleine de défis et de rebondissements. Grâce à ma détermination et à la grâce de Dieu, j'ai réussi à franchir toutes les étapes et à apporter les dernières touches à ce mémoire.

J'adresse mes profonds remerciements à mes parents, dont l'amour et l'encouragement ont été une source inépuisable de soutien. Ma réussite leur revient de droit, car ils ont toujours cru en moi et m'ont poussée à donner le meilleur de moi-même.

Je souhaite également dédier ce mémoire à ma soeur Lilia, ainsi qu'à mes deux frères Massinissa et Hichem. Leur présence, leurs encouragements et leur soutien indéfectible ont été essentiels tout au long de mon parcours.

Je tiens à exprimer ma reconnaissance envers Katia, Djedjiga, Chahinez, Nawal, Khalida, Hadda, Cinhinane, Lina, Asma et tant d'autres qui ont contribué à ce que je ne perde pas le lien avec le monde extérieur au cours de ces cinq années. Leurs échanges, leurs conseils et leur amitié ont été précieux pour moi.

Enfin, je souhaite partager une leçon que j'ai apprise cette année :
Les bonnes décisions sont le fruit de l'expérience que l'on a forgé en prenant de mauvaises décisions.

Cette dédicace est un témoignage de ma gratitude envers tous ceux qui ont

contribué à mon parcours, et elle représente également un rappel de

l'importance de l'apprentissage continu et de la persévérance.

Sarah

Remerciements

En tout premier lieu, je tiens à exprimer ma profonde gratitude envers le Bon Dieu, tout puissant, pour m'avoir accordé la force et la persévérance nécessaires pour surmonter les épreuves et aboutir à la réalisation de ce mémoire.

Je voudrais ensuite adresser mes sincères remerciements à Dr Djabri Rabah, mon directeur de mémoire, pour son soutien indéfectible, son encadrement attentif et ses conseils éclairés. La présence bienveillante et sa complémentarité ont joué un rôle essentiel dans la réussite de ce travail de recherche.

Je tiens également à exprimer ma reconnaissance envers les honorables membres du jury pour avoir accepté de consacrer leur temps et leur expertise à l'évaluation et à la correction de ce mémoire. Leurs remarques et leurs suggestions constructives ont contribué à améliorer la qualité de mon travail.

Mes remerciements les plus chaleureux vont à ma famille, mes parents, mes frères et sœurs, pour leur soutien inconditionnel et leur présence constante à mes côtés tout au long de ces années d'études. Leur amour, leurs encouragements et leurs sacrifices ont été une source de motivation et de force dans mon parcours académique.

Enfin, je souhaite exprimer ma gratitude envers toutes les personnes qui ont contribué de près ou de loin à la réalisation de ce mémoire. Leur aide, leurs conseils et leur soutien moral ont été précieux et ont nourri ma réflexion.

Je suis consciente que cette liste ne saurait être exhaustive, mais je tiens à

adresser mes remerciements les plus sincères à chacune de ces personnes qui

ont joué un rôle significatif dans mon parcours.

TABLE DES MATIÈRES

TABLE DES FIGURES

LISTE DES TABLEAUX

INTRODUCTION GÉNÉRALE

La cryptographie est une discipline essentielle pour la protection des données et la sécurisation des communications. Elle permet de garantir la confidentialité, l'intégrité et l'authenticité des informations échangées, que ce soit sur les réseaux informatiques ou dans d'autres domaines où la sécurité est primordiale [21]. Cependant, même les systèmes de cryptographie les plus robustes peuvent être sujets à des attaques visant à briser leur sécurité.

C'est là qu'intervient la cryptanalyse, une discipline complémentaire de la cryptographie qui se focalise sur l'étude et la recherche de vulnérabilités dans les systèmes cryptographiques [1]. En tant que processus visant à casser des codes ou à décoder des messages chiffrés sans avoir connaissance des clés secrètes, la cryptanalyse joue un rôle crucial dans la compréhension et l'amélioration de la sécurité des systèmes cryptographiques.

La structure de ce mémoire se présente de la manière suivante :

Le premier chapitre de ce mémoire est consacré à l'arithmétique et à la théorie des nombres. Il explore plusieurs concepts essentiels tels que la division d'Euclide, le plus grand commun diviseur (PGCD), l'algorithme d'Euclide, la décomposition en facteurs premiers, l'algorithme d'Euclide étendu et l'anneau $\mathbb{Z}_n$, etc. Cette section posera les fondements essentiels pour appréhender les concepts mathématiques qui seront explorés tout au long du mémoire.

Dans le deuxième chapitre, nous explorerons le jargon spécifique utilisé en cryptographie, ainsi que ses fondements théoriques. De plus, nous examinerons quelques systèmes de chiffrement basiques de la cryptographie tels que le chiffrement par décalage, le chiffrement de Vigenère, le chiffrement affine et le chiffrement de Hill [23].

Dans le troisième chapitre, nous nous pencherons aux cryptosystèmes modernes qui jouent un rôle central dans la préservation de la confidentialité et de l'intégrité des données. Nous examinerons en détail à la fois le chiffrement symétrique et le chiffrement asymétrique qui sont largement utilisés de nos jours. De plus, nous aborderons les aspects liés aux fonctions de hachage et aux signatures numériques. Outre l'étude des différentes techniques de la cryptographie, nous nous intéresserons également à la cryptanalyse qui consiste à analyser les systèmes de chiffrement afin de les compromettre.

Au quatrième chapitre, nous explorerons en détail les principales méthodes de cryptanalyse utilisées, telles que l'analyse de fréquence, la recherche exhaustive de clé, la cryptanalyse linéaire et la cryptanalyse différentielle, en fournissant des exemples concrets d'application [18].

Le cinquième chapitre abordera l'importance de la programmation dans le domaine de la cryptographie. Nous explorerons l'utilité de la programmation et présenterons des exemples d'attaques par analyse de fréquence et d'attaque exhaustive réalisées à l'aide du logiciel MATLAB. Tout d'abord, vous découvrirez un exemple de chiffrement par substitution mono-alphabétique d'un cryptogramme à l'aide d'une table de substitution aléatoire. Ce cryptogramme sera ensuite utilisé dans la fonction "analyseDeFrequence", qui permettra d'appliquer une attaque par analyse de fréquence afin de récupérer le texte original. Le résultat correspondant sera ensuite affiché.

De même, nous présenterons un exemple de fonction qui applique une attaque par force brute sur un cryptogramme chiffré par décalage. Le résultat de cette attaque sera également affiché.

En fin, la conclusion fournira une réflexion globale sur les exemples d'application présentés et leurs implications dans le domaine de la cryptanalyse.

À travers ce mémoire, je m'efforcerai d'approfondir mes connaissances sur la cryptanalyse et d'explorer de manière rigoureuse les méthodes qui permettent de remettre en question la sécurité des systèmes cryptographiques, tout en contribuant ainsi à la protection des données et à la sécurité des communications dans notre monde numérique en constante évolution.

CHAPITRE 1

ARITHMÉTIQUES ET THÉORIE DES NOMBRES

Introduction

L'objectif de ce chapitre est de fournir un aperçu rapide des principaux outils utilisés pour manipuler les nombres et leurs opérations de base en cryptographie.

1.1 Division d'Euclide

Pour tout $a \in \mathbb{N}^*$ et $b \in \mathbb{Z}$, il existe de manière unique $q, r \in \mathbb{Z}$ satisfaisant les propriétés suivantes :

- $a = b \cdot q + r$,
- $0 \leq r < b$.

On dit que "b divise a" (noté $b|a$) ou que "a est un multiple de b" ou encore que "b est un diviseur de a" lorsque le reste de la division de a par b est zéro ($r = 0$). Pour deux nombres $m, n \in \mathbb{Z}$, un diviseur commun est un nombre naturel d qui divise simultanément m et n.

1.2 Plus grand commun diviseur (PGCD)

Soient $a, b \in \mathbb{Z}^*$. Le (PGCD) entre deux entiers a et b est le nombre unique $g \in \mathbb{N}$ qui possède les propriétés suivantes :

- $g|a$ et $g|b$,
- Pour tout $c \in \mathbb{N}$: si $c|a$ et $c|b$, alors $c|g$.

La notation pour $g = \mathrm{PGCD}(a, b)$, ou simplement $g = (a, b)$. Si $(a, b) = 1$, alors a et b sont appelés des nombres premiers entre eux. Il convient de noter que les nombres premiers entre eux sont également

appelés nombres premiers relatifs. Une notion liée au (PGCD) est le PPCM (Plus petit commun multiple) entre deux nombres entiers a et b. Il est noté comme $\mathrm{PPCM}(a, b)$, ou simplement $[a, b]$ et représente le plus petit nombre naturel divisible par a et b.

Il existe plusieurs méthodes pour trouver le (PGCD) de deux nombres. Voici deux méthodes couramment utilisées :

1.3 Algorithme d'Euclide

Il s'agit d'une méthode simple et efficace pour trouver le (PGCD) de deux nombres. Les étapes de l'algorithme sont les suivantes :

1– Divisez le plus grand nombre par le plus petit. Notez le reste.

2– Divisez le diviseur précédent par le reste. Notez le reste.

3– Continuez ce processus de division en prenant le dernier reste comme nouveau diviseur et le précédent comme nouveau reste jusqu'à obtenir un reste nul.

4– Le dernier diviseur est le (PGCD) des deux nombres.

Voici un exemple pour trouver le (PGCD) de 56 et 48 :

$56 = 1 \times 48 + 8,$

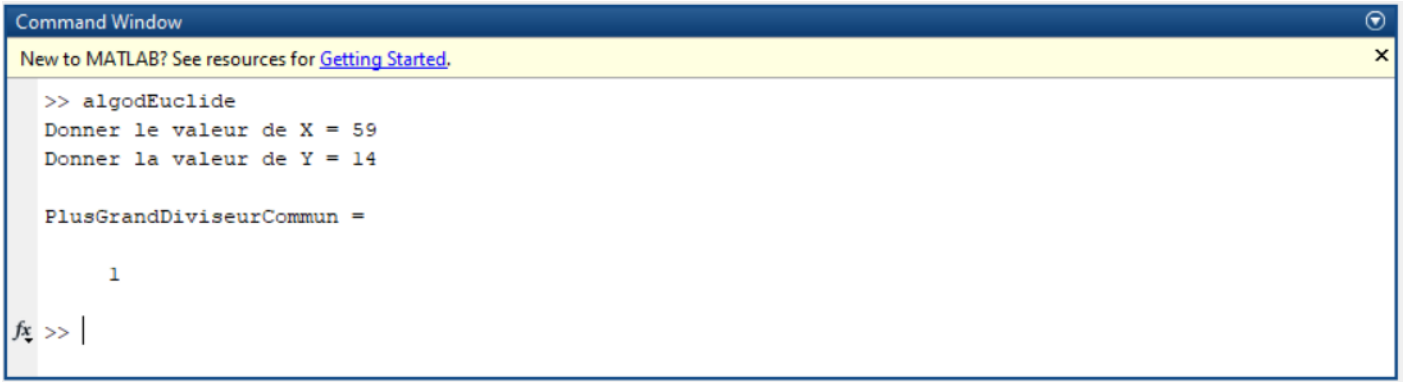
$48 = 6 \times 8 + 0.$

Le dernier diviseur est 8, donc le (PGCD) de 56 et 48 est 8.

La fonction ci-dessous permet de retourner le PGCD entre deux entiers positifs x et y :

```
% L'algorithme d'Euclide sous MATLAB
% Les nombres a saisir doivent etre des nombres positifs.
 x = input('Donner le valeur de X = ');
 y = input('Donner la valeur de Y = ');

 % Validation de X
 if isempty(x)
 error 'Nombre X -> Vous devez entrer une valeur.
 Les valeurs vides ne sont pas autorisees';
 else
 x = abs(x);
 %La valeur de x est prise en valeur absolue en utilisant la fonction abs().
 Cela garantit que x est un nombre positif,
 meme si une valeur negative a ete initialement entree.
 end
```

```
% Validation de Y
if isempty(y)
error 'Nombre Y -> Vous devez entrer une valeur.
Les valeurs vides ne sont pas autorisees';
else
y = abs(y);

end

% Calcul du reste
reste = x - y*floor(x/y); % floor(x/y) renvoie la partie entiere
de la division de x par y, en ignorant la partie decimale.
while reste ~= 0
x = y;
y = reste ;
reste = x - y*floor(x/y);
end
% Afficher le resultat
PlusGrandDiviseurCommun = y
```



1.4 Décomposition en facteurs premiers

Cette méthode consiste à décomposer les deux nombres en facteurs premiers, puis à trouver les facteurs communs. Le PGCD est le produit de ces facteurs communs.

Voici un exemple pour trouver le PGCD de 84 et 120 :

$84 = 2 \times 2 \times 3 \times 7$,

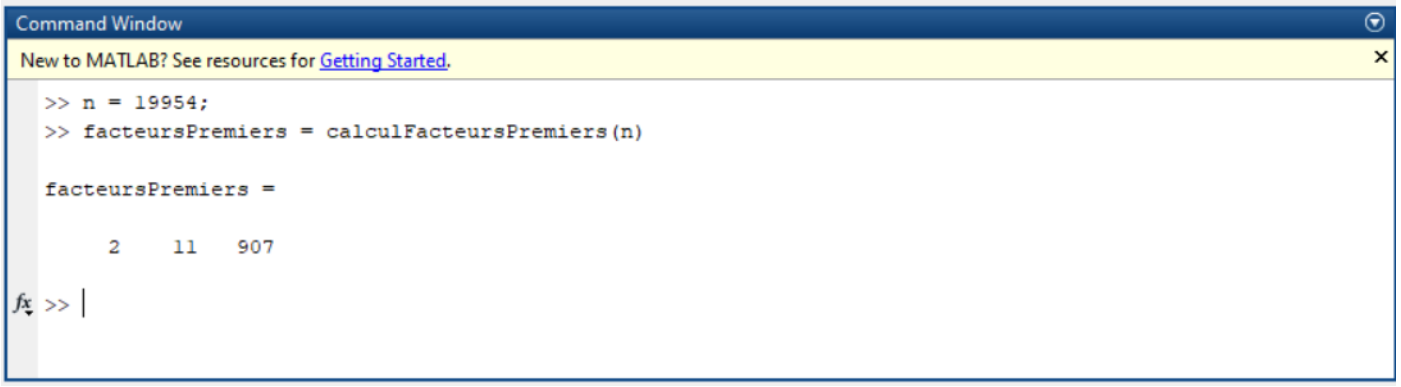
$120 = 2 \times 2 \times 2 \times 3 \times 5$.

Les facteurs communs sont $2 \times 2 \times 3 = 12$, donc le PGCD de 84 et 120 est 12.

Ces deux méthodes sont équivalentes, mais l'algorithme d'Euclide peut être plus rapide pour les grands nombres.

La fonction "facteursPremiers" permet de retourner le vecteur des facteurs premiers d'un entier n donné :

```
function facteursPremiers = calculFacteursPremiers(n)
% Initialisation du vecteur des facteurs premiers
facteursPremiers = [];
% Diviseur initial
diviseur = 2;
while n > 1
if rem(n, diviseur) == 0 % Verifie si le diviseur est un facteur premier.
'rem(n , diviseur)' est utilisee pour calculer le reste de
la division entiere de n par diviseur.
facteursPremiers = [facteursPremiers, diviseur]; % Ajoute le diviseur a
 la liste des facteurs premiers.
n = n / diviseur; % Ajoute le diviseur a la liste des facteurs premiers.
else
diviseur = diviseur + 1; % Incremente le diviseur pour passer au nombre suivant.
end
end
end
```



1.5 Algorithme d'Euclide étendu

L'algorithme d'Euclide étendu est une extension de l'algorithme d'Euclide qui permet non seulement de trouver le plus grand commun diviseur (PGCD) de deux nombres, mais aussi de trouver les coefficients de Bézout.

Définition 1.1. Les coefficients de Bézout sont deux entiers qui permettent d'exprimer le PGCD de deux nombres entiers comme une combinaison linéaire de ces deux nombres.
Soient a et b deux entiers non nuls. Les coefficients de Bézout sont deux entiers u et v tels que :

$$\mathrm{PGCD}(a, b) = u \times a + v \times b,$$

En utilisant l'algorithme d'Euclide étendu, on trouve les coefficients de Bézout u et v en remontant les étapes de l'algorithme, en utilisant les relations :

$$r_n = r_{n-2} - q_n \times r_{n-1}.$$

Où r_n est le reste à l'étape n, q_n est le quotient à l'étape n et $r_0 = a$, $r_1 = b$.

Exemple 1.1. Prenons l'exemple de $a = 17$ et $b = 5$. Nous voulons trouver les coefficients de Bézout u et v tels que : $\mathrm{PGCD}(17, 5) = u \times 17 + v \times 5 = 1$.
On commence par appliquer l'algorithme d'Euclide pour trouver le $\mathrm{PGCD}(17, 5)$:
$17 = 3 \times 5 + 2$,
$5 = 2 \times 2 + 1$,
$2 = 2 \times 1 + 0$. Le $\mathrm{PGCD}(17, 5)$ est donc 1.
Ensuite, on utilise l'algorithme d'Euclide étendu pour trouver les coefficients de Bézout u et v. On part de l'équation où le reste est égal à 1 et on remonte les équations précédentes en exprimant le reste comme une combinaison linéaire de a et b :
$1 = 5 - 2 \times 2$,
$1 = 5 - 2 \times (1 \times 7 - 3 \times 5)$,
$1 = -2 \times 17 + 7 \times 5$.
Donc, les coefficients de Bézout pour $a = 17$ et $b = 5$ sont $u = -2$ et $v = 7$.

1.6 Anneau $\mathbb{Z}_n$

L'anneau $\mathbb{Z}_n$, également connu sous le nom d'anneau des entiers modulo n, est une structure algébrique importante en cryptographie. Il est construit à partir de l'ensemble des entiers relatifs $\mathbb{Z}$ en prenant les classes d'équivalence modulo n.

↪ $\mathbb{Z}$: Le symbole $'\mathbb{Z}'$ représente l'ensemble des entiers relatifs. Cet ensemble comprend tous les nombres entiers positifs, négatifs et zéro $\{\ldots, -3, -2, -1, 0, 1, 2, 3, \ldots\}$.

↪ n : 'n' est un entier positif donné. Il représente le modulo ou le diviseur par lequel les entiers sont divisés pour former des classes d'équivalence. Les classes d'équivalence sont basées sur le reste de la division des entiers par n.

Supposons que nous travaillions avec $\mathbb{Z}_6$. Cela signifie que nous divisons les entiers relatifs en classes d'équivalence modulo 6. Les classes d'équivalence seraient :

- [0] : L'ensemble des entiers multiples de 6 $(\ldots, -12, -6, 0, 6, 12, \ldots)$,

- [1] : L'ensemble des entiers qui ont un reste de 1 lorsqu'ils sont divisés par 6 $(\dots, -11, -5, 1, 7, 13, \dots)$,
- [2] : L'ensemble des entiers qui ont un reste de 2 lorsqu'ils sont divisés par 6 $(\dots, -10, -4, 2, 8, 14, \dots)$,
- [3] : L'ensemble des entiers qui ont un reste de 3 lorsqu'ils sont divisés par 6 $(\dots, -9, -3, 3, 9, 15, \dots)$,
- [4] : L'ensemble des entiers qui ont un reste de 4 lorsqu'ils sont divisés par 6 $(\dots, -8, -2, 4, 10, 16, \dots)$,
- [5] : L'ensemble des entiers qui ont un reste de 5 lorsqu'ils sont divisés par 6 $(\dots, -7, -1, 5, 11, 17, \dots)$.

Ainsi, $\mathbb{Z}_n$ est l'anneau des entiers modulo 6 avec ces six classes d'équivalence distinctes.

1.6.1 Entiers modulo n

Définition 1.2.

Pour deux entiers a et b, avec $n > 1$, on dit qu'ils sont congruents $modulo\ n$ si la différence $a - b$ est un multiple de n. En d'autres termes, il existe un entier q tel que $a - b = qn$.

Les expressions suivantes sont synonymes et utilisées pour décrire cette relation : être dans la même classe résiduelle $modulo\ n$, être dans la même classe de congruence $modulo\ n$, ou plus simplement, être égaux $modulo\ n$.

On utilise indifféremment les notations :

$$a \equiv b\ mod\ n,\ a \equiv b\ (n),\ a = b\ mod\ n,\ a = b(n).$$

Exemple 1.2. Considérons les entiers $a = 15$ et $b = 22$. Pour vérifier s'ils sont congruents modulo 7, nous devons calculer la différence $a - b$:

$$a - b = 15 - 22 = -7,$$

Maintenant, nous devons vérifier si cette différence est un multiple de $n = 7$. Dans ce cas, puisque $-7 = (-1) \times 7$, nous avons trouvé un entier $q = -1$ qui satisfait $a - b = qn$.

Remarque 1.1. La relation $a \equiv b\ mod\ n$ est une relation d'équivalence sur $\mathbb{Z}$. Les classes d'équivalence sont appelées les classes de congruence $modulo\ n$. Leur ensemble est noté $\mathbb{Z}$.

Les restes possibles dans la division euclidienne par n sont : $0, 1, \dots, (n-1)$. Il y a donc exactement n classes distinctes $modulo\ n$ et $\mathbb{Z}_n = \{0, 1, 2, \dots, (n-1)\ mod\ n\}$.

1.6.2 Addition et multiplication dans $\mathbb{Z}_n$

Propriété 1.1. *L'addition et la multiplication sont compatibles avec la relation de congruence $modulo\ n$, c'est-à-dire :*

$$si \begin{cases} a_1 \equiv b_1(\ mod\ n), \\ a_2 \equiv b_2(\ mod\ n), \end{cases} \qquad alors \begin{cases} a_1 + a_2 \equiv b_1 + b_2\ (mod\ n), \\ a_1.a_2 \equiv b_1.b_2\ (mod\ n). \end{cases}$$

Corollaire 1.1. *On définit l'addition et la multiplication dans $\mathbb{Z}_n$ par :*

$$si \begin{cases} ((a\ mod\ n) + (b\ mod\ n))\ mod\ n \equiv (a + b)\ mod\ n, \\ (a\ mod\ n)(b\ mod\ n) \equiv ab\ (mod\ n). \end{cases}$$

Les règles de calcul usuelles dans $\mathbb{Z}$ restent valables dans $\mathbb{Z}_n$. En particulier, on a :

$\hookrightarrow$ $x + 0 = x \ mod \ n$,

$\hookrightarrow$ $x + (-x) = 0 \ mod \ n$ (*l'opposé de* $x \ mod \ n$ *est* $-x \ mod \ n$),

$\hookrightarrow$ $x.1 = x \ mod \ n$, $x.0 = 0 \ mod \ n$,

$\hookrightarrow$ $x(y + z) = xy + xz \ mod \ n$,

$\hookrightarrow$ $(x \ mod \ n) \ k = xk \ mod \ n$,

On dit que ($\mathbb{Z}_n$, +, $\cdot$) est un anneau commutatif et unitaire.

1.6.3 Inversibles dans $\mathbb{Z}_n$

Définition 1.3. Pour un entier a donné, on dit que a est inversible $modulo \ n$, ou que $a \ mod \ n$ est inversible dans $\mathbb{Z}_n$, si et seulement s'il existe un entier b tel que le produit ab est congruent à $1 \ modulo \ n$, c'est-à-dire $ab \equiv 1 \ (mod \ n)$.

$\mathbb{Z}_n^*$ représente l'ensemble des éléments de $\mathbb{Z}_n$ pour lesquels il existe un inverse $modulo \ n$.

L'inverse de $a \ modulo \ n$ est noté $a^{-1} \ mod \ n$.

Proposition 1.1. *Si un entier a est inversible $modulo \ n$, cela signifie qu'il existe un unique inverse $modulo \ n$ pour a.*

Remarque 1.2. 0 n'est jamais inversible $modulo \ n$ puisque $0.b = 0 \ mod \ n$ et 1 est toujours inversible puisque $1.1 = 1 \ mod \ n$.

Propriété 1.2. *1. Si p est un nombre premier, tout élément non nul de $\mathbb{Z}_p$ est inversible.*

2. Réciproquement, si n est un entier, tel que tout élément non nul de $\mathbb{Z}_n$ est inversible, alors n est premier.

3. Un diviseur de zéro de $\mathbb{Z}_n$ est un entier a tel que $a \neq \ mod \ n$ et tel qu'il existe $b \neq 0 \ mod \ n$ tel que $ab = 0 \ mod \ n$. Si a est un diviseur de zéro $modulo \ n$, alors il ne peut pas être inversible $modulo \ n$.

Conclusion

Ce chapitre a abordé des outils essentiels pour manipuler les opérations arithmétiques de base, notamment l'algorithme d'Euclide étendu, le calcul des facteurs premiers et de leurs produits, ainsi que le calcul de l'inverse modulaire d'un nombre. Ces concepts théoriques clés constituent un point de départ dans le domaine de la théorie des nombres.

CHAPITRE 2

NOTIONS FONDAMENTALES DE LA CRYPTOGRAPHIE

Introduction

La cryptographie est l'art de protéger les informations en les rendant illisibles pour les personnes non autorisées. Depuis l'Antiquité, les civilisations ont développé des méthodes de cryptographie pour protéger leurs messages militaires, diplomatiques et commerciaux. De nos jours, la cryptographie est utilisée dans de nombreux domaines, tels que la sécurité des communications sur Internet, la protection des données personnelles et bancaires, ainsi que dans la sécurisation des transactions financières. Dans ce chapitre, nous allons explorer les bases de la cryptographie, les différents types de chiffrement, ainsi que les concepts et les techniques utilisées pour la cryptographie moderne.

2.1 Jargon de la cryptographie

Avant d'explorer les différents types de chiffrement, il est important de comprendre les termes et concepts de base de la cryptographie.

- **Cryptologie :** La cryptologie est la science de la communication sécurisée. Elle englobe à la fois la cryptographie, qui vise à protéger l'information en la rendant incompréhensible, et la cryptanalyse, qui cherche à décoder les informations protégées [20].
- **Cryptographie :** La cryptographie se charge de protéger les données sensibles lors de leurs transmission entre deux interlocuteurs à travers un canal peu sûr en utilisant des techniques de chiffrement [20].
- **Cryptanalyse :** La cryptanalyse est l'étude et l'analyse des systèmes de cryptographie dans le but de développer des techniques permettant de déchiffrer les textes chiffrés et de déduire les messages clairs. En outre, elle permet également d'évaluer la sécurité des mécanismes de chiffrement en

identifiant les vulnérabilités et les faiblesses que les attaquants peuvent exploiter pour compromettre un système [20].

- **Chiffrement :** Le chiffrement est le processus de transformation d'un message clair en un message chiffré, rendant ainsi son contenu incompréhensible pour les tiers non autorisés. Le chiffrement peut être réalisé à l'aide d'un algorithme mathématique et une clé qui doit être connue des deux parties impliquées dans la communication [20].
- **Déchiffrement :** Le déchiffrement est le processus inverse du chiffrement. Il permet de restaurer les données originales à partir des données chiffrées en utilisant une clé de déchiffrement appropriée [20].
- **Texte en clair :** Le texte en clair est simplement le message original ou non chiffré qui doit être protégé contre la lecture ou la modification non autorisée [20].
- **Texte chiffré :** Le texte chiffré est le résultat du processus de chiffrement, où le texte en clair original a été transformé à l'aide d'un algorithme de chiffrement et d'une clé de chiffrement pour produire un texte qui est illisible [20].
- **Clé :** La clé fait référence à une information secrète utilisée pour chiffrer ou déchiffrer les données. Il s'agit généralement d'une chaîne de caractères, souvent aléatoire, qui est utilisée pour transformer un message en un texte chiffré, ou pour décoder un texte chiffré en un message compréhensible.

Il existe deux types de clés en cryptographie : Les clés symétriques et les clés asymétriques. Les clés symétriques sont utilisées dans les algorithmes de chiffrement symétrique, où une même clé est utilisée pour chiffrer et déchiffrer les données. Les clés asymétriques, quant à elles, sont utilisées dans les algorithmes de chiffrement asymétrique, où une paire de clés distinctes est utilisée : une clé publique pour chiffrer les données et une clé privée pour les déchiffrer.

La sécurité des données chiffrées dépend largement de la robustesse de la clé utilisée pour les chiffrer. Par conséquent, les clés doivent être suffisamment complexes et difficiles à deviner pour éviter toute tentative de déchiffrement non autorisée [20].

- **Chiffrement par substitution :** Le chiffrement par substitution est une technique de chiffrement dans laquelle chaque lettre du texte clair est remplacée par une autre lettre selon une règle de substitution. La règle de substitution est généralement une table de correspondance qui associe chaque lettre de l'alphabet à une autre lettre. On reconnaît deux types de chiffrement par substitution :
  - ↪ **Substitution mono-alphabétique :** La substitution mono-alphabétique est une méthode de chiffrement dans laquelle chaque lettre du texte clair est remplacée par une seule lettre de substitution, qui reste constante tout au long du message. Cela signifie que la même lettre du texte clair sera toujours remplacée par la même lettre de substitution dans le texte chiffré. Cependant, cette méthode est vulnérable aux attaques de fréquence, car les lettres

qui apparaissent fréquemment dans le texte clair seront toujours remplacées par la même lettre de substitution dans le texte chiffré [14].

↪ **Substitution poly-alphabétique :** La substitution poly-alphabétique est une méthode de chiffrement dans laquelle chaque lettre du texte clair peut être remplacée par plusieurs lettres de substitution, selon une clé de chiffrement qui indique quelle lettre de substitution utiliser pour chaque lettre du texte clair. Cela signifie que la même lettre du texte clair peut être remplacée par différentes lettres de substitution dans le texte chiffré, ce qui rend l'analyse de fréquence et d'autres méthodes d'attaque plus difficiles. Cette méthode est souvent considérée comme plus sécurisée que la substitution mono-alphabétique [14].

- **Chiffrement par transposition :** Le chiffrement par transposition est une technique de chiffrement dans laquelle les lettres du texte clair sont simplement réarrangées ou permutées selon une certaine règle pour obtenir le texte chiffré. Contrairement au chiffrement par substitution qui remplace chaque lettre par une autre lettre, le chiffrement par transposition ne change pas les lettres du message clair, mais change leur ordre ou leur position [14].
- **Systèmes de chiffrement :** Traditionnellement, la cryptographie est divisée en deux familles, la cryptographie classique et la cryptographie moderne.

  ↪ **Cryptographie classique :** La cryptographie classique est l'ensemble des techniques de chiffrement qui ont été utilisées depuis l'Antiquité jusqu'à la fin du XIXe siècle. Ces techniques étaient principalement basées sur des substitutions et des transpositions de lettres, des chiffres ou des symboles. Les méthodes de chiffrement classiques comprennent, entre autres, la substitution mono-alphabétique, la substitution poly-alphabétique [14].

  ↪ **Cryptographie moderne :** La cryptographie moderne quant à elle, est l'ensemble des techniques de chiffrement qui ont été développées depuis le début du XXe siècle jusqu'à nos jours. La cryptographie moderne utilise des algorithmes mathématiques plus sophistiqués et des clés de chiffrement plus longues pour garantir une sécurité plus élevée. Les méthodes de chiffrement modernes incluent la cryptographie à clé publique, la cryptographie symétrique, les fonctions de hachage, les signatures numériques [14].

Voici un schéma synthétique qui illustre les différentes catégories de cryptosystèmes :

2.2 Histoire de la cryptographie

Dès que les hommes apprirent à communiquer, ils durent trouver des moyens d'assurer la confidentialité de leurs communications. En effet, la première idée qui vient à l'esprit consiste à dissimuler le message, afin que son existence même soit ignorée. Vers -600 avant Jésus-Christ, Nabuchodonosor, le roi de Babylone, avait recours à une méthode originale pour dissimuler des messages confidentiels. Il écrivait des messages sur le crâne rasé de ses esclaves, puis attendait que leurs cheveux aient repoussé avant de les envoyer à ses généraux. Une fois le message transmis, il suffisait de raser à nouveau le

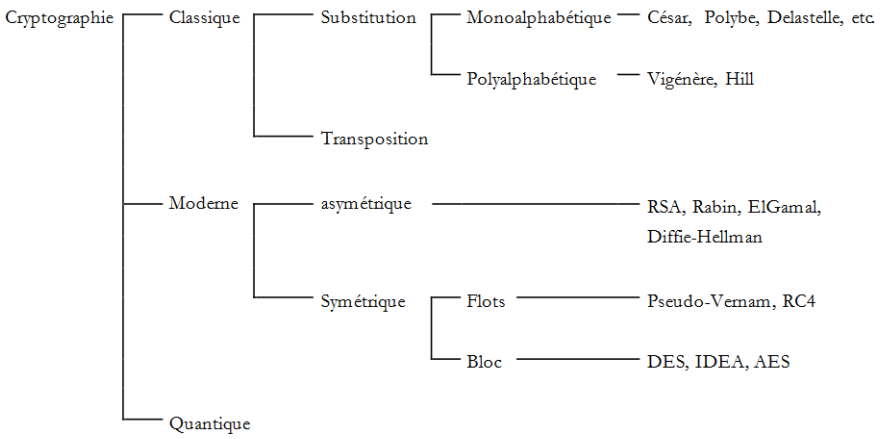


FIGURE 2.1 – Classification des systèmes cryptographiques [14]

crâne du messager pour révéler le texte. Pline l'Ancien a également décrit une autre méthode classique qui consiste à utiliser de l'encre sympathique. Cette encre est invisible à température normale, mais elle peut être révélée par la chaleur d'une bougie. Ces techniques utilisées constituent le domaine de la stéganographie (l'information est cachée et non pas codée).

L'inconvénient de la stéganographie est que même si elle permet de cacher un message, mais elle ne le protège pas de manière sécurisée contre les attaques des personnes malveillantes qui cherchent à intercepter le message. De plus, elle peut être découverte si la méthode de dissimulation est connue ou suspectée, ce qui en fait une méthode de sécurité moins fiable que la cryptographie qui en principe garde le secret d'un message même s'il tombe dans les mains d'une personne autre que son destinataire.

Au commencement la cryptographie était loin d'être aussi complexe et astucieuse qu'à notre époque. La plupart des méthodes de chiffrement reposaient sur deux principes fondamentaux : la substitution (on remplace certains lettres par d'autres) et la transposition (on permute les lettres du message afin de le rendre inintelligible). Les civilisations anciennes ont utilisé plusieurs techniques pour protéger leurs messages, en voici quelques exemples :

- **Scytale :** La scytale est une technique de chiffrement par transposition utilisée dans la Grèce antique. Elle impliquait l'usage d'un bâton de bois ou de métal, sur laquelle un message était écrit en spirale. Pour lire le message, il fallait enrouler une bande de cuir autour de celui-ci de la même manière que le message avait été écrit, de sorte que les lettres du message apparaissaient dans l'ordre correct. La sécurité de la technique reposait sur la longueur de la scytale, qui devait être connue uniquement par les deux parties impliquées. Si une personne interceptait le message, elle ne pourrait pas le lire sans la scytale appropriée [7].
- **Atbash :** Atbash est une méthode de chiffrement par substitution alphabétique inversée utilisée dans l'écriture hébraïque ancienne. Elle consiste à remplacer chaque lettre de l'alphabet par la

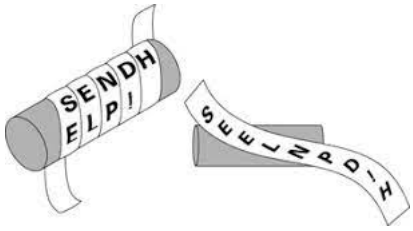


FIGURE 2.2 – Scytale

lettre se trouvant à l'opposé de l'alphabet. Par exemple, la première lettre de l'alphabet, Aleph, est remplacée par la dernière lettre, Tav, la deuxième lettre, Bet, est remplacée par la lettre avant-dernière, Shin, et ainsi de suite. Atbash peut être appliqué à n'importe quel alphabet, y compris l'alphabet latin.

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
Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A

TABLE 2.1 – L'alphabet suivant la technique des Hébreux

Ainsi, le mot "Bonjour" serait chiffré en "Ylivmr" en utilisant la méthode Atbash sur l'alphabet latin.

Au Moyen Âge, les techniques de cryptographie ont continué à évoluer. La substitution de lettres était toujours utilisée, mais des méthodes plus sophistiquées ont également été développées. La plus connue d'entre elles est probablement le code de César (ou bien chiffrement par décalage), consistait à changer les lettres du message que l'on voulait crypter en les remplaçant par des lettres décalées d'un certain nombre de places dans l'ordre de l'alphabet latin, le décalage le plus souvent utilisé est de 3 lettres. Le destinataire pouvait décoder avec succès le message, s'il connaissait le système et le décalage à effectuer [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
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

TABLE 2.2 – Alphabet de César

Au cours de la Renaissance, la cryptographie a connu une nouvelle évolution. Le chiffrement par transposition, a été développé. Des méthodes de chiffrement par substitution plus complexes ont également été créées, notamment le chiffre de Vigenère qui utilise plusieurs chiffrements par décalages et permet de chiffrer une même lettre du texte clair de manière différente en fonction de la clé utilisée. La clé, qui est généralement un mot-clé plus court que le texte à chiffrer, est répétée autant de fois que nécessaire pour atteindre la longueur du message à chiffrer [8]. Le processus de chiffrement est le suivant :

↪ Chaque caractère du message est combiné avec le caractère correspondant de la clé en utilisant le

tableau de Vigenère correspondant. Pour trouver le caractère chiffré, on cherche la lettre de la ligne correspondant au caractère du message et la lettre de la colonne correspondant à la lettre de la clé.

↪ Le caractère chiffré est alors ajouté au message chiffré.

Lettre de la clé \ Lettre du message 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
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

FIGURE 2.3 – Tableau de Vigenère [7]

Au début du 20e siècle, la radio était une technologie relativement nouvelle et en pleine expansion, qui permettait aux gens de communiquer sur de longues distances de manière rapide et efficace. Cependant, cette nouvelle technologie a également créé un nouveau défi en matière de sécurité des communications. Les messages envoyés par radio pouvaient être interceptés par des ennemis potentiels, et il était donc nécessaire de développer des méthodes de chiffrement pour protéger les informations sensibles. C'est dans ce contexte que la machine Enigma a été développée. En utilisant des rotors et une configuration de chiffrement complexe, la machine Enigma était capable de coder des messages de manière efficace et rapide, ce qui a permis à l'armée allemande de communiquer de manière sécurisée pendant la Seconde Guerre mondiale [8].

La cryptographie a connu de nombreux développements au fil des siècles, avec des systèmes de chiffrement de plus en plus sophistiqués. Toutefois, l'arrivée de l'informatique et d'Internet a marqué un tournant décisif dans l'histoire de la cryptographie. En effet, les enjeux de la sécurité informatique ont poussé les chercheurs à perfectionner les techniques de chiffrement pour protéger les données échangées en ligne. Ainsi, la cryptographie est devenue une discipline essentielle pour garantir la confidentialité et la sécurité des informations dans notre monde numérique.

FIGURE 2.4 – Machine Enigma

2.3 Un peu de formalisme mathématique

Un cryptosystème peut être formellement défini comme un quintuplet $\Pi = (\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ qui répond aux critères suivants [6] :

- $\mathcal{P}$: l'espace des messages clairs sur un alphabet $\mathcal{A}$ (qui peut être l'alphabet français, mais qui sera dans la pratique 0, 1 car tout message sera codé en binaire pour pouvoir être traité par ordinateur),
- $\mathcal{C}$: l'espace des messages chiffrés sur un alphabet $\mathcal{B}$ (en général égal à $\mathcal{A}$) ;
- $\mathcal{K}$: l'espace des clés,
- $\mathcal{E}$ = $\{e_k : k \in \mathcal{K}\}$ est l'espace des fonctions de chiffrement $e_k : \mathcal{P} \longrightarrow \mathcal{C}$.
- $\mathcal{D}$ = $\{d_k : k \in \mathcal{K}\}$ est l'espace des fonctions de déchiffrement $d_k : \mathcal{P} \longrightarrow \mathcal{C}$, tel que :

$$d_k(e_k(M)) = M, \forall\, M \in \mathcal{P}.$$

Identifions l'alphabet usuel avec $\mathbb{Z}$ par $A = 0, B = 1, \ldots, Z = 25$. On a alors les systèmes cryptographiques suivants :

2.3.1 Chiffrement par décalage

Définition 2.1.

Le chiffrement par décalage se définit dans $\mathbb{Z}_{26}$, car on utilise 26 lettres dans l'alphabet, on pose [20] :

$$\mathcal{P} = \mathcal{C} = \mathcal{K} = \mathbb{Z}_{26}, \forall\, 0 \leqslant K \leqslant 25,$$

La fonction de chiffrement $e_k : \mathbb{Z}_{26} \longrightarrow \mathbb{Z}_{26}$ est donnée par :

$$e_k = (m + k)\ mod\ 26,$$

La fonction de déchiffrement $d_k : \mathbb{Z}_{26} \longrightarrow \mathbb{Z}_{26}$ est donnée par :

$$d_k = (c - k)\ mod\ 26.$$

Exemple 2.1. Nous allons appliquer le chiffrement de César à la citation de Gandhi qui suit :

"La vie est un mystère qu'il faut vivre, et non un problème à résoudre."

Tout d'abord, nous allons convertir chaque lettre du message en clair en un nombre correspondant à sa position dans l'alphabet. Ainsi, "L" sera remplacé par 11, "a" par 0, "v" par 21, etc.
Ensuite, nous allons appliquer la fonction de chiffrement $e_k(m) = m + k = (m + 3)\ mod\ 26$ à chaque nombre correspondant à une lettre du message. Cela signifie que nous allons ajouter 3 à chaque nombre, en veillant à ce que le résultat reste compris entre 0 et 25. Voici le tableau qui récapitule les étapes de chiffrement :

Lettre	L	A	V	I	E	E	S	T	U	N	M	Y	S	T	E	R	E	...
Nombre	11	0	21	8	4	4	18	19	20	13	12	24	18	19	4	17	4	...
Chiffré	O	D	Y	L	H	H	V	W	X	Q	P	B	V	W	H	U	H	...

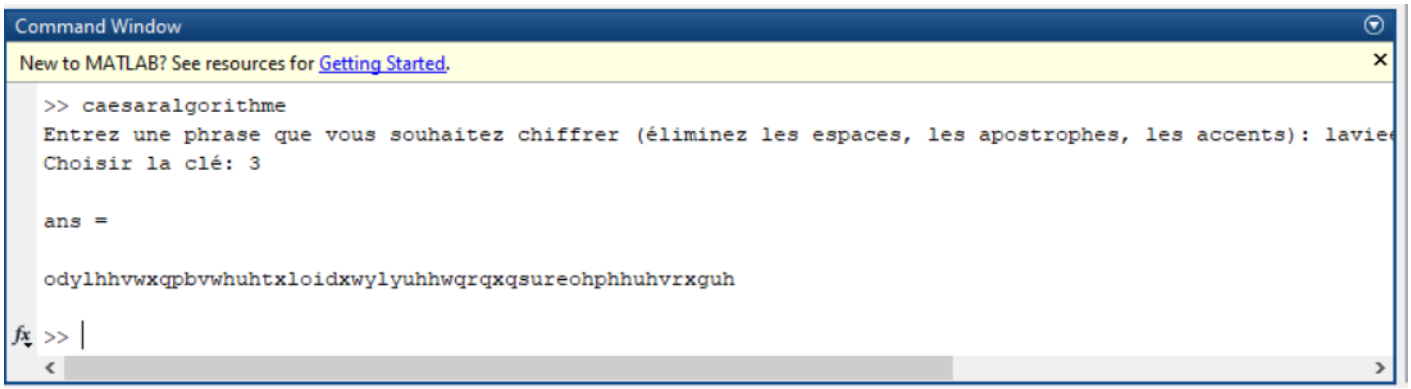
Le texte chiffré complet : "ODYLHHVWXQPBVWHUHTXLOIDXWYLYUHHWQRQXQSUREOHPHDUHVRXGUH".

Pour aller plus vite on propose la fonction suivante "caesaralgorithme()" pour effectuer le chiffrement :

```
function cryptogramme = caesaralgorithme()
alphabet = ['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'];
 % Entrer la phrase a chiffrer
texte_en_clair = input('Entrez une phrase que vous souhaitez chiffrer
(eliminez les espaces, les apostrophes, les accents.) : ','s');
% Choisir la cle de chiffrement
cle = input('Choisir la cle : ');
 % Message d'erreur a afficher en cas d'entree invalide
message_d_erreur = 'Il y a une entree invalide. Veuillez redemarrer le programme.';
% Si la cle est superieure a 26 ou inferieure a 0, elle est
ramenee dans la plage valide de 0 a 25
if cle > 26 || cle < 0
 cle = mod(cle,26);
end
breaking_loop = 0; % Variable pour indiquer si la boucle doit etre interrompue
iteration = 1; % Compteur pour iterer a travers les caracteres du texte en clair
% Boucle pour iterer a travers chaque caractere du texte en clair
while iteration <= length(texte_en_clair)
```

```
% Verifie si le caractere est une lettre majuscule
if ismember(texte_en_clair(iteration),'A':'Z') == 1
% Definit breaking_loop a 1 pour interrompre la boucle
breaking_loop = 1;
% Affecter le message d'erreur au cryptogramme
cryptogramme = message_d_erreur;
 % Verifie si le caractere est une lettre
elseif isletter(texte_en_clair(iteration)) == 1
% Trouve l'indice de la lettre dans l'alphabet
 y = strfind(alphabet,texte_en_clair(iteration));
% Applique le decalage de la cle a l'indice
z = y+cle;
% Si l'indice depasse 26, il est ajuste en soustrayant 26 pour revenir
 a la plage valide de l'alphabet
if z > 26
z = z - 26;
end
% Affecte la lettre chiffree au cryptogramme
cryptogramme(iteration) = alphabet(z);
 % Verifie si le caractere n'est pas une lettre
elseif isletter(texte_en_clair(iteration)) == 0
if plain(iteration) == ' '
cryptogramme(iteration) = texte_en_clair(iteration);
% Si le caractere est un espace, il est simplement copie dans le cryptogramme
else
% Definit breaking_loop a 1 pour interrompre la boucle
breaking_loop = 1;
% Affecte le message d'erreur au cryptogramme
cryptogramme = message_d_erreur;
end
end
% Incremente le compteur d'iteration
iteration = iteration+1;
if breaking_loop == 1
 break
% Si breaking_loop est egal a 1, la boucle est interrompue
end
end
```

end



Remarque 2.1. Ce système de chiffrement n'est pas sûr du tout puisque l'espace des clés ne contient que 26 possibilités. Ainsi, il est possible de les essayer une par une jusqu'à trouver la clé correcte, ce qui rend le système facilement vulnérable aux attaques par force brute.

2.3.2 Chiffrement de Vigenère

Définition 2.2.

Mathématiquement, le chiffrement de Vigenère peut être défini de la manière suivante [20] :

Soit $\mathcal{P}$ = $\mathcal{C}$ = $\mathbb{Z}_{26}^n$ ($n \in N^*$) l'ensemble des messages clairs et chiffrés, où chaque lettre est représentée par un entier entre 0 et 25.

Soit $K = (k_1, \ldots, k_m) \in \mathcal{K}$ l'ensemble des clés, où chaque élément k_i est un entier entre 0 et 25.

La fonction de chiffrement $e_k : \mathbb{Z}_{26}^n \longrightarrow \mathbb{Z}_{26}^n$ est définie par :

$$e_k(x_1, \ldots, x_m) = (x_1 + k_1, \ldots, x_m + k_m) \ mod \ 26,$$

La fonction de déchiffrement $d_k : \mathbb{Z}_{26}^n \longrightarrow \mathbb{Z}_{26}^n$ est définie par :

$$d_k(y_1, \ldots, y_m) = (y_1 - k_1, \ldots, y_m - k_m) \ mod \ 26.$$

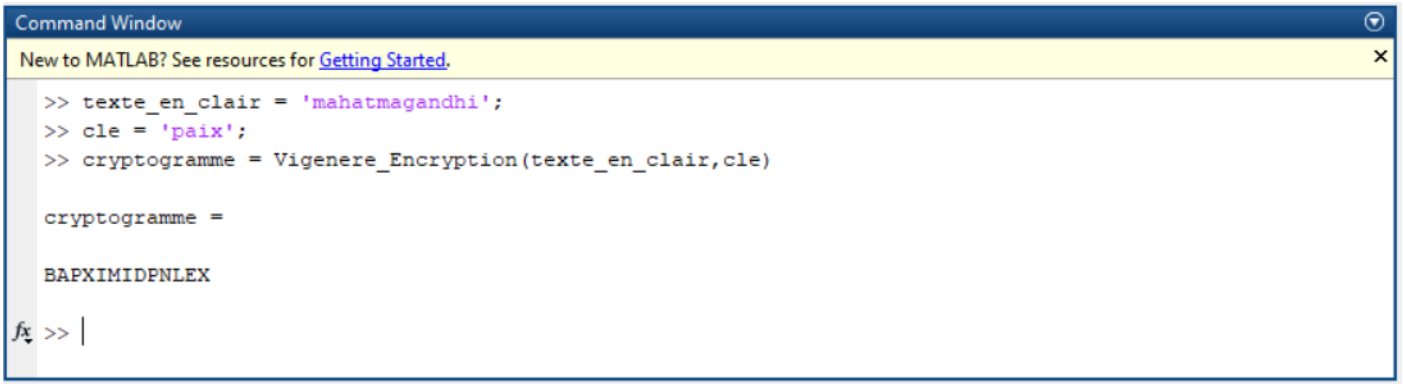
Exemple 2.2. Supposons que nous voulons chiffrer le message "MAHATMAGANDHI" avec la clé "PAIX". Nous représentons chaque lettre du message et de la clé par un nombre entre 0 et 25 selon l'ordre alphabétique, Dans ce cas, nous avons :

Message	M	A	H	A	T	M	A	G	A	N	D	H	I
Nombre	12	0	7	0	19	12	0	6	0	13	3	7	8
Clé	P	A	I	X	P	A	I	X	P	A	I	X	P
Nombre	15	0	8	23	15	0	8	23	15	0	8	23	15

Ensuite, nous ajoutons chaque élément de la clé étendue à l'élément correspondant du message clair, en effectuant les additions modulo 26 pour obtenir le message chiffré "BAPXIMIDPNLEX".

Nombre	1	0	15	23	8	12	8	3	15	13	11	4	23
chiffré	B	A	P	X	I	M	I	D	P	N	L	E	X

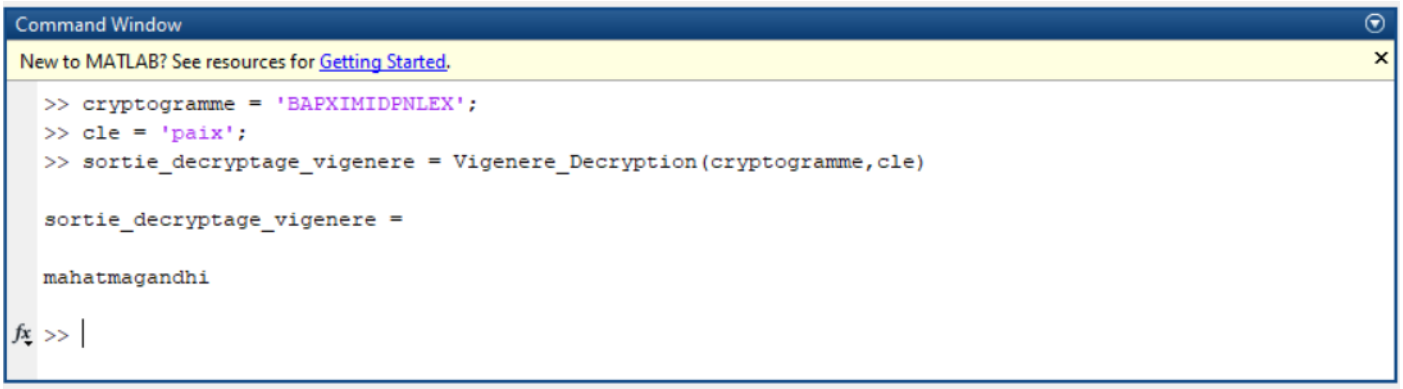
```
function ChaineDeSortie = Shift_Encryption(texte_en_clair,entierModulo)
% Conversion des caracteres du texte en clair en valeurs numeriques
correspondant a leur position dans l'alphabet
 tab = texte_en_clair - 'a';
% Calcul du decalage et application de l'operation modulo 26
 tabDeSortie = mod( tab + entierModulo, 26);
 % Conversion des valeurs numeriques en caracteres correspondant a la
 position dans l'alphabet majuscule
 ChaineDeSortie = char(tabDeSortie + 'A');

function cryptogramme = Vigenere_Encryption(texte_en_clair,cle)
% Conversion de la cle en une sequence de nombres correspondant
a la position dans l'alphabet
 tab = cle - 'a';
long_cle = length(cle);
for i=1 : length(texte_en_clair)
% Calcul de l'indice dans la cle pour le decalage
 ishift = mod(i,long_cle);
 if ishift == 0, ishift = long_cle; end
 % Chiffrement du caractere actuel
cryptogramme(i) = Shift_Encryption(texte_en_clair(i),tab(ishift));
end
```



Le code comporte quelques étapes simples : l'une des étapes principales consiste à convertir la clé de chiffrement en lettres minuscules (voir ligne 3) et à les représenter sous forme de tableau. L'étape suivante consiste à calculer la longueur de la clé de chiffrement (voir ligne 4). De plus, la procédure de chiffrement est basée sur le calcul de l'opération modulo entre l'index de chaque caractère du texte

en clair et la longueur de la clé de chiffrement (voir lignes 5 et 6). La fonction $Shift - Encryption$, son principal objectif est d'effectuer l'opération de décalage, comme décrit dans le contexte théorique.

```
function sortie_decryptage_vigenere = Vigenere_Decryption(cryptogramme, cle)
% Conversion de la cle en une sequence de nombres
correspondant a la position dans l'alphabet
tab = cle - 'a';
cle_decryption = length(cle);
% Conversion du cryptogramme en minuscules
chaine_en_min = char((cryptogramme - 'A') + 'a');
for i = 1 : length(cryptogramme)
% Calcul de l'indice dans la cle pour le decalage
ishift = mod(i, cle_decryption);
if ishift == 0
ishift = cle_decryption;
end
% Decryptage du caractere actuel
sortie_decryptage_vigenere(i) = Shift_Encryption(chaine_en_min(i), -tab(ishift));
end
% Conversion du resultat en minuscules
sortie_decryptage_vigenere = char((sortie_decryptage_vigenere - 'A') + 'a');
end
```



2.3.3 Chiffrement affine

Définition 2.3.

Le chiffrement affine est une méthode de cryptage qui utilise une fonction mathématique linéaire pour encoder un message. Cette fonction est définie par une paire de clés : une clé de chiffrement et une clé de déchiffrement [20].

La clé de chiffrement est une paire de nombres (a,b) où a est un nombre entier positif et b est un nombre

entier. Il faut choisir les valeurs de a et b, qui doivent être des nombres entiers premiers entre eux. Ensuite, chaque lettre du message clair est remplacée par une autre lettre selon la formule suivante :

$$e_k(x) = y = (a \times x + b)\ mod\ 26, \text{ avec } k = (a, b),$$

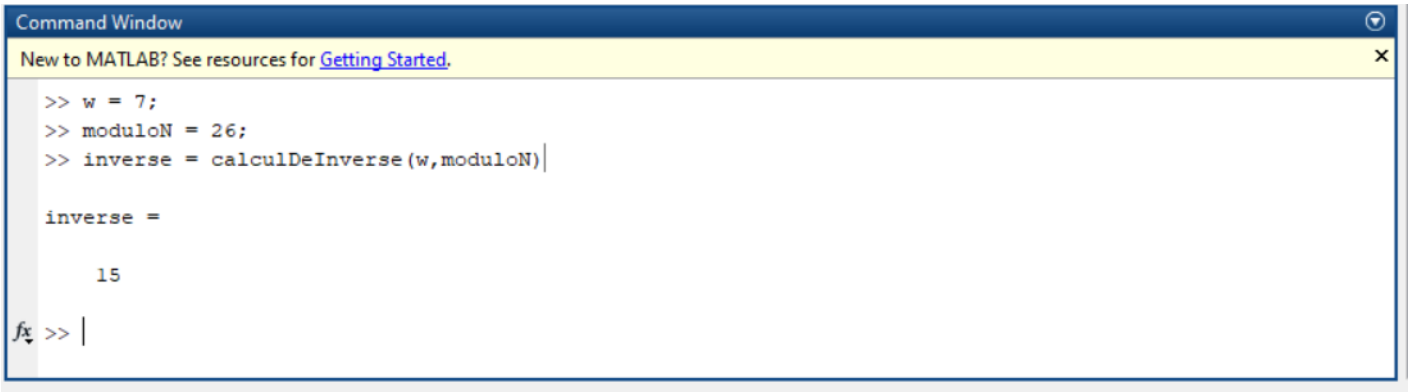
où x est la position de la lettre dans l'alphabet (A=0, B=1, etc.), y est la position de la lettre chiffrée.
La clé de déchiffrement est une paire de nombres (a^{-1}, b) où a^{-1} est l'inverse multiplicatif de a modulo 26, c'est-à-dire un nombre tel que $a \times a^{-1} = 1\ mod\ 26$. Cette clé est utilisée pour transformer chaque lettre du message chiffré en une lettre en clair en utilisant la formule suivante :

$$d_k(y) = x = (a^{-1} \times (y - b))\ mod\ 26.$$

Remarque 2.2. Pour trouver l'inverse multiplicatif de a modulo 26, nous cherchons à trouver le PGCD de a et 26, ainsi que les coefficients de Bézout correspondants. Si le PGCD est différent de 1, alors il n'y a pas d'inverse multiplicatif de a modulo 26. Si le PGCD est égal à 1, alors le coefficient de Bézout correspondant à a est l'inverse multiplicatif de a modulo 26. Si le coefficient de Bézout est négatif, ajoutez 26 pour obtenir l'inverse multiplicatif positif.
Par exemple, pour trouver l'inverse multiplicatif de 7 modulo 26 :
Utilisez l'algorithme d'Euclide pour trouver le PGCD de 7 et 26, ainsi que les coefficients de Bézout correspondants :
$26 = 3 \times 7 + 5$,
$7 = 1 \times 5 + 2$,
$5 = 2 \times 2 + 1$,
$2 = 1 \times 2 + 0$. Comme le PGCD de 7 et 26 est 1, nous pouvons procéder à l'étape suivante.
En remontant les équations précédentes, nous avons :
$1 = 5 - 2 \times 2$,
$1 = 5 - 2 \times (7 - 1 \times 5)$,
$1 = 3 \times 5 - 2 \times 7$,
$1 = 3 \times (26 - 3 \times 7) - 2 \times 7$,
$1 = 3 \times 26 - 11 \times 7$.
Le coefficient de Bézout correspondant à 7 est -11, donc l'inverse multiplicatif de 7 modulo 26 est -11 (négatif). Comme le coefficient de Bézout est négatif, on ajoute 26 pour obtenir l'inverse multiplicatif positif qui vaut 15.

```
function inverse = calculDeInverse(w,moduloN)
 % La fonction gcd retourne le pgcd de w et moduloN (d), ainsi que les
 % coefficients de Bezout (x et y) qui satisfont l'equation de
 Bezout (ax + by = gcd(a,b)).
 [d, x, y] = gcd(w,moduloN);
 % On calcule le reste de la division de x par moduloN pour s'assurer que le
 % resultat est compris dans l'intervalle [0, moduloN-1].
```

```
inverse = mod(x,moduloN);
end
```



Exemple 2.3. Pour chiffrer le message "petit à petit l'oiseau fait son nez" avec le chiffrement affine, nous avons besoin d'une clé de chiffrement. On choisie une clé de chiffrement au hasard, par exemple $k = (a, b) = (7, 13)$.
Tout d'abord, nous devons convertir le message en clair en nombres correspondant à leur position dans l'alphabet. Nous ignorons les espaces et les apostrophes. Voici le message en clair converti :

$$X = (15, 4, 19, 8, 19, 0, 15, 4, 19, 8, 19, 11, 14, 8, 18, 4, 0, 20, 5, 0, 8, 19, 18, 14, 13, 13, 4, 25),$$

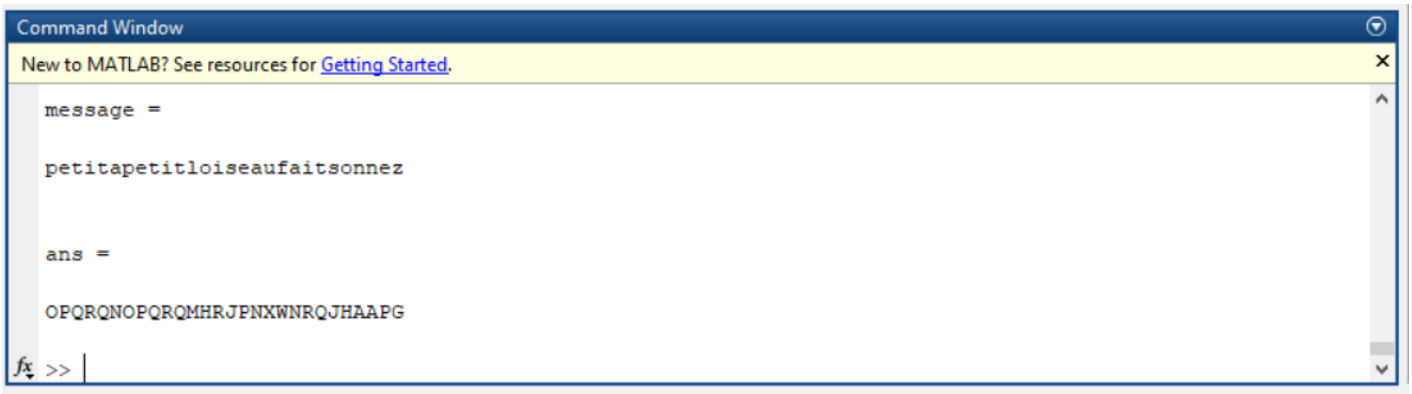
Maintenant, nous pouvons utiliser la clé pour chiffrer chaque nombre en utilisant la formule de chiffrement vu précédemment, et on obtient les nombres chiffrés correspondants :

$$Y = (14, 15, 16, 17, 16, 13, 14, 15, 16, 17, 16, 12, 7, 17, 9, 15, 13, 23, 22, 13, 17, 9, 7, 0, 0, 15, 6).$$

Nous avons maintenant le message chiffré. Nous pouvons le convertir en lettres en utilisant la correspondance des positions des lettres dans l'alphabet : C = "OPQRQNOPQRQMHRJPNXWNRQJHAAPG".

```
function code = codage_affine(a, b, message)
message = 'petitapetitloiseaufaitsonnez'
a = 7; b = 13;
alphabet = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'; % mise en memoire de l'alphabet
code = '';
for i = 1 : length(message)
caractere = upper(message(i)); % convertir en majuscules
if ismember(caractere, alphabet) % si le caractere est dans l'alphabet
x = double(caractere) - 65; % calculer la position du caractere dans l'alphabet
y = mod(a*x + b, 26); % calcul de y
code = [code, char(y + 65)]; % ajouter le caractere correspondant e y au code
```

```
else
code = [code, '␣']; % ajouter un espace au code si le caractere n'est pas dans l'alphabet
end
end
end
```



Exemple 2.4. Supposons que le message chiffré soit "NMDPFRP". Pour déchiffrer ce message, nous devons d'abord convertir chaque lettre en un nombre entre 0 et 25.

chiffré	N	M	D	P	C	R	P
Nombre	13	12	3	15	2	17	15

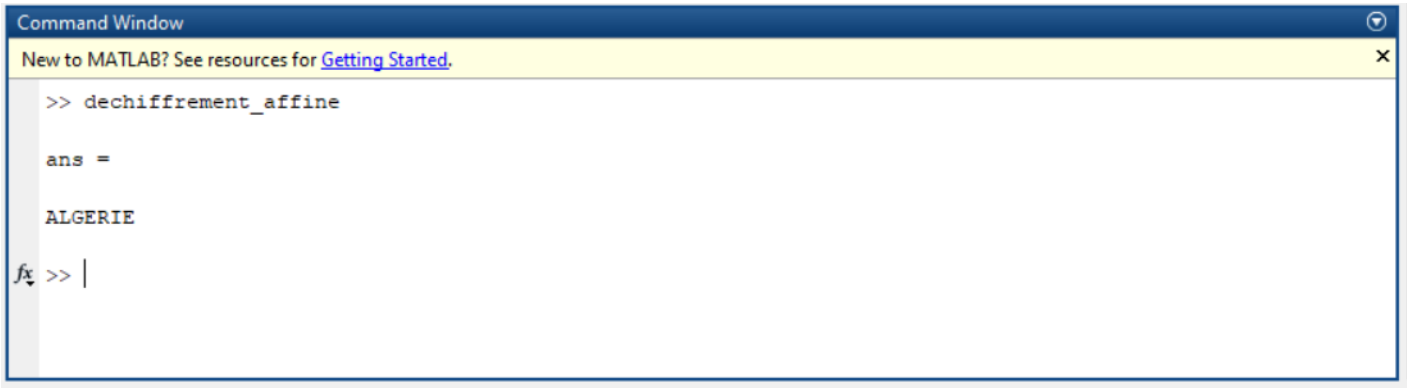
En utilisant la formule de déchiffrement affine et la valeur de a^{-1} que nous avons trouvée précédemment (15), nous pouvons déchiffrer chaque lettre pour obtenir le message original :

Nombre de chaque lettre chiffrée	13	12	3	15	2	17	15
Nombre de chaque lettre déchiffrée	0	11	6	4	17	8	4
Lettre originale	A	L	G	E	R	I	E

Le message original est donc "ALGERIE".

```
function message = dechiffrement_affine(inverse_de_a, b, code)
inverse_de_a = 15;
b = 13;
code ='NMDPCRP';
alphabet = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'; % mise en memoire de l'alphabet
message = '';
for i = 1 : length(code)
caractere = upper(code(i)); % convertir en majuscules
if ismember(caractere, alphabet) % si le caractere est dans l'alphabet
y = double(caractere) - 65; % calculer la position du caractere dans l'alphabet
```

```
x = mod(inverse_de_a*(y - b + 26), 26); % calcul de x
% ajouter le caractere correspondant a x au message dechiffre
message = [message, char(x + 65)];
else
message = [message, ' ']; % ajouter un espace au message dechiffre si le
caractere n'est pas dans l'alphabet
end
end
end
```



2.3.4 Chiffrement de Hill

Définition 2.4.

$GL_m(\mathbb{Z})$ est l'ensemble des matrices inversibles d'ordre m à coefficients dans $\mathbb{Z}$.

Le G dans $GL_m(\mathbb{Z})$ signifie que cet ensemble forme un groupe de matrices, c'est-à-dire qu'il est muni d'une opération binaire souvent la multiplication qui est associative, possède un élément neutre (la matrice identité) et pour chaque matrice de l'ensemble, il existe une matrice inverse qui est également dans l'ensemble.

Le L fait référence au fait que cet ensemble est constitué de matrices linéaires, c'est-à-dire que l'opération de multiplication de matrices satisfait la propriété de linéarité. Cela signifie que pour toutes les matrices A, B et C de $GL_m(\mathbb{Z})$ et tous les scalaires α, β de $\mathbb{Z}$, on a :

$$(\alpha A + \beta B)C = \alpha(AC) + \beta(BC),$$
$$C(\alpha A + \beta B) = \alpha(CA) + \beta(CB).$$

Enfin, le m représente l'ordre des matrices dans l'ensemble. Cela signifie que chaque matrice dans $GL_m(\mathbb{Z})$ est carrée et possède un nombre fixe de lignes et de colonnes, qui est égal à m.

Définition 2.5.

Le chiffrement de Hill est un algorithme de cryptographie symétrique qui utilise des matrices pour chiffrer des messages. Il repose sur l'utilisation d'une matrice clé défini dans l'ensemble $GL_m(\mathbb{Z})$ des

matrices inversibles d'ordre m à coefficients dans $\mathbb{Z}$. Autrement dit, l'espace des clés est constitué de toutes les matrices carrées d'ordre m dont les coefficients sont des entiers $modulo\ 26$, et qui sont inversibles.
Soit $k \in \mathcal{K}$ une clé de chiffrement. Pour chiffrer un bloc de m lettres $(x_1, \ldots, x_m)^T$, on calcule le produit matriciel entre la matrice de chiffrement k et le vecteur colonne $(x_1, \ldots, x_m)^T$, c'est-à-dire :

$$e_k(x) = k \times x, \text{ avec } x = (x_1, \ldots, x_m)^T$$

Le résultat est un nouveau vecteur colonne $(y_1, \cdots, y_m)$ qui correspond au texte chiffré. Pour déchiffrer le texte, il suffit de multiplier le vecteur colonne chiffré $(y_1, \cdots, y_m)$ par l'inverse de la matrice de chiffrement $k\ modulo\ 26$, c'est-à-dire :

$$d_k(y) = k^{-1} \times y, \text{ avec } (y_1, \ldots, y_m)^T.$$

Le résultat est un nouveau vecteur colonne $(x_1, \ldots, x_m)^T$ qui correspond au texte clair.

Il est important de choisir une matrice de chiffrement qui ait une inverse dans $\mathbb{Z}_{26}$, afin de pouvoir déchiffrer le message de manière cohérente [20].

Exemple 2.5. Prenons l'exemple d'un chiffrement de Hill avec une clé de dimension $m = 2$. Nous choisissons une clé de chiffrement $k = \begin{pmatrix} 9 & 4 \\ 5 & 7 \end{pmatrix}$, ce qui signifie que pour chiffrer un bloc de deux lettres (x_1, x_2), nous allons multiplier la clé k par le vecteur colonne correspondant :
Pour chiffrer le bloc "AL", nous allons tout d'abord transformer les lettres en nombres correspondant à leur position dans l'alphabet : $A = 0$, $L = 11$. Le vecteur colonne correspondant est donc $(0, 11)$.
Nous multiplions maintenant la clé k par le vecteur colonne pour obtenir le vecteur chiffré :

$$e_k(0,11)^T = \begin{pmatrix} 9 & 4 \\ 5 & 7 \end{pmatrix} \begin{pmatrix} 0 \\ 11 \end{pmatrix} mod\ 26 = \begin{pmatrix} 44 \\ 77 \end{pmatrix} mod\ 26 = \begin{pmatrix} 18 \\ 25 \end{pmatrix}.$$

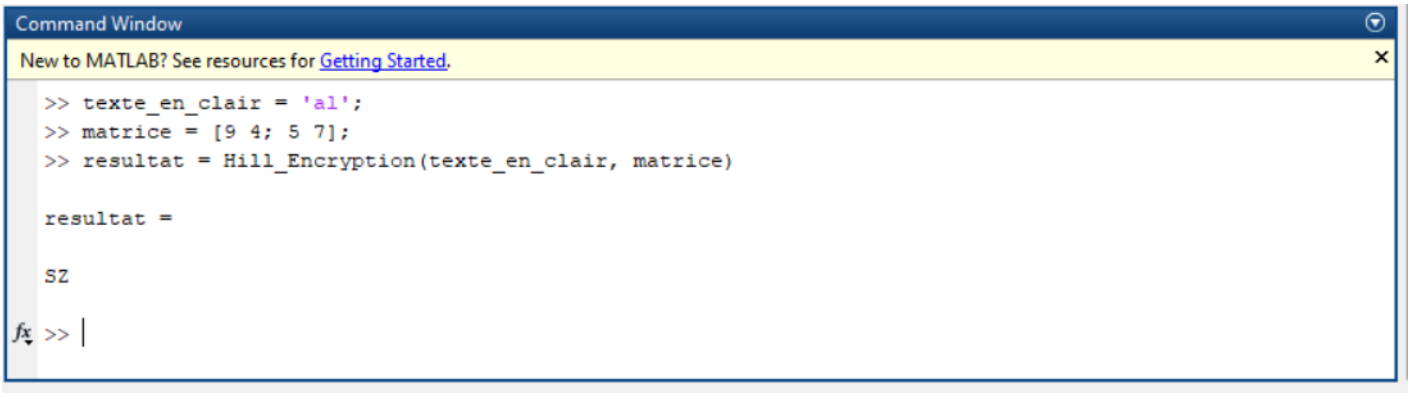
Le vecteur chiffré est donc $(18, 25)$, qui correspond aux lettres "S" et "Z" dans l'alphabet.
Pour déchiffrer ce bloc, nous devons multiplier le vecteur chiffré par l'inverse de la clé k modulo 26. Dans ce cas, l'inverse de la clé est $k^{-1} = (9 \times 7 - 5 \times 4)^{-1} \begin{pmatrix} 7 & -4 \\ -5 & 9 \end{pmatrix}$. On calcul donc l'inverse multiplicatif de 17^{-1}, qui est égal à 23, nous multiplions donc :

$$d_k(18,25)^T = \begin{pmatrix} 5 & 12 \\ 15 & 25 \end{pmatrix} \begin{pmatrix} 18 \\ 25 \end{pmatrix} mod\ 26 = \begin{pmatrix} 390 \\ 895 \end{pmatrix} mod\ 26 = \begin{pmatrix} 0 \\ 11 \end{pmatrix}.$$

Le vecteur déchiffré est donc $(0, 11)$, qui correspond aux lettres "A" et "L" dans l'alphabet.

```
function resultat = Hill_Encryption(texte_en_clair, matrice)
% Obtention de la taille de la matrice de chiffrement
[n n] = size(matrice);
% Conversion du texte en clair en une sequence de nombres
correspondant a la position dans l'alphabet
```

```
tab = texte_en_clair - 'a';
taille_tab = length(tab);  % Calcul de la taille du tableau
 % Calcul du reste de la division de la taille du tableau
 par la taille de la matrice.
concatener_long_tab = mod(taille_tab, n);
if concatener_long_tab > 0
% Calcul du nombre d'elements a ajouter pour completer
le tableau a la taille requise
ajout_taille = n - concatener_long_tab;
% Ajout de valeurs arbitraires pour completer le tableau
tab(taille_tab + 1 : taille_tab + ajout_taille) = 13;
end
% Calcul du nombre de colonnes de la matrice a partir de la taille du tableau
nombre_de_colonne = length(tab) / n;
% Remodelage du tableau en une matrice de dimensions [n, nombre_de_colonne]
tab_remodele = reshape(tab, [n, nombre_de_colonne]);
 % Calcul du cryptogramme en effectuant la multiplication matricielle
 et en appliquant l'operation modulo 26
 cryptogramme = mod(matrice * tab_remodele, 26);
sortie_tableau = cryptogramme(:)';  % Conversion de la matrice en un tableau 1D
% Conversion des valeurs numeriques en caracteres correspondant a
la position dans l'alphabet majuscule
resultat = char(sortie_tableau + 'A');
end
```



2.3.5 Chiffrement de Hill affine

Définition 2.6.

Le chiffrement de Hill Affine est basé sur le chiffrement de Hill, cependant il ajoute une étape sup-

plémentaire en appliquant une fonction affine à chaque bloc de lettres avant de les permuter avec la matrice de chiffrement.

Voici comment procéder pour chiffrer un message m en utilisant le chiffrement de Hill affine :

- Choisir une matrice de chiffrement k_1 de taille $n \times n$ appartenant au groupe linéaire général $GL(n, \mathbb{Z}_d)$, où d est un entier positif qui détermine la taille de l'alphabet,
- Choisir une constante de chiffrement k_2 appartenant à $\mathbb{Z}_d^n$,
- Diviser le message clair m en blocs de longueur n, en ajoutant éventuellement des caractères de remplissage si la longueur de m n'est pas un multiple de n,
- Pour chaque bloc de longueur n, appliquer la fonction de chiffrement $e_k(m) = k_1 m + k_2$ pour obtenir le texte chiffré c,
- Concaténer tous les blocs chiffrés pour former le texte chiffré final.

Pour déchiffré un message suivez les étapes suivantes :

- Calculer la matrice inverse k_1^{-1} de la matrice de chiffrement k_1 $modulo$ d. Si k_1 n'est pas inversible $modulo$ d, alors il n'est pas possible de déchiffrer le texte chiffré,
- Diviser le texte chiffré c en blocs de longueur n,
- Pour chaque bloc de longueur n, appliquer la fonction de déchiffrement $d_k(c) = k_1^{-1}(c - k_2)$ pour retrouver le message clair m [20].

Exemple 2.6. Supposons que nous voulions chiffrer le message "UNIVERSITE" en utilisant une matrice de chiffrement $k_1 = \begin{pmatrix} 5 & 8 \\ 3 & 17 \end{pmatrix}$, et $k_2 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$.

1– Tout d'abord, nous devons convertir chaque lettre en son nombre correspondant dans l'alphabet :

$U = 20$, $N = 13$, $I = 8$, $V = 21$, $E = 4$, $R = 17$, $S = 18$, $I = 8$, $T = 19$, $E = 4$,

2– Nous divisons ensuite le message en blocs de deux lettres : $\begin{pmatrix} 20 \, , 13 \end{pmatrix}^T$; $\begin{pmatrix} 8 \, , 21 \end{pmatrix}^T$; $\begin{pmatrix} 4 \, , 17 \end{pmatrix}^T$; $\begin{pmatrix} 18 \, , 8 \end{pmatrix}^T$; $\begin{pmatrix} 19 \, , 4 \end{pmatrix}^T$;

3– Nous appliquons ensuite la fonction de chiffrement affine pour chaque bloc en utilisant k_1 et k_2 :

$$e_k = \begin{pmatrix} 5 & 8 \\ 3 & 17 \end{pmatrix} . \begin{pmatrix} 20 \\ 13 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 205 \\ 282 \end{pmatrix} (mod\ 26) = \begin{pmatrix} 23 \\ 22 \end{pmatrix},$$

$$e_k = \begin{pmatrix} 5 & 8 \\ 3 & 17 \end{pmatrix} . \begin{pmatrix} 8 \\ 21 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 209 \\ 382 \end{pmatrix} (mod\ 26) = \begin{pmatrix} 1 \\ 18 \end{pmatrix},$$

$$e_k = \begin{pmatrix} 5 & 8 \\ 3 & 17 \end{pmatrix} . \begin{pmatrix} 4 \\ 17 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 157 \\ 302 \end{pmatrix} (mod\ 26) = \begin{pmatrix} 1 \\ 16 \end{pmatrix},$$

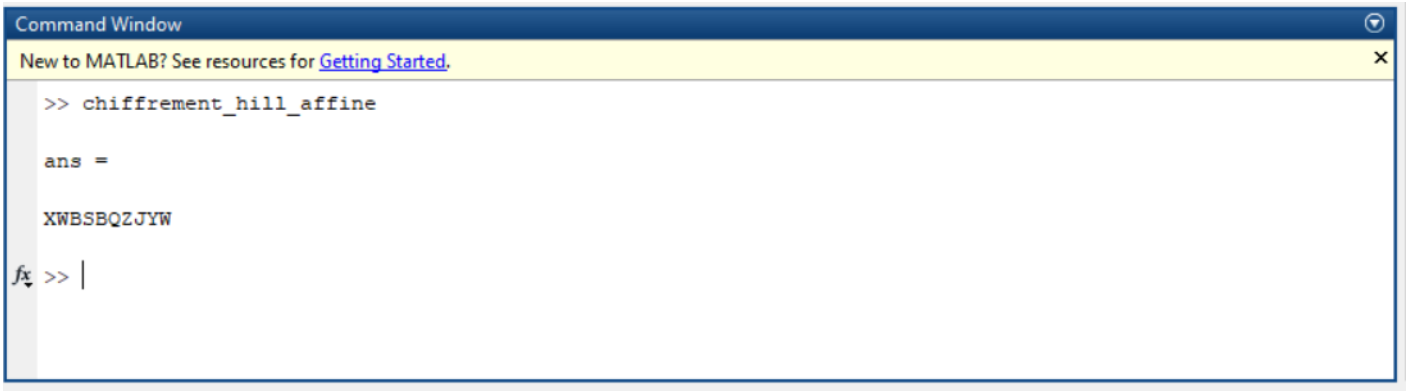
$$e_k = \begin{pmatrix} 5 & 8 \\ 3 & 17 \end{pmatrix} . \begin{pmatrix} 18 \\ 8 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 155 \\ 191 \end{pmatrix} (mod\ 26) = \begin{pmatrix} 25 \\ 9 \end{pmatrix},$$

$$e_k = \begin{pmatrix} 5 & 8 \\ 3 & 17 \end{pmatrix} . \begin{pmatrix} 19 \\ 4 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 128 \\ 126 \end{pmatrix} (mod\ 26) = \begin{pmatrix} 24 \\ 22 \end{pmatrix}.$$

4– Nous convertissons ensuite chaque nombre chiffré en sa lettre correspondante dans l'alphabet,

5– Enfin, nous concaténons les lettres chiffrées pour former le message chiffré : "XWBSBQZJYW".

```
function code = chiffrement_hill_affine(k1, k2, message, n)
message = 'universite'; n=2; k1 = [5 8;3 17]; k2 = [1 1];
alphabet = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'; % mise en memoire de l'alphabet
code = '';
% nombre de blocs necessaires pour couvrir le message
num_blocks = ceil(length(message)/n);
% message complete avec des espaces si necessaire
padded_message = [message, repmat(' ', 1, num_blocks*n - length(message))];
for i = 1 : num_blocks
block = upper(padded_message((i-1)*n+1 : i*n)); % extraire un bloc du message
% convertir les caracteres du bloc en position dans l'alphabet
x = double(block) - 65;
x_vec = mod(k1 * x' + k2', 26);% calcul du vecteur chiffre
vecteur_ligne = x_vec';
% ajouter les caracteres correspondants au vecteur chiffre au code
code = [code, char(vecteur_ligne + 65)];
end
end
```



Pour le déchiffrement on applique la fonction suivante : $d_k(c) = k_1^{-1}(c - k_2)$, et on refait les calculs de la même manière, on obtient :

1– $k_1^{-1} = (5 \times 17 - 3 \times 8)^{-1} \begin{pmatrix} 17 & -8 \\ -3 & 5 \end{pmatrix} = 9^{-1} \begin{pmatrix} 17 & -8 \\ -3 & 5 \end{pmatrix} = 3 \begin{pmatrix} 17 & -8 \\ -3 & 5 \end{pmatrix} = \begin{pmatrix} 25 & 2 \\ 17 & 15 \end{pmatrix}$.

2– Nous pouvons déchiffrer le message caractère par caractère en effectuant les calculs suivants :

$$d_k = \begin{pmatrix} 25 & 2 \\ 17 & 15 \end{pmatrix} . \begin{pmatrix} 23 & -1 \\ 22 & -1 \end{pmatrix} = \begin{pmatrix} 592 \\ 689 \end{pmatrix} (mod\ 26) = \begin{pmatrix} 20 \\ 13 \end{pmatrix},$$

$$d_k = \begin{pmatrix} 25 & 2 \\ 17 & 15 \end{pmatrix}. \begin{pmatrix} 1 & -1 \\ 18 & -1 \end{pmatrix} = \begin{pmatrix} 34 \\ 255 \end{pmatrix} (mod\ 26) = \begin{pmatrix} 8 \\ 21 \end{pmatrix},$$
$$d_k = \begin{pmatrix} 25 & 2 \\ 17 & 15 \end{pmatrix}. \begin{pmatrix} 1 & -1 \\ 16 & -1 \end{pmatrix} = \begin{pmatrix} 30 \\ 255 \end{pmatrix} (mod\ 26) = \begin{pmatrix} 4 \\ 17 \end{pmatrix},$$
$$d_k = \begin{pmatrix} 25 & 2 \\ 17 & 15 \end{pmatrix}. \begin{pmatrix} 25 & -1 \\ 9 & -1 \end{pmatrix} = \begin{pmatrix} 616 \\ 528 \end{pmatrix} (mod\ 26) = \begin{pmatrix} 18 \\ 8 \end{pmatrix},$$
$$d_k = \begin{pmatrix} 25 & 2 \\ 17 & 15 \end{pmatrix}. \begin{pmatrix} 24 & -1 \\ 22 & -1 \end{pmatrix} = \begin{pmatrix} 617 \\ 706 \end{pmatrix} (mod\ 26) = \begin{pmatrix} 19 \\ 4 \end{pmatrix}.$$

On convertit chaque nombre en sa lettre correspondante et on obtient le message originale m = UNIVERSITE.

Conclusion

Le deuxième chapitre nous a introduits aux bases essentielles de la cryptographie. Nous avons exploré le jargon spécifique utilisé dans ce domaine et avons jeté un regard sur son histoire fascinante. De plus, nous avons plongé dans un peu de formalisme mathématique, en examinant en détail des techniques de chiffrement classiques telles que le chiffrement par décalage, le chiffrement de Vigenère, le chiffrement affine, et le chiffrement de Hill. Ces concepts et méthodes serviront de fondement solide pour notre exploration ultérieure de la cryptographie moderne et de la cryptanalyse. Dans les chapitres à venir, nous approfondirons notre compréhension des systèmes de chiffrement actuels et des méthodes de cryptanalyse, tout en explorant leur application concrète dans le monde numérique en constante évolution.

CHAPITRE 3

CRYPTOSYSTÈMES MODERNES

Introduction

L'arrivée de l'ordinateur a révolutionné la cryptographie et la cryptanalyse en permettant la manipulation de séquences binaires c'est-à-dire des suites de bits, pour le chiffrement et le déchiffrement de messages. Contrairement à la cryptographie classique, les règles de chiffrement et de déchiffrement sont connues de tous, la sécurité de ces méthodes reposent uniquement sur le secret de la clé qui est un paramètre facile à changer. La clé de chiffrement peut être secrète ou non, en fonction de la méthode de chiffrement utilisée : une méthode symétrique utilise une clé secrète tandis qu'une méthode asymétrique utilise une clé publique et une clé privée. Ces deux types de chiffrement vont être étudié en profondeur ci-dessus.

3.1 Chiffrement symétrique

Le chiffrement symétrique est une technique de cryptographie qui utilise une clé partagée entre deux interlocuteurs pour chiffrer et déchiffrer les données. Le même algorithme de chiffrement est utilisé pour le chiffrement et le déchiffrement, mais la clé est nécessaire pour accéder aux données chiffrées. Le chiffrement symétrique est également appelé chiffrement conventionnel ou chiffrement à clé privée.

Le principal avantage du chiffrement symétrique est sa rapidité et son efficacité, il offre un niveau élevé de sécurité si la clé est suffisamment longue et complexe. Toutefois, le partage de la clé secrète peut être un défi, car il doit être effectué de manière sécurisée, si la clé secrète est compromise, toutes les données chiffrées avec cette clé peuvent être déchiffrées. Enfin, le chiffrement symétrique nécessite la gestion de nombreuses clés différentes pour chaque paire de correspondants (nombre de clés à gérer $n(\frac{n-1}{2})$).

On distingue deux catégories de chiffrement symétrique : le chiffrement par flots et le chiffrement par blocs [12].

3.1.1 Chiffrement par flots

Le chiffrement par flots (également appelée chiffrement par flux) consiste à chiffrer le message clair bit à bit (aucun découpage n'est effectué) à l'aide d'une clé secrète et d'un générateur de clé qui produit une séquence de bits pseudo-aléatoires. Les bits de la séquence pseudo-aléatoire sont combinés avec les bits de données clairs à l'aide d'une opération XOR (ou exclusif) pour produire les bits de données chiffrés. Le chiffre de Vernam est un exemple de chiffrement par flots qui utilise un masque jetable ou "One Time Pad" où la clé n'est utilisée qu'une seule fois pour chiffrer le message. Il est considéré comme incassable théoriquement car il a été prouvé qu'il est impossible de retrouver le message chiffré sans la clé. Cependant, le chiffrement par flots pose des défis pratiques tels que la nécessité d'un canal sécurisé pour la distribution des clés, la taille encombrante des clés qui doit être de la même taille que le message, et l'imprévisibilité des générateurs de bits de clé utilisés. Un avantage du chiffrement par flots est qu'il est insensible aux erreurs de transmission car un bit erroné ne compromet pas les bits suivants [12].

3.1.2 Technique du masque jetable

Le chiffrement par la méthode du masque jetable repose sur l'utilisation de deux éléments : le message en clair à chiffrer m et une clé k, qui est une suite de caractères aléatoires de même longueur que le message. Ces deux éléments sont combinés à l'aide de l'opérateur XOR pour obtenir le message chiffré c (c'est à dire : $c = m\ XOR\ k$). Il est essentiel que les caractères de la clé soient choisis de manière totalement aléatoire pour garantir la sécurité du chiffrement. De plus, chaque clé doit être utilisée une seule fois pour assurer la sécurité du système et éviter toute tentative de décryptage. Le processus de déchiffrement est similaire à celui du chiffrement, en utilisant l'opérateur XOR pour combiner le message chiffré et la clé (c'est à dire : $m = c\ XOR\ k$) [12].

Remarque 3.1. Pour avoir le message m sous forme de suite de bits nous utilisons la table de code ASSCII.

Exemple 3.1. Supposons que nous voulons chiffrer le message "HELLO" en utilisant la clé secrète "WORLD". La clé secrète doit avoir la même longueur que le message, donc nous devons répéter la clé pour qu'elle ait la même longueur. Nous combinons chaque caractère du message clair avec le caractère correspondant de la clé répétée en utilisant l'opération XOR :

H (01001000) XOR W (01010111) = 00011111 (1F en hexadécimal)
E (01000101) XOR O (01001111) = 00001010 (0A en hexadécimal)
L (01001100) XOR R (01010010) = 00011100 (1C en hexadécimal)
L (01001100) XOR L (01001100) = 00000000 (00 en hexadécimal)

O (01001111) XOR D (01000100) = 00001011 (0B en hexadécimal)

Les données chiffrées sont donc : 1F0A1C000B.

3.1.3 Chiffrement par blocs

Dans le chiffrement par blocs, le message à chiffrer est converti en une séquence de bits et ensuite divisé en blocs de taille fixe. Ensuite, chaque bloc est transformé en effectuant une opération XOR avec tout ou une partie de la clé. En utilisant la théorie de l'information de Shannon, cette étape de chiffrement permet d'introduire de la confusion (une lettre est transformée en une autre) et de la diffusion (les lettres se retrouvent mélangées).
Le mode d'opération est un élément important du chiffrement par blocs car il définit la façon dont les blocs de données sont chiffrés. Les modes d'opération les plus couramment utilisés sont :

- **ECB (Electronic Codebook)** : Chaque bloc est chiffré indépendamment des autres blocs,
- **CBC (Cipher Block Chaining)** : Chaque bloc est chiffré en utilisant le bloc précédent chiffré,
- **CFB (Cipher Feedback)** : Les données chiffrées sont réinjectées dans l'algorithme de chiffrement pour chiffrer les blocs suivants,
- **OFB (Output Feedback)** : L'algorithme de chiffrement est utilisé pour générer une séquence de bits pseudo-aléatoires qui sont ensuite combinés avec les données d'entrée pour chiffrer les blocs.

Le schéma de chiffrement par bloc repose généralement sur une méthode itérative qui consiste en plusieurs tours ou rondes. À l'exception de la première et de la dernière ronde, chaque itération est généralement identique. La clé de chiffrement de chaque tour est créée à partir de la clé secrète k à l'aide d'un algorithme de génération de clé, également appelé algorithme de cadencement de clé.

Les schémas de Feistel sont l'un des types de chiffrement par blocs les plus connus. Ils ont été développés en 1973 et ont la particularité d'être réversibles même si la fonction utilisée n'est pas inversible. En raison de cette caractéristique, ils ont été largement acceptés par la communauté des experts en cryptographie et ont été normalisés en 1977 comme standard de chiffrement symétrique dans l'algorithme Data Encryption Standard (DES) [10].

3.1.4 Schéma de Feistel

Le schéma de construction de Feistel est une méthode de chiffrement par blocs qui divise un bloc de message clair en deux blocs de taille identique. À chaque tour, l'un des deux blocs est combiné avec une version transformée de l'autre bloc. La transformation d'un bloc par le schéma de Feistel peut être détaillée de la manière suivante :

► Supposons que le bloc de message clair x soit de taille n paire et soit découpé en deux blocs de longueur $n/2$, notés L_0 et R_0. Ainsi, on a $x = L_0 \| R_0$, où $\|$ représente la concaténation entre L_0 et R_0,

- À chaque tour $1 \leq i \leq r$, le bloc d'entrée (L_{i-1}, R_{i-1}) de taille n est transformé en un bloc (L_i, R_i) de taille n. Cela se fait grâce à un algorithme de cadencement de clé, qui génère r clés de tours, notées r_{ki} , de taille $n/2$ à partir de la clé cryptographique secrète originale,
- On utilise ensuite une fonction de tour F, définie pour des blocs de taille $n/2$ et prenant en entrée les blocs (L_{i-1}, R_{i-1}) et la clé de tour r_{ki}. La transformation du bloc (L_{i-1}, R_{i-1}) est illustrée dans la figure ci-dessous, et se fait par les formules suivantes :

$$L_i = R_{i-1},$$
$$R_i = L_{i-1} \oplus F(R_{i-1}, r_{ki}),$$

  où $\oplus$ représente l'opération XOR (ou exclusif).
- À la fin des r tours, les deux blocs L_i et R_i sont fusionnés pour produire le bloc de sortie chiffré $y = L_r \| R_r$. Cette méthode de chiffrement est simple, efficace et résistante à certaines attaques cryptographiques, telles que la cryptanalyse différentielle.

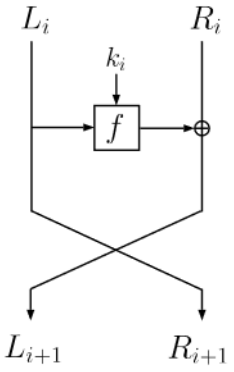


FIGURE 3.1 – Fonction de tour d'un schéma de Feistel.

Le schéma de construction de Feistel est utilisé dans de nombreux algorithmes de chiffrement, notamment dans le célèbre chiffrement symétrique DES (Data Encryption Standard) [11].

Algorithme Data Encryption Standard (DES)

L'algorithme de chiffrement symétrique le plus connu est le Data Encryption Standard (DES), qui a été conçu par IBM (International Business Machines) dans les années 1970 et révisé par la NSA (National Security Agency) en 1976. Depuis, il est devenu la norme de chiffrement symétrique de référence. Initialement appelé "Lucifer", le DES opérait sur des blocs de 128 bits et avait une clé de taille similaire. Cependant, en raison d'une vulnérabilité de chiffrement découverte en 1996 par Ben-Aroya et Biham, la NSA a normalisé la taille de la clé à sa valeur actuelle.
Le chiffrement DES est conçu sur la base des réseaux de Feistel et fonctionne sur des blocs de 64 bits en utilisant une clé de chiffrement de 56 bits. Il utilise une transformation f à 16 tours, chaque tour ayant sa propre sous-clé de 48 bits générée à partir de la clé principale, en fonction du tour. Le chiffrement

est terminé par deux permutations : une permutation initiale et son inverse finale.
Aujourd'hui, en raison de la puissance de calcul disponible, le DES n'est plus considéré comme sécurisé en raison de sa taille de clé jugée trop petite et de sa lenteur d'exécution. De plus, il a été victime de nombreuses attaques réussies. C'est pourquoi une version renforcée, le Triple DES, a été créée pour utiliser deux clés (k_1, k_2) et une fonction de chiffrement dérivée du DES donnée par $E_{k1}(D_{k2}(E_{k2}(m)))$, avec E et D désignent les opérations de chiffrement et déchiffrement. Cependant, l'Advanced Encryption Standard (AES) est maintenant recommandé car il utilise une clé de taille variable de 128 à 256 bits et est considéré comme résistant à toutes les attaques connues. L'AES est également utilisé dans les protocoles de chiffrement des réseaux mobiles [14].

Structure

1− **Clé :** la clé de chiffrement du DES est composée de 64 bits, avec le dernier bit de chaque octet étant utilisé comme un code de parité pour les autres bits du même octet. Pour générer les 16 sous-clés différentes de 56 bits chacune utilisées lors du chiffrement, la clé principale est d'abord réduite à 56 bits en supprimant les bits de parité et en effectuant une permutation appelée PC_1, qui est décrite par le tableau suivant :

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	25	44	36

TABLE 3.1 – Demi bloc gauche

63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

TABLE 3.2 – Demi bloc droit

À chaque itération du chiffrement avec DES, les bits de la clé principale sont décalés de façon circulaire vers la gauche d'un ou deux bits, selon chaque itération de la manière suivante :

Itération	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Nbre de décalage	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

TABLE 3.3 – Décalage relatifs au sous-clés

A titre d'exemple, si la clé principale de 64 bits est :
$k^{64} = 0000000000000000000000000000000000000000000000001010000000010000000$
Après suppression des caractères de parité et permutation à l'aide du tableau PC_1, on obtient :
$k^{56} = 11000000000000000000000000000\ 00100000000000000000000000000$
La clé relative à la première itération subira un seul décalage à gauche de chacune de ses deux parties d'une position :
$k_1^{56} = 10000000000000000000000000001\ 01000000000000000000000000000$
Pour chaque itération i, la clé k_i subit une transformation qui la réduit à 48 bits. Cette transformation est obtenue en supprimant certains bits de la clé et en permutant les autres, selon une permutation appelée PC_2. Cette permutation est définie par le tableau suivant :

14	17	11	24	1	5	3	28
15	6	21	10	23	19	12	4
26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40
51	45	33	48	44	49	39	56
34	53	46	42	50	36	29	32

TABLE 3.4 – Permutation PC2

En appliquant cette permutation, la clé en question sera :

$k_1^{48} = 000010010000000000000000000000100000000000000000$

Pour calculer k_2 par exemple, le processus est repris à partir de k^{56}.

2– **Permutation initiale IP :** La permutation initiale IP permet de déplacer chaque bit du bloc de texte clair d'une position à une autre selon la table de permutation suivante :

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

TABLE 3.5 – Permutation initiale IP

L'opération implique le déplacement de différents bits du texte clair vers de nouvelles positions, créant ainsi une disposition aléatoire. Par exemple, le 58 ème bit peut être déplacé à la première position, le 50 ème à la deuxième position, et le 60 ème à la neuvième position. Cette nouvelle disposition des bits crée de la confusion et empêche toute structure linéaire ou algébrique d'être détectée dans le système [14].

3– **Fonction d'expansion E :** La fonction d'expansion **E** augmente la taille du demi-bloc R de 32 bits à la taille de la clé, qui est de 48 bits. Cette expansion est réalisée en répétant certains bits du demi-bloc, en utilisant le tableau fourni :

32	1	2	3	4	5	4	5
6	7	8	9	8	9	10	11
12	13	12	13	14	15	16	17
16	17	18	19	20	21	20	21
22	23	24	25	24	25	26	27
28	29	28	29	30	31	32	1

TABLE 3.6 – Fonction d'expansion E

Le tableau de la fonction d'expansion E comprend des répétitions de certains bits, par exemple, le 32 ème bit sera répété à la première position et à la 47 ème position. De même, le 25 ème bit sera

répété simultanément à la 36 ème et 38 ème position.

4– **Fonction de substitution S :** La fonction **S**, également appelée les S-boîtes, est une fonction non linéaire qui prend un bloc d'entrée de 48 bits et renvoie un bloc de sortie de 32 bits. Le principe de cette fonction consiste à diviser le bloc d'entrée en 8 sous-blocs de 6 bits chacun. Pour chaque sous-bloc i, les quatre bits 2 à 5 représentent la ligne dans la S-boîtes i, tandis que les deux bits 1 et 6 représentent la colonne. Ces coordonnées sont utilisées pour identifier une valeur de 4 bits dans la S-boîte i, qui remplacera les six bits du sous-bloc i [14].

S_0															
14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	8	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S_1															
15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

S_2															
10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	12	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

S_3															
7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

S_4															
2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

S_5															
12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	8	2	12	9	5	15	10	11	14	1	7	6	0	8	13

S_6															
4	11	2	14	15	0	8	13	3	12	9	7	5	10	3	1
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

S_7															
13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

FIGURE 3.2 – S-boîtes

5– **Fonction de permutation P :** Il s'agit d'une fonction de permutation classique qui agit sur un bloc de données en modifiant la position de ses bits sans altérer sa longueur. Cette fonction est mise en œuvre à l'aide du tableau de transposition suivant :

16	7	20	21	29	12	28	17
1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9
19	13	30	6	22	11	4	25

TABLE 3.7 – Fonction de permutation P

6– **Permutation finale** IP^{-1} **:** La permutation finale effectue une perturbation supplémentaire des bits du bloc pour annuler la permutation initiale à travers une transposition inverse qui est spécifiée dans le tableau suivant :

7– **Algorithme :** les grandes lignes de l'algorithme sont :

- **Diversification de la clé :** le texte est découpé en blocs x de 64 bits. En même temps, la clé de 64 bits est diversifiée en 16 sous-clés distinctes $(K_1, \ldots, K_{16})$, chacune étant constituée de 48 bits issus de la clé initiale K.
- **Permutation initiale :** la deuxième phase de l'algorithme consiste à effectuer une permutation initiale $y = P(x)$ sur chaque bloc de 64 bits du texte à chiffrer. y s'écrit sous la forme $y = L_0R_0$, L_0 étant les 32 bits à gauche de y, et R_0 les 32 bits à droite.

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

TABLE 3.8 – Permutation finale IP^{-1}

- **Itération :** cette étape consiste à appliquer 16 rondes d'une même fonction pour chaque bloc de 64 bits y. À partir des valeurs L_{i-1} et R_{i-1} (pour i allant de 1 à 16), on calcule les valeurs L_i et R_i en utilisant les équations suivantes :

$$L_i = R_{i-1},$$
$$R_i = L_{i-1} \oplus f(R_{i-1}, K_i).$$

- **Permutations finale :** la quatrième et la dernière phase de l'algorithme DES est la permutation finale, qui consiste à appliquer l'inverse de la permutation initiale à la valeur finale $L_{16}R_{16}$, $Z = P^{-1}(L_{16}, R_{16})$.

Le déchiffrement est similaire à l'algorithme de chiffrement, sauf que les sous-clés sont utilisées dans l'ordre inverse pour déchiffrer le texte chiffré.

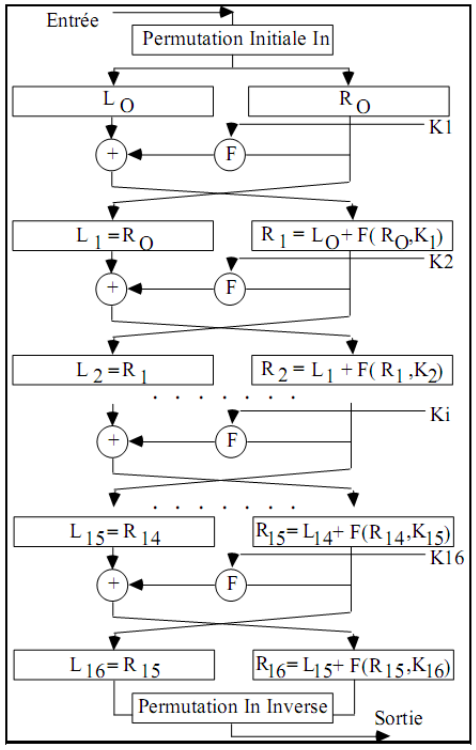


FIGURE 3.3 – Tours de l'algorithme DES

Exemple 3.2. [20] Soit le bloc P de données claires suivant :

$P_1^{64} = 0101111001110100111110001010000001111110110000100001100100110011$

Il sera subdivisé en deux sous-blocs :

$L_1^{32} = 01011110011101001111100010100000$ et $R_1^{32} = 01111110110000100001100100110011$

Le bloc droit R_1 sera étendu à la l'aide de la fonction d'expansion E qui deviendra :

$R_1^{48} = 100101011001010011111100001010101011110110100010$

En utilisant la clé : $k_1^{48} = 000010010000000000000000000000100000000000000000$

La combinaison du bloc de données avec la clé donne :

$R_1^{48} \oplus k_1^{48} = 100111001001010011111100001010001011110110100010$.

La fonction S réduit chaque sous-bloc de 6 bits en un sous-bloc de 4 bits en utilisant les S-boîtes. Par exemple, pour le premier sous-bloc, **1**00111, les deux bits en gras (11) déterminent la ligne et les autres bits déterminent la colonne de la S-boîte S_0, soit S_0(11,0011). La valeur correspondante est 2, donc la représentation binaire est 0011.

Le sous-bloc de 32 bits obtenu à partir de l'application de la fonction S subit une nouvelle permutation en suivant le tableau 1.12, sans affecter sa longueur. Ensuite, il est X-oré avec l'autre demi-bloc R_1. Le résultat est transféré à R_2 tandis que L_2 est équivalent à R_1. Ce processus est répété de manière similaire pour chaque itération suivante.

Remarque 3.2. Les deux fonctions, E et P, contribuent à la diffusion, c'est-à-dire qu'un changement d'un seul bit en entrée entraîne plusieurs changements de bits en sortie. Cette propriété permet de prévenir les attaques statistiques, qui cherchent à exploiter des motifs récurrents dans les données chiffrées pour déduire des informations sur les données claires correspondantes.

Par contre, les S-boîtes permettent d'apporter de la confusion : le but est de rendre complexe les relations entre bits de chiffré et bits de clef, changer un bit de clef, change complètement le chiffré. Les boîtes S ont été conçues pour résister à la cryptanalyse différentielle, une méthode qui nécessite 2^{47} clairs choisis, et elles sont également à grande distance de Hamming des fonctions affines, ce qui les rend résistantes à la cryptanalyse linéaire, qui nécessite 2^{43} clairs connus.

Bien que le DES n'ait pas de réelle vulnérabilité connue, il est considéré comme faible en raison de l'utilisation de clés de 64 bits, dont seulement 56 bits fournissent une sécurité efficace (les 8 bits restants sont utilisés pour des contrôles de parité). Cela signifie que la sécurité du DES peut être brisée en quelques heures avec des ordinateurs modernes. Pour renforcer sa sécurité, le Triple-DES a été adopté en 1998.

Peu de temps après la normalisation du Triple-DES, le National Institute of Standards and Technology (NIST) a lancé le concours AES (Advanced Encryption Standard), appelant la communauté cryptographique à concevoir le prochain standard de chiffrement par blocs. Le gagnant du concours a été Joan Daemen et Vincent Rijmen, dont la proposition Rijndael a été renommée AES lors de sa normalisation en 2001. Contrairement au DES, Rijndael n'utilise pas de réseau de Feistel, mais stocke l'information à chiffrer dans un état interne. Cependant, le principe de réseau de substitutions-permutations sur

plusieurs tours reste le même. Aujourd'hui, l'AES est largement utilisé, intégré dans des processeurs, des cartes bancaires et même certains téléphones. Avec des clés allant de 128 à 256 bits, l'AES devrait être résistant aux attaques pendant de nombreuses années, à moins qu'une faille de conception ne soit découverte ou que des avancées technologiques ne permettent de compromettre sa sécurité [14].

3.1.5 Chiffrement de type substitution-permutation

Une autre grande famille de systèmes de chiffrement itératif que nous allons traiter sont les schémas de type substitution-permutation (SPN : "Substitution Permutation Network").

Définition 3.1. Si la fonction de tour d'un système de chiffrement par bloc itératif peut être décomposée en trois étapes principales, à savoir l'ajout de clé, une substitution non linéaire et une permutation linéaire, on parle alors d'un système de type substitution-permutation.

Pour qu'un système de chiffrement par bloc itératif soit déchiffrable, il est nécessaire que la fonction de tour soit une bijection, c'est-à-dire qu'elle soit à la fois injective (chaque entrée est associée à une seule sortie) et surjective (chaque sortie est atteinte au moins une fois) .

Cette définition très générale nous permet d'inclure un grand nombre de systèmes de chiffrement dans la catégorie des SPN. L'un des exemples les plus connus de système de chiffrement de ce type est le "Rijndael", qui a été standardisé sous le nom de "Advanced Encryption Standard" (AES) par le NIST en 2000.

Dans un système de chiffrement de type substitution-permutation, la phase de confusion correspond à la partie de substitution, tandis que la phase de diffusion correspond à la permutation. Il existe plusieurs techniques pour la diffusion de l'information entre les tours.

Transformation d'un bloc par le schéma SPN supposons qu'un bloc de données de taille n soit divisé en l sous-blocs de taille identique. Le nombre de tours est déterminé par la valeur r. Nous déduisons les clés de tour rk_i $(1 \leqslant i \leqslant r)$ à partir de la clé cryptographique secrète originale k en utilisant un algorithme de génération de clé. On pose ensuite $x_0 = x \oplus k$, puis, pour $(1 \leqslant i \leqslant r)$, on calcule $x_i = F(x_{i-1}, rk_i)$ où F est la fonction de tour.

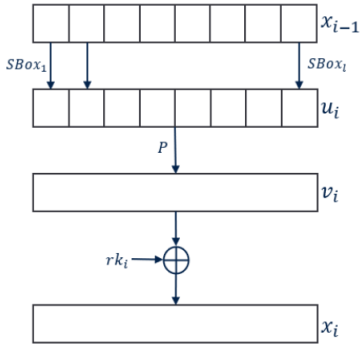


FIGURE 3.4 – Fonction de tour par le schéma SPN [11]

Commentaires : avant l'application de la fonction de tour F, le bloc précédent x_{i-1} de taille n est divisé en l sous-blocs de taille plus petite. La fonction de tour F commence par l'application de l boîtes de substitution S-boîtes sur chacun des sous-blocs. Les S-boîtes sont des boîtes-S qui effectuent des opérations non-linéaires pour assurer la propriété de confusion du chiffrement. Le résultat de cette opération de substitution est un bloc de sortie u_i de taille n.
Une permutation linéaire (notée P) sur les bits du bloc u_i est ensuite appliquée. On note v_i le résultat.
Enfin, la clé de tour rk_i est additionnée au bloc v_i . On obtient $x_i = v_i \oplus rk_i$.
Au bout des r tours, le bloc chiffré est représenté par $X = x_r$.
L'algorithme symétrique le plus connu basé sur le schéma SPN est l'algorithme AES [11].

Algorithme Advanced Encryption Standard (AES)

Comme mentionné précédemment, l'AES utilise une structure SPN pour chiffrer un message clair de 128 bits avec une clé secrète de 128, 192 ou 256 bits. C'est un algorithme de chiffrement symétrique itératif qui utilise un nombre de tours déterminé par la longueur de la clé secrète : 10 tours pour une clé de 128 bits, 12 tours pour une clé de 192 bits, et 14 tours pour une clé de 256 bits.

Structure algébrique de l'AES : l'AES chiffre un bloc de message clair de 128 bits à l'aide d'un bloc de clé cryptographique de 128 bits. Ces deux blocs sont représentés par des tableaux de 16 octets, chacun composé de quatre lignes et quatre colonnes. Ces tableaux sont appelés "états de données" ou "états d'AES".

Structure d'état dans AES : un bloc est représenté comme un tableau de 4 lignes et N_b colonnes, où N_b est égal à la taille du bloc en octets divisée par 32. La clé de chiffrement est également représentée sous forme de tableau, le nombre de colonnes égal à N_k = longeur de la clé$/32$.

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$

$k_{0,0}$	$k_{0,1}$	$k_{0,2}$	$k_{0,3}$
$k_{1,0}$	$k_{1,1}$	$k_{1,2}$	$k_{1,3}$
$k_{2,0}$	$k_{2,1}$	$k_{2,2}$	$k_{2,3}$
$k_{3,0}$	$k_{3,1}$	$k_{3,2}$	$k_{3,3}$

FIGURE 3.5 – Exemple d'état (avec des blocs de 128 bits, $N_b = 4$) et de clef (de longueur 128 bits, $N_k = 4$) [11]

Nombre de tour : Le nombre de tours effectués dans l'algorithme AES dépend à la fois de la taille des blocs et de la longueur de la clé de chiffrement. Le nombre de tours r est déterminé par un tableau spécifique :

Pour déchiffrer le bloc chiffré, les clés de tours sont utilisées dans l'ordre inverse et les calculs effectués dans la fonction de tour sont appliqués de manière inverse.
Fonction de tour : la fonction de tour de l'AES consiste en 4 opérations : AddRoundKey, SubBytes,

N_r	$N_b = 4$ (128 bits)	$N_b = 6$ (192 bits)	$N_b = 8$ (256 bits)
$N_k = 4$ (128 bits)	10	12	14
$N_k = 6$ (192 bits)	12	12	14
$N_k = 8$ (256 bits)	14	14	14

FIGURE 3.6 – Nombre de tour

ShiftRows et MixColumns, qui sont appliquées dans un ordre précis à l'état clair d'AES. Le processus de chiffrement commence par l'opération AddRoundKey, qui ajoute la clé cryptographique à l'état clair. Ensuite, neuf tours consécutifs sont effectués, chacun impliquant une clé de tour et appliquant les quatre opérations mentionnées précédemment. Finalement, un dixième tour est effectué où l'opération MixColumns n'est pas utilisée. Les rôles de ces quatre opérations sont détaillés ci-dessous.

AddRoundKey : consiste à ajouter la clé de tour à l'état de données en effectuant un XOR entre chaque octet de l'état de données et son octet correspondant dans la clé de tour. La taille de la clé de tour est la même que celle de l'état de données, à savoir 128 bits. Il est important de noter que la clé de tour utilisée lors de la première application de AddRoundKey est identique à la clé cryptographique originale.

SubBytes : la propriété de confusion est assurée dans l'algorithme AES grâce à une opération appelée substitution de boîte-S. Cette opération non-linéaire consiste à substituer chaque octet de l'état interne par un autre octet, en utilisant une même boîte-S définie sur $GF(2^8)$. La boîte-S est appliquée individuellement sur chacun des 16 octets de l'état interne.

Avant l'exécution de l'algorithme AES, la boîte de substitution S est souvent pré-calculée et stockée dans une table de recherche appelée LUT, qui permet de stocker toutes les substitutions possibles d'un octet, représentées sous forme d'un tableau à double entrée dans la notation hexadécimale, comme illustré dans la figure ci-dessous :

Autrement dit, SubBytes s'applique individuellement à tous les octets $a_{i,j}$ d'un état en deux étapes distinctes :

- Tout d'abord, on considère l'octet $a_{i,j}$ comme un polynôme dans le corps fini $GF(2^8)$ et on calcule son inverse $a_{i,j}^{-1}$ dans ce corps,
- Ensuite, pour obtenir l'image correspondante on applique la fonction $y = f(x)$ à l'inverse $a_{i,j}^{-1}$ de la manière suivante :

	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f
00	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
10	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
20	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
30	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
40	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
50	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
60	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
70	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
80	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
90	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a0	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b0	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c0	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d0	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e0	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f0	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

FIGURE 3.7 – Table de correspondance de la boîte-S de l'AES pour un octet en notation hexadécimale, en commençant par la ligne, puis la colonne. Par exemple, l'octet 9a est substitué par l'octet b8 [11]

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}$$

ShiftRows : l'opération ShiftRow dans l'algorithme AES consiste à décaler les octets des lignes de l'état courant. Cette opération agit sur les quatre lignes de l'état de la manière suivante :

- La ligne 0 n'est pas décalée,
- La ligne 1 est décalée de C_1 octets,
- La ligne 2 est décalée de C_2 octets,
- La ligne 3 est décalée de C_3 octets.

Les valeurs de C_1, C_2 et C_3 dépendent de la taille du bloc et sont données dans le tableau suivant :

N_b	C_1	C_2	C_3
4	1	2	3
6	1	2	3
8	1	3	4

MixColumns : consiste à appliquer une multiplication matricielle à chaque colonne de l'état courant en utilisant la matrice suivante :

$$\begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \otimes \begin{pmatrix} a_{0,x} \\ a_{1,x} \\ a_{2,x} \\ a_{3,x} \end{pmatrix} = \begin{pmatrix} b_{0,x} \\ b_{1,x} \\ b_{2,x} \\ b_{3,x} \end{pmatrix}$$

AddRoundKey : consiste en un OU exclusif de l'état courant et de la clef du tour [11].

3.2 Chiffrement asymétrique

La cryptographie asymétrique est née de la nécessité de trouver une solution aux problèmes posés par la cryptographie symétrique, notamment celui de la distribution des clés. En 1970 Whitfield Diffie et Martin Hellman, ont proposé un système dans lequel chaque utilisateur dispose d'une paire de clés : une clé publique et une clé privée. La clé publique peut être diffusée largement, tandis que la clé privée doit être gardée secrète. Avec ce système, chaque utilisateur peut chiffrer un message à l'aide de la clé publique de son destinataire, mais seul ce dernier peut le déchiffrer à l'aide de sa clé privée.

3.2.1 Algorithme RSA

En 1977, Ron Rivest, Adi Shamir et Len Adleman ont inventé l'un des algorithmes à clé publique les plus célèbres, après que Diffie et Hellman aient publié l'idée de la cryptographie à clé publique. Cet algorithme a été nommé RSA d'après les initiales de ses inventeurs [**?**]. L'algorithme fonctionne de la manière suivante :

- **Création des clés :** l'étape de génération des clés est de la responsabilité de l'entité A qui souhaite chiffrer ses messages à l'aide de sa clé privée. Cette étape ne se répète pas à chaque chiffrement, car les clés peuvent être utilisées de manière répétée. Le renouvellement des clés n'est nécessaire que si la clé privée est compromise ou pour des raisons de sécurité, après une certaine période de temps, qui peut s'étendre sur plusieurs années. L'entité A doit :
  - $\hookrightarrow$ Choisir p et q, deux grands nombres premiers distincts,
  - $\hookrightarrow$ Calculer $n = p \times q$, appelé module de chiffrement,
  - $\hookrightarrow$ Calculer la fonction d'Euler $\phi(n) = (p-1)(q-1)$,
  - $\hookrightarrow$ Choisir un entier e tel que $1 < e < \phi(n)$ et $pgcd\ (e, \phi(n)) = 1$,
  - $\hookrightarrow$ Calculer d, l'inverse modulaire de $e\ modulo\ \phi(n)$, c'est-à-dire $e \times d \equiv 1 (mod\ \phi(n))$,
  - $\hookrightarrow$ La clé publique est (n,e) et la clé privée est (n,d).
- **Chiffrement :** M est un entier strictement inférieur à n qui représente un message, alors le message chiffré sera exprimé sous la forme : $C = M^e\ mod\ n$,

• **Déchiffrement :** Pour déchiffrer C, on utilise la clé privée (d, n) comme suit : $M = C^d \ mod \ n$.

La sécurité de l'algorithme RSA repose sur la difficulté de factoriser un nombre entier très grand en ses facteurs premiers. Plus le nombre est grand, plus il est difficile à factoriser, et plus la sécurité de l'algorithme est élevée.

3.3 Fonction de hachage et signature numérique

Dans les communications où les messages sont chiffrés, il est essentiel de garantir que le contenu n'a pas été altéré ou manipulé par une personne malveillante pendant la transmission. Les fonctions de hachage sont utilisées pour vérifier l'authenticité du message et s'assurer que son contenu est resté intact.

3.3.1 Fonction de hachage

Une fonction de hachage h est une application prenant en entrée des messages, des suites de bits de longueurs quelconques et retournant un haché ou une empreinte de longueur fixé 160 bits en général. Pour des raisons de sécurité on tend à augmenter la taille de l'empreinte [6].

$$h : \{0,1\}^* \longrightarrow \{0,1\}^n,$$

La valeur de n représente le degré de sécurité de la fonction de hachage.
Cette fonction doit être à sens unique, c'est-à-dire que la transformation du message en hash doit être facile, mais la transformation du hash en message d'origine doit être très difficile, voire impossible.

La fonction de hachage doit également être sensible aux modifications apportées au message, de sorte que la modification d'une seule partie du message entraîne une modification significative de l'empreinte numérique ou du hash. Cela garantit que si le destinataire reçoit le message accompagné de son hash, il peut recalculer l'empreinte numérique à l'arrivée et la comparer à celle reçue. Si les deux empreintes numériques sont différentes, cela signifie que le message a été altéré ou modifié par un tiers. Les fonctions de hachage les plus couramment utilisées incluent MD5, SHA-1, SHA-2, SHA-3 et BLAKE2, chacune avec ses propres avantages et inconvénients en termes de sécurité et de performance. Le choix de la fonction de hachage appropriée dépend des exigences de sécurité et de performance de l'application [6].

3.3.2 Propriétés

Une fonction de hachage doit avoir les propriétés suivantes pour être considérée comme sécurisée :

↪ Propriété de compression et facilité de calcul,

↪ Résistance à la préimage : si h est une fonction de hachage et y est une empreinte donnée, il doit être difficile de trouver un message x tel que $h(x) = y$(fonction à sens unique),

↪ Résistance à la deuxième préimage : étant donné y et un x tel que $y = h(x)$, il est calculatoirement difficile de trouver un autre $x' \neq x$ tel que $y = h(x')$,

↪ Résistance à la collision, étant donné un message x, il est très difficile de trouver un autre message x' différent de x tel que la fonction h produise la même valeur de hachage pour les deux messages, c'est-à-dire que $h(x) = h(x')$. En d'autres termes, il est difficile de trouver deux messages différents qui ont la même empreinte numérique [6].

3.3.3 Applications des fonctions de hachage

Les fonctions de hachage sont largement utilisées dans les applications de sécurité informatique pour garantir l'intégrité des données, protéger les mots de passe et stocker les informations de manière sécurisée. Voici quelques exemples courants d'application des fonctions de hachage :

- **Stockage de mots de passe :** les fonctions de hachage sont souvent utilisées pour stocker les mots de passe de manière sécurisée. Au lieu de stocker le mot de passe en clair, un hachage est calculé à partir du mot de passe et stocké dans la base de données. Lorsqu'un utilisateur entre son mot de passe, le hachage est recalculé et comparé avec le hachage stocké dans la base de données pour vérifier si le mot de passe est correct,
- **Vérification de l'intégrité des données :** avant de transmettre ou de stocker des données, un hachage est calculé à partir des données et envoyé avec les données. Lorsque les données sont reçues ou lues, un nouveau hachage est calculé et comparé avec le hachage envoyé pour s'assurer que les données n'ont pas été altérées en transit ou en stockage,
- **Signature numérique :** les données à signer sont hachées, puis la signature numérique est créée à partir du hachage et d'une clé privée. Lorsque la signature est vérifiée, le hachage est recalculé à partir des données d'origine, puis la signature est vérifiée à l'aide de la clé publique correspondante,
- **Stockage de données sensibles :** les fonctions de hachage sont utilisées pour stocker les données sensibles de manière sécurisée, telles que les numéros de carte de crédit, les informations de compte bancaire et les numéros de sécurité sociale. Les données sont hachées avant d'être stockées, de sorte qu'en cas de violation de sécurité, les données ne peuvent pas être récupérées en clair,
- **Authentification :** les fonctions de hachage sont utilisées pour l'authentification dans les protocoles de sécurité, tels que TLS/SSL. Les fonctions de hachage sont utilisées pour calculer les empreintes digitales de certificats et les clés publiques, qui sont ensuite utilisées pour vérifier l'authenticité des certificats et des clés publiques.

3.4 Signature numérique

Les fonctions de hachage sont utiles pour s'assurer que les données d'un message n'ont pas été altérées pendant la transmission. Cependant, l'authentification de l'expéditeur reste un problème à résoudre. Comment pouvons-nous être sûrs que le message provient bien de la personne qui prétend l'avoir envoyé, et que cette personne ne pourra pas nier l'avoir envoyé plus tard ?
Pour répondre à ce problème, la solution est d'utiliser une signature numérique.

La signature numérique utilise généralement un algorithme de cryptographie asymétrique, comme RSA, pour créer une paire de clés publique-privée. La clé privée est utilisée pour signer le document et la clé publique est utilisée pour vérifier la signature.

En utilisant un algorithme de hachage, comme SHA-256 ou SHA-3, le document original est transformé en une empreinte numérique de longueur fixe. Cette empreinte est ensuite signée avec la clé privée de l'auteur de la signature.

Lorsque quelqu'un vérifie la signature, il calcule également l'empreinte numérique du document et compare cette empreinte avec celle qui a été signée. Si les deux empreintes correspondent, cela prouve que le document n'a pas été altéré depuis la signature et que l'auteur de la signature possède la clé privée associée à la clé publique utilisée pour la vérification.

En utilisant cette combinaison de techniques, une signature numérique est considérée comme authentique (provenant de l'auteur présumé), non réutilisable (car elle est spécifique au document signé), infalsifiable (car toute modification du document altère l'empreinte numérique), inaltérable (car toute altération est détectée lors de la vérification de la signature) et irrévocable (car la clé privée utilisée pour la signature est en possession de l'auteur) [5].

3.4.1 Signature numérique avec RSA

Le procédé de génération des paramètres est identiques a celui utilisé pour l'algorithme de cryptage RSA, l'expéditeur A doit disposer d'une clé publique (e, n) et d'une clé privée (d, n).

Les étapes de la création de la signature numérique du message m avec RSA sont :

Étape 1 : calculer le hash de m tel que : $y = h(m)$,

Étape 2 : calculer $S = y^d \ mod \ n$,

Étape 3 : envoyer la signature numérique (m, S).

Les étapes de vérification de l'entité B de la signature numérique transmise par A :

Étape 1 : calcul de hash y_1, tel que y_1 = $h(m)$,

Étape 2 : calcul de $h_2 = S^e \ mod \ n$,

Étape 3 : comparer les deux signatures y_1 et y_2, si les deux signatures sont égales, alors le message m est authentique, sinon le message m est altéré.

3.5 Objectifs de la cryptographie

La cryptographie a pour but de garantir la sécurité des communications entre l'expéditeur et le destinataire de l'information. Cette sécurité est assurée à différents niveaux :

- Confidentialité : seul le destinataire (et éventuellement l'expéditeur) peut comprendre l'information transmise,
- Intégrité : l'information transmise ne peut pas être altérée ou modifiée durant son trajet,
- Authentification : chaque interlocuteur est identifié de manière sûre et fiable, de sorte qu'il n'y a pas de risque d'usurpation d'identité,
- Non-Répudiation : l'expéditeur ne peut pas nier avoir envoyé l'information, ce qui assure la responsabilité de chaque partie dans la communication,
- Contrôle d'accès : l'accès à l'information est limité aux personnes autorisées, ce qui empêche toute divulgation non autorisée ou fuite de données sensibles.

3.6 Nivaux de sécurité d'un cryptosystème

Les différents niveaux de sécurité d'un cryptosystème dépendent des capacités de l'attaquant à violer la confidentialité, l'intégrité, l'authentification et la non-répudiation des données transmises, voici une brève explication de chacune de ces notions :

- **La sécurité inconditionnelle :** il s'agit d'une notion théorique selon laquelle le cryptosystème est considéré comme complètement inviolable et ne peut être compromis quelle que soit la puissance de calcul de l'attaquant. Cette notion est souvent associée à la cryptographie quantique qui utilise les propriétés de la physique quantique pour garantir la sécurité,
- **La sécurité calculatoire :** cette notion de sécurité repose sur l'idée qu'il est impossible pour un attaquant de déchiffrer un message en un temps raisonnable, même s'il possède une puissance de calcul importante. Les algorithmes de chiffrement modernes tels que AES ou RSA sont basés sur la sécurité calculatoire,
- **La sécurité prouvée :** elle est basée sur des preuves mathématiques et la théorie de la complexité des algorithmes pour démontrer que le cryptosystème est sûr et résistant aux attaques. Les cryptosystèmes à clé publique tels que RSA et Diffie-Hellman sont souvent soumis à des preuves mathématiques pour garantir leur sécurité,
- **La sécurité parfaite :** cette notion est basée sur la théorie de la cryptographie classique, qui stipule que la sécurité est garantie si la clé de chiffrement est aussi longue que le message à chiffrer et que la clé est générée de manière aléatoire. Cette notion est rarement utilisée dans les systèmes de chiffrement modernes, car elle nécessite une clé très longue et n'est pas pratique pour la plupart des applications [10].

3.7 Cryptanalyse

Il est indéniable que la sécurité est la qualité la plus importante pour tout cryptosystème, car il doit résister à toute tentative de déchiffrement qui viserait à récupérer le message original en clair. Malheureusement, la cryptanalyse constitue un véritable défi pour les cryptographes. Toutefois, sans les attaques et les méthodes de cryptanalyse, la cryptographie n'aurait jamais connu autant de progrès et serait restée au même niveau qu'à l'époque de Jules César. C'est pour cela que pour des systèmes aussi utilisés que RSA, AES, la cryptanalyse de ces algorithmes est essentielle pour en trouver les failles. Ainsi, il est souvent recommandé que le code source de l'algorithme soit rendu public afin de permettre aux cryptographes de le tester et d'en améliorer la sécurité si nécessaire.
Il est fréquent que les attaques contre les systèmes de chiffrement ne ciblent pas directement les algorithmes de chiffrement utilisés. Les failles peuvent être liées à une conception ou une réalisation défectueuse, ou une installation mal configurée. On peut distinguer deux types réels d'attaque : les attaques passives qui consistent à intercepter des messages et à exploiter les informations qu'ils contiennent sans les modifier, et les attaques actives qui cherchent à perturber la communication en ralentissant, altérant ou supprimant des informations, ou en envoyant des données parasites pour saturer les systèmes. Les types d'attaques varient en fonction du type de système de chiffrement utilisé (symétrique, asymétrique, fonction de hachage, etc.)[25]. Voici une liste de différents types d'attaques pouvant être utilisés contre les algorithmes de chiffrement :

- **Attaque à chiffré seul :** est une attaque où l'attaquant n'a accès qu'au texte chiffré sans avoir connaissance du texte clair correspondant. Dans ce type d'attaque, l'attaquant tente d'analyser les différences entre le texte clair et le texte chiffré pour déterminer les relations entre les bits de la clé de chiffrement et les bits de données. L'objectif final de l'attaquant est de deviner la clé de chiffrement et de déchiffrer d'autres messages chiffrés avec la même clé. Cette technique est utilisée pour casser des algorithmes de chiffrement à clé secrète en exploitant les caractéristiques de l'algorithme et en testant les différentes clés jusqu'à ce que la bonne clé soit trouvée. Cependant, pour les systèmes de chiffrement modernes, cette attaque est souvent inefficace en raison de l'utilisation de clés de chiffrement longues et aléatoires.
- **Attaque à clair connu :** est une technique de cryptanalyse où un attaquant a accès à la fois au texte clair et à son chiffrement correspondant. L'objectif de l'attaque est de déterminer la clé de chiffrement utilisée pour chiffrer le texte clair en question. Cette attaque peut être appliquée à différents types de chiffrements, y compris les chiffrements symétriques et asymétriques, mais elle est particulièrement efficace contre les chiffrements par substitution et les chiffrements par permutation.
- **Attaque à clair choisi :** semblable à la précédente, mais plus puissante que l'attaque à clair connu, car l'attaquant peut sélectionner des messages clairs spécifiques pour obtenir des informations sur la clé de chiffrement.
- **Attaque à chiffré choisi :** dans une attaque à texte chiffré choisi, le cryptanalyste choisit des

messages chiffrés spécifiques et demande leur version en clair. En obtenant les versions en clair des messages chiffrés, l'attaquant peut analyser la manière dont le chiffrement est appliqué et tenter de déduire des informations sur la clé de chiffrement utilisée.

Échelle de succès d'une attaque

L'échelle de succès d'une attaque de cryptographie peut être divisée en quatre niveaux :

- Cassage complet : l'attaquant parvient à découvrir la clé de chiffrement, ce qui lui permet de déchiffrer tous les messages cryptés,
- Déduction globale : l'attaquant découvre des fonctions équivalentes aux fonctions de chiffrement et de déchiffrement sans connaître la clé. Cela lui permet de déchiffrer tous les messages cryptés qui ont été chiffrés avec ces fonctions équivalentes,
- Déduction locale : l'attaquant peut déchiffrer un ou plusieurs nouveaux messages cryptés en utilisant des techniques spécifiques qu'il a développées,
- Déduction d'information : l'attaquant obtient des informations sur la clé de chiffrement ou sur des messages cryptés, ce qui lui permet de déchiffrer d'autres messages cryptés.

Mesure de robustesse d'un chiffrement

L'efficacité d'une attaque dépend de la robustesse du système de chiffrement ciblé, et peut être évaluée selon plusieurs critères, notamment :

- **Complexité des données :** la taille et le type de données nécessaires pour mener l'attaque,
- **Complexité en temps :** le volume de traitement temporel nécessaire pour mener l'attaque, mesuré en termes d'opérations élémentaires par unité de temps,
- **Complexité en mémoire :** l'espace de ressources nécessaire pour mener l'attaque,
- **Coût :** la qualité du résultat obtenu en fonction du coût de l'attaque, en tenant compte des ressources nécessaires et des conditions de succès fixées.

Ces critères peuvent être utilisés pour évaluer la performance et la faisabilité d'une attaque, ainsi que pour identifier les vulnérabilités d'un système de chiffrement et proposer des améliorations.

Conclusion

En conclusion, la cryptographie est un domaine fascinant qui englobe diverses techniques telles que le chiffrement symétrique, asymétrique, les fonctions de hachage et la signature numérique. Cependant, malgré la complexité de ces méthodes de cryptage, elles peuvent être exposées à des attaques malveillantes. C'est là qu'intervient la cryptanalyse, qui est l'étude des méthodes de décryptage et des attaques contre les systèmes de cryptage. Dans le prochain chapitre, on va présenté certains outils fondamentaux de la théorie des nombres, l'algorithme d'Euclide étendu, le calcul des facteurs premiers

et de leurs produits, ainsi que le calcul de l'inverse modulaire qui sont des concepts essentiels pour les applications cryptographiques. Ensuite, nous explorerons en profondeur les différentes techniques de cryptanalyse et comment elles peuvent être utilisées pour briser la sécurité des systèmes cryptographiques.

CHAPITRE 4

CRYPTANALYSE

Introduction

La première technique de cryptanalyse connue remonte à l'époque de Jules César, qui utilisait une méthode de chiffrement simple connue sous le nom de 'chiffrement par décalage'. Bien que cette méthode soit très simple et facile à utiliser, elle est également facile à casser à l'aide d'une analyse de fréquence, qui consiste à compter le nombre d'apparitions de chaque lettre dans le texte chiffré et à comparer ces fréquences aux fréquences typiques de la langue dans laquelle le texte est écrit. Cette méthode permet de déterminer la clé de chiffrement, c'est-à-dire le nombre de positions de décalage utilisé. Depuis lors, de nombreuses autres techniques de cryptanalyse ont été développées pour briser différents types de chiffrements, et la cryptographie est devenue un domaine de recherche et de développement majeur pour la sécurité des communications. Dans ce chapitre, nous allons explorer les différentes méthodes et techniques utilisées en cryptanalyse pour décrypter des messages, ainsi que les enjeux et les défis de cette discipline en constante évolution.

4.1 Quatre opérations de base en cryptanalyse

Pratiquement la résolution de chaque cryptogramme implique quatre opérations ou étapes fondamentales :

- Identifier la langue utilisée dans le texte en clair,
- Déterminer le système de cryptographie utilisé,
- Reconstruire la clé spécifique du système de chiffrement,
- Reconstruire ou établir le texte en clair.

Ces opérations seront abordées dans l'ordre dans lequel elles sont présentées ci-dessus et dans lequel elles sont généralement effectuées dans la résolution de cryptogrammes, bien que parfois la deuxième

étape puisse précéder la première [10].

4.2 Analyse de fréquence

Les Arabes ont été les premiers à découvrir la méthode de cryptanalyse par analyse de fréquence, qui consiste à déchiffrer les messages codés en utilisant des règles de substitution. Cette méthode a été décrite dans le 'Manuscrit sur le déchiffrement des messages cryptographiques' écrit par le savant Al-Kindi au IXème siècle.

4.2.1 Principe de l'analyse de fréquence

Le principe de l'analyse de fréquence consiste à déterminer la fréquence d'apparition de chaque lettre, chiffre ou symbole dans un texte chiffré, puis de comparer ces fréquences à celles de la langue utilisée. En effectuant cette comparaison, il est souvent possible de déterminer la clé de chiffrement utilisée pour chiffrer le message [14]. Pour que l'attaque par analyse de fréquence soit efficace, les conditions suivantes doivent être remplies :

↪ L'algorithme de chiffrement doit utiliser des substitutions mono-alphabétiques ou poly-alphabétiques qui préservent la distribution des fréquences, mais pas des transpositions,

↪ Le message chiffré doit être suffisamment long, car un texte trop court ne reflète pas la répartition générale des fréquences des lettres,

↪ La langue du message doit être connue pour pouvoir utiliser les statistiques de fréquence appropriées,

↪ La clé de chiffrement doit être relativement courte par rapport à la longueur du message pour permettre suffisamment de répétitions de lettres.

4.2.2 Outils utilisés en analyse de fréquence

Les tableaux de fréquences

La construction de tables de fréquences est une tâche délicate qui dépend de la méthode utilisée pour collecter et analyser les textes. Les résultats peuvent varier en fonction de la nature des textes choisis, qu'il s'agisse de textes anciens, scientifiques actuels, courriers électroniques, textes commerciaux, etc. La qualité du résultat dépend également de certaines spécificités du texte, telles que la présence de chiffres, d'abréviations et de caractères spéciaux, qui peuvent affecter la fréquence d'apparition des lettres. Tout le problème est donc de choisir le bon corpus pour établir le tableau de fréquences. En littérature, on rencontre une multitude de corpus ayant des caractéristiques plus ou moins différents. Les corpus les plus utilisés sont les suivants :

↪ pour la littérature française :

- Corpus Thomas Tempé,

- Corpus français,
- Corpus Wikipedia-FR.

↪ Pour la littérature anglaise :

- NYT Corpus,
- Concise Oxford Dictionary,
- Leipzig English Corpora.

	Français		Anglais		
	Thomas Tempé	Corpus Français	Concise OD	NYT	Leipzig Corpora
A	7.246	9.38	8.167	8,43	8.55
B	0.855	1.54	1.492	1,57	1.6
C	3.094	1.49	2.782	3,33	3.16
D	3.481	4.70	4.253	3,80	3.87
E	13.980	10.15	12.702	11,98	12.1
F	1.012	2.03	2.228	2,12	2.18
G	0.822	2.86	2.015	1,98	2.09
H	0.699	2.09	6.094	4,68	4.96
I	7.144	5.82	6.966	7,22	7.33
J	0.517	0.61	0.153	0,22	0.22
K	0.046	3.14	0.772	0,77	0.81
L	5.177	5.28	4.025	4,04	421
M	2.816	3.47	2.406	2,62	2.53
N	6.732	8.54	6.749	7,20	7.17
O	5.121	4.48	7.507	7,35	7.47
P	2.867	1.84	1.929	2,13	2.07
Q	1.292	0.02	0.095	0,10	0.1
R	6.218	8.43	5.987	6,51	6.33
S	7.542	6.59	6.327	6,83	6.73
T	6.874	7.69	9.056	8,86	8.94
U	5.988	1.92	2.758	2,54	2.68
V	1.545	2.42	0.978	1,04	1.06
W	0.108	0.14	2.360	1,71	1.83
X	0.367	0.16	0.150	0,20	0.19
Y	0.292	0.71	1.974	1,76	1.72
Z	0.129	0.07	0.074	1,02	.11

FIGURE 4.1 – Fréquence de caractères de certains Corpus

Indice de coïncidence

L'IC mesure la probabilité que deux lettres choisies au hasard dans un échantillon de texte soient identiques. Il est utilisé en cryptanalyse pour aider à déterminer si un chiffrement est mono-alphabétique ou poly-alphabétique. Un IC proche de 0,0385 est indicatif d'un chiffrement mono-alphabétique, tandis qu'un IC plus élevé est indicatif d'un chiffrement poly-alphabétique. Les chiffrements poly-alphabétiques, tels que le chiffre de Vigenère, ont tendance à masquer les fréquences des lettres dans le texte, ce qui entraîne un IC plus élevé.

Soit p un texte écrit dans un langage L. Si le texte $'p'$ est de taille $'n'$ et qu'il contient tous les caractères

de l'alphabet '$\mathcal{A}_a$', où '$\mathcal{A}$' appartient au langage 'L' et 'a' est la taille de l'alphabet. Supposons que le nombre de caractères dans 'p' est distribué uniformément, alors le nombre d'occurrences de n'importe quel caractère 'i' de l'alphabet '$\mathcal{A}$' peut être défini par la relation suivante :

$$n_i = \frac{n}{a},$$

Si est n est suffisamment grand, on peut supposer que $n(n-1) = n^2$, et de même pour n_i qui devient $\frac{n}{a}(\frac{n}{a}-1) = (\frac{n}{a})^2$. Si A contient les caractères de l'alphabet courant, ce qui est le cas en général, on aura $a = 26$. L'indice de coïncidence minimal du texte p sera défini par la relation suivante :

$$I_p = \frac{1}{n^2}\sum_{i=1}^{26}(\frac{n}{26})^2 \approx \frac{1}{26} = 0.385.$$

Indice de coïncidence réel

L'indice de coïncidence réel (ICR) est une mesure utilisée en cryptanalyse pour évaluer la similarité entre la distribution des fréquences d'apparition des lettres dans un texte chiffré et la distribution des fréquences d'apparition des lettres dans la langue utilisée dans le texte original. Il défini par la relation $I_l = I_A + I_B + \cdots + I_Z$ où I_A est la probabilité de tirer aléatoirement deux 'A', I_B, la probabilité de tirer deux 'B' et de même pour les autres caractères de $\mathcal{A}$. Puisqu'il y a C_n^2 possibilités de tirer deux caractères identiques d'un texte de taille n, l'indice du caractère 'A' sera donc :

$$I_A = \frac{C_{n_A}^2}{C_n^2} = \frac{\frac{n_A!}{2(n_A-2)!}}{\frac{n!}{2(n-2)!}} = \frac{n_A(n_A-1)}{n(n-1)},$$

L'indice du texte p contenant les 26 caractères de $\mathcal{A}$ devient :

$$I_p = \sum_{i=1}^{26}\frac{n_i(n_i-1)}{n(n-1)}.$$

L'ICR peut prendre des valeurs entre 0 et 1, où une valeur proche de 0 indique une distribution de fréquences uniforme (c'est-à-dire une absence de similarité avec la distribution de fréquences attendue), tandis qu'une valeur proche de 1 indique une forte similarité entre les deux distributions de fréquences. En pratique, l'ICR peut être utilisé pour déterminer la longueur de la clé d'un chiffrement à substitution monoalphabétique. En effet, si la clé est de longueur k, alors les lettres du texte chiffré sont simplement décalées selon un motif de longueur k. Cela signifie que la distribution des fréquences d'apparition des lettres dans le texte chiffré sera similaire à la distribution des fréquences d'apparition des lettres dans le texte original, mais décalée selon un motif de longueur k. En utilisant l'ICR, on peut essayer de déterminer la longueur k en trouvant une longueur qui maximise la valeur de l'ICR.

Test de Friedman

L'idée de Friedman, nommée également le test de Kappa en référence à sa formulation mathématique, repose sur le principe suivant : pour un chiffrement mono-alphabétique, l'indice de coïncidence I_l propre à chaque langue ne change pas si l'on remplace un caractère par un autre caractère différent, car chaque caractère est codé de la même manière. En revanche, pour un chiffrement poly-alphabétique, l'indice de coïncidence I_m prendra une valeur minimale largement inférieure à 0,04 car chaque portion

du texte (égale à la longueur de la clé) est codée avec un alphabet différent.
Supposons que nous ayons un texte chiffré c composé de n caractères et qu'une séquence t_p de p caractères apparaît plusieurs fois dans le texte avec une distance multiple d'une constante d. Il existe deux possibilités pour expliquer ce fait :

- La séquence provient de la même séquence du texte clair, qui a été chiffrée avec la même partie de la clé utilisée pour le chiffrement de la séquence t_p,
- La séquence pourrait provenir de différentes séquences du texte clair, mais par coïncidence, elles ont été chiffrées avec la même partie de la clé.

Dans le premier cas, si la longueur de la clé est k, alors pour que deux séquences soient chiffrées avec la même partie de la clé, il faut que k divise d, avec $(p < k \leq d << n)$. Le $PGCD$ de k et d peut être considéré comme la longueur de la clé. Chaque séquence t_{ik} de k caractères consécutifs (avec $i = 1, 2, \ldots, t/k$) correspond à une substitution mono-alphabétique de clé k. Chaque caractère $j (j = 1, 2, \ldots, k)$ de la clé a été utilisé pour chiffrer la séquence t_{ik}^{j} du texte t, avec i appartenant à l'ensemble $\{1, 2, \ldots, t/k - 1\}$. En appliquant une analyse de fréquences à chaque séquence t_{ik}, nous pouvons déterminer le caractère correspondant de la clé.
L'idée de Kasiski repose sur ce principe et consiste à rechercher les séquences identiques (de préférence des polygrammes longs) au sein d'un texte chiffré. Si ces séquences sont trouvées, les distances qui les séparent sont relevées. Le diviseur commun de ces distances pourrait être, dans la plupart des cas, la longueur de la clé de chiffrement.

4.2.3 Limitations de l'analyse de fréquence

L'analyse de fréquence est une technique de cryptanalyse utile pour déduire des informations sur un texte chiffré, mais elle présente des limitations importantes et ne doit pas être utilisée comme seule méthode pour casser des codes. Les restrictions de l'analyse de fréquence sont les suivantes :

- L'analyse de fréquence ne fonctionne que si le texte chiffré a été chiffré à l'aide d'une substitution mono-alphabétique. Si le texte a été chiffré à l'aide d'une substitution poly-alphabétique, où chaque lettre peut être chiffrée de manière différente en fonction de sa position dans le texte, l'analyse de fréquence sera beaucoup plus difficile, voire impossible,
- L'analyse de fréquence ne permet pas de récupérer les espaces ou la ponctuation dans le texte chiffré, car ces symboles n'ont pas de fréquence d'apparition caractéristique. Cela peut rendre la tâche de déchiffrer le texte chiffré plus difficile,
- L'analyse de fréquence ne prend pas en compte le contexte du texte. Par exemple, la fréquence d'apparition de la lettre 'q' peut être très faible dans un texte en français, mais elle peut être plus fréquente dans un texte en allemand. Ainsi, l'analyse de fréquence peut être trompeuse si elle est appliquée à un texte chiffré dans une langue différente de la langue utilisée pour établir les fréquences de référence,

- L'analyse de fréquence peut être contournée en utilisant des techniques de chiffrement plus sophistiquées, telles que la substitution homophonique, où plusieurs symboles sont utilisés pour représenter chaque lettre, ou la transposition, où les lettres sont simplement réorganisées dans le texte sans être chiffrées individuellement.

4.2.4 Les contre-mesures

Pour empêcher ou rendre plus difficile l'analyse de fréquence :

- ► Pour rendre l'analyse de fréquence moins efficace, les chiffrements poly-alphabétiques sont souvent utilisés, tels que le code de Vigenère ce qui rend plus difficile la détermination des fréquences des lettres,
- ► Mélange de caractères : pour éviter que les fréquences de caractères soient facilement déterminables, il est possible de mélanger les caractères du message crypté, en utilisant par exemple des techniques de transposition,
- ► Ajout de bruit : une autre technique consiste à ajouter des caractères aléatoires dans le message crypté pour brouiller les fréquences des lettres. Cela peut être fait en ajoutant des caractères de remplissage ou en utilisant des techniques de padding,
- ► Utilisation de chiffrement moderne : les chiffrements modernes, tels que AES (Advanced Encryption Standard), sont conçus pour résister à l'analyse de fréquence et à d'autres types d'attaques cryptographiques. Ces algorithmes sont basés sur des techniques de chiffrement avancées et sont considérés comme étant très sécurisés.

4.2.5 Exemple d'application

Le texte suivant résulte de chiffrement d'un texte français par une substitution monoalphabétique [24].

"V ubcfb osu ymoqsuu n cxqfj dqmfnu ub vjcfqu juz amqjmruz zmsscfusb bquflu auoquz hfszbms zwfba ju wusbms qusbqu ncsz ju vmo z uddmqvcfb n uxfbuq ju xusb wcoxcfz fj eczzc qcefnuwusb jc emqbu xfbquu no ijmv nuz wcfzmsz nu jc xfvbmfqu ecz czzul qcefnuwusb vueusncsb emoq uweuvauq kou jof os bmoqifjjms nu emozzfuqu ub nu zciju.

Ju acjj zusbcfb ju vamo vofb ub ju xfuog bcefz c j osu nu zuz ugbquwfbuz osu cddfvau nu vmojuoq bqme xczbu emoq vu nuejmfuwusb fsbuqfuoq ubcfb vjmouu co woq ujju quequzusbcfb zfwejuwusb os usmqwu xfzcru jcqru nu ejoz n os wubqu ju xfzcru n os amwwu n usxfqms kocqcsbu vfsk czz c j uecfzzu wmozbcvau smfqu cog bqcfbz cvvusbouz ub iucog.

Hfszbms zu nfqfruc xuqz j uzvcjfuq fj ubcfb fsobfju n uzzcpuq nu equsnqu j czvuszuoq wuwuw cog wufjjuoquz uemkouz fj dmsvbfmsscfb qcquwusb cvboujjuwusb n cfjjuoqz ju vmoqcsb ujuvb q fkou ubcfb osu nuz wuzoquz n uvmsmwfu eqfzuz us xou nu jc zuwcfsu nu jc acfsu. "

Le tableau ci-dessous présente le nombre d'occurrences de chaque caractère dans le texte chiffré :

a	b	c	d	e	f	g	h	i	j	k	l	m
10	59	60	8	22	60	5	2	4	48	6	2	38
n	o	p	q	r	s	t	u	v	w	x	y	z
29	49	1	53	6	57	0	157	27	28	13	2	57

Il semble probable que le caractère 'u' dans le texte chiffré corresponde au caractère 'e' dans le texte clair (noté en gras). Les autres lettres les plus fréquentes sont 'c' et 'f', mais elles ont le même nombre d'occurrences, rendant difficile la détermination des caractères qu'elles représentent à ce stade.

Les bigrammes les plus courants de la langue française, sont : 'es', 'le', 're', 'de', 'en', 'et', 'ai', 'te', 'ou', 'nt', 'it', 'er' et 'la'. Les bigrammes commençant par 'u' dans le texte chiffré ne sont pas assez fréquents pour décider quel caractère correspond à la lettre 's' dans le texte clair. Les bigrammes les plus fréquents se terminant par 'u' dans le texte chiffré sont 'ju' (18 occurrences) et 'nu' (15 occurrences), suggérant que 'j' représente 'l' et 'n' représente 'd'.

En utilisant ces trois substitutions, le premier paragraphe du texte chiffré peut être reformulé comme suit : "V **e**bcfb os**e** ymoqs**ee d** cxqf**l** dqmf**de e**b v**l**cfq**e le**z amq**l**mr**e**z zmsscf**e**sb bq**e**f**l**e a**e**oq**e**z hf-szbms zwfba **le** w**e**sbms q**e**sbq**e** ncsz **le** vmo z **e**ddmqvcfb **d e**xfb**e**q **le** x**e**sb wcoxcfz f**l** eczzc qcef**de**w**e**sb **l**c emqb**e** xfbq**ee d**o i**l**mv n**e**z wcfzmsz **de l**c xfvbmfq**e** ecz czz**e**l qcef**de**w**e**sb v**eee**sncsb emoq **e**wee**e**va**e**q ko**e l**of os bmoqif**ll**ms **de** emozzf**e**q**e e**b **de** zci**le**."

La suggestion selon laquelle le caractère 'o' dans le mot 'do' pourrait représenter 'u' nous conduit à considérer le mot 'ose' comme 'use', ce qui suggère que le caractère 's' représente 'n'. En utilisant cette déduction, on peut reformuler le premier paragraphe du texte de la manière suivante :

"V **e**bcfb **une** ym**u**q**nee d** cxqf**l** dqmf**de e**b v**l**cfq**e le**z amq**l**mr**e**z zm**nn**cf**en**b bq**e**f**l**e a**eu**q**e**z hf**n**zbm**n** zwfba **le** w**en**bm**n** q**en**bq**e** nc**n**z **le** vm**u** z **e**ddmqvcfb **d e**xfb**e**q **le** x**en**b wc**u**xcfz f**l** eczzc qcef**de**w**en**b **l**c emqb**e** xfbq**ee du** i**l**mv n**e**z wcfzm**n**z **de l**c xfvbmfq**e** ecz czz**e**l qcef**de**w**en**b v**eeen**nc**n**b em**u**q **e**wee**e**va**e**q ko**e l**of **un** bm**u**qif**ll**m**n de** em**u**zzf**e**q**e e**b **de** zci**le**."

La répétition du mot 'kue' dans le texte suggère que la lettre 'k' pourrait représenter 'q'. De plus, les mots 'qcefdewenb' et 'eb' nous indiquent que la lettre 'b' pourrait représenter 't'. Enfin, les mots 'luf' et 'fl' suggèrent que la lettre 'f' pourrait représenter 'i'. En appliquant ces déductions, nous obtenons le texte reformulé suivant :

"V **et**cf**t une** ym**u**q**nee d** cxq**il** dqm**ide et** v**l**ci**q**e **le**z amq**l**mr**e**z zm**nn**c**ient** t**q**e**i**l**e** a**eu**q**e**z h**i**n**z**tm**n** zw**i**t**a le** w**ent**m**n** q**ent**q**e** nc**n**z **le** vm**u** z **e**ddmqvc**it d exite**q **le** x**ent** wc**u**xc**iz il** eczzc qce**ide**w**ent l**c emq**te** x**i**tq**ee du** i**l**mv n**e**z wc**i**zm**n**z **de l**c x**i**v**t**mi**q**e ecz czz**e**l qce**ide**w**ent** v**eeen**nc**nt** em**u**q **e**wee**e**va**e**q **que lui un** t**m**u**q**i**ill**m**n de** em**u**zz**ie**q**e et de** zci**le**."

La présence du mot 'etcit' dans le texte suggère que la lettre 'c' pourrait représenter 'a'. De plus, les mots 'wewe' et 'qceidewent' suggèrent que la lettre 'w' pourrait représenter 'm'. En appliquant ces déductions, le texte chiffré devient :

"V **etait une** ym**u**q**nee d a**xq**il** dqm**ide et** v**lai**q**e le**z amq**l**mr**e**z zm**nnaient t**q**ei**l**e** a**eu**q**e**z h**i**n**z**tm**n** z**mit**a **le ment**m**n** q**ent**q**e** n**an**z **le** vm**u** z **e**ddmqv**ait d exite**q **le** x**ent mau**x**ai**z **il** e**a**zz**a** q**ae**i**dement**

la emq**te** x**it**q**ee du** i**l**mv n**ez ma**i**z**m**nz de la** x**iv**t**mi**q**e** e**az a**zz**e**l q**ae**i**dement** v**eeen**n**ant** em**u**q **em**ee**v**a**e**q **que lui un t**m**u**qi**illm**n **de** em**uzzie**q**e et de** z**aile**."

L'observation du mot 'mentmn' suggère que la lettre 'm' pourrait représenter 'o'. De plus, le mot 'danz' suggère que la lettre 'z' pourrait représenter 's'. En considérant le mot 'mauxaiz', nous pourrions déduire que la lettre 'x' représente 'v'. En utilisant ces déductions, nous pouvons également conclure que le mot 'exiteq' suggère que la lettre 'q' représente 'r'. Enfin, l'expression 'en meme temez' suggère que 'e' représente 'p'. À partir de ces informations, nous avons établi la table de correspondance suivante :

a	b	c	d	e	f	g	h	i	j	k	l	m
	t	a		p	i				l	q		o
n	o	p	q	r	s	t	u	v	w	x	y	z
d	u		r		n		e		m	v		s

Et le texte suivant :

"V **eait une** y**ournee d avril** d**roide et** v**laire les** a**orlo**r**es sonnaient trei**l**e** a**eures** h**inston smit**a **le menton rentre** n**ans le** v**ou s e**dd**orvait d eviter le vent mauvais il** e**assa ra**e**idement la** em**rte vitree du** i**lo**v n**es mais**m**ns de la vi**v**toire** e**as asse**l **ra**e**idement** v**eeendant** e**our eme**e**v**a**er que lui un tour**i**illon de** e**oussiere et de sa**i**le**.

Le a**all sentait le** va**ou** v**uit et le vieu**g **tapis a l une de ses e**g**tremites une a**ddiva**e de** v**ouleur trop vaste pour** v**e deploiement interieur etait** v**louee au mur elle representait simplement un enorme visa**r**e lar**r**e de plus d un metre le visa**r**e d un** a**omme d environ quarante** v**inq assa l epaisse mousta**v**ae noire au**g **traits a**vv**entues et** i**eau**ucog h**inston se diri**r**ea vers l esvalier il etait inutile d essa**p**er de prendre l as**v**enseur meme au**g **meilleures epoques il d**on**vtionnait rarement a**v**tuellement d ailleurs le** v**ourant elevtr**textbfique **etait une des mesures d e**v**onomie prises en vue de la semaine de la** a**aine**. "

En terminant l'analyse de façon similaire, nous obtenons la table de correspondance :

a	b	c	d	e	f	g	h	i	j	k	l	m
h	t	a	f	p	i	x	w	b	l	q	z	o
n	o	p	q	r	s	t	u	v	w	x	y	z
d	u	y	r	g	n	k	e	c	m	v	j	s

4.3 Recherche exhaustive de la clé

La recherche exhaustive sur la clé est une attaque élémentaire utilisée pour tout algorithme cryptographique ayant une taille de clé fixe. L'attaque consiste à tester toutes les clés possibles jusqu'à ce que la bonne soit trouvée. Pour cette attaque, l'attaquant a besoin des spécifications de l'algorithme cryptographique et d'un test d'arrêt pour détecter la clé utilisée avec une faible probabilité de fausse alarme. Pour les algorithmes de chiffrement à clé secrète, l'attaquant doit également disposer d'une quantité suffisante de texte chiffré correspondant à la taille de la clé.

Pour se prémunir contre la recherche exhaustive, le nombre de clés d'un algorithme cryptographique doit être beaucoup plus grand que le nombre d'opérations réalisables en pratique avec les moyens de calcul actuels et futurs. Les calculs requérant environ 2^{64} opérations ont été réalisés dans le passé par des projets de calcul distribué, et il est estimé qu'un calcul nécessitant 2^{128} opérations n'est pas réalisable à moyen terme. Par conséquent, une taille de clé de 128 bits est recommandée pour se protéger contre cette attaque.
Il est important de noter que, dans de nombreux cas, la recherche exhaustive reste la meilleure attaque en pratique contre un algorithme, même s'il existe des attaques théoriques plus sophistiquées. Cela est dû au fait que cette attaque nécessite très peu de données et se parallélise facilement, ce qui permet de tirer pleinement parti de l'augmentation de la puissance de calcul disponible. De plus, de nombreux cryptosystèmes pratiques sont sous-dimensionnés ou utilisent des clés de faible entropie, ce qui les rend vulnérables à la recherche exhaustive.
L'attaque par recherche exhaustive peut être utilisée contre toute forme de chiffrement, à condition que l'attaquant ait accès au texte chiffré et qu'il connaisse les spécifications de l'algorithme de chiffrement utilisé. Les cibles courantes d'une attaque par recherche exhaustive sont les systèmes de chiffrement symétriques, tels que le chiffrement par bloc (comme AES) ou le chiffrement par flux (comme RC4).
L'attaque par recherche exhaustive peut également être utilisée contre des systèmes de chiffrement asymétriques, tels que le RSA, mais en raison des clés plus grandes utilisées dans ces systèmes, elle est beaucoup plus difficile à réaliser en pratique [10].

4.3.1 Complexité théorique de l'attaque

En théorie la complexité d'une attaque par force brute est une fonction exponentielle de la longueur de la clé utilisée, ainsi pour une clé K codée sur N bits uniformément distribués, le nombre maximum d'essais nécessaires est égal à 2^N essais. Pour une clé codée sur 64 bits (ce qui est rarement le cas aujourd'hui), le nombre maximum de clés à tester est 2^{64} = 18 446 744 073 709 551 616 = 18,446 milliards de milliards de clés. Ainsi pour un super ordinateur pouvant tester un milliard de clés par seconde il lui faut plus de 584 ans de calcul sans arrêt pour atteindre ce nombre.
Il reste à noter que les clés d'aujourd'hui sont codées sur plus de 128 bits. Des solutions ont été proposées pour s'affronter à ce problème, les plus importants sont :

1− **Attaque par dictionnaire :**

L'attaque par dictionnaire est une technique de cryptanalyse utilisée pour essayer de trouver un mot de passe ou une clé. Elle implique de tester une liste de mots de passe potentiels les uns après les autres, en espérant que le mot de passe utilisé pour le chiffrement soit contenu dans le dictionnaire. Cette méthode est basée sur le fait que de nombreuses personnes utilisent des mots de passe courants, tels qu'un nom, une couleur ou le nom d'un animal. Pour cette raison, il est recommandé de ne pas utiliser de mots courants comme mot de passe. L'attaque par dictionnaire est souvent combinée à l'attaque par force brute, qui consiste à tester systématiquement toutes les combinaisons de mots de passe possibles,

en particulier pour les mots de passe de 5 ou 6 caractères. Contrairement à la génération aléatoire de clés d'essai, l'attaque par dictionnaire permet de sélectionner des clés qui sont significatives, ce genre d'attaque est employé pour réduire d'une manière intelligente le nombre de clés d'essai sous réserve que le dictionnaire soit conforme à l'environnement du chiffrement utilisé [14].

Comment éviter une attaque par dictionnaire

Il est possible d'arrêter les attaques en ligne en utilisant des techniques de sécurité telles que l'utilisation de captchas, la mise en place de l'authentification à deux facteurs obligatoire et la limitation du nombre de tentatives de connexion. Cependant, les attaques hors ligne sont plus difficiles à contrer.

Pour lutter contre les attaques hors ligne, il est conseillé d'utiliser l'authentification à deux facteurs et d'établir des règles strictes en matière de choix de mots de passe, telles que l'interdiction d'utiliser des mots de passe courants, de simples mots ou phrases, et un minimum de 12 caractères. Il est également important de ne pas stocker les mots de passe en clair et de les chiffrer à l'aide d'algorithmes de hachage sécurisés.

2– **Technique du mot probable :**

Lorsque l'on intercepte un message chiffré, il est possible que l'on ait des informations sur le contenu d'origine du message, par exemple le nom de l'expéditeur ou du destinataire, une formule de politesse, ou d'autres termes. Ces informations sont appelées « mots probables » car elles sont susceptibles d'apparaître dans le texte original. Lorsqu'on dispose de ces mots probables, cela peut aider à réduire l'espace des clés possibles et ainsi faciliter la tâche de décryptage. Par exemple, si l'on sait que le message a été envoyé par Alice à Bob, cela peut aider à déterminer la clé de chiffrement car on peut supposer que le nom « Alice » et le nom « Bob » apparaîtront dans le message chiffré. Cependant, il est important de noter que plus la longueur du message chiffré est grande et plus le chiffrement est complexe, moins ces mots probables seront utiles pour déterminer la clé de chiffrement. Par exemple, si un message est chiffré avec une clé de chiffrement forte et qu'il est de plusieurs pages, même si en connaissant quelques mots dans le texte original, cela ne nous aidera probablement pas beaucoup à retrouver la clé de chiffrement. Il semble que ce type d'attaque soit efficace contre la plupart des méthodes de chiffrement classiques, telles que le chiffre de Vigenère, les substitutions homophoniques et le chiffre de Hill [14].

4.3.2 Exemple d'application

La recherche exhaustive peut être utilisée pour décrypter un message chiffré avec le chiffrement de César.

Par exemple, si le message chiffré est " at rdst rthpg, r'thi uprxat p rphhtg", on teste toutes les clés possibles, on trouve que la clé correcte est 11.

Décalage de 0 : at rdst rthpg, r'thi uprxat p rphhtg,

Décalage de 1 : bu setu suiqh, s'uij vqsybu q sqiiuh,
Décalage de 2 : cv tfuv tvjri, t'vjk wrtzcv r trjjvi,
Décalage de 3 : dw ugvw uwksj, u'wkl xsuadw s uskkwj,
Décalage de 4 : ex vhwx vxltk, v'xlm ytvbex t vtllxk,
Décalage de 5 : fy wixy wymul, w'ymn zuwcfy u wummyl,
Décalage de 6 : gz xjyz xznvm, x'zno avxdgz v xvnnzm,
Décalage de 7 : ha ykza yaown, y'aop bwyeha w ywooan,
Décalage de 8 : ib zlab zbpxo, z'bpq cxzfib x zxppbo,
Décalage de 9 : jc ambc acqyp, a'cqr dyagjc y ayqqcp,
Décalage de 10 : kd bncd bdrzq, b'drs ezbhkd z bzrrdq,
Décalage de 11 : le code cesar, c'est facile a casser,
Décalage de 12 : mf dpef dftbs, d'ftu gbdjmf b dbttfs,
Décalage de 13 : ng eqfg eguct, e'guv hceknq c ecuugt,
Décalage de 14 : oh frgh fhvdu, f'hvw idfloh d fdvvhu,
Décalage de 15 : pi gshi giwev, g'iwx jegmpi e gewwiv,
Décalage de 16 : qj htij hjxfw, h'jxy kfhnqj f hfxxjw,
Décalage de 17 : rk iujk ikygx, i'kyz lgiork g igyykx,
Décalage de 18 : sl jvkl jlzhy, j'lza mhjpsl h jhzzly,
Décalage de 19 : tm kwlm kmaiz, k'mab nikqtm i kiaamz,
Décalage de 20 : un lxmn lnbja, l'nbc ojlrun j ljbbna,
Décalage de 21 : vo myno mockb, m'ocd pkmsvo k mkccob,
Décalage de 22 : wp nzop npdlc, n'pde qlntwp l nlddpc,
Décalage de 23 : xq oapq oqemd, o'qef rmouxq m omeeqd,
Décalage de 24 : yr pbqr prfne, p'rfg snpvyr n pnffre,
Décalage de 25 : zs qcrs qsgof, q'sgh toqwzs o qoggsf,

4.4 Cryptanalyse linéaire

La cryptanalyse linéaire est une technique de cryptanalyse inventée en 1993 par Mitsuru Matsui, qui visait initialement à casser l'algorithme de chiffrement symétrique DES. Cette technique repose sur le concept d'expressions linéaires probabilistes étudiées auparavant par Henri Gilbert et Anne Tardy-Corfdir dans le cadre d'une attaque sur FEAL. Bien que la cryptanalyse linéaire soit plus efficace que la cryptanalyse différentielle, elle est moins pratique car elle suppose que l'attaquant ne dispose pas de la boîte noire représentant l'algorithme de chiffrement et ne peut pas soumettre ses propres textes. En augmentant le nombre de paires de texte chiffré et de texte en clair disponibles, la précision de l'approximation linéaire de l'algorithme de chiffrement est améliorée, permettant ainsi d'extraire la clé de chiffrement. Les nouveaux algorithmes de chiffrement doivent être conçus pour être résistants à ce type d'attaque, car DES présentait certaines propriétés linéaires dues à ses tables de substitution

(S-boîtes) qui étaient destinées à ajouter de la non-linéarité à l'algorithme. La cryptanalyse linéaire a également été appliquée avec succès à d'autres algorithmes, tels que LOKI, FEAL et une version simplifiée de Serpent. Cependant, les algorithmes plus récents comme AES (Rijndael) et IDEA sont conçus pour être insensibles à une attaque linéaire, car leur complexité est largement supérieure à celle d'une recherche exhaustive [6].

4.4.1 Grandes lignes d'une attaque linéaire

Les étapes de la cryptanalyse linéaire peuvent être résumées comme suit :

- **Collecte des données :** pour mener une cryptanalyse linéaire, l'attaquant doit collecter un grand nombre de paires de texte en clair et texte chiffré, pour pouvoir détecter des relations linéaires entre les bits de texte en clair et de texte chiffré,
- **Approximation linéaire :** l'attaquant doit choisir une approximation linéaire qui relie des bits de texte en clair et de texte chiffré. Cette approximation est utilisée pour récupérer des informations sur la clé de chiffrement,
- **Calcul du coefficient de corrélation :** l'attaquant utilise l'approximation linéaire pour calculer le coefficient de corrélation entre les bits de texte en clair et de texte chiffré. Le coefficient de corrélation mesure la force de la relation linéaire entre ces bits,
- **Construction de l'équation de clé :** en utilisant les coefficients de corrélation, l'attaquant peut construire une équation de clé qui relie la clé de chiffrement aux bits de texte en clair et de texte chiffré. L'attaquant peut ensuite résoudre cette équation pour déterminer la valeur de la clé de chiffrement,
- **Vérification de la clé :** une fois que l'attaquant a obtenu une clé possible, il peut vérifier si cette clé est correcte en utilisant d'autres paires de texte en clair et texte chiffré. Si la clé est correcte, l'attaquant peut l'utiliser pour décrypter des messages chiffrés [6].

4.4.2 Principe

L'attaque linéaire est une attaque à clairs connus visant les chiffrements par bloc itératifs. L'approche générale de cette attaque consiste à essayer d'approximer le chiffrement en utilisant des équations linéaires qui relient les bits du message clair, les bits du message chiffré et les bits de la clé.

On peut exprimer une équation linéaire en utilisant le produit scalaire entre un vecteur $x = (x_1, \ldots, x_n) \in F_2^n$ et un vecteur de coefficients $\alpha = (\alpha_1, \ldots, \alpha_n) \in F^n$, c'est-à- dire : $(\alpha, x) = \alpha_1 x_1 \oplus \cdots \oplus \alpha_n x_n$.

Si nous choisissons x de manière équiprobable dans F_2^n, alors l'équation $(\alpha, x) = 0$ est satisfaite avec une probabilité de $1/2$ pour tout vecteur non nul α. Cela est dû au fait que la forme linéaire $f : F_2^n \longrightarrow F_2, x \longrightarrow (\alpha, x)$ est non nulle et que son noyau a une dimension de $n - 1$. Selon le théorème du rang, nous avons $n = dim(Imf) + dim(kerf)$. Par conséquent, la moitié des valeurs de x satisfont l'équation $(\alpha, x) = 0$.

Dans ce contexte, on cherche à trouver des équations du type $(\alpha, x) \oplus (\beta, c) \oplus (\gamma, k) = 0$ qui sont vraies avec une probabilité de $1/2 + \varepsilon$, où m est choisi uniformément. Cette équation nous permet de construire une équation linéaire sur les bits de la clé secrète k.

Supposons maintenant que nous connaissions certains couples (m, c), nous pouvons alors calculer $(\alpha, m) \oplus (\beta, c)$ pour chaque couple. Si le résultat est généralement égal à zéro, nous pouvons en déduire que $(\gamma, k) = 0$, car l'équation $(\alpha, x) \oplus (\beta, c) \oplus (\gamma, k) = 0$ est vraie avec une probabilité de $1/2 + \varepsilon$, où $\varepsilon > 0$.

En d'autres termes, en exploitant les équations que nous avons construites, nous pouvons déterminer certains bits de la clé secrète k [6].

Exemple 4.1. Soit une table de substitution S représentée par le tableau ci-dessous :

Input	000	001	010	011	100	101	110	111
Output	010	100	000	111	001	110	101	011

Cette boite prend en entrée un nombre hexadécimal composé de 3 bits X_1, X_2, X_3 et fournie en sortie un autre nombre hexadécimal constitué de 3 bits aussi Y_1, Y_2, Y_3.

Soient les deux équations linéaires suivantes :

$X_1 \oplus X_2 \oplus X_3 = Y_1 \oplus Y_2 (1)$

$X_2 \oplus X_3 = Y_3 (2)$

La figure ci-dessous donne les différents cas pour lesquels les deux équations sont satisfaites :

Première équation

X	Y	$X_1 \oplus X_2 \oplus X_3$	$Y_1 \oplus Y_2$
000	010	0	1
001	100	1	1
010	000	1	0
011	111	0	0
100	001	1	0
101	110	0	0
110	101	0	1
111	011	1	1

Deuxième équation

X	Y	$X_2 \oplus X_3$	Y_3
000	010	0	0
001	100	1	0
010	000	1	0
011	111	0	1
100	001	0	1
101	110	1	0
110	101	1	1
111	011	0	1

On remarque que la probabilité de satisfaction de la 1 ère équation est 4/8, et pour la 2 ème équation est égale à 2/8.

L'approche de cryptanalyse linéaire consiste à chercher des approximations qui ont des probabilités d'occurrence très élevées ou bien très faibles.

Lemme d'empilement Soit $X_1, X_2, \ldots, X_N$ des variables aléatoires binaires indépendantes telles que $P(X_i = 0) = \frac{1}{2} \pm \varepsilon i$, pour tout $i \in \{1, \ldots, N\}$. Alors,

$$P(X_1 \oplus \cdots \oplus X_N = 0) = \tfrac{1}{2} \pm 2^{N-1} \prod_{i=1}^{N} \varepsilon_i.$$

4.4.3 Exemples concrets de l'application de la cryptanalyse linéaire sur des systèmes de chiffrement connus

DES (Data Encryption Standard) En 1994, Matsui a utilisé la cryptanalyse linéaire pour attaquer DES avec une complexité de 2^{47} opérations, ce qui était considéré comme une percée majeure à l'époque.

AES (Advanced Encryption Standard) AES est un algorithme de chiffrement symétrique qui utilise des blocs de 128 bits et une clé de 128, 192 ou 256 bits. En 2002, Biryukov et al. ont utilisé la cryptanalyse linéaire pour attaquer une version réduite de AES avec 6 rounds et une clé de 128 bits.

GOST GOST est un algorithme de chiffrement symétrique développé par le gouvernement russe qui utilise des blocs de 64 bits et une clé de 256 bits. En 2007, Bogdanov et al. ont utilisé la cryptanalyse linéaire pour attaquer GOST avec une complexité de 2^{199} opérations.

RC4 RC4 est un algorithme de chiffrement symétrique utilisé dans de nombreux protocoles de sécurité, tels que SSL et WEP. En 2001, Fluhrer, Mantin et Shamir ont utilisé la cryptanalyse linéaire pour attaquer WEP en exploitant une faiblesse dans l'initialisation de la clé de chiffrement RC4.

4.4.4 Limites de l'attaque linéaire

Bien que la cryptanalyse linéaire soit une technique puissante pour attaquer certains types de chiffrements, elle présente également des limites importantes :

- Elle ne fonctionne que pour les chiffrements symétriques : la cryptanalyse linéaire ne peut être appliquée que pour les chiffrements symétriques, c'est-à-dire les chiffrements qui utilisent la même clé pour le chiffrement et le déchiffrement,
- Elle nécessite souvent un grand nombre de paires de textes clairs et chiffrés pour fonctionner efficacement. Cela peut être difficile à obtenir dans des situations réelles, où les données peuvent être limitées ou coûteuses à collecter,
- la cryptanalyse linéaire peut être sensible aux erreurs dans les mesures des probabilités de différentes entrées et sorties. De petites erreurs dans les estimations de probabilité peuvent entraîner des estimations de biais incorrectes, ce qui peut rendre l'attaque impraticable.
- Il est possible pour les concepteurs de chiffrements de renforcer la sécurité de leur algorithme en ajoutant des mécanismes anti-linéarité pour compliquer l'analyse linéaire. Ces mécanismes peuvent inclure l'ajout de non-linéarités dans le processus de chiffrement, comme l'utilisation de S-boîtes (substitution) ou de fonctions de permutation pour mélanger les bits, ou encore l'ajout de tours supplémentaires pour augmenter la complexité de l'algorithme.

4.4.5 Exemple d'application

Considérons un algorithme de chiffrement très simple qui prend 3 bits en entrée et donne 3 bits chiffrés en sortie. Le processus se déroule sur 3 tours et utilise 4 sous-clés [**?**].

Soit P la donnée en clair de 3 bits et soit le résultat final C chiffré de 3 bits.

Tour 1 Le texte P est chiffré avec la sous-clé K_1 (XOR), on obtient le texte A_1, tel que : $A_1 = P \oplus K_1$,

Le résultat passe dans une table de substitution S_1 : $B_1 = S_1(A_1)$.

Tour 2 Le résultat du premier tour de chiffrement est mélangé avec la sous-clé K_2, puis est substitué par la table S_2 pour obtenir :

$A_2 = B_1 \oplus K_2$,

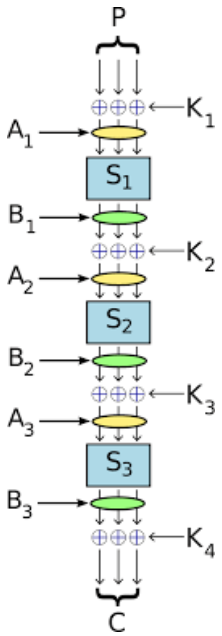
$B_2 = S_2(A_2)$.

Tour 3 Le même processus est appliqué et on obtient :

$A_3 = B_2 \oplus K_3$,

$B_3 = S_3(A_3)$.

Finalisation A la fin un dernier mixage est appliqué avec la sous-clé K_4, c'est-à-dire : $C = B_3 \oplus K_4$.
On suppose avoir les approximations suivantes :



$S_1 : X_1 \oplus X_2 \oplus X_3 = Y_2$.

$S_2 : X_2 = Y_1 \oplus Y_3$.

1^{er} **tour** $A_1 = P \oplus K_1$ donc $A_1 = (A_{1,1} = P_{1,1} \oplus K_{1,1}, A_{1,2} = P_{1,2} \oplus K_{1,2}, A_{1,3} = P_{1,3} \oplus K_{1,3})$,
$B_1 = S_1(A_1) \Rightarrow B_{1,2} = A_{1,1} \oplus A_{1,2} \oplus A_{1,3}$,
$\Rightarrow B_{1,2} = (P_{1,1} \oplus K_{1,1}) \oplus (P_{1,2} \oplus K_{1,2}) \oplus (P_{1,3} \oplus K_{1,3})$. **(I)**

$2^{ème}$ **tour** $A_2 = B_1 \oplus K_2$, donc $A_2 = (A_{2,1} = B_{1,1} \oplus K_{2,1}, A_{2,2} = B_{1,2} \oplus K_{2,1}, A_{2,3} = B_{1,3} \oplus K_{2,3})$.
$B_2 = S_2(A_2)$ et à partir de l'approximation $(X_2 = Y_1 \oplus Y_3)$ donc $(A_{2,2} = B_{2,1} \oplus B_{2,3})$.
$\Rightarrow B_{2,1} \oplus K_{2,3} = B_{1,2} \oplus K_{2,2}$, en remplaçant $B_{1,2}$ par sa valeur dans (I) on obtient :
$B_{2,1} \oplus B_{2,3} = ((P_{1,1} \oplus K_{1,1}) \oplus (P_{1,2} \oplus K_{1,2}) \oplus (P_{1,3} \oplus K_{1,3})) \oplus K_{2,2}$. **(II)**

$3^{ème}$ **tour** $A_3 = B_2 \oplus K_3$ donc $A_3 = (A_{3,1} = B_{2,1} \oplus K_{3,1}, A_{3,2} = B_{2,2} \oplus K_{3,2}, A_{3,3} = B_{2,3} \oplus K_{3,3})$
$\Rightarrow (B_{2,1} = A_{3,1} \oplus K_{3,1}$ et $B_{2,3} = A_{3,3} \oplus K_{3,3}) \Rightarrow B_{2,1} \oplus B_{2,3} = (A_{3,1} \oplus K_{3,1}) \oplus (A_{3,3} \oplus K_{3,3})$.
On remplace $B_{2,1} \oplus B_{2,3}$ par sa valeur dans (II) on obtient :
$(A_{3,1} \oplus K_{3,1}) \oplus (A_{3,3} \oplus K_{3,3}) = ((P_{1,1} \oplus K_{1,1}) \oplus (P_{1,2} \oplus K_{1,2}) \oplus (P_{1,3} \oplus K_{1,3})) \oplus K_{2,2}$.
En rassemblant les différents termes, nous obtenons finalement l'équation suivante :
$(K_{1,1} \oplus K_{1,2} \oplus K_{1,3} \oplus K_{2,2} \oplus K_{3,1} \oplus K_{3,3}) \oplus (P_{1,1} \oplus P_{1,2} \oplus P_{1,3}) \oplus (A_{3,1} \oplus A_{3,3}) = 0$.**(III)**
En définissant $\sum K$ comme étant égal à la somme suivante : $(K_{1,1} \oplus K_{1,2} \oplus K_{1,3} \oplus K_{2,2} \oplus K_{3,1} \oplus K_{3,3})$, nous obtenons alors le résultat suivant :
$\sum K \oplus (P_{1,1} \oplus P_{1,2} \oplus P_{1,3}) \oplus (A_{3,1} \oplus A_{3,3}) = 0$.**(IV)**
En utilisant le lemme du Piling-Up de Matsui, nous pouvons déterminer la probabilité qu'une approximation soit valide en fixant la valeur de $\sum K$ à 0 ou à 1. Cette approximation dépend d'une partie des trois clés intermédiaires, du texte en clair et d'une partie de l'entrée de la dernière table de substitution.

Récupération des clés Dans cette analyse, nous disposons d'une approximation des trois premiers tours de l'algorithme de chiffrement, mais la clé du dernier tour, K_4, est manquante. Nous utilisons les messages chiffrés pour tenter de retrouver cette clé en essayant toutes les valeurs possibles de la sous-clé K_4, ce qui correspond à une attaque par force brute sur K_4.
En calculant $C \oplus K_4$, nous obtenons en fait la sortie de la troisième table de substitution, qui correspond à B_3 (car $C = B_3 \oplus K_4$ implique que $B_3 = C \oplus K_4$).
Ensuite, nous calculons la substitution inverse de S_3, c'est-à-dire $S_3^{-1}(C \oplus K_4)$. Cette valeur correspond à A_3, ce qui signifie que $A_3 = S_3^{-1}(C \oplus K_4)$. Nous pouvons alors estimer la validité des clés testées en comparant la valeur exacte retournée par la substitution inverse avec l'approximation linéaire sur tout ou une partie des bits.
En utilisant un grand nombre de paires de messages, nous pouvons affiner les estimations et les rendre plus précises.
Pour découvrir les autres clés intermédiaires, nous continuons l'attaque en remontant progressivement dans les tours de chiffrement jusqu'à arriver à la première sous-clé.

Lemme d'empilement Le lemme d'empilement est un résultat statistique introduit par Mitsuru Matsui en 1993 dans le cadre de la cryptanalyse linéaire. Ce lemme permet de quantifier le biais linéaire présent dans une approximation linéaire d'un algorithme de chiffrement symétrique par bloc.

Formulation mathématique Dans le contexte de la cryptanalyse linéaire, une équation linéaire est formulée sous la forme d'un XOR (ou-exclusif) de variables binaires. Considérons N variables binaires aléatoires et indépendantes $X_1, X_2, \ldots, X_N$. La probabilité que l'équation linéaire formée par ces variables soit vraie est définie par :

$$P(X_1 \oplus X_2 \oplus \cdots \oplus X_N = 0) = \tfrac{1}{2} + 2^{N-1} \prod_{i=1}^{N} \varepsilon_i,$$

Avec ε_i est le biais linéaire de la variable aléatoire X_i.

Raisonnement Dans le contexte du lemme d'empilement, les variables utilisées sont des variables aléatoires binaires et indépendantes, notées $X_1, X_2, \ldots, X_N$.
Soit $P(X_i = 0)$, la probabilité que la variable binaire X_i soit égale à 0. Cette probabilité soit égale à 1 si $X_i = 0$ et égale à 0 si $X_i = 1$.
Le lemme d'empilement peut être appliqué pour deux variables aléatoires de la façon suivante :

$$P(X_1 = i) = \begin{cases} p_1 & , si\ i = 0, \\ 1 - p_1 & , si\ i = 1. \end{cases}$$

$$P(X_2 = j) = \begin{cases} p_2 & , si\ j = 0, \\ 1 - p_2 & , si\ j = 1. \end{cases}$$

Soit l'équation $X_1 \oplus X_2 = 0$, grâce aux propriétés du XOR on a : $P(X_1 \oplus X_2 = 0) \iff P(X_1 = X_2)$.
Les événements "$X_1 = X_2 = 0$" et "$X_1 = X_2 = 1$" sont mutuellement exclus (ils ne peuvent pas se produire simultanément), par conséquent :

$$\begin{aligned} P(X_1 = X_2) &= P(X_1 = X_2 = 0) + P(X_1 = X_2 = 1) \\ &= P(X_1 = 0) \times P(X_2 = 0) + P(X_1 = 1) \times P(X_2 = 1) \\ &= p_1 p_2 + (1 - p_1)(1 - p_2) = p_1 p_2 + 1 - p_1 - p_2 + p_1 p_2. \end{aligned}$$

$P(X_1 \oplus X_2 = 0) = 2p_1 p_2 - p_1 - p_2 + 1$.(V)
Soit $p_1 = \frac{1}{2} + \varepsilon_1$ et $p_2 = \frac{1}{2} + \varepsilon_2$ où ε_1 et ε_2 sont respectivement les biais des probabilités des deux variables X_1 et X_2.
En substituant les valeurs de p_1 et p_2 dans la formule (V), on peut quantifier le degré de déviation de la probabilité par rapport à $\frac{1}{2}$:
$P(X_1 \oplus X_2 = 0) = 2(\frac{1}{2} + \varepsilon_1)(\frac{1}{2} + \varepsilon_2) - (\frac{1}{2} + \varepsilon_1) - (\frac{1}{2} + \varepsilon_2) + 1$
$= 2(\frac{1}{4} + \frac{1}{2}\varepsilon_1 + \frac{1}{2}\varepsilon_2 + \varepsilon_1\varepsilon_2) - \frac{1}{2} - \varepsilon_1 - \frac{1}{2} - \varepsilon_2 + 1$
$= \frac{1}{2} + \varepsilon_1 + \varepsilon_2 + 2\varepsilon_1\varepsilon_2 - \varepsilon_1 - \varepsilon_2 - \frac{1}{2} - \frac{1}{2} + 1$
$= \frac{1}{2} + 2\varepsilon_1\varepsilon_2$.

La formule précédente $(X_1 \oplus X_2 = 0)$ possède un biais linéaire, qui est égal à $2\varepsilon_1\varepsilon_2$.
En généralisant la formule pour N variables aléatoires binaires et indépendantes on obtient :

$$P(X_1 \oplus X_2 \oplus \cdots \oplus X_N = 0) = \frac{1}{2} + 2^{N-1} \prod_{i=1}^{N} \varepsilon_i.$$

Si un seul biais linéaire ε_i est égal à 0 (une variable X_i est non biaisée), la probabilité de toute la formule est égale à $\frac{1}{2}$ (toute la formule sera non biaisée).

Remarque 4.1. Le biais linéaire peut être positif ou négatif,il permet de mesurer l'écart entre la distribution de probabilité d'une variable aléatoire binaire et une distribution uniforme dans laquelle l'espérance est de $\frac{1}{2}$. Lorsque le biais est important, l'algorithme de chiffrement peut être facilement compromis.

4.5 Cryptanalyse différentielle

En 1990, Biham and Shamir ont présenté la cryptanalyse différentielle. Cette technique consiste à utiliser des messages clairs choisis pour attaquer les constructions itératives de chiffrement par bloc. Le principe de base de la cryptanalyse différentielle implique l'utilisation de deux messages clairs différents, notés m et m^*, pour créer une somme appelée différence, représentée par $\alpha = m \oplus m^*$. On analyse ensuite comment cette différence évolue au fil des tours de chiffrement, en se concentrant sur ce qui est appelé le "chemin différentiel".

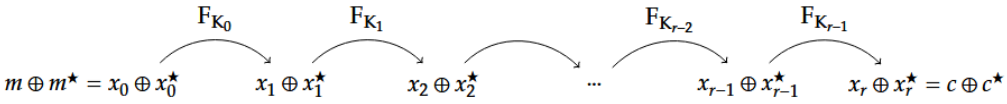


FIGURE 4.2 – Le chemin différentiel

Si la fonction de tour était linéaire, chaque tour produirait toujours la même sortie pour une différence donnée entre l'entrée et la clé secrète, indépendamment des valeurs de l'entrée et de la clé secrète elles-mêmes (c'est-à-dire : $x_{i+1} \oplus x^*_{i+1} = F_{Ki}(x_i) \oplus F_{Ki}(x^*_i) = F_{Ki}(x_i \oplus x^*_i)$). Cela signifierait, que pour tout couple de données en entrée m et m^* qui ont une différence α, le chemin différentiel suivi pour crypter ces données serait toujours le même : $\alpha, F_{K0}(\alpha), F_{K1}(F_{K0}(\alpha))$, et ainsi de suite.
Cependant, comme la fonction de tour n'est pas linéaire, le chemin différentiel suivi ne sera pas toujours le même avec une probabilité de 1 pour toutes les paires de messages m et m^* qui satisfont une différence donnée, mais il peut varier.
Le focus est mis spécifiquement sur la probabilité :

$$P_{\alpha,\beta} = P(x_{r-1} \oplus x^*_{r-1} = \beta \,|m \oplus m^* = \alpha).$$

Supposant que l'on ait identifié une paire de différences (α, β) pour lesquelles la probabilité $P(\alpha, \beta)$ est significativement plus élevée que l'uniforme ($\frac{1}{2^n}$ où n est la taille des blocs), on peut utiliser cette

information pour évaluer des candidats potentiels pour la clé K_{r-1} du dernier tour, tout comme dans le processus de cryptanalyse linéaire.
Dans le cadre d'une attaque à clairs choisis, un grand nombre de paires de messages chiffrés (m, c) et (m^*, c^*) sont récupérées, tels que la différence entre les messages est $m \oplus m^* = \alpha$. Pour chacune de ces paires de messages, la différence à l'entrée du dernier tour, $x_{r-1} \oplus x^*_{r-1}$, doit être égale à β avec une probabilité $P_{\alpha,\beta}$. Nous devons ensuite remonter le dernier tour en calculant toutes les valeurs possibles de la clé de dernier tour K_{r-1} pour chaque paire de messages (m, c) et (m^*, c^*).

$$Z_K = F_K^{-1}(c) \text{ et } Z_K^* = F_K^{-1}(c^*).$$

Si $Z_K \oplus Z_K^* = \beta$, alors nous augmentons le compteur associé à la clé K. Pour la clé "correcte", $K = K_{r-1}$, le compteur doit être incrémenté avec une probabilité $P_{\alpha,\beta}$. En revanche, pour les clés incorrectes, les valeurs Z_K et Z_K^* sont indépendantes des messages m et m^*, et nous nous attendons à avoir $Z_K \oplus Z_K^* = \beta$ avec une probabilité de $\frac{1}{2^n}$ [6].

4.5.1 Exemple d'application sur un chiffre de type SPN

La cryptanalyse différentielle est une technique qui exploite la forte probabilité d'apparition de certaines différences entre des messages chiffrés résultant de différences spécifiques entre les messages en clair [9].
Dans un chiffrement parfait, la probabilité qu'une différence donnée $\triangle X$ en entrée produise une certaine différence $\triangle Y$ en sortie est de $\frac{1}{2^n}$, où n est le nombre de bits dans l'entrée X. Cependant, cette perfection est rarement atteinte dans les chiffrements existants. En effet, l'analyse des composants de chaque chiffrement révèle souvent des caractéristiques différentielles, qui sont utilisées pour mener des attaques.

Caractéristiques différentielles des S-boîtes : les S-boîtes utilisées dans ce chiffrement sont des boîtes de 4×4 bits. Cela signifie qu'elles prennent une entrée codée sur 4 bits et produisent en sortie un code également codé sur 4 bits.
Supposons que nous avons deux entrées de 4 bits : $X' = [X'_1, X'_2, X'_3, X'_4]$ et $X'' = [X''_1, X''_2, X''_3, X''_4]$. La différence en entrée entre les deux est calculée comme suit : $X = X' \oplus X''$ où $\oplus$ représente l'opération de XOR. De même, supposons que $Y' = [Y'_1, Y'_2, Y'_3, Y'_4]$ et $Y'' = [Y''_1, Y''_2 Y''_3, Y''_4]$ sont les sorties correspondantes produites par une boîte-S après avoir chiffré X' et X'' respectivement. Dans ce cas, la différence de sortie est calculée comme suit : $\triangle Y = Y' \oplus Y''$.
Le tableau ci-dessous présente toutes les valeurs possibles de $\triangle Y$ qui peuvent apparaître pour toutes les valeurs de X, en relation avec les valeurs de $\triangle X = 1011$ (B en hexadécimal), $\triangle X = 1000$ (8 en hexadécimal) et $\triangle X = 0100$ (4 en hexadécimal).
On peut calculer les valeurs de $\triangle Y$ en utilisant :
$Y' = S(X'), Y'' = S(Y'')$, et $\triangle Y = Y' \oplus Y''$ ou bien par : $\triangle Y = S(X) \oplus S(X \oplus \triangle X)$.

X	Y	ΔY		
		$\Delta X = 1011$	$\Delta X = 1000$	$\Delta X = 0100$
0000	1110	0010	1101	1100
0001	0100	0010	1110	1011
0010	1101	0111	0101	0110
0011	0001	0010	1011	1001
0100	0010	0101	0111	1100
0101	1111	1111	0110	1011
0110	1011	0010	1011	0110
0111	1000	1101	1111	1001
1000	0011	0010	1101	0110
1001	1010	0111	1110	0011
1010	0110	0010	0101	0110
1011	1100	0010	1011	1011
1100	0101	1101	0111	0110
1101	1001	0010	0110	0011
1110	0000	1111	1011	0110
1111	0111	0101	1111	1011

FIGURE 4.3 – Exemples de paires de différences de la boîte S

Ainsi, on peut observer les probabilités suivantes pour les valeurs de $\triangle Y$ en fonction des différentes valeurs de $\triangle X$:

La probabilité de $P(\triangle Y = 0010, \triangle X = 1011)$ est de $\frac{8}{16}$.

La probabilité de $P(\triangle Y = 1011, \triangle X = 1000)$ est de $\frac{4}{16}$.

La probabilité de $P(\triangle Y = 0110, \triangle X = 0100)$ est de $\frac{6}{16}$.

La probabilité de $P(\triangle Y = 1011, \triangle X = 0100)$ est de $\frac{4}{16}$.

Pour réaliser une analyse complète pour toutes les valeurs possibles de $\triangle X$ (de $\triangle X = 0000$ à $\triangle X = 1111$), il est nécessaire de dresser un tableau présentant toutes les distributions possibles de $\triangle Y$ en fonction de $\triangle X$.

Output / Input	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	2	0	0	0	2	0	2	4	0	4	2	0	0
2	0	0	0	2	0	6	2	2	0	2	0	0	0	0	2	0
3	0	0	2	0	2	0	0	0	0	4	2	0	2	0	0	4
4	0	0	0	2	0	0	6	0	0	2	0	4	2	0	0	0
5	0	4	0	0	0	2	2	0	0	0	4	0	2	0	0	2
6	0	0	0	4	0	4	0	0	0	0	0	0	2	2	2	2
7	0	0	2	2	2	0	2	0	0	2	2	0	0	0	0	4
8	0	0	0	0	0	0	2	2	0	0	0	4	0	4	2	2
9	0	2	0	0	2	0	0	4	2	0	2	2	2	0	0	0
A	0	2	2	0	0	0	0	0	6	0	0	2	0	0	4	0
B	0	0	8	0	0	2	0	2	0	0	0	0	0	2	0	2
C	0	2	0	0	2	2	2	0	0	0	0	2	0	6	0	0
D	0	4	0	0	0	0	0	4	2	0	2	0	2	0	2	0
E	0	0	2	4	2	0	0	0	6	0	0	0	0	0	2	0
F	0	2	0	0	6	0	0	0	0	4	0	2	0	0	2	0

TABLE 4.1 – Tableau de distribution des différences

Le tableau présenté dispose les différentes valeurs possibles de $\triangle X$ sur les lignes et les différentes valeurs possibles de $\triangle Y$ (exprimées en hexadécimal) sur les colonnes. Chaque case du tableau

contient le nombre d'occurrences de la valeur de $\triangle Y$ pour une valeur donnée de $\triangle X$.
En utilisant ce tableau, il est possible de choisir les bonnes S-boîtes à combiner pour obtenir une probabilité élevée d'occurrence de la valeur souhaitée pour $\triangle Y$ à la fin du dernier tour de chiffrement.
Il convient de noter que cette analyse porte sur les entrées et sorties des S-boîtes. Toutefois, les entrées des S-boîtes au premier tour ne correspondent pas exactement aux valeurs du texte en clair, car elles ont subi un mélange avec la sous-clé K_1. Heureusement, en étudiant les différences, on peut conclure que cette différence n'a pas d'importance pour la suite de l'analyse.
Supposons que W' et W'' soient deux textes en clair de 4 bits qui sont entrés dans une S-boîtes. Les résultats correspondants après le mixage sont X' et X''. On a alors $X' = W' \oplus K_1$ et $X'' = W'' \oplus K_1$.
Par conséquent, la différence entre X' et X'' peut être calculée comme suit :
$\triangle X = X' \oplus X'' = (W' \oplus K_1) \oplus (W'' \oplus K_1) = W' \oplus W'' \oplus K_1 \oplus K_1 = W' \oplus W''$.

Choix des S-boîtes : pour élaborer le parcours nécessaire, on ne prendra en compte que les valeurs non nulles qui ont été obtenues lors du calcul des différences.
Considérons P comme étant un bloc de texte en clair de 16 bits, et C_1, C_2, C_3, et C_4 comme étant les résultats des tours 1, 2, 3, et 4 respectivement. Pour le premier tour, nous avons sélectionné la S-Boîte S_{12} avec une différence $\triangle X = 1101$, ce qui nous a permis d'obtenir une valeur de $\triangle Y = 0010$, avec une probabilité de $\frac{8}{16}$.
Ainsi, en partant de la différence $\triangle P = [0000101100000000]$, nous avons pu obtenir la différence $\triangle C = [0000001000000000]$ au niveau 1, avec une probabilité de $\frac{8}{16}$.
En sélectionnant la S-Boîte S_{23} au deuxième tour avec la différence $\triangle X = 0100$, nous avons pu obtenir le résultat $\triangle Y = 0110$ avec une probabilité de $\frac{6}{16}$.
Ainsi, en partant de la différence $\triangle P = [0000101100000000]$, nous avons pu obtenir la différence $\triangle C = [0000000001100000]$ au niveau 2 avec une probabilité de $(\frac{8}{16}) \times (\frac{6}{16}) = \frac{48}{256} = \frac{3}{16}$.
Ce résultat conduit à choisir les S-boîtes S_{32} et S_{33} au niveau 3 avec une entrée $\triangle X = 0010$ pour chacune. Pour chaque S-Boîte, le résultat pour $\triangle Y$ est $\triangle Y = 0101$ avec une probabilité de $\frac{6}{16}$.
Ainsi, en partant de $\triangle P = [0000101100000000]$, en suivant les chemins de S-boîtes et de différences non nulles, on obtient$\triangle C = [0000010101010000]$ au niveau 3, avec une probabilité de $(\frac{3}{16}) \times (\frac{6}{16}) \times (\frac{6}{16}) = \frac{27}{1024}$.
Au dernier tour, les S-boîtes S_{42} et S_{44} sont sélectionnées avec une entrée $\triangle X = 0110$ pour chacune. Cette combinaison donne deux résultats possibles : $\triangle Y = 0011$ et $\triangle Y = 0101$, chacun ayant une probabilité de $\frac{4}{16}$.
On observe que les bits de la dernière sous-clé que l'on peut potentiellement déduire sont ceux correspondant aux positions $K_{5,5}$ à $K_{5,8}$ et $K_{5,13}$ à $K_{5,16}$.
Au lieu de chercher à exploiter les nouvelles caractéristiques différentielles des S-boîtes S_{42} et S_{44} qui nous conduisent à deux résultats pour chacune, chacun ayant la même probabilité, il est préférable d'attaquer cette étape dans l'ordre inverse. Nous pouvons nous baser sur les résultats de chiffrement des textes obtenus jusqu'à présent et procéder à une recherche exhaustive (force brute) des valeurs

des bits des sous-clés $K_{5,5}$ à $K_{5,8}$ et $K_{5,13}$ à $K_{5,16}$.

Conclusion

La cryptanalyse représente un domaine essentiel pour évaluer et briser les systèmes de chiffrement. Les différentes techniques abordées dans ce chapitre, telles que l'analyse statistique et les méthodes avancées, offrent des approches variées pour décrypter les informations chiffrées.

CHAPITRE 5

APPLICATION DE QUELQUES MÉTHODES DE CRYPTANALYSE

Introduction

Dans ce chapitre, nous mettrons en pratique les concepts théoriques de la cryptanalyse en utilisant le logiciel MATLAB R2015a. Nous explorerons différentes méthodes de cryptanalyse, telles que l'analyse de fréquence, l'attaque par force brute, et les appliquerons à des exemples concrets de chiffrements.

5.1 Utilité de la programmation

Voici quelques raisons pour lesquelles on ne peut pas simplement utiliser des méthodes manuelles pour effectuer des cryptanalyses et pourquoi la programmation et l'utilisation de machines sont nécessaires :

- **Complexité des calculs :** la cryptanalyse peut impliquer des calculs mathématiques complexes, tels que la factorisation de grands nombres premiers, la recherche de collisions dans des fonctions de hachage, ou l'application d'algorithmes de recherche d'espaces de clés. Ces calculs sont souvent extrêmement longs et fastidieux à réaliser à la main, mais peuvent être exécutés rapidement par des ordinateurs,
- **Vitesse de traitement :** les systèmes cryptographiques modernes utilisent des clés de chiffrement très longues et des algorithmes robustes, ce qui les rend extrêmement résistants aux attaques manuelles. Les ordinateurs, grâce à leur vitesse de traitement élevée, peuvent effectuer un grand nombre d'opérations en un temps très court, ce qui accélère considérablement le processus de cryptanalyse,
- **Analyse de grandes quantités de données :** dans certains types d'attaques cryptographiques, il est nécessaire d'analyser de grandes quantités de données chiffrées pour détecter des modèles ou des faiblesses. Les machines sont bien plus efficaces que les humains pour traiter de telles

quantités de données rapidement et de manière systématique.

- **Automatisation des attaques :** la programmation permet de développer des scripts et des programmes qui automatisent les attaques cryptographiques, ce qui permet de les exécuter de manière répétée et contrôlée. Cela facilite la recherche de vulnérabilités et accélère le processus de décryptage.

En somme, la cryptanalyse nécessite l'utilisation de machines et la programmation en raison de la complexité des calculs, de la vitesse de traitement requise, de l'analyse de grandes quantités de données, etc. Les outils et les langages de programmation permettent aux cryptanalystes d'exploiter pleinement les capacités des machines pour résoudre des problèmes cryptographiques complexes [12].

5.2 Programmation avec MATLAB

MATLAB est un logiciel de calcul numérique très puissant et polyvalent largement utilisé dans de nombreux domaines, y compris la cryptographie. Voici quelques raisons pour lesquelles MATLAB est utilisé dans la cryptographie :

- **Bibliothèques de fonctions :** MATLAB dispose de nombreuses bibliothèques de fonctions pré-construites qui facilitent la mise en œuvre de différents algorithmes de cryptographie. Ces bibliothèques comprennent des fonctions pour la génération de nombres aléatoires, le chiffrement et le déchiffrement, les opérations sur les polynômes, etc. Cela permet aux développeurs de cryptographie de gagner du temps en utilisant des fonctions éprouvées plutôt que de créer chaque algorithme à partir de zéro.
- **Calcul matriciel :** MATLAB est connu pour sa puissance en matière de calcul matriciel. Dans la cryptographie, de nombreux algorithmes sont basés sur des opérations matricielles, tels que les calculs de multiplication modulaire, les transformations linéaires, les opérations sur les vecteurs, etc. MATLAB facilite l'implémentation de ces opérations complexes et permet d'optimiser les performances en utilisant des calculs matriciels efficaces.

Il convient de noter que bien que MATLAB soit utilisé dans la cryptographie, il existe également d'autres langages et bibliothèques de programmation populaires pour la cryptographie, tels que Python avec la bibliothèque Cryptography, C/C++ avec OpenSSL, Java avec Bouncy Castle, etc. Le choix de l'outil dépend souvent des préférences personnelles, de la disponibilité des bibliothèques et des performances requises pour une application spécifique [12].

5.3 Attaque par analyse de fréquence

Nous allons prendre une lettre écrite par Gandhi et l'encrypter en utilisant une table de substitution quelconque. Ensuite, nous allons essayer de la décrypter pour la rendre lisible.

"Le symbole de la non-violence d'un côté, le visage de l'horreur absolue de l'autre et, pourtant, l'un a tenté d'entrer en contact avec l'autre. Le but ? Empêcher de faire entrer le monde dans une guerre. La guerre se faisant de plus en plus proche, Gandhi décide de prendre la plume, le 23 juillet 1939, depuis Wardha, pour adresser ces quelques mots au Führer."
Soit le passage suivant :

"Mon cher ami,

Des amis m'ont encouragé à vous écrire pour le bien de l'humanité. J'ai résisté à leur requête, pensant qu'une lettre de ma part serait une impertinence. Mais quelque chose me dit que je ne dois pas faire de calcul et que je dois faire cet appel, quel qu'en soit le prix.
Il est assez clair que vous êtes aujourd'hui la seule personne au monde qui puisse empêcher le déclenchement d'une guerre pouvant réduire l'humanité à l'état sauvage. Devez-vous payer ce prix pour atteindre votre objectif, aussi précieux vous semble-t-il ? Écouterez-vous l'appel d'un homme qui a délibérément évité la solution de la guerre, non sans un certain succès ?
Je sollicite néanmoins votre pardon au cas où j'aurais commis une erreur en vous écrivant.

Je reste votre ami sincère, MK Gandhi."

Lorsque Gandhi envoie sa seconde lettre au Führer, le 24 décembre 1940, la Seconde Guerre Mondiale est déjà bien entamée. Sur un ton des plus bienveillants, le pacifiste souhaite moins questionner les objectifs d'Hitler que remettre en cause sa méthode.

"Cher ami,

Que je m'adresse à vous en tant qu'ami n'est pas une simple formalité. Je n'ai pas d'ennemi. Depuis ces trente-trois dernières années, mon entreprise dans l'existence a été d'obtenir l'amitié de toute l'humanité en me liant avec le genre humain, quelle que soit sa race, sa couleur ou sa croyance.
J'espère que vous aurez le temps et le souhait de savoir ce qu'une bonne portion de l'humanité, qui a en vue de vivre sous l'influence de cette doctrine d'amitié universelle, pense de votre action. Nous n'avons aucun doute sur votre courage et votre dévotion envers votre patrie, nous ne croyons pas non plus que vous êtes le monstre décrit par vos opposants.
Mais vos écrits et vos déclarations et ceux de vos amis et admirateurs ne laissent aucune place au doute que beaucoup de vos actes sont monstrueux et étrangers à toute dignité humaine, spécialement du point de vue de personnes qui comme moi croient à l'amitié universelle.
Je veux parler de l'humiliation de la Tchécoslovaquie, du viol de la Pologne et de l'invasion du Danemark. Je suis conscient que votre conception de l'existence vous fait considérer ces actes comme vertueux. Mais nous avons été éduqués depuis notre enfance à les considérer comme des actes déshonorant l'humanité. En conséquence, nous ne pouvons pas souhaiter le succès de vos armes."
Pour le chiffrement de cette lettre, nous ne tiendrons pas compte des apostrophes et des accents :

Première lettre : Uhw tzsy pua, osm puam uwhi shtwfypls p cwfm styays vwfy bs xash os bzfuphais. Epa ysmamis p bsfy ysrfsis, vjhmphi rffhs bsiiys os up vpyi msypai fhs auvsyiahshts. Upam rfsbrfs tzwms us oai rfs es hs owai vpm npays os tpbtfb si rfs esowai npays tsi pvvsb, rfsb rfsh mwai bs vyad.
Ab smi pmmsj tbpay rfs cwfm sism pfeafyo zfa bp msfbs vsymwhhs pf uwhos fa vfamms suvstzsy bs ostbphtzsushi ofhs lfsyys vwfcphi ysofays bzfuphais p bsipi mpfcpls.
Oscsj cwfm vpksy ts vyad vwfy piisahoys cwiys wxestian, pfmma vystasfd ewfm msuxbsiab. Stwfisysj cwfm b pvvsb ofh zwuus rfa p osbaxsysushi scais bp mwbfiawh os bp lfsyys, hwh mphm fh tsyipah mfttsm.
Es mwbbatais hsphuwahm cwiys vpyowh pf tpm wf e pfypam twuuam fhs syysfy sh cwfm styacphi. Es ysmis cwiys pua mahtsys, ug lphoza.

Deuxième lettre : Tzsy pua, rfs es upoysmms p cwfm sh iphi rfpua hsmi vpm fhs mauvbs nwyupbais. Es hpa vpm oshhsua. Osvfam tsm iyshis iywam osyhasysm phhssm, umh shiysvyams ophm bsdamiphts psis owxishay bpuaias os iwfis bzfuphais sh us baphi pcst bs lshys zfupahs, rfsbbs rfsh mwai mp ypts, mp twfbsfy wf mp tywkphts esmvsys rfs cwfm pfysj bs isuvm si bs mwfzpai os mpcway ts rffhs xwhhs vwyiawh os bzfuphais, rfap sh cfs os cacys mwfm bahnbfshts os tsiis owtiyahs opuaias fhacsymsbbs, vshms os cwiys ptiawh. Hwfm hpcwhm pftfh owfis mfy cwiys twfypls si cwiys oscwiawh shcsym cwiys vpyias, hwfm hs tywkwhm vpm hwh vbfm rfs cwfm sism bs umhmiys ostyai vpy cwm wvvwmphis.
Upam cwm stxaim si cwm ostbpxpiawhm si tsfd os cwm puam si pouaxpisfxm hs bpammshi pftfhs vbpts pf owfis rfs xspftwfv os cwm ptism mwhi uwhmixsfd si sixphlsxm p iwfis oalhais zfupahs, mvstapbsushi of vwahi os cfs os vsxmwhhsm rfa twuus uwa txwashi p bpuaias fhacsxmsbbs.
Es csfd vpybsy os bzfuabapiawh os bpitzstwmbwcprfas, of cawb os bp vwbwlhs si os bahcpmawh of ophsupyg. Es mfam twhmtashi rfs cwiys twhtsviawh os bsdamiphts cwfm npai twhmaosysy tsm ptism twuus csyifsfd. Upam hwfm pcwhm sis sofrfsm osvpafm hwiys shnphts p bsm twhmaosysy twuus osmptism osmzwhwyphi bzfuphais. Sh twhmsrfshts, hwfmhs vwfcwhm vpm mwfzpaisy bs mfttsm os cwm pyusm.
Voici une fonction en MATLAB qui effectue une analyse de fréquence sur le cryptogramme ci-dessus :

```
function texteDechiffre = analyseDeFrequence(cryptogramme)
cryptogramme = 'Uhw tzsy pua, osm puam u whi shtwfypls p cwfm styays
vwfy bs xash os b zfuphais. Epa ysmamis p bsfy ysrfsis,
vjhmphi rffhs bsiiys os up vpyi msypai fhs auvsyiahshts.
Upam rfsbrfs tzwms us oai rfs es hs owai vpm npays os
tpbtfb si rfs esowai npays tsi pvvsb, rfsb rfsh mwai bs vyad.
Ab smi pmmsj tbpay rfs cwfm sism pfeafyo zfa bp msfbs vsymwhhs
pf uwhos fa vfamms suvstzsy bs ostbphtzsushi ofhs lfsyys vwfcphi
ysofays bzfuphais p bsipi mpfcpls.Oscsj cwfm vpksy ts vyad vwfy
```

```
piisahoys cwiys wxestian, pfmma vystasfd ewfm msuxbsiab. Stwfisysj cwfm
b pvvsb ofh zwuus rfa p osbaxsysushi scais bp mwbfiawh os bp lfsyys,
hwh mphm fh tsyipah mfttsm. Es mwbbatais hsphuwahm cwiys vpyowh
pf tpm wf e pfypam twuuam fhs syysfy sh cwfm styacphi. Es ysmis
cwiys pua mahtsys, ug lphoza. Tzsy pua, rfs es upoysmms p cwfm
sh iphi rfpua hsmi vpm fhs mauvbs nwyupbais. Es hpa vpm oshhsua. Osvfam
tsm iyshis iywam osyhasysm phhssm, umh shiysvyams ophmbsdamiphts psis
owxishay bpuaias os iwfis bzfuphais sh us baphi pcstbs lshys zfupahs,
rfsbbs rfsh mwai mp ypts, mp twfbsfy wf mp tywkphts esmvsys rfs cwfm
pfysj bs isuvm si bs mwfzpai os mpcway ts rffhs xwhhs vwyiawh os
bzfuphais, rfap sh cfs os cacys mwfm bahnbfshts os tsiis owtiyahs
opuaias fhacsymsbbs, vshms os cwiys ptiawh. Hwfm hpcwhm pftfh owfis
mfy cwiys twfypls si cwiys oscwiawh shcsym cwiys vpyias, hwfm hs
tywkwhm vpm hwh vbfm rfs cwfm sism bs umhmiys ostyai vpy cwm
wvvwmphis. Upam cwm stxaim si cwm ostbpxpiawhm si tsfd os cwm
puam si pouaxpisfxm hs bpammshi pftfhs vbpts pf owfis rfs xspftwfv
os cwm ptism mwhi uwhmixsfd si sixphlsxm p iwfis oalhais zfupahs,
mvstapbsushi of vwahi os cfs os vsxmwhhsm rfa twuus uwa txwashi p
bpuaias fhacsxmsbbs. Es csfd vpybsy os bzfuabapiawh os bpitzstwmbwcprfas,
of cawb os bp vwbwlhs si os bahcpmawh of ophsupyg. Es mfam twhmtashi
rfs cwiys twhtsviawh os bsdamiphts cwfm npai twhmaosysy tsm ptism twuus
csyifsfd. Upam hwfm pcwhm sis sofrfsm osvpafm hwiys shnphts p bsm
twhmaosysy twuus osmptism osmzwhwyphi bzfuphais. Sh twhmsrfshts,
hwfmhs vwfcwhm vpm mwfzpaisy bs mfttsm os cwm pyusm.';
freqRef = [0.0711 0.0047 0.0415 0.0356 0.1796 0.0041 0.0059 0.0119 0.0670 0.0077
0.0012 0.0397 0.0332 0.0753 0.0670 0.0267 0.0148 0.0593 0.0782 0.0658 0.0723
0.0279 0.0000 0.0053 0.0018 0.0024];
% Convertir le cryptogramme en lettres majuscules
cryptogramme = upper(cryptogramme);
% Compter les occurrences de chaque lettre dans le cryptogramme
occurrences = zeros(1, 26);
for i = 1 : length(cryptogramme)
if isletter(cryptogramme(i))
index = double(cryptogramme(i)) - 64;  % ASCII 'A' est 65
occurrences(index) = occurrences(index) + 1;
end
end
% Normaliser les frequences observees
```

```
frequencesObs = occurrences / sum(occurrences);
% Trier les frequences observees par ordre decroissant
[~, indices] = sort(frequencesObs, 'descend');
[~, freq_fr] = sort (freqRef, 'descend');
% Associer les lettres les plus frequentes du cryptogramme aux lettres de reference
tableSubstitution = containers.Map;
for i = 1 : 26
lettreCryptogramme = char(indices(i) + 64);  % Convertir en lettre majuscule
lettreReference = char(freq_fr(i) + 64);  % Convertir en lettre majuscule
tableSubstitution(lettreCryptogramme) = lettreReference;
end
% Dechiffrer le cryptogramme en appliquant la table de substitutions
texteDechiffre = '';
for i = 1 : length(cryptogramme)
if isKey(tableSubstitution, cryptogramme(i))
lettreDechiffree = tableSubstitution(cryptogramme(i));
texteDechiffre = [texteDechiffre lettreDechiffree];
else
texteDechiffre = [texteDechiffre cryptogramme(i)];
end
end
end
```

ans = MNT CHER UMI, DES UMIS M TNO ENCTARUXE U VTAS ECRIRE PTAR LE JIEN DE L HAMUNIOE. G UI RESISOE U LEAR REQAEOE, PZNSUNO QA ANE LEOORE DE MU PURO SERUIO ANE IMPEROINENCE. MUIS QAELQAE CHTSE ME DIO QAE GE NE DTIO PUS FUIRE DE CULCAL EO QAE GE DTIO FUIRE CEO UPPEL, QAEL QA EN STIO LE PRIB. IL ESO USSEZ CLUIR QAE VTAS EOES UAGIARD HAI LU SEALE PERSTNNE UA MTNDE QAI PAISSE EMPECHER LE DECLUNCHEMENO DANE XAERRE PTAVUNO REDAIRE LHAMUNIOE U L EOUO SUAVUXE. DEVEZ VTAS PUYER CE PRIB PTAR UOOEINDRE VTORE TJGECOIF, UASSI PRECIEAB GTAS SEMJLE O IL. ECTAOEREZ VTAS L UPPEL D AN HTMME QAI U DELIJEREMENO EVIOE LU STLAOITN DE LU XAERRE, NTN SUNS AN CEROUIN SACCES. GE STLLICIOE NEUNMTINS VTORE PURDTN UA CUS TA G UARUIS CTMMIS ANE ERREAR EN VTAS ECRIVUNO. GE RESOE VTORE UMI SINCERE, MK XUNDHI. CHER UMI, QAE GE M UDRESSE U VTAS EN OUNO QA UMI NESO PUS ANE SIMPLE FTRMULIOE. GE N UI PUS DENNEMI. DEPAIS CES ORENOE ORTIS DERNIERES UNNEES, MSN ENOREPRISE DUNS L EBISOUNCE U EOE D TJOENIR L UMIOIE DE OTAOE L HAMUNIOE EN ME LIUNO UVEC LE XENRE HAMUINE, QAELLE QAEN STIO SU RUCE, SU CTALEAR TA SU CRTYUNCE. G

ESPERE QAE VTAS UAREZ LE OEMPS EO LE STAHUIO DE SUVTIR CE QA ANE JTNNE PTROITN DE L HAMUNIOE, QAI U EN VAE DE VIVRE STAS L INFLAENCE DE CEOOE DTCORINE D UMIOIE ANIVERSELLE, PENSE DE VTORE UCOITN. NTAS N UVTNS UACAN DTAOE SAR VTORE CTARUXE EO VTORE DEVTOITN ENVERS VTORE PUROIE, NTAS NE CRTYTNS PUS NTN PLAS QAE VTAS EOES LE MSNSORE DECRIO PUR VTS TPPTSUNOE. MUIS VTS ECJIOS EO VTS DECLUJUOITNS EO CEAB DE VTS UMIS EO UDMIJUOEAJS NE LUISSENO UACANE PLUCE UA DTAOE QAE JEUACTAP DE VTS UCOES STNO MTNSOJEAB EO EOJUNXEJS U OTAOE DIXNIOE HAMUINE, SPECIULEMENO DA PTINO DE VAE DE PEJSTNNES QAI CTMME MTI CJTIENO U L UMIOIE ANIVEJSELLE. GE VEAB PURLER DE LHAMILIUOITN DE LU OCHECTSLTVUQAIE, DA VITL DE LU PTLTXNE EO DE L INVUSITN DA DUNEMURK. GE SAIS CTNSCIENO QAE VTORE CTNCEPOITN DE LEBISOUNCE VTAS FUIO CTNSIDERER CES UCOES CTMME VEROAEAB. MUIS NTAS UVTNS EOE EDAQAES DEPUIAS NTORE ENFUNCE U LES CTNSIDERER CTMME DESUCOES DESHTNTRUNO L HAMUNIOE. EN CTNSEQAENCE, NTASNE PTAVTNS PUS STAHUIOER LE SACCES DE VTS URMES.

Remarque 5.1. Il est important de noter que cette méthode d'attaque sur un texte chiffré avec substitution monoalphabétique est une approche basique et ne garantit pas la déchiffrement complet ou précis. La substitution monoalphabétique peut être plus complexe à déchiffrer lorsqu'elle utilise des techniques de mélange ou de substitution non uniforme.
L'analyse de fréquence repose sur la comparaison des fréquences observées dans le texte chiffré avec les fréquences de référence de la langue. Cependant, les fréquences de référence peuvent varier en fonction du contexte, du style d'écriture ou de la langue utilisée. Par conséquent, il peut être nécessaire d'ajuster les fréquences de référence en fonction du texte spécifique que vous essayez de déchiffrer.
De plus, la taille d'un texte peut avoir un impact significatif sur l'efficacité d'une attaque par analyse de fréquence, plus la taille du texte est grande, plus les fréquences observées seront précises et représentatives. Les petites quantités de texte peuvent présenter des fluctuations importantes dans les fréquences observées, ce qui rend difficile l'estimation précise des fréquences de référence. Ainsi, un texte plus grand offre une base de données plus fiable pour l'analyse de fréquence.

Le tableau ci-dessous présente le nombre d'occurrences de chaque caractère dans le texte chiffré :

A	B	C	D	E	F	G	H	I	J	K	L	M
114	67	46	9	14	121	2	128	113	5	3	10	131
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
7	60	124	0	24	301	70	56	45	110	17	90	20

TABLE 5.1 – Nombre d'occurrences de l'alphabet dans le cryptogramme

Ce tableau présente les associations entre chaque caractère du texte chiffré et la lettre de référence correspondante, basées sur les résultats obtenus à l'aide de la fonction 'analyseDeFrequence' :
Maintenant, nous pouvons poursuivre le processus de déchiffrement manuel du texte.

A	B	C	D	E	F	G	H	I	J	K	L	M
F	D	T	O	S	N	E	Z	X	A	T	B	U
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
H	I	V	R	Y	M	C	P	C	Q	J	K	J

TABLE 5.2 – Table de correspondance primaire

À partir des mots 'UMI', 'MUIS', 'UPPEL', 'USSEZ' et 'CLUIR, nous pouvons déduire que la lettre 'U' est susceptible de représenter la lettre 'A'.

MNT CHER AMI, DES AMIS M TNO ENCTARAXE A VTAS ECRIRE PTAR LE JIEN DE L HAMANIOE. G AI RESISOE A LEAR REQAEOE, PZNSANO QA ANE LEOORE DE MA PARO SERAIO ANE IMPEROINENCE. MAIS QAELQAE CHTSE ME DIO QAE GE NE DTIO PAS FAIRE DE CALCAL EO QAE GE DTIO FAIRE CEO APPEL, QAEL QA EN STIO LE PRIB. IL ESO ASSEZ CLAIR QAE VTAS EOES AAGIARD HAI LA SEALE PERSTNNE AA MTNDE QAI PAISSE EMPECHER LE DECLANCHEMENO DANE XAERRE PTAVANO REDAIRE LHAMANIOE A L EOAO SAAVAXE. DEVEZ VTAS PAYER CE PRIB PTAR AOOEINDRE VTORE TJGECOIF, AASSI PRECIEAB GTAS SEMJLE O IL. ECTAOEREZ VTAS L APPEL D AN HTMME QAI A DELIJEREMENO EVIOE LA STLAOITN DE LA XAERRE, NTN SANS AN CEROAIN SACCES. GE STLLICIOE NEANMTINS VTORE PARDTN AA CAS TA G AARAIS CTMMIS ANE ERREAR EN VTAS ECRIVANO. GE RESOE VTORE AMI SINCERE, MK XANDHI.

On peut conjecturer que la lettre 'A' dans le texte chiffré correspond à la lettre 'U' en regardant les mots 'ANE', 'QAELQAE', 'QAE', 'CALCAL', 'SACCES', 'ERREAR', et 'QAI'. Le texte devient alors :

MNT CHER AMI, DES AMIS M TNO ENCTURAXE A VTUS ECRIRE PTUR LE JIEN DE L HUMANIOE. G AI RESISOE A LEUR REQUEOE, PZNSANO QU UNE LEOORE DE MA PARO SERAIO UNE IMPEROINENCE. MAIS QUELQUE CHTSE ME DIO QUE GE NE DTIO PAS FAIRE DE CALCUL EO QUE GE DTIO FAIRE CEO APPEL, QUEL QU EN STIO LE PRIB. IL ESO ASSEZ CLAIR QUE VTUS EOES AUGIURD HUI LA SEULE PERSTNNE AU MTNDE QUI PUISSE EMPECHER LE DECLANCHEMENO DANE XUERRE PTAVANO REDUIRE L HUMANIOE A L EOAO SAUVAXE. DEVEZ VTUS PAYER CE PRIB PTUR AOOEINDRE VTORE TJGECOIF, AUSSI PRECIEUB GTAS SEMJLE O IL. ECTAOEREZ VTAS L APPEL D AN HTMME QUI A DELIJEREMENO EVIOE LA STLUOITN DE LA XUERRE, NTN SANS UN CEROAIN SUCCES. GE STLLICIOE NEANMTINS VTORE PARDTN Au CAS TA G AURAIS CTMMIS UNE ERREUR EN VTUS ECRIVANO. GE RESOE VTORE AMI SINCERE, MK XANDHI.

L'observation des mots 'EO', 'DECLANCHEMENO', 'VTUS', 'PTUR', 'ESO', 'DIO' suggèrent que la lettre 'O' pourrait représenter 'T'. De plus, 'PERSTNNE', 'MTNDE', 'HTMME' suggèrent que la lettre 'T' pourrait représenter 'O'. En considérant le mot 'PRIB', nous pourrions déduire que la lettre 'B' représente 'X'.

MNT CHER AMI, DES AMIS M ONT ENCOURAXE A VOUS ECRIRE POUR LE BIEN DE L HUMANITE. G AI RESISTE A LEUR REQUETE, PZNSANT QU UNE LETTRE DE MA PART

SERAIT UNE IMPERTINENCE. MAIS QUELQUE CHOSE ME DIT QUE GE NE DOIT PAS FAIRE DE CALCUL ET QUE GE DOIT FAIRE CET APPEL, QUEL QU EN SOIT LE PRIX. IL EST ASSEZ CLAIR QUE VOUS ETES AUGIURD HUI LA SEULE PERSONNE AU MONDE QUI PUISSE EMPECHER LE DECLANCHEMENT D UNE XUERRE POUVANT REDUIRE L HUMANITE A L ETAT SAUVAXE. DEVEZ VOUS PAYER CE PRIX POUR ATTEINDRE VOTRE OJGECTIF, AUSSI PRECIEUX GOUS SEMJLE T IL. ECOUTEREZ VOUS L APPEL D UN HOMME QUI A DELIJEREMENT EVITE LA SOLUTION DE LA XUERRE, NON SANS UN CERTAIN SUCCES. GE SOLLICITE NEANMOINS VOTRE PARDON AU CAS TA G AURAIS COMMIS UNE ERREUR EN VOUS ECRIVANT. GE RESTE VOTRE AMI SINCERE, MK XANDHI.

On continue de la même manière pour le reste du texte et on obtient :

MON CHER AMI, DES AMIS M ONT ENCOURAGE A VOUS ECRIRE POUR LE BIEN DE L HUMANITE. J AI RESISTE A LEUR REQUETE, PENSANT QU UNE LETTRE DE MA PART SERAIT UNE IMPERTINENCE. MAIS QUELQUE CHOSE ME DIT QUE JE NE DOIT PAS FAIRE DE CALCUL ET QUE JE DOIT FAIRE CET APPEL, QUEL QU EN SOIT LE PRIX. IL EST ASSEZ CLAIR QUE VOUS ETES AUJOURD HUI LA SEULE PERSONNE AU MONDE QUI PUISSE EMPECHER LE DECLANCHEMENT D UNE GUERRE POUVANT REDUIRE L HUMANITE A L ETAT SAUVAGE. DEVEZ VOUS PAYER CE PRIX POUR ATTEINDRE VOTRE OJGECTIF, AUSSI PRECIEUX VOUS SEMBLE T IL. ECOUTEREZ VOUS L APPEL D UN HOMME QUI A DELIBEREMENT EVITE LA SOLUTION DE LA GUERRE, NON SANS UN CERTAIN SUCCES. JE SOLLICITE NEANMOINS VOTRE PARDON AU CAS OU J AURAIS COMMIS UNE ERREUR EN VOUS ECRIVANT. JE RESTE VOTRE AMI SINCERE, MK GANDHI.

A	B	C	D	E	F	G	H	I	J	K	L	M
P	X	T	O	S	N	L	Z	A	E	G	B	U
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
H	W	V	R	Y	M	I	F	C	Q	D	K	J

TABLE 5.3 – Table de correspondance finale

5.4 Attaque exhaustive

Pour tenter de décrypter un cryptogramme chiffré par décalage, une approche courante est l'attaque par recherche exhaustive. Cette méthode consiste à tester toutes les clés possibles afin de trouver celle qui donne le message déchiffré le plus significatif. Dans cette section, nous présenterons une fonction en MATLAB qui implémente cette attaque en automatisant le processus de test des clés, et en effectuant une simple inspection visuelle, il est possible de déduire directement le décalage utilisé dans le chiffrement.

```
function texte_decrypte = attackeExhaustive(cryptogramme)
cryptogramme= 'prq fkhu dpl ghv dplv prqw hqfrxudjh d yrxv hfuluh srxu
```

```
oh elhq gh okxpdqlwh, mdl uhvlvwh d ohxu uhtxhwh, shqvdqw
txxqh ohwwuh gh pd sduw vhudlw xqh lpshuwlqhqfh.';
alphabet = 'abcdefghijklmnopqrstuvwxyz';
texte_decrypte = cell(1, 26);
for shift = 1 : 26
texte_decrypte{shift} = '';
for i = 1 : length(cryptogramme)
if isletter(cryptogramme(i))
% Decalage inverse du caractere
decryptedChar = alphabet(mod(find(alphabet == cryptogramme(i))-shift-1, 26) + 1);
texte_decrypte{shift} = [texte_decrypte{shift} decryptedChar];
else
% Conserver les caracteres non alphabetiques
texte_decrypte{shift} = [texte_decrypte{shift} cryptogramme(i)];
end
end
end
% Afficher les resultats
for shift = 1 : 26
fprintf('Cle de decalage %d : %s\n', shift, texte_decrypte{shift});
end
end
```

Conclusion

En conclusion de ce quatrième chapitre axé sur l'application de différentes méthodes de cryptanalyse à l'aide de MATLAB R2015a, nous avons pu expérimenter et évaluer ces techniques sur des exemples concrets. En utilisant les fonctionnalités avancées de MATLAB, nous avons pu mettre en pratique les concepts théoriques et analyser l'efficacité de ces méthodes dans la résolution de problèmes cryptographiques.

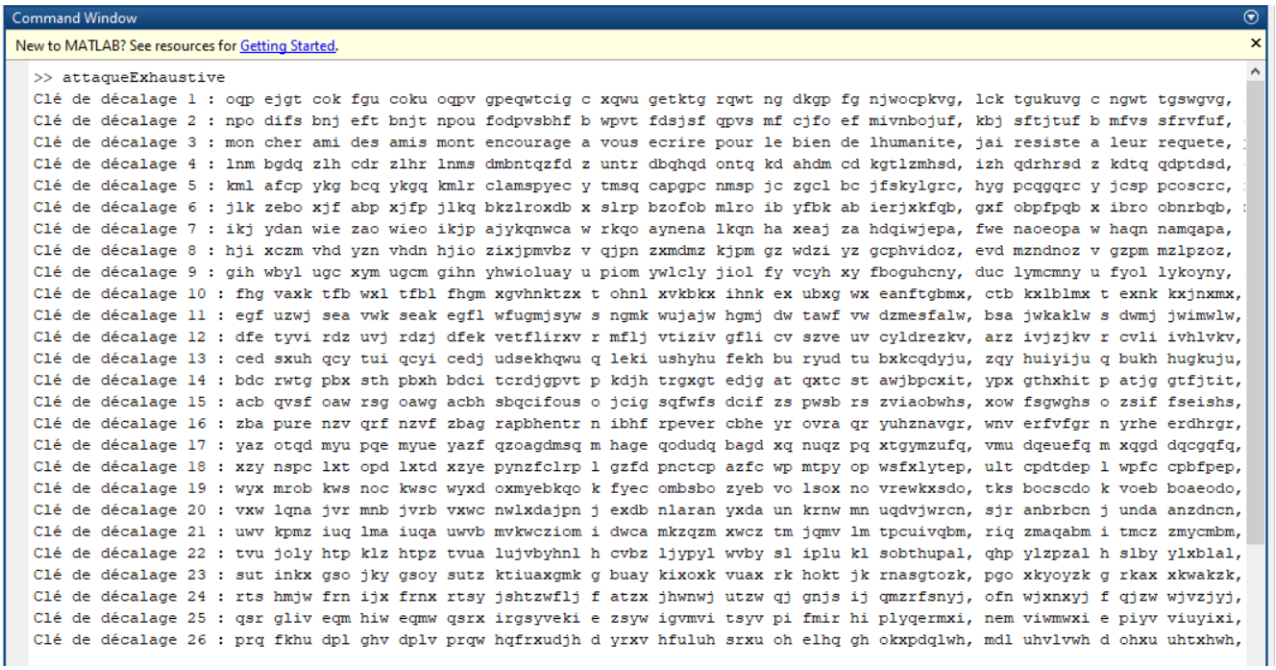


FIGURE 5.1 – Résultats d'exécution

CONCLUSION GÉNÉRALE

En conclusion, ce mémoire a exploré différents aspects de la cryptographie, de la cryptanalyse et de la programmation dans le domaine de la sécurité informatique. Les chapitres ont jeté les bases en abordant les concepts mathématiques fondamentaux, les systèmes de chiffrement basiques et les cryptosystèmes modernes.

La cryptographie continue d'évoluer pour répondre aux défis croissants de sécurité. Bien que la cryptanalyse puisse représenter un risque potentiel, elle joue également un rôle crucial dans l'amélioration des systèmes de chiffrement existants. Cependant, de nouvelles méthodes de chiffrement plus avancées et résistantes sont nécessaires pour faire face aux futures avancées en cryptanalyse.

Il est crucial de prendre des mesures pour assurer la sécurité de demain. La cryptographie quantique offre une perspective prometteuse, car elle repose sur des principes inviolables et offre des solutions plus robustes pour la confidentialité des communications et la protection des données sensibles.

L'importance de la programmation dans le domaine de la cryptographie a été soulignée, avec des exemples concrets d'attaques par analyse de fréquence et par force brute réalisées à l'aide du logiciel MATLAB R2015a. La programmation offre des outils puissants pour analyser, décrypter et renforcer la sécurité des systèmes.

En conclusion, la cryptographie et la cryptanalyse sont des domaines essentiels pour assurer la sécurité des données et des communications à l'ère numérique. La recherche continue, l'innovation et l'utilisation de la programmation sont nécessaires pour faire face aux défis actuels et futurs en matière de sécurité informatique. En gardant à l'esprit l'évolution des technologies et en adoptant des approches avancées telles que la cryptographie quantique, nous pourrons mieux protéger nos données et préserver l'économie mondiale de la dépendance croissante aux technologies numériques.

BIBLIOGRAPHIE

[1] A. Joux, *Algorithmic Cryptanalysis*, CRC Press, 2009.

[2] ADEL M. R., *Attaques cryptanalytiques sur les cryptosystèmes à clé publique*, mémoire de Master 2, Université Larbi Ben M'Hidi de Oum El Bouaghi, 2020.

[3] Biham E., Shamir A., *Differential cryptanalysis of DES-like cryptosystems*, Journal of Cryptology, pp. 3-72 , 1991.

[4] Biham E., Shamir A., *Differential cryptanalysis of full 16-round DES*, Springer, 1998.

[5] CANTEAUT A., *Attaques de cryptosystèmes à mots de poids faible et constructions de fonctions t-résilientes*, thèse de doctorat, Université Paris 6, 1996.

[6] CASTAGNOS G., *Cours de cryptanalyse*, Université de Bordeaux, 2022.

[7] DENEUVILLE J. C., *Contributions à la cryptographie post-quantique*, thèse de doctorat, Université de Limoges, 2016.

[8] Gilles B. M., *Arithmétique et cryptologie*, ellipses, 2e édition, 2021.

[9] Howard M. H., *Linear and differential cryptanalysis*, Electrical and Computer Engineering, Faculty of Engineering and Applied Science, Memorial University of Newfoundland, Canada.

[10] RFIEDMAN W., *Military cryptanalysis Part 1*, National Security Agency Washington 25, D. C, 1952.

[11] LANDRY S., *Étude de la résistance des algorithmes cryptographiques symétriques face à la cryptanalyse moderne*, thèse de doctorat, Université Sorbonne, 2021.

[12] LAXRENCE C. W., *Introduction to cryptography with coding theory*, Prentice Hall, 2e édition, 2006.

[13] Matsui M., *Linear cryptanalysis method for DES*, Cipher Computer and Information Systems Laboratory Mitsubishi Electeric Corporation, 1998.

[14] MEKHAZNIA T., *Analyse cryptographique par les méthodes heuristiques*, mémoire de Master 2, Université de Batna 2, 2017.

[15] National Institute of Standards and Technology, *DES Modes of Operation*, Federal Information Processing Standards Publication 81, NIST, 1980.

[16] National Institute of Standards and Technology, *Data Encryption Standard (DES)*, Federal Information Processing Standards Publication U.S, NIST, 1999.

[17] National Institute of Standards and Technology, *Advanced Encryption Standard (AES)*, Federal Information Processing Standards Publication, NIST, 2001.

[18] P. Paar, J. Pelzl,*Understanding Cryptography : A Textbook for Students and Practitioners*, Springer, 2010.

[19] Welschenbach M., *Cryptography in C and C++*, Apress, 2e édition, 2005.

[20] REZKALLAH L., *De la cryptographie classique à la cryptographie moderne théorie et application*, mémoire de Master 2, Université USTHB, 2007.

[21] R. L. Rivest, A. Shamir, L. Adleman, *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems*, Communications of the ACM, Vol. 21, No. 2, 1978, pp. 120-126.

[22] STALLINGS W., *Cryptography and network security principles and practices*, Prentice Hall, 7e édition, 2005.

[23] W. Trappe, L. C. Washington, *Introduction to Cryptography with Coding Theory*, Pearson, 2006.

[24] VERGNAUD D., *Exercices et problèmes de cryptographie*, Dunod, 2018.

[25] VIDEAU M., *Critères de sécurité des algorithmes de chiffrement à clé secrète*, thèse de doctorat, Université Paris 6, 2005.

[26] Welschenbach M., *Cryptography in C and C++*, Apress, 2e édition, 2005.

[27] WILLIAM J. P., *Intoduction to MATLAB for engineers*, RRDonnelly, 3e édition, 2011.

Résumé :

Dans l'ensemble, ce mémoire vise à fournir une compréhension approfondie de la cryptographie, de la cryptanalyse et de l'utilisation de la programmation pour explorer ces domaines. À l'aide d'exemples concrets, nous montrerons comment la programmation peut être utilisée pour mener des attaques par analyse de fréquence et des attaques par force brute sur des cryptogrammes. Le logiciel MATLAB R2015a sera utilisé comme outil de programmation dans ce mémoire.

Mots clés : Cryptographie, Cryptanalyse, Attaques cryptographiques, Cryptographie classique, Cryptographie moderne, Cryptologie, Cryptanalyse différentielle, Cryptanalyse linéaire.

Abstract :

Overall, this thesis aims to provide an in-depth understanding of cryptography, cryptanalysis, and the use of programming to explore these domains. Using concrete examples, we will demonstrate how programming can be employed to conduct attacks such as frequency analysis and brute-force attacks on cryptograms. The MATLAB R2015a software will be utilized as a programming tool in this thesis.

Keywords : Cryptography, Cryptanalysis, Cryptographic attacks, Classical cryptography, Modern cryptography, Cryptology, Differential cryptanalysis, Linear cryptanalysis.