***SAE 3.01 : Compétence 2***

**1. Contexte du projet**

L'Université Sorbonne Paris Nord possède une collection de près de 17 000 jeux de société, certains datant du XIXe siècle. La gestion actuelle de cette collection repose sur un fichier Excel unique, contenant le legs initial, qui est insuffisant pour répondre aux besoins de curation de l'équipe.

**Objectif du projet**

L’objectif est de développer une application web avec une base de données relationnelle et des scripts de nettoyage et d'importation de données pour :

- Faciliter la gestion des informations des jeux.

- Gérer l’administration et une recherche avancée au sein de la collection.

- Gérer des extensions pour des fonctionnalités futures (statistiques, API, etc.).

**Périmètre du projet**

Le projet comprend trois phases :

- Nettoyage et importation des données depuis le fichier Excel.

- Développement de l'application web pour gérer la collection.

- Sécurisation du site web/ Analyse des failles de sécurité

- Documentation pour les utilisateurs et l'administration.

**2. Préoccupation de sécurité :**

La protection des données consiste à mettre en place des mesures de sécurité pour garantir la confidentialité, l'intégrité des données sensibles. L'objectif est d'empêcher les accès non autorisés, les fuites des données.L'un des principaux outils utilisés pour protéger les données est la cryptologie.

La cryptographie utilise des algorithmes et des clés pour transformer les messages clairs et lisibles en un message crypté.Ils existent deux types de chiffrement :

* Le chiffrement asymétrique
* Le chiffrement symétrique

### En ce qui concerne la différences entres le chiffrements asymétriques et symétriques est que le chiffrement symétriques possède :

* Une même clé qui est utilisée pour chiffrer et déchiffrer les données comme utiliser dans le chiffrement affine qui va vous être présentée plus tard.
* Avantage : Rapide et efficace pour des grandes quantités de données.
* Inconvénient : La clé doit être partagée en toute sécurité, ce qui peut poser problème car si la clé est interceptée le message peut être attaqué par une personne malveillante et corrompre la confidentialité des données .

### Pour ce qui s’agit du chiffrement asymétrique :

* Utilise une clé publique pour chiffrer et une clé privée pour déchiffrer ce qui fait une clé de plus que le chiffrement symétrique parmis différentes méthodes de chiffrement je peux vous citer le chiffrement RSA qui utilise cette méthode.
* Avantage : Plus sécurisé, car les deux parties n'ont pas besoin de partager une clé secrète.
* Inconvénient : Plus lent que le chiffrement symétrique

Authentification : nous devons nous assurer que seuls les administrateurs/ utilisateurs autorisés peuvent accéder à certaines fonctionnalités/informations.

Nous voulons avant tout protéger les données sensibles ( mots de passe, informations personnelles etc..) contre les accès non autorisés. Nous souhaitons empêcher qu’un utilisateur malveillant puisse avoir toutes ces informations et donc accéder à notre site de manière illégale.

Concernant la procédure de prospection des failles, premièrement nous identifions les points d’entrées possibles sur le site web, que ce soit formulaire à remplir, page de connexion etc..

Par la suite, nous examinons le code, en recherchant les mauvaises pratiques comme par exemple la non validation des entrées. Par exemple, si un utilisateur veut se créer un compte et qu’il peut avoir comme mot de passe tout type de chose, cela risque d’être compliqué. Son mot de passe doit vérifier des conditions comme par exemple sa longueur ou l’utilisation de caractères spéciaux afin de le sécuriser.

Dans cette continuité , nous vérifions les requêtes SQL, sont-elles paramétrées ? Est-ce que la valeur saisie par l’utilisateur est protégée à l’aide d’un marqueur de place (« : »). Toutes ces vérifications permettent d’assurer la « sécurité du site ».

Comme évoqué précédemment, les solutions apportés sont les marqueurs de place dans les requêtes SQL lorsque les utilisateurs ont à saisir une donnée, la mise en place d’un HTTPS pour chiffrer les communications entre le serveur et les utilisateurs ainsi que la validation des entrées saisies par l’utilisateur par pour le mot de passe lors de la création d’un compte.

**3. Présentation du crypto système affine:**

Le cryptosystème affine est un système de chiffrement symétrique ( une clé pour chiffrer et déchiffrer le message). Il est symétrique car il n’y a aucune distinction entre une clé publique et une clé privée comme RSA (Rivest-Shamir-Adleman) et que toute personne connaissant la clé peut à la fois chiffrer et déchiffrer le message.

Ce cryptosystème repose sur des transformations mathématiques simples(modulo) pour coder et décoder un message. Il correspond au chiffrement de César mais de façon plus développée.

Le cryptosystème affine débute par la transformation de chaque lettre du message à son code ascii. Par exemple A= 0, B = 1 etc…

Par la suite nous allons coder chaque lettre à l’aide d’une fonction linéaire qui est : u

C(x) = (a \* x + b) modulo 26

Avec • C(x) : la lettre codée (sous forme de nombre),

• x : la lettre claire (sous forme de nombre),

• a : le multiplicateur (doit être premier avec 26 pour garantir l’inversibilité),

• b : le décalage ou le terme constant.

ou a et b sont un couple prédéfini par exemple (19,1).

L’opération est Modulo 26 car on est par paquet de 1, on transforme une lettre par une.

Pour déchiffrer une lettre, la fonction inverse est :

D(x) = 1/a (x - b) mod 26

où :

• a^{-1} est l’inverse multiplicatif de a modulo 26.

Le x correspond au code ascii de chaque lettre chiffrée, codée.

Le b correspond au b dans l’algorithme D’Euclide

Alors pour déchiffrer on a besoin de l’inverse de A, du coup nous allons utiliser l’algorithme d’Euclide pour déterminer l’inverse de A. Avant tout cela nous devons déterminer le plus grand diviseur commun entre le modulo (26) et le a (19). Si le pgcd est égal à 1, cela signifie que a est inversible modulo 26. Alors nous pourrons lancer l’algorithme d’Euclide étendu afin de découvrir l’inverse de a qui est le dernier V dans l’algorithme d’Euclide étendu.

Maintenant que nous avons l’inverse de a, nous pouvons déchiffrer chaque lettre du message codé/chiffré afin de le décoder. Dans notre formule de déchiffrement, le x correspond au code ascii de la lettre à décoder, le b correspond au premier nombre de la colonne b dans l’algorithme d’euclide et 1/a à l’inverse de a. par la suite nous faisons le nombre obtenu modulo 26 et ainsi nous retournons sur le code ascii de la lettre déchiffrée. Un exemple d’utilisation de ce cryptosystème sera présenté dans notre vidéo.

**4. Présentation du protocole de chiffrement AES**

Advanced Encryption Standard ou AES est un protocole de chiffrement à cryptographie symétrique. Il est composé de plusieurs étapes :

* ByteSub
* ShiftRow
* MixColumn
* AddKey
* RoundKey.

Ainsi l'algorithme AES prend en entrée un bloc de 128 bits (16 octets) et une clé de 128, 192 ou 256 bits. Les octets sont permutés et placés dans une matrice 4x4. Chaque ligne subit une rotation et une transformation linéaire via une multiplication binaire. Un XOR entre la matrice et une matrice de clé génère une matrice intermédiaire. Et les opérations sont répétées sur plusieurs tours : 10 tours pour une clé de 128 bits, 12 pour 192 bits, et 14 pour 256 bits.

**Shift Row**

**ShiftRow** est une étape du chiffrement AES qui repose sur l'idée de confusion, un concept clé en cryptographie. Elle consiste à réarranger les octets d’un bloc de données en appliquant des décalages circulaires sur les lignes d'une matrice 4x4 appelée "state" (état). Ce processus perturbe les relations locales entre les octets et contribue ainsi à renforcer la sécurité du chiffrement.

1. **Matrice d'état (State Matrix)** : AES manipule un bloc de 128 bits (16 octets), organisé sous la forme d’une matrice de 4 lignes et 4 colonnes, où chaque octet occupe une position. Cela donne un total de 16 octets à traiter.
2. **Décalage circulaire des lignes** : Lors de l'application de **ShiftRow**, chaque ligne de la matrice subit un décalage circulaire :
   * La première ligne (ligne 0) reste inchangée.
   * La deuxième ligne (ligne 1) subit un décalage d'une position vers la gauche.
   * La troisième ligne (ligne 2) subit un décalage de deux positions.
   * La quatrième ligne (ligne 3) subit un décalage de trois positions.
3. Le décalage circulaire signifie que les octets en dehors des limites du côté gauche de la ligne sont re introduits au début.
4. **Formule mathématique** : Le décalage de chaque octet peut être exprimé par la formule suivante : S'\_i,j = S\_i,(j+i) mod 4

L’élément [S\_i,j] est l'octet initial de la i-ème ligne et de la j-ème colonne, et [S’\_i,j] est sa nouvelle position. En changeant l’ordre des bits, on crée de la confusion.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ByteSub**

ByteSub est l'une des étapes fondamentales dans l'algorithme AES. AES est structuré en plusieurs tours, et ByteSub est une des premières transformations appliquées à chaque tour. Elle consiste en une substitution de chaque byte du bloc de données chiffré.

Le rôle principal de ByteSub est d'introduire de la non-linéarité dans l'algorithme. Cela signifie qu'il rend la relation entre les données d'entrée et de sortie plus complexe et imprévisible. Cette complexité est essentielle pour renforcer la sécurité du chiffrement et le rendre plus résistant aux attaques cryptographiques, notamment les attaques différentielles et linéaires.

L'objectif de ByteSub est d'empêcher toute structure simple dans les données chiffrées qui pourrait être exploitée par un attaquant. En perturbant la relation directe entre les bytes d'entrée et de sortie, ByteSub complique considérablement le travail des attaquants essayant de déduire la clé ou les données d'origine.

La transformation ByteSub repose sur une table de substitution appelée **S-box**. Cette table est fixe et contient 256 valeurs pré-calculées. Chaque byte d'entrée est remplacé par un byte de sortie déterminé par cette table. Le S-box est conçu de manière à maximiser la non-linéarité et à minimiser les corrélations entre les données d'entrée et de sortie, augmentant ainsi la sécurité du chiffrement.

Pour vous donner un exemple concret, prenons un byte d'entrée avec la valeur hexadécimale **0x53**. Après l'application du S-box, il pourrait être transformé en une nouvelle valeur, par exemple **0xED**. Chaque byte subit une transformation unique et non linéaire, ce qui contribue à la robustesse globale de l'AES.

#### **Forces de l'AES**

**1. Sécurité Élevée :** L'une des principales forces d'AES est sa sécurité. AES repose sur des principes mathématiques solides et une structure en plusieurs tours qui inclut des étapes telles que ByteSub, ShiftRows, MixColumns, et AddRoundKey. Ces transformations compliquent considérablement les tentatives de décryptage non autorisé. Depuis son adoption en 2001 par le NIST, AES a résisté à de nombreuses tentatives de cassage, prouvant sa robustesse contre les attaques directes.

**2. Flexibilité :** AES est flexible grâce à sa prise en charge de trois tailles de clés : 128, 192, et 256 bits. Cette flexibilité permet aux utilisateurs d'adapter le niveau de sécurité à leurs besoins spécifiques. Par exemple, pour les applications nécessitant une sécurité maximale, une clé de 256 bits peut être utilisée, tandis que des clés plus courtes peuvent offrir un bon compromis entre performance et sécurité.

**3. Performance :** AES est conçu pour être rapide et efficace, que ce soit sur du matériel spécialisé ou sur des systèmes généraux. Il est bien adapté aux applications nécessitant un chiffrement rapide, comme les communications sécurisées, les transactions en ligne, ou encore le stockage de données. AES est également optimisé pour les systèmes embarqués, les smartphones, et les serveurs.

#### **Faiblesses de l'AES**

Passons maintenant aux faiblesses d'AES. Bien que l'algorithme soit robuste, certaines failles potentielles existent.

**1. Vulnérabilités à l'implémentation :** L'algorithme AES en lui-même est sécurisé, mais une mauvaise implémentation peut introduire des vulnérabilités. Par exemple, des erreurs dans la gestion des clés, l'utilisation de générateurs de nombres aléatoires non fiables, ou l'emploi de mauvais modes de chiffrement peuvent compromettre la sécurité. Il est crucial de suivre les meilleures pratiques lors de l'implémentation pour éviter ces erreurs.

**2. Attaques par Canal Auxiliaire :** AES peut être vulnérable aux attaques par canal auxiliaire, qui exploitent des informations secondaires telles que la consommation d'énergie, les variations de température, ou les temps d'exécution pour déduire des informations sur les clés de chiffrement. Ces attaques ne visent pas directement l'algorithme mais tirent parti des faiblesses du système dans lequel il est implémenté.

**3. Dépendance au S-box :** L'étape ByteSub d'AES repose sur la S-box, une table de substitution non linéaire. Bien que cette table soit conçue pour maximiser la sécurité, toute faiblesse ou mauvaise implémentation de celle-ci pourrait être exploitée pour compromettre l'algorithme. Heureusement, la S-box standard d'AES est bien conçue et a résisté à l'analyse cryptographique jusqu'à présent.