Cryptographie et Chiffrement Symétrique

Les prémisses de la cryptographie

Introduction

La cryptographie est l'art de protéger les informations en les rendant inaccessibles à toute personne non autorisée. Depuis l'Antiquité, les techniques de cryptographie ont évolué pour répondre aux besoins de protection des communications. Les premières méthodes étaient souvent basées sur des techniques de substitution simples, où les lettres ou symboles d'un message étaient remplacés par d'autres pour masquer le contenu. Avec le temps, ces techniques se sont sophistiquées, menant aux méthodes modernes que nous utilisons aujourd'hui.

Le carré de Polybe

Le carré de Polybe est une méthode de chiffrement ancienne, inventée par le philosophe grec Polybe. Il s'agit d'une technique de substitution où les lettres sont remplacées par une paire de chiffres correspondant à leur position dans une grille 5x5. Cette méthode permet non seulement de chiffrer des messages mais aussi de les transmettre sous forme numérique, ce qui était particulièrement utile dans l'Antiquité pour simplifier la communication à distance.

Exemple :

On utilise une grille 5x5 pour les 25 lettres de l'alphabet (on fusionne généralement I et J).

	1	**2**	**3**	**4**	**5**
1	A	B	C	D	E
2	F	G	H	I/J	K
3	L	M	N	O	P
4	Q	R	S	T	U
5	V	W	X	Y	Z

Pour chiffrer le mot "CHAT", on remplace chaque lettre par les coordonnées correspondantes :

- C -> 13
- H -> 23
- A -> 11
- T -> 44

Le message chiffré est donc : 13 23 11 44.

Le chiffre de César

Le chiffre de César est l'une des techniques de chiffrement les plus simples et les plus célèbres. Utilisé par Jules César, ce chiffre est basé sur le décalage de chaque lettre d'un certain nombre de positions dans l'alphabet. Ce chiffre est un exemple basique de chiffrement par substitution monoalphabétique.

Exemple :

Avec un décalage de 3, la lettre A devient D, B devient E, etc. Pour chiffrer "CHAT" avec un décalage de 3 lettres vers la droite :

- C -> F
- H -> K
- A -> D
- T -> W

Le message chiffré est donc : "FKDW".

Les congruences

Les congruences sont des outils mathématiques utilisés pour modéliser le comportement des nombres sous certaines opérations, en particulier pour les opérations dans un cadre modulaire. En cryptographie, les congruences sont essentielles pour comprendre et implémenter des algorithmes de chiffrement, comme le chiffre de César ou d'autres techniques plus complexes.

Exemple :

Si l'on considère un alphabet de 26 lettres, le chiffre de César peut être formalisé comme suit :
$C = (P + k) \ mod \ 26$
où C est la lettre chiffrée, P la lettre en clair, et k le décalage.

Le chiffrement symétrique

Introduction

Le chiffrement symétrique est une méthode de cryptographie où une même clé est utilisée à la fois pour chiffrer et déchiffrer un message. La sécurité de cette méthode repose sur la confidentialité de la clé : si une personne non autorisée obtient la clé, elle peut déchiffrer le message. Le chiffrement symétrique est généralement plus rapide et moins gourmand en ressources que le chiffrement asymétrique, ce qui en fait un choix populaire pour chiffrer de grandes quantités de données.

Le chiffre de Vigenère

Le chiffre de Vigenère est une méthode de chiffrement par substitution polyalphabétique, ce qui signifie qu'il utilise plusieurs alphabets de substitution pour chiffrer le message. Contrairement au chiffre de César, qui déplace chaque lettre d'un nombre fixe de positions, le chiffre de Vigenère utilise une clé pour déterminer le décalage de chaque lettre du message.

Comment fonctionne le chiffre de Vigenère ?

1. **Choix de la clé** :
   - La clé est un mot ou une phrase, où chaque lettre de la clé indique combien de positions chaque lettre du message doit être décalée.
   - Exemple : Clé = "CLE".
2. **Répétition de la clé** :
   - Si la clé est plus courte que le message, elle est répétée autant de fois que nécessaire pour couvrir tout le message.
   - Exemple : Pour chiffrer "CHAT" avec la clé "CLE", la clé répétée devient "CLEC".
3. **Chiffrement** :
   - Chaque lettre du message est décalée d'un nombre de positions dans l'alphabet correspondant à la valeur de la lettre de la clé.
   - Exemple :
     - Message en clair : "CHAT"
     - Clé répétée : "CLEC"
     - C -> décalé de 2 positions (C) -> E
     - H -> décalé de 11 positions (L) -> S
     - A -> décalé de 4 positions (E) -> E
     - T -> décalé de 2 positions (C) -> V
   - Le message chiffré est donc "ESEV".

Déchiffrement :

Pour déchiffrer le message, on soustrait le décalage indiqué par la clé :

- Exemple : Pour "ESEV" avec la clé "CLEC" :
  - E -> soustrait de 2 positions -> C
  - S -> soustrait de 11 positions -> H
  - E -> soustrait de 4 positions -> A
  - V -> soustrait de 2 positions -> T
- Le message déchiffré est donc "CHAT".

Le chiffre de Vernam

Le chiffre de Vernam, également connu sous le nom de chiffre à masque jetable (One Time Pad), est une méthode de chiffrement extrêmement sécurisée. Il s'agit d'une amélioration du chiffre de Vigenère, où la clé utilisée pour chiffrer le message est aussi longue que le message lui-même et est entièrement aléatoire.

Comment fonctionne le chiffre de Vernam ?

1. **Clé aléatoire** :
   - Une clé aléatoire est générée, de la même longueur que le message à chiffrer.
   - Cette clé est utilisée une seule fois (d'où le nom de "masque jetable").
2. **Chiffrement** :
   - Chaque lettre du message en clair est combinée avec la lettre correspondante de la clé à l'aide de l'opération XOR (ou "OU exclusif"), qui est une opération binaire.
   - Dans le cas de textes, cette opération est réalisée en convertissant chaque lettre en sa représentation binaire (bits), puis en appliquant le XOR bit à bit.
   - Exemple :
     - Message en clair : "CHAT"
     - Clé aléatoire : "XMCK"
     - C -> XOR avec X -> résultat : une nouvelle lettre
     - H -> XOR avec M -> résultat : une nouvelle lettre
     - A -> XOR avec C -> résultat : une nouvelle lettre
     - T -> XOR avec K -> résultat : une nouvelle lettre
   - Le message chiffré est une séquence de lettres apparemment aléatoires.

3. **Déchiffrement** :
    - Le déchiffrement est réalisé de la même manière que le chiffrement : le message chiffré est combiné avec la clé aléatoire à l'aide de l'opération XOR.
    - Étant donné que le XOR d'une valeur avec elle-même donne toujours zéro, et que le XOR d'une valeur avec zéro redonne la valeur initiale, le message original est restauré.

Exemple :

Supposons que nous voulions chiffrer "CHAT" avec la clé aléatoire "XMCK" :

- C -> XOR avec X -> K
- H -> XOR avec M -> F
- A -> XOR avec C -> C
- T -> XOR avec K -> Q
- Le message chiffré devient "KFCQ".

Pour déchiffrer "KFCQ" avec la même clé "XMCK" :

- K -> XOR avec X -> C
- F -> XOR avec M -> H
- C -> XOR avec C -> A
- Q -> XOR avec K -> T
- On retrouve ainsi le message original : "CHAT".

Les opérations sur Z/nZ

Dans le contexte de la cryptographie, les opérations sur Z/nZ (l'anneau des entiers modulo n) jouent un rôle crucial, notamment pour les techniques de chiffrement comme les chiffres de César, de Vigenère, et d'autres algorithmes plus avancés.

Qu'est-ce que Z/nZ ?

Z/nZ représente l'ensemble des classes d'équivalence des entiers modulo n. En termes simples, cela signifie que dans cet ensemble, deux nombres sont considérés comme égaux s'ils ont le même reste lorsqu'ils sont divisés par n. Par exemple, dans Z/nZ (qui est souvent utilisé en cryptographie pour des alphabets de 26 lettres), les nombres 0 et 26 sont équivalents, de même que 1 et 27, et ainsi de suite.

Les opérations dans Z/nZ

1. **Addition** :
    - L'addition dans Z/nZ est définie comme l'addition classique suivie de la prise du modulo n.
    - Formule : (a + b) \mod n\
    - Exemple : Dans Z/nZ, (23 + 5 = 28), et (28 \mod 26 = 2).
2. **Soustraction** :
    - La soustraction suit la même logique que l'addition, avec le modulo n appliqué après la soustraction classique.
    - Formule : (a - b) \mod n\
    - Exemple : (5 - 23 = -18), et (-18 \mod 26 = 8) dans Z/nZ.
3. **Multiplication** :
    - La multiplication dans Z/nZ est définie comme la multiplication classique suivie de la prise du modulo n.
    - Formule : (a \times b) \mod n\
    - Exemple : (7 \times 4 = 28), et (28 \mod 26 = 2) dans Z/nZ.

Application à la cryptographie

Les opérations sur Z/nZ sont fondamentales pour les algorithmes de chiffrement car elles permettent de formaliser le processus de chiffrement et de déchiffrement. Par exemple :

- **Chiffre de César** : Le chiffrement d'une lettre peut être vu comme une addition dans Z/nZ, où chaque lettre est associée à un nombre (A = 0, B = 1, …, Z = 25). Le chiffrement consiste à ajouter un décalage k à la valeur de la lettre et à prendre le résultat modulo 26.
- **Chiffre de Vigenère** : Ici, la clé est une suite de nombres (représentant les lettres) et chaque lettre du message est chiffrée en ajoutant la lettre correspondante de la clé (en termes de valeur numérique) modulo 26.
- **Chiffre de Vernam** : Le chiffre de Vernam peut être vu comme une opération XOR entre les bits du message et les bits de la clé, mais en termes de Z/nZ, il peut être considéré comme une opération d'addition ou de soustraction modulo 2.

Les congruences modulo n sont donc au cœur des mécanismes qui permettent de sécuriser l'information en cryptographie, garantissant que les opérations restent dans un ensemble fermé et évitant ainsi des dépassements ou des erreurs mathématiques dans les calculs.

AES expliqué

L'Advanced Encryption Standard (AES) est un algorithme de chiffrement symétrique adopté comme standard par le gouvernement des États-Unis. Il remplace l'ancien standard DES (Data Encryption Standard). AES fonctionne sur des blocs de données de 128 bits, avec des clés de 128, 192 ou 256 bits. L'algorithme

applique plusieurs transformations sur les données, incluant des substitutions, des permutations, et des opérations mathématiques sur des champs finis, pour assurer un haut niveau de sécurité.

Chiffrer des fichiers en Python avec AES

Pour chiffrer des fichiers en Python avec AES, on peut utiliser la bibliothèque `cryptography`. Voici un exemple de code pour chiffrer un fichier :

```
from cryptography.hazmat.primitives.ciphers import Cipher, algorithms, modes
from cryptography.hazmat.primitives.kdf.pbkdf2 import PBKDF2HMAC
from cryptography.hazmat.backends import default_backend
from cryptography.hazmat.primitives import hashes
from base64 import urlsafe_b64encode
import os

# Génération de la clé à partir d'un mot de passe
password = b"mon_mot_de_passe"
salt = os.urandom(16)
kdf = PBKDF2HMAC(
    algorithm=hashes.SHA256(),
    length=32,
    salt=salt,
    iterations=100000,
    backend=default_backend()
)
key = urlsafe_b64encode(kdf.derive(password))

# Chiffrement des données
cipher = Cipher(algorithms.AES(key), modes.CFB8(salt), backend=default_backend())
encryptor = cipher.encryptor()

# Lecture du fichier à chiffrer
with open('mon_fichier.txt', 'rb') as f:
    fichier_clair = f.read()

fichier_chiffre = encryptor.update(fichier_clair) + encryptor.finalize()

# Écriture du fichier chiffré
with open('fichier_chiffre.txt', 'wb') as f:
    f.write(fichier_chiffre)

print("Fichier chiffré avec succès !")
```

Ce code génère une clé à partir d'un mot de passe, puis utilise cette clé pour chiffrer le contenu d'un fichier en utilisant AES en mode CFB (Cipher Feedback). Le fichier chiffré est ensuite enregistré.