Crypto

- A reviser
- Je sais pas trop qu'est ce que je fou ici :) : Je vise 0 pour cette matière
- Cours numéro 2 :
- Nouvelle page

A reviser

Revoir toute les merdes binaires et compagnie
Expliquer ce qu'est un groupe (algèbre)
Revoir le Log2&10
Un logarithme deterministe
3DES => De la merde
Revoir les fonction appliqués dans l'AES + son schéma CBC
LE PADDING UTILE DANS UN CADRE ? jsp wallah
La malléabilitée en cryptographie

Je sais pas trop qu'est ce que je fou ici :) : Je vise 0 pour cette matière

1. Non-répudiation d'un chiffré :

La **non-répudiation** est une propriété d’un système cryptographique qui garantit que l'auteur d'un message ne peut pas nier l’avoir envoyé. Cela est particulièrement important dans les systèmes **asymétriques** (où deux clés distinctes sont utilisées, une pour chiffrer et l'autre pour déchiffrer). Dans ce contexte :

- Un attaquant ne peut pas accéder à la donnée chiffrée, et la personne qui l'a envoyée ne peut pas nier être l'auteur.
- **Valable seulement en asymétrique**, car les clés privées assurent cette non-répudiation.

2. Deux valeurs pour chiffrement et déchiffrement :

- Dans les systèmes asymétriques, **deux clés** sont nécessaires :
  - Une clé publique pour chiffrer.
  - Une clé privée pour déchiffrer.

Dans ce type de système, seule la personne possédant la clé privée peut déchiffrer le message, assurant ainsi la confidentialité.

3. Différence entre encodage et chiffrement :

- **Encodage** : Il s'agit simplement de transformer des données dans un autre format, mais sans objectif de confidentialité ou de sécurité. Tout le monde peut encoder et décoder les données. C'est souvent utilisé pour assurer que des données puissent être correctement traitées par des systèmes (exemples : **Base64**, **UTF-8**, etc.). **N'importe qui peut décoder** un message encodé, car il ne s'agit pas d'un processus de sécurité.
  Exemple : Encodage d'un texte en Base64 ou en UTF-8 pour le rendre compatible avec un autre système.
- **Chiffrement** : Il s'agit de sécuriser les données en les rendant illisibles sans une clé spécifique. Seule une personne ayant la clé appropriée peut déchiffrer le contenu. Le chiffrement a pour objectif de protéger la confidentialité et la sécurité de l'information.
  Exemple : Un fichier chiffré avec **AES** ne peut être déchiffré que par une personne ayant la clé secrète.

4. Binaire et les puissances de 2 :

Ligne	Triangle de Pascal								Somme sur la ligne	
n = 0	1								= 1	$= 2^0$
n = 1	1	+ 1							= 2	$= 2^1$
n = 2	1	+ 2	+ 1						= 4	$= 2^2$
n = 3	1	+ 3	+ 3	+ 1					= 8	$= 2^3$
n = 4	1	+ 4	+ 6	+ 4	+ 1				= 16	$= 2^4$
n = 5	1	+ 5	+ 10	+ 10	+ 5	+ 1			= 32	$= 2^5$
n = 6	1	+ 6	+ 15	+ 20	+ 15	+ 6	+ 1		= 64	$= 2^6$
n = 7	1	+ 7	+ 21	+ 35	+ 35	+ 21	+ 7	+ 1	= 128	$= 2^7$

Le **système binaire** est un système de numération qui utilise **deux chiffres**, 0 et 1, pour représenter des nombres. Il est utilisé en informatique car il correspond directement à la façon dont les ordinateurs fonctionnent avec des états électriques (0 = éteint, 1 = allumé).

Représentation des nombres binaires :

- **Système décimal** : Nous utilisons 10 chiffres (de 0 à 9).
- **Système binaire** : Nous utilisons 2 chiffres (0 et 1).

Chaque **position** dans un nombre binaire représente une **puissance de 2**, comme dans le système décimal où chaque position représente une puissance de 10.

Exemple des puissances de 2 :

$$2^0 = 1 \quad \text{(premier chiffre à droite dans un nombre binaire)}$$

$$2^1 = 2$$

$$2^2 = 4$$

$$2^3 = 8$$

$$2^4 = 16$$

$$2^5 = 32$$

$$2^6 = 64 \quad \text{(et ainsi de suite...)}$$

Exemple de conversion binaire en décimal :

Prenons le nombre binaire **1010** :

- La première position à droite représente $2^0 = 1$
- La deuxième position représente $2^1 = 2$
- La troisième position représente $2^2 = 4$
- La quatrième position représente $2^3 = 8$

Maintenant, décomposons le nombre **1010** :

$$1010 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$$

$$= 1 \times 8 + 0 \times 4 + 1 \times 2 + 0 \times 1$$

$$= 8 + 0 + 2 + 0 = 10 \text{ en base 10 (décimal)}$$

Conversion décimal en binaire :

Pour convertir un nombre décimal en binaire, on divise par 2 et on prend les restes.

Exemple : Convertir **13** en binaire.

1. 13 ÷ 2 = 6 reste 1
2. 6 ÷ 2 = 3 reste 0
3. 3 ÷ 2 = 1 reste 1
4. 1 ÷ 2 = 0 reste 1

Le résultat est **1101** en binaire (en lisant les restes de bas en haut).

Puissance de 2 dans le binaire :

Les puissances de 2 sont très importantes en binaire car chaque bit représente une de ces puissances. Par exemple, le bit tout à gauche dans un nombre binaire de 4 bits représente $2^3 = 8$, le suivant $2^2 = 4$, et ainsi de suite.

En résumé, chaque bit dans un nombre binaire représente une puissance de 2, et la somme des valeurs de ces puissances détermine le nombre en base 10.

5. Algorithme déterministe :

Un **algorithme déterministe** est un algorithme qui, pour un même jeu d'entrées, produit **toujours le même résultat** et suit le même chemin d'exécution à chaque fois. Il n'y a aucune forme de hasard dans ses étapes de calcul : à chaque étape de l'algorithme, l'état suivant est complètement déterminé par l'état actuel.

Caractéristiques d'un algorithme déterministe :

- **Prédictibilité** : On peut prévoir exactement ce que l'algorithme fera pour un ensemble donné d'entrées.
- **Répétabilité** : Il donnera toujours le même résultat pour les mêmes entrées, sans variations.
- **Pas d'aléa** : Il n'y a aucun facteur aléatoire ou probabiliste dans l'exécution.

> **Exemple :** L'algorithme de tri **insertion** est déterministe. Si tu lui donnes la même liste de nombres deux fois, il produira exactement la même liste triée.

Avantages :

- **Prévisibilité** : Idéal dans des systèmes où la fiabilité et la répétabilité sont critiques (comme dans les systèmes bancaires, les systèmes embarqués, etc.).

- **Simplicité de débogage** : Puisqu’il produit toujours les mêmes résultats avec les mêmes entrées, il est plus facile de déboguer et de tester ce type d'algorithme.

5.1. Algorithme probabiliste(ou inverse d'un algorithme déterministe)

Un **algorithme non déterministe** est un algorithme qui peut produire différents résultats pour un même ensemble d'entrées, même si aucune variation dans les données d'entrée n'a lieu. L'algorithme inclut un **élément de hasard** ou de **choix multiple** dans son fonctionnement. Cela signifie qu'il peut explorer plusieurs solutions possibles à partir d'une même situation initiale.

Caractéristiques d'un algorithme probabiliste :

- **Résultats variables** : L'algorithme peut produire des résultats différents pour les mêmes entrées d'une exécution à l'autre.
- **Composante aléatoire ou choix multiples** : L'algorithme peut inclure des étapes où des choix aléatoires ou multiples sont faits.
- **Probabiliste** : Certains algorithmes non déterministes sont basés sur des probabilités, comme ceux utilisés dans la cryptographie ou les algorithmes de Monte Carlo.

> **Exemple :** L'algorithme de tri **Quicksort** avec le choix aléatoire du pivot est un algorithme non déterministe. Selon le pivot choisi aléatoirement à chaque itération, l'exécution peut suivre des chemins différents, bien que le résultat final (la liste triée) soit généralement le même.

Avantages :

- **Efficacité potentielle** : Dans certains cas, les algorithmes non déterministes, comme ceux qui utilisent des choix aléatoires, peuvent être plus efficaces pour explorer de grandes solutions ou pour trouver des résultats approximatifs rapidement.
- **Résolution de problèmes complexes** : Certains problèmes, comme ceux relevant de la classe **NP** (problèmes pour lesquels il est difficile de vérifier une solution), peuvent nécessiter des algorithmes non déterministes pour explorer efficacement l’espace des solutions.

Comparaison :

- **Algorithme déterministe** : Même résultat à chaque exécution avec les mêmes entrées (ex. tri par insertion).
- **Algorithme non déterministe** : Peut donner des résultats différents pour les mêmes entrées, souvent basé sur des choix aléatoires ou multiples (ex. Quicksort avec pivot aléatoire).

Applications :

- **Déterministe** : Utilisé dans des systèmes critiques nécessitant une haute fiabilité, où chaque exécution doit être reproductible.
- **Non déterministe** : Utilisé dans des domaines comme l'intelligence artificielle, la cryptographie, ou les simulations où les résultats exacts ne sont pas toujours nécessaires, mais plutôt des approximations ou des résultats diversifiés.

> En résumé, les algorithmes déterministes sont prévisibles et répétables, tandis que les algorithmes non déterministes peuvent explorer plusieurs chemins d'exécution, souvent avec un élément d'aléatoire ou de choix multiple.

6. LE CBC :

Le **CBC** (Cipher Block Chaining) est un mode de fonctionnement utilisé en cryptographie pour chiffrer des données par blocs. Il est principalement utilisé avec des algorithmes de chiffrement symétrique comme **AES** ou **DES**.

Principe général :

Le CBC prend un message et le divise en **blocs de taille fixe** (par exemple, 128 bits pour AES). Chaque bloc est chiffré de manière à dépendre du bloc précédent, créant ainsi un "enchaînement" (chaining) entre les blocs.

Étapes principales :

1. **Initialisation** :
   - Le premier bloc du message est combiné avec un vecteur d'initialisation (**IV**), qui est aléatoire et unique.
   - Cette combinaison est ensuite chiffrée avec l'algorithme de chiffrement (AES, DES, etc.).
2. **Chiffrement en chaîne** :

- Chaque bloc suivant est combiné avec le **bloc chiffré précédent** (grâce à l'opération XOR).
- Le résultat de cette combinaison est ensuite chiffré.

3. **Déchiffrement** :
   - Pour déchiffrer, on fait le processus inverse : on déchiffre chaque bloc puis on applique un XOR avec le bloc chiffré précédent.

Caractéristiques principales du CBC :

- **Sécurité renforcée** : Chaque bloc chiffré dépend du précédent, ce qui rend plus difficile la déduction du message clair même si certains blocs sont connus.
- **Vecteur d'initialisation (IV)** : Un IV unique et aléatoire garantit que même si le même message est chiffré plusieurs fois, les résultats seront différents à chaque fois.

Avantages :

- **Protection contre les répétitions** : Chaque bloc chiffré est unique grâce à l'utilisation de l'IV et à la dépendance avec les blocs précédents.
- **Facile à implémenter** : Le mode CBC est simple à comprendre et à implémenter avec les algorithmes de chiffrement symétriques classiques.

Inconvénients :

- **Pas de parallélisme** : Chaque bloc dépend du précédent, ce qui empêche le chiffrement de plusieurs blocs en parallèle.
- **Propagation des erreurs** : Si un bloc est corrompu lors de la transmission, il peut affecter le déchiffrement des blocs suivants.

En résumé :

CBC est une méthode de chiffrement qui lie chaque bloc de données au bloc précédent pour renforcer la sécurité. Il est largement utilisé mais peut être remplacé par des modes plus récents pour des raisons d'efficacité et de sécurité.

7. La malléabilité en cryptographie :

Modifier de manière intéligente le chiffre et de connaitre l'impact sur le clair sans avoir la possibilitée d'avoir le clair.

Cours numéro 2 :

```markdown
# Titre de votre cours
Voici du texte en *italique* et en **gras**.
```

Nouvelle page

Guide Grille et Espacements - Site SEMAYAS

1. Grille (Grid)

- **Grille de base** : 12 colonnes (flexible, moderne).
- **Largeur maximale du conteneur** : `1200px` à `1440px` (centré).
- **Marges latérales** :
  - **Desktop** : `40px`.
  - **Mobile** : `20px`.

Breakpoints recommandés :

- **Mobile** : `≤ 767px`
- **Tablette** : `768px – 1024px`
- **Desktop** : `≥ 1025px`

---

2. Dimensions pour chaque balise

Titres

- **H1** :
  - Taille : `48px`
  - Espacement (line-height) : `1.2`
  - Marge bas : `24px`
- **H2** :
  - Taille : `36px`
  - Espacement (line-height) : `1.3`
  - Marge bas : `20px`
- **H3** :
  - Taille : `28px`
  - Espacement (line-height) : `1.4`
  - Marge bas : `16px`

Paragraphes (balise `<p>`)

- Taille : `18px`
- Espacement (line-height) : `1.6`
- Marge bas entre paragraphes : `16px à 24px`

Boutons / Call-to-Action (CTA)

- **Padding** :
  - Vertical : `16px`
  - Horizontal : `32px`
- **Taille de police** : `16px` (en gras).
- **Espacement avec les autres éléments** : `32px à 48px`.

Sections principales

- **Espacements haut/bas** : `80px` à `120px` pour créer de l’air.
- **Entre les sections mineures** : `48px` à `64px`.

---

3. Alignements

- **Texte principal** :
  - **Gauche aligné** pour les paragraphes.

    - **Centré** uniquement pour les titres des sections clés (ex. H1 dans la section héro).
- **Visuels et contenu** :
    - Sur Desktop :
        - Texte : `6 colonnes`
        - Image : `6 colonnes`
    - Sur Mobile :
        - Disposition en **stacked layout** (empilé).

---

4. Espacement général (Spacing)

Utilise une **échelle de 8px** pour garantir la cohérence :

- **Petit espacement** : `8px`, `16px` (pour les marges internes des éléments comme boutons).
- **Moyen espacement** : `24px`, `32px` (entre paragraphes, icônes).
- **Grand espacement** : `48px`, `64px` (pour séparer les sections principales).

---

5. Résumé rapide des valeurs

- **H1** : `48px`
- **H2** : `36px`
- **H3** : `28px`
- **Paragraphe** : `18px`
- **Boutons** : Padding : `16px / 32px`
- **Espacement vertical (sections)** : `80px à 120px`
- **Grille** : 12 colonnes, conteneur `1200px` à `1440px`