

Cryptographie et sécurité

Cours 3: compléments

Mickaël Bettinelli
(mickael.bettinelli@univ-smb.fr)



Prérequis et objectifs

Compétences nécessaire pour ce cours:

- Comprendre le fonctionnement des chiffrements asymétriques
- Effectuer des opérations sur $\mathbb{Z}/n\mathbb{Z}$

Compétences maîtrisées à la fin du cours:

- Connaître le fonctionnement d'une infrastructure de gestion de clés
- Comprendre les codes détecteurs et correcteurs
- Savoir comment stocker des mots de passe



Cours interactif

1. Connectez vous au quizz
1. Choisissez votre pseudo (= nom de famille + prénom)
2. Répondez aux questions
3. Chaque bonne réponse vous rapporte des points
4. Un classement des participants est effectué à la fin du questionnaire
5. Chaque étudiant recevra un bonus de point sur sa moyenne proportionnel à son score dans le questionnaire (max $\frac{1}{3}$)

Connectez vous avec votre téléphone sur:

<https://quizizz.com/>

Game code:





Sommaire

1. Infrastructures de gestion de clés
2. Codes détecteurs
 - a. CRC
3. Codes correcteurs
 - a. Bit de parité
 - b. Codage de hamming
4. Stocker des mots de passe
 - a. Le hachage



Infrastructures de gestion clés



Pourquoi gérer des clés

Problèmes:

1. Dans un cas réaliste, Bob et Alice n'ont pas les clés publiques de l'autre
2. Les deux partis communicants doivent être certains de s'adresser à la bonne personne



Comment gérer des clés

Les services proposés par les outils de gestion de clés:

1. Lier des clés publiques à des identités
2. Stocker des clés et les distribuer
3. Maintenance des clés

Quelques noms d'outils:

- L'autorité de certification
- La toile de confiance
- DANE



Autorité de certification (AC) - introduction

Objectif. authentifier l'identité de correspondants

Conditions d'utilisation. sécurisation des communications avec le protocole TLS (anciennement SSL)

Fonctions.

1. Assure que les données entre le serveur et le client n'ont pas été modifiées durant le transfert
2. Protège de l'usurpation d'identité
3. Assure la confidentialité des données transmises

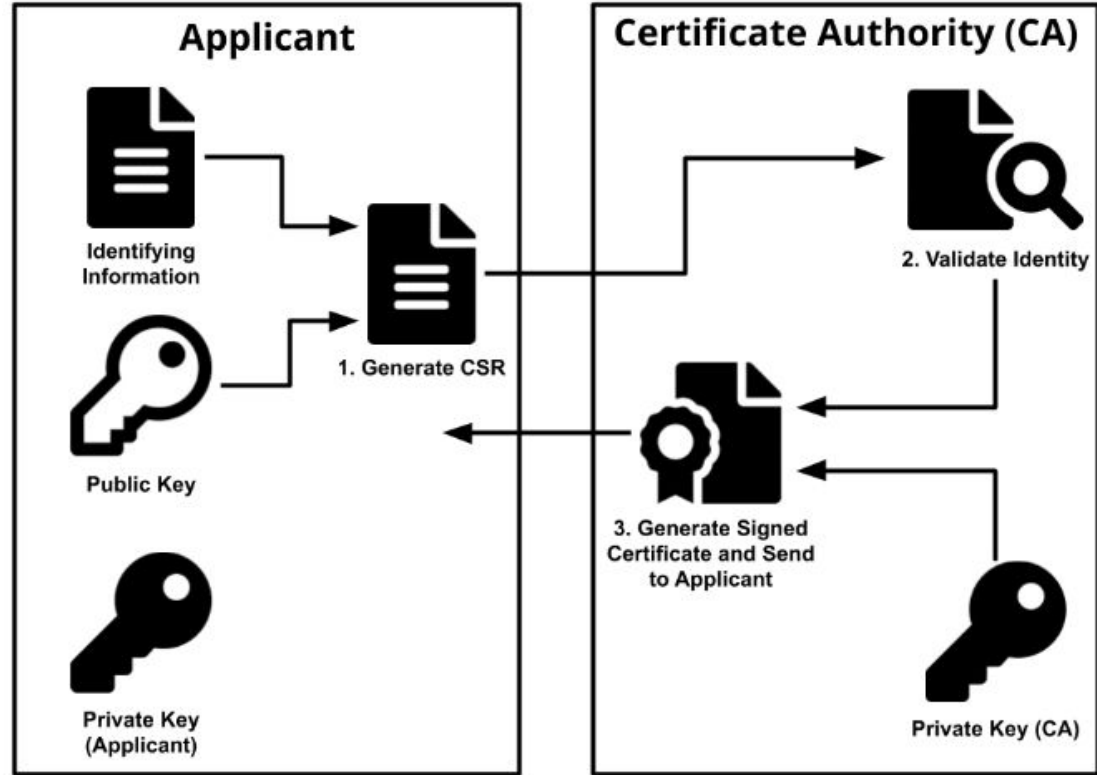


Obtenir un certificat



Fonctionnement.

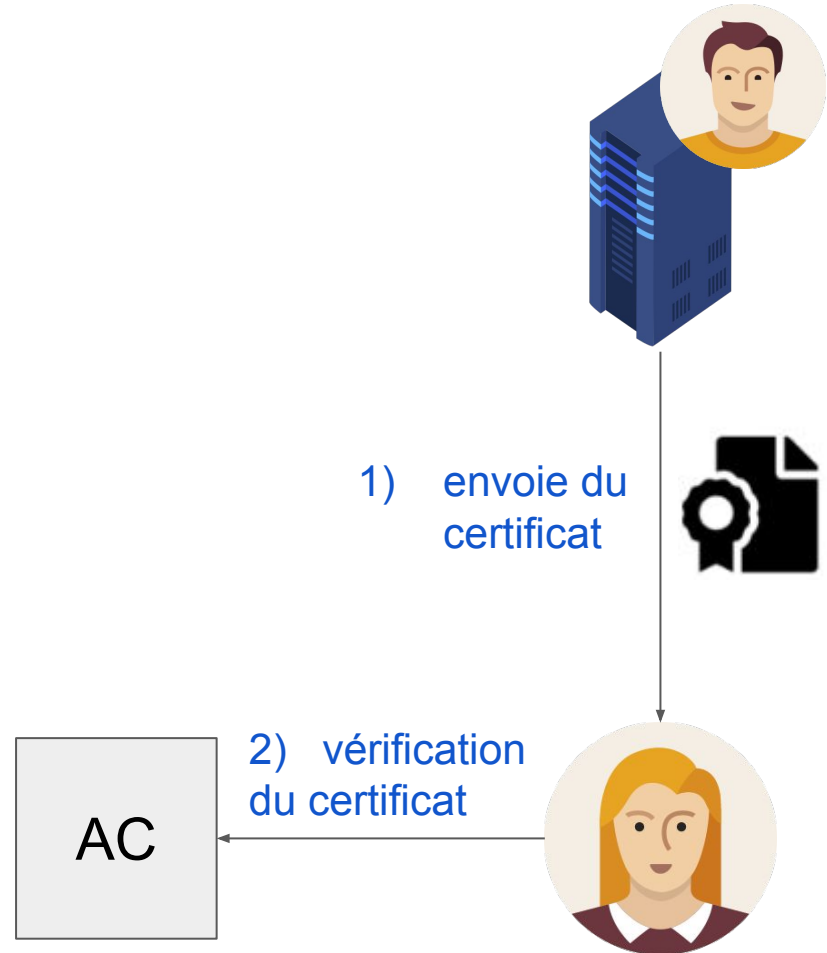
1. Génération d'une clé publique et privée
2. Envoie de la clé publique + identité (coordonnées postales, téléphonique, etc.) à l'autorité de certification
3. L'AC vérifie l'identité
4. Elle génère et envoie un certificat au candidat
5. Le candidat intègre la clé sur son serveur web





Utiliser le certificat

1. Alice se connecte au serveur web de Bob
2. Le serveur lui envoie son certificat signé par la clé privée de l'autorité
3. Le navigateur de Alice vérifie le certificat grâce à la clé publique de la certification
4. Le navigateur de Alice contacte l'AC pour vérifier la validité du certificat

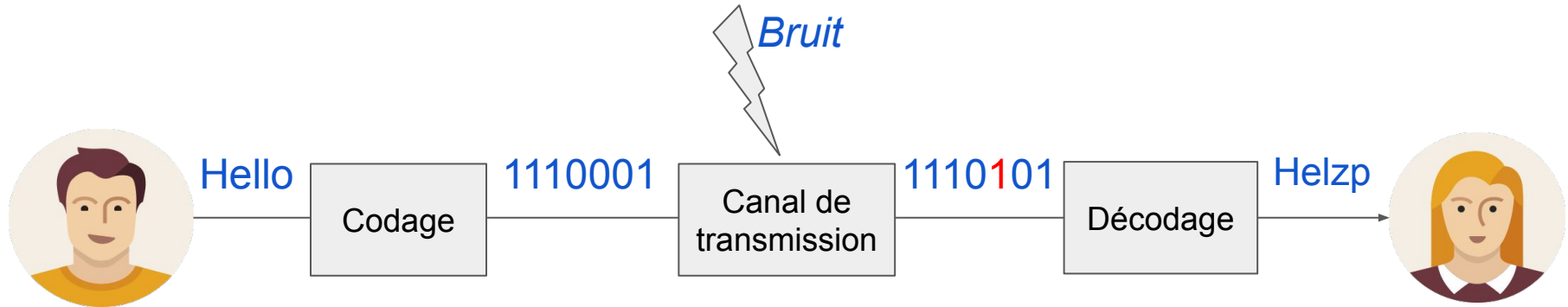




Code détecteurs et correcteurs



Problématique



Des erreurs peuvent se glisser dans les message:

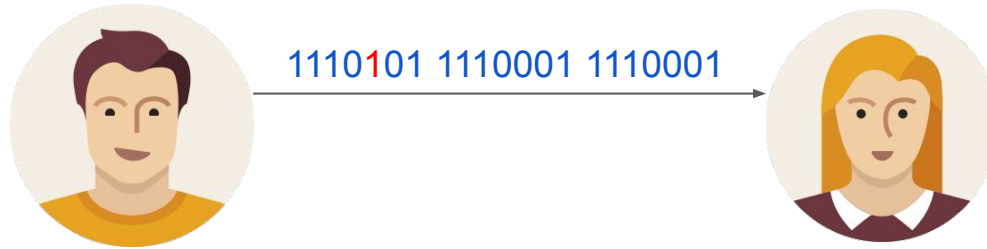
- Comment les détecter ?
- Comment les localiser ?
- Comment les corriger ?



Détecter des erreurs

Approche naïve: la redondance d'information

Bob envoie un message contenant 3 fois son message initial



Le bon message est celui qui existe en double exemplaire.

Inconvénients: risque d'erreur si l'erreur intervient deux fois d'affilées au même endroit du message (1110101 1110101 1110001)



Détecter des erreurs

Autre solution (également naïve):

- Ajouter de la redondances aux messages des utilisateurs:
 - Exemple: pour envoyer HELLO -> “H comme Hélice, E comme Ecran, L comme Laure, L comme Laure, O comme Orange”
 - Si mauvaise réception: “**T** comme Hélice, E comme **E**dfan, L **co**eme Laure, L comme Laure, O comme **P**rangle”, on peut quand même reconstituer le message

La redondance semble être une solution pratique, mais peu fiable.



Détecter des erreurs

Autre solution (un peu plus efficace):

- Code à contrôle de parité: on ajoute des bits en fin de message (binaire) pour que le nombre de 1 soit pair ou impair.
 - Exemple:

Lettre	Code ASCII	Parité paire	Parité impaire
A	1000001	10000010	10000011
B	1000010	10000100	10000101
C	1000011	10000111	10000110



Détecter des erreurs

Lettre	Code ASCII	Parité paire	Parité impaire
A	1000001	10000010	10000011
B	1000010	10000100	10000101
C	1000011	10000111	10000110

Inconvénients:

- Impossible à détecter en cas de double erreurs
- Simple détection sans possibilité de correction



Détecter des erreurs

Autre solution (moins intuitive):

- Codes polynômiaux:
 - Aussi appelé Code à Redondance Cyclique ou **CRC**
 - On utilise un polynôme $G(x)$ connu à l'avance par l'émetteur et le récepteur pour coder et décoder la séquence de bits à transmettre.
 - Une séquence de n bits est représenté comme un polynôme à n termes de x^0 à x^{n-1} (10101 $\rightarrow x^4 + x^2 + 1$)



CRC - coder un message

Principe. Soit M le message à envoyer et $G(X)$ un polynôme générateur dont le plus grand terme est x^k .

L'algorithme pour coder M est le suivant:

1. Calculer $M(x)$ le polynôme correspondant à M
2. Calculer $P(x) = M(x) * x^k$

Exemple. $M = 10101$, $G(x) = x^3 + x + 1$

1. $M(x) = x^4 + x^2 + 1$
2. $P(x) = M(x) * x^3 = x^7 + x^5 + x^3$



CRC - coder un message

Principe. Soit M le message à envoyer et $G(X)$ un polynôme générateur dont le plus grand terme est k .

L'algorithme pour coder M est le suivant:

3. Diviser $P(x)$ par $G(x)$, le reste de la division est appelé $R(x)$
4. Calculer $M'(x) = P(x) + R(x)$
5. Reformuler le message M'
6. Transmettre M' à la place de M



CRC - coder un message

Exemple. $M = 10101$, $G(x) = x^3 + x + 1$

1. $M(x) = x^4 + x^2 + 1$
2. $P(x) = M(x) * x^3 = x^7 + x^5 + x^3$
3. Division **modulo 2** de $P(x)$ par $G(x)$:
 - a. À chaque étape on élimine le terme le plus élevé du numérateur
 - b. La division s'arrête lorsqu'il est impossible d'éliminer ce terme

$$\begin{array}{r|l} x^7 + x^5 + x^3 & x^3 + x + 1 \\ \hline \end{array}$$



CRC - coder un message

Exemple. $M = 10101$, $G(x) = x^3 + x + 1$

1. $M(x) = x^4 + x^2 + 1$
2. $P(x) = M(x) * x^3 = x^7 + x^5 + x^3$
3. Division de $P(x)$ par $G(x)$:
 - a. À chaque étape on élimine le terme le plus élevé du numérateur
 - b. La division s'arrête lorsqu'il est impossible d'éliminer ce terme

$$\begin{array}{r|l} x^7 + x^5 + x^3 & x^3 + x + 1 \\ x^7 + x^5 + x^4 & \hline \hline & x^4 \end{array}$$



CRC - coder un message

Exemple. $M = 10101$, $G(x) = x^3 + x + 1$

1. $M(x) = x^4 + x^2 + 1$
2. $P(x) = M(x) * x^3 = x^7 + x^5 + x^3$
3. Division de $P(x)$ par $G(x)$:
 - a. À chaque étape on élimine le terme le plus élevé du numérateur
 - b. La division s'arrête lorsqu'il est impossible d'éliminer ce terme

$$\begin{array}{r|l} x^7 + x^5 + x^3 & x^3 + x + 1 \\ x^7 + x^5 + x^4 & \hline \hline x^4 + x^3 & \end{array}$$



CRC - coder un message

Exemple. $M = 10101$, $G(x) = x^3 + x + 1$

1. $M(x) = x^4 + x^2 + 1$
2. $P(x) = M(x) * x^3 = x^7 + x^5 + x^3$
3. Division de $P(x)$ par $G(x)$:
 - a. À chaque étape on élimine le terme le plus élevé du numérateur
 - b. La division s'arrête lorsqu'il est impossible d'éliminer ce terme

$$\begin{array}{r|l} x^7 + x^5 + x^3 & x^3 + x + 1 \\ x^7 + x^5 + x^4 & \hline \hline x^4 + x^3 & x^4 \\ x^4 + x^2 & x \end{array}$$



CRC - coder un message

Exemple. $M = 10101$, $G(x) = x^3 + x + 1$

1. $M(x) = x^4 + x^2 + 1$
2. $P(x) = M(x) * x^3 = x^7 + x^5 + x^3$
3. Division de $P(x)$ par $G(x)$:
 - a. À chaque étape on élimine le terme le plus élevé du numérateur
 - b. La division s'arrête lorsqu'il est impossible d'éliminer ce terme

$$\begin{array}{r|l} x^7 + x^5 + x^3 & x^3 + x + 1 \\ \hline x^7 + x^5 + x^4 & x^4 \\ \hline x^4 + x^3 & \\ x^4 + x^2 & x \\ \hline x^3 + x^2 & \end{array}$$



CRC - coder un message

Exemple. $M = 10101$, $G(x) = x^3 + x + 1$

1. $M(x) = x^4 + x^2 + 1$
2. $P(x) = M(x) * x^3 = x^7 + x^5 + x^3$
3. Division de $P(x)$ par $G(x)$:
 - a. À chaque étape on élimine le terme le plus élevé du numérateur
 - b. La division s'arrête lorsqu'il est impossible d'éliminer ce terme

$$\begin{array}{r|l} x^7 + x^5 + x^3 & x^3 + x + 1 \\ \hline x^7 + x^5 + x^4 & x^4 \\ \hline x^4 + x^3 & x \\ x^4 + x^2 & \\ \hline x^3 + x^2 & \\ x^3 + x + 1 & 1 \end{array}$$



CRC - coder un message

Exemple. $M = 10101$, $G(x) = x^3 + x + 1$

1. $M(x) = x^4 + x^2 + 1$
2. $P(x) = M(x) * x^3 = x^7 + x^5 + x^3$
3. Division de $P(x)$ par $G(x)$:
 - a. À chaque étape on élimine le terme le plus élevé du numérateur
 - b. La division s'arrête lorsqu'il est impossible d'éliminer ce terme

$$\begin{array}{r|l} x^7 + x^5 + x^3 & x^3 + x + 1 \\ \hline x^7 + x^5 + x^4 & x^4 \\ \hline x^4 + x^3 & \\ x^4 + x^2 & x \\ \hline x^3 + x^2 & \\ x^3 + x + 1 & 1 \\ \hline x^2 + x + 1 & \end{array}$$



CRC - coder un message

Exemple. $M = 10101$, $G(x) = x^3 + x + 1$

1. $M(x) = x^4 + x^2 + 1$
2. $P(x) = M(x) * x^3 = x^7 + x^5 + x^3$
3. Division de $P(x)$ par $G(x)$:
 - a. À chaque étape on élimine le terme le plus élevé du numérateur
 - b. La division s'arrête lorsqu'il est impossible d'éliminer ce terme

$$\begin{array}{r|l} x^7 + x^5 + x^3 & x^3 + x + 1 \\ \hline x^7 + x^5 + x^4 & x^4 \\ \hline x^4 + x^3 & \\ x^4 + x^2 + x & x \\ \hline x^3 + x^2 + x & \\ x^3 + x + 1 & 1 \\ \hline x^2 + 1 & \end{array}$$

Le reste est : $x^2 + 1$



CRC - coder un message

Exemple. $M = 10101$, $G(x) = x^3 + x + 1$

1. $M(x) = x^4 + x^2 + 1$
2. $P(x) = M(x) * x^3 = x^7 + x^5 + x^3$
3. Division de $P(x)$ par $G(x)$:
 - a. À chaque étape on élimine le terme le plus élevé du numérateur
 - b. La division s'arrête lorsqu'il est impossible d'éliminer ce terme

$$\begin{array}{r|l} x^7 + x^5 + x^3 & x^3 + x + 1 \\ \hline x^7 + x^5 + x^4 & x^4 \\ \hline x^4 + x^3 & \\ x^4 + x^2 + x & x \\ \hline x^3 + x^2 + x & \\ x^3 + x + 1 & 1 \\ \hline x^2 + 1 & \end{array}$$





CRC - coder un message

Exemple. $M = 10101$, $G(x) = x^3 + x + 1$

1. $M(x) = x^4 + x^2 + 1$
2. $P(x) = M(x) * x^3 = x^7 + x^5 + x^3$
3. Division de $P(x)$ par $G(x)$: le reste $R(x)$ est égal à $x^2 + x + 1$
4. $M'(x) = P(x) + R(x) = x^7 + x^5 + x^3 + x^2 + x + 1$
5. $M' = 10101111$
6. Transmettre M' !



CRC - décoder un message

Principe.

1. Calculer le polynôme $M'(x)$ à partir de M'
2. Diviser $M'(x)$ par $G(x)$
3. Si le résultat vaut 0, alors il n'y a pas d'erreur détectée
4. Sinon, il y a une erreur



CRC - coder un message

Exemple. Nous recevons $M' = 10101111$ et, $G(x) = x^3 + x + 1$

1. $M'(x) = x^7 + x^5 + x^3 + x^2 + x + 1$
2. Division de $M'(x)$ par $G(x) \rightarrow$ le reste est 0

Résultat: pas d'erreur détectée !



CRC - choisir le générateur

La qualité de la détection dépend du polynôme générateur.

Quel polynôme générateur choisir ?

- Plus le générateur a un degré élevé, plus la probabilité de laisser passer une erreur est faible
- Plus le générateur a un degré élevé, plus il est coûteux en temps et en énergie de coder et décoder les messages



Détecter des erreurs en groupes

Activité de groupe. 40 minutes (25 exercice + 15 correction)

1. Formez des groupes de 3 (*déplacez vous si nécessaire*) (**3mn**)
2. L'un des groupes doit encoder un message de 4 bits avec CRC et le transmettre **avec une erreur ou pas** à l'autre groupe (**10mn max**)
3. Trouvez un autre groupe de 3 avec qui travailler (**2mn**)
4. L'autre groupe doit le décoder et dire s'il y a (ou non) une erreur (**10mn max**).

A la fin des 25 minutes:

1. 2 groupes descendent pour montrer comment ils ont coder leur message
2. Leur 2 partenaires viennent le décoder

Générateur à utiliser:

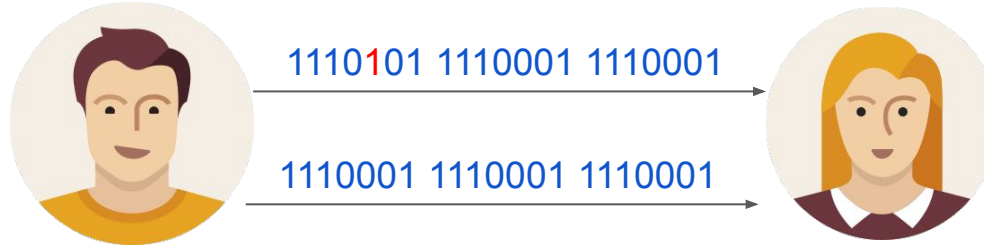
- $G(x) = x^3 + x + 1$



Correction des erreurs

Approche naïve:

- Renvoi du message après la détection de l'erreur

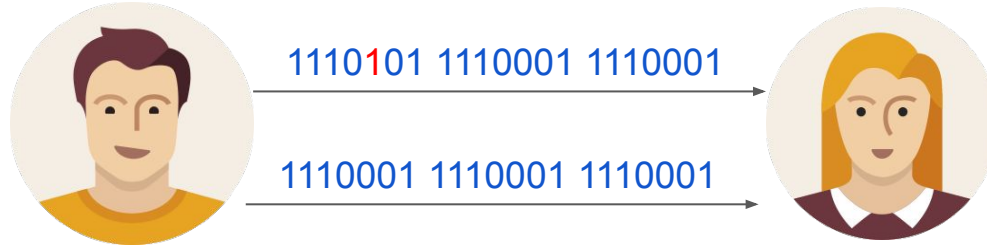




Correction des erreurs

Approche naïve:

- Renvoi du message après la détection de l'erreur



- Inconvénient: pas sûr que le second message n'aura pas d'erreur



Code de Hamming

Une autre solution plus fiable est le code de Hamming.

- Présentation:
 - Mathématicien américain récompensé par le prix Turing en 1968
 - Le code de Hamming est inventé en 1948
- Principe général:
 - ajouter un nombre de bits de contrôle dans le message pour détecter et corriger les erreurs



Fonctionnement

Le fonctionnement général est le suivant:

1. Préparer des emplacements pour les bits de contrôle sur les emplacements de puissances de 2 (... , 8, 4, 2, 1)
2. Compléter le message sur les emplacements non réservés aux bits de contrôle
3. Calculer la valeur des bits de contrôle en fonction du message
4. Insérer les bits de contrôle dans le message

Exemple. Message = 1010

bit	controle	1	0	1	controle	0	controle	controle
index	8	7	6	5	4	3	2	1



Préparation des bits de contrôle

Comment calculer les bits de contrôle ?

1. Faire une matrice bits message (seulement les bits à 1) / bits de contrôle
2. Remplir la matrice en décomposant les positions en poids

Exemple. Message = 1010

Poids / Positions	8	4	2	1
5	0	1	0	1
7	0	1	1	1



Préparation des bits de contrôle

Comment calculer les bits de contrôle ?

1. Tracer les colonnes et calcul du bit de parité (impaire)

Exemple. Message = 1010

Poids / Positions	8	4	2	1
5	0	1	0	1
7	0	1	1	1
Bit de controles	1	1	0	1



Préparation des bits de contrôle

Dernière étape: on remplit les bits de contrôle avec ceux que l'on vient de calculer

Poids / Positions	8	4	2	1
5	0	1	0	1
7	0	1	1	1
Bit de controles	1	1	0	1

bit	1	1	0	1	1	0	0	1
index	8	7	6	5	4	3	2	1



Préparation des bits de contrôle

Message initial = 1010

Message encodé à transmettre = 11011001

bit	1	1	0	1	1	0	0	1
index	8	7	6	5	4	3	2	1



Détection des erreurs

Le fonctionnement est le suivant:

- Numérotation des bits dans le message

bit	1	1	0	1	1	0	0	1
index	8	7	6	5	4	3	2	1



Détection des erreurs

Le fonctionnement est le suivant:

- Faire une matrice bits à 1 / numéro des bits de contrôle

Poids / Positions	8	4	2	1
1	0	0	0	1
4	0	1	0	0
5	0	1	0	1
7	0	1	1	1
8	1	0	0	0



Détection des erreurs

Le fonctionnement est le suivant:

- Vérifier la cohérence de la parité des colonnes (parité impaire)

Poids / Positions	8	4	2	1
1	0	0	0	1
4	0	1	0	0
5	0	1	0	1
7	0	1	1	1
8	1	0	0	0
Bit de parité	OK	OK	OK	OK



Détection des erreurs

Attention:

- Si une erreur est détectée alors il y a bien une erreur !
- Si aucune erreur n'est détectée, il peut quand même y en avoir une !



Correction des erreurs

Voici le même message que précédemment avec l'ajout d'une erreur sur à l'index 4.

bit	1	1	1	1	1	0	0	1
index	8	7	6	5	4	3	2	1



Correction des erreurs

On reconstruit la matrice

Poids / Positions	8	4	2	1
1	0	0	0	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0



Correction des erreurs

Et on re-calcule les bits de parités.

On sait qu'il y a au moins une erreur dans le message.

Poids / Positions	8	4	2	1
1	0	0	0	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
Bit de parité	OK	PAS OK	PAS OK	OK



Correction des erreurs

Pour déterminer où se trouve l'erreur il faut calculer la somme des indices des bits de contrôle où une erreur se trouve. Ici $4 + 2 = 6$

Poids / Positions	8	4	2	1
1	0	0	0	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
Bit de parité	OK	PAS OK	PAS OK	OK

Correction des erreurs



Poids / Positions	8	4	2	1
1	0	0	0	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
Bit de parité	OK	PAS OK	PAS OK	OK

bit	1	1	1 -> 0	1	1	0	0	1
index	8	7	6	5	4	3	2	1



En résumé

Hamming permet de:

- Détecter jusqu'à deux erreurs simultanées dans un message
- Corriger un maximum d'une erreur



Stockage des mots de passes



Stockage des mots de passes

Pourquoi ?

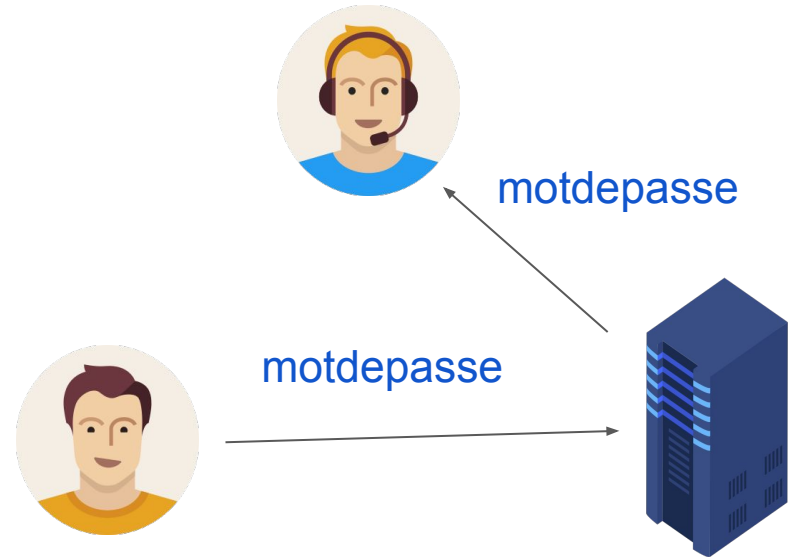
- Accéder à ses espaces personnels

Comment ?

- Stockage des mots de passe en base de données
- Besoin de les chiffrer en cas de fuite



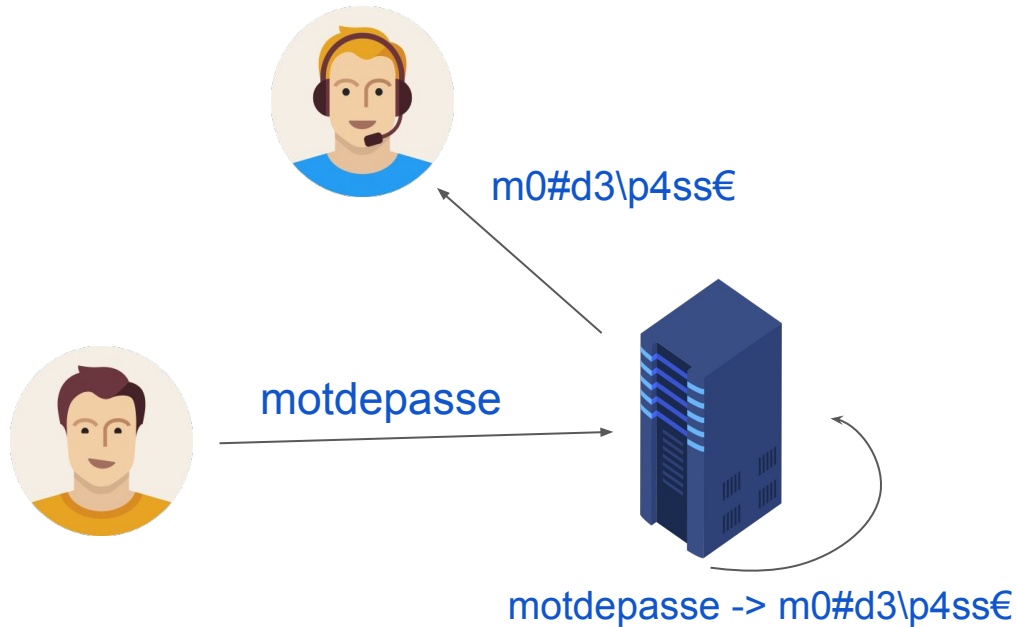
Instagram





Problématique

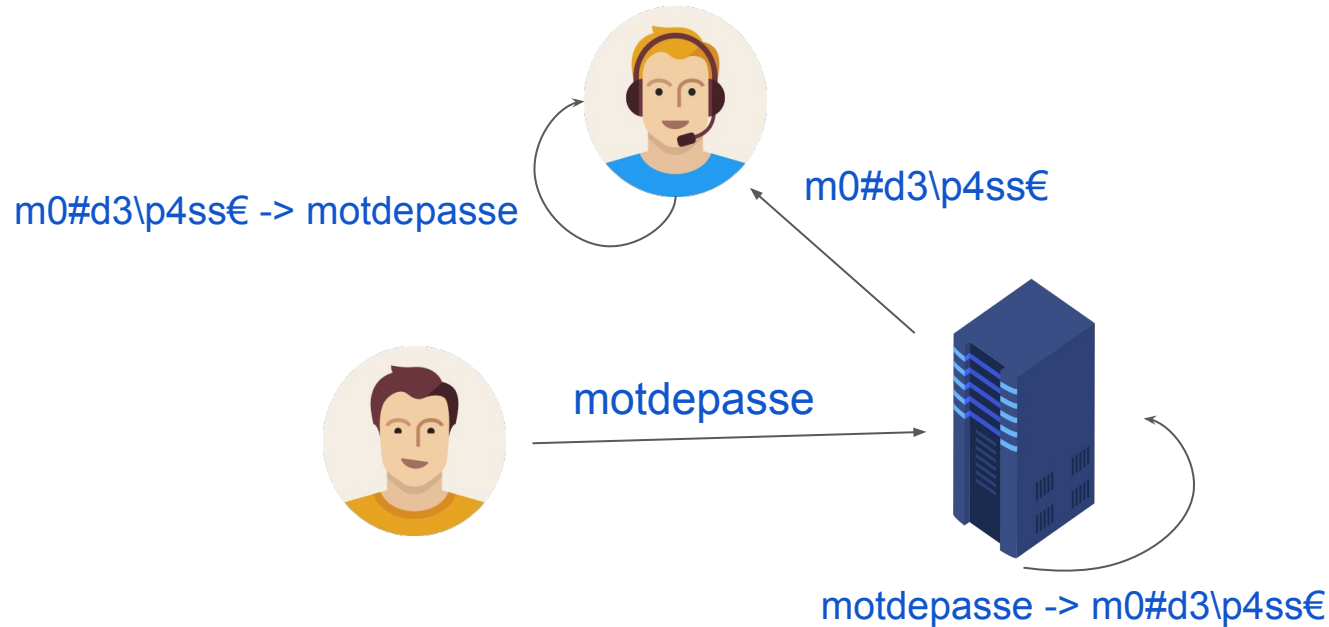
Chiffrer les mots de passe pour les rendre inutilisable en cas de fuite





Problématique

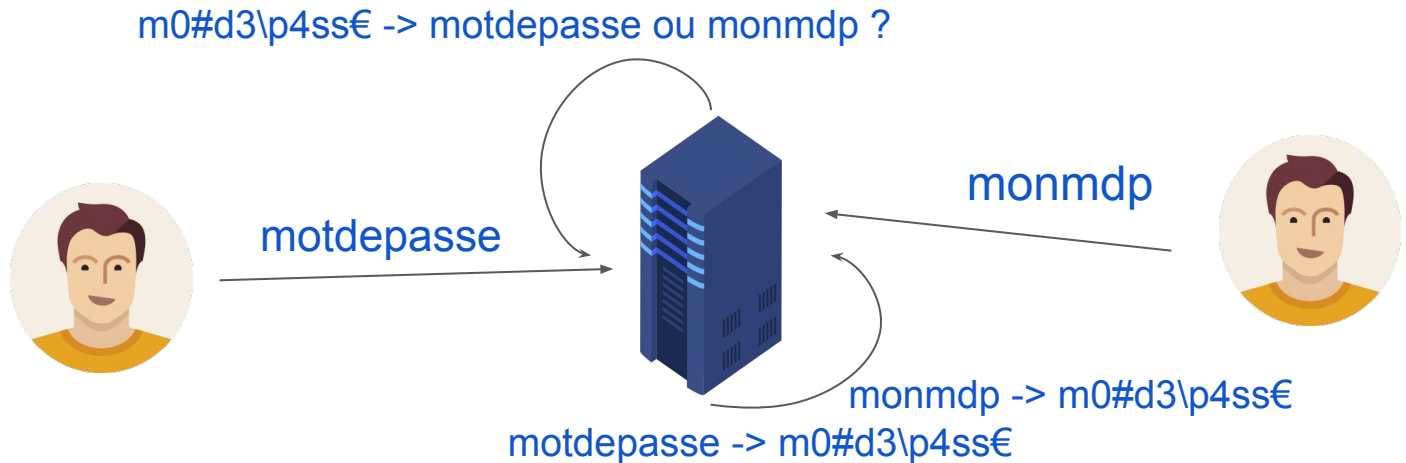
1. Mais les mots de passe ne doivent pas être déchiffrables





Problématique

1. Mais les mots de passe ne doivent pas être déchiffrables
2. Et ils ne doivent pas être confondus les uns avec les autres

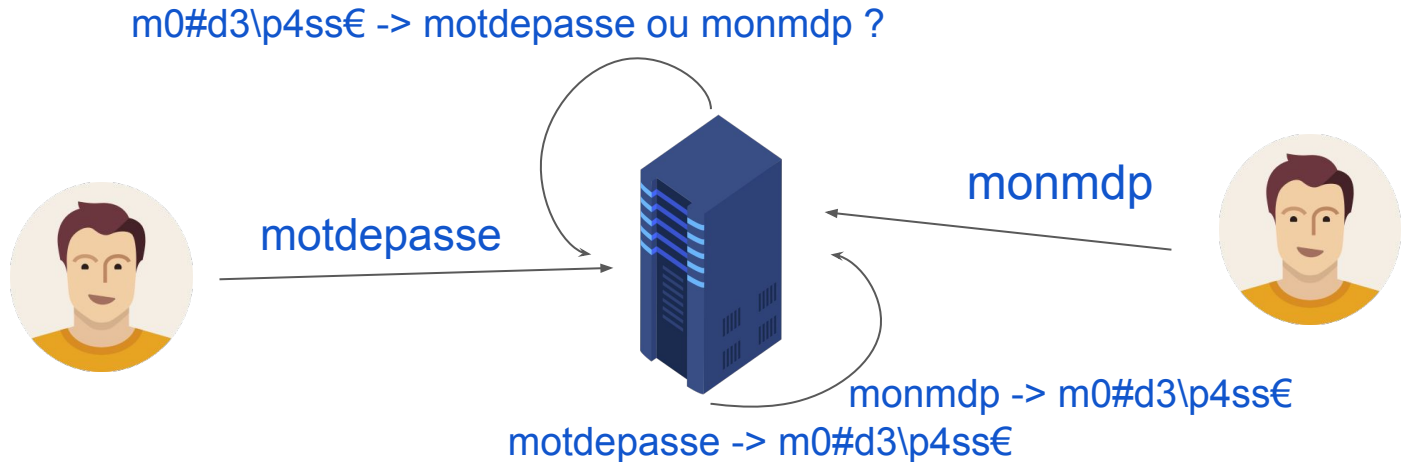




Solution

Chiffrer les mots de passe de façon à ce que:

- Les mots de passe soient impossible à déchiffrer
- Un mot de passe = un mot de passe chiffré
 - (bijection entre l'ensemble des mdp et l'ensemble des mdp chiffrés)





Les fonctions de hachage

Définition 1. Une fonction de hachage est une fonction particulière qui, **à partir d'une donnée fournie** en entrée, **calcule une empreinte numérique** servant à identifier rapidement la donnée initiale.

Définition 2. Le retour de la fonction est appelée un *hash* ou une *empreinte*.

Les caractéristiques des fonctions de hachage.

1. Fonction à sens unique: impossible de déchiffrer une empreinte
2. Déterminisme: hacher deux fois le même message conduit au même résultat
3. Unicité de la signature: un même message va correspondre une signature unique

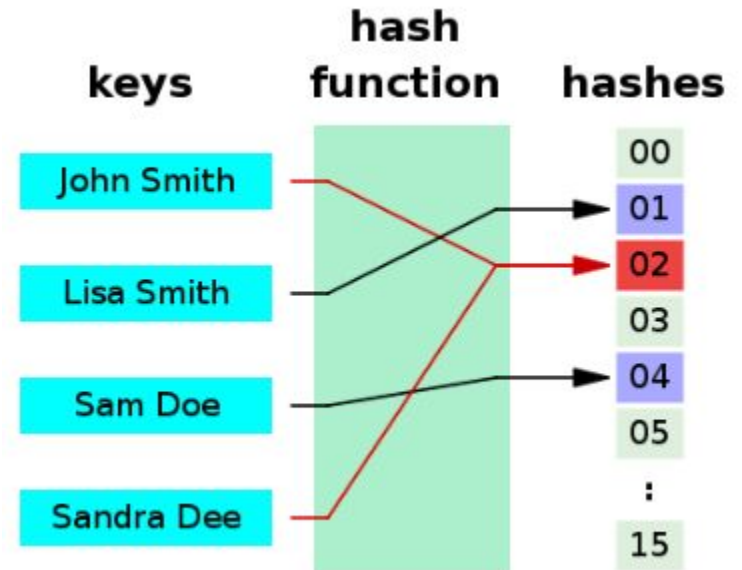


Les fonctions de hachage

Exemple simple de fonction de hachage:

- Une fonction qui retourne le premier octet de votre mot de passe
 - Exemple:
 - motdepasse = 01101101
 - chat = 01100011

Cette fonction est très peu efficace !

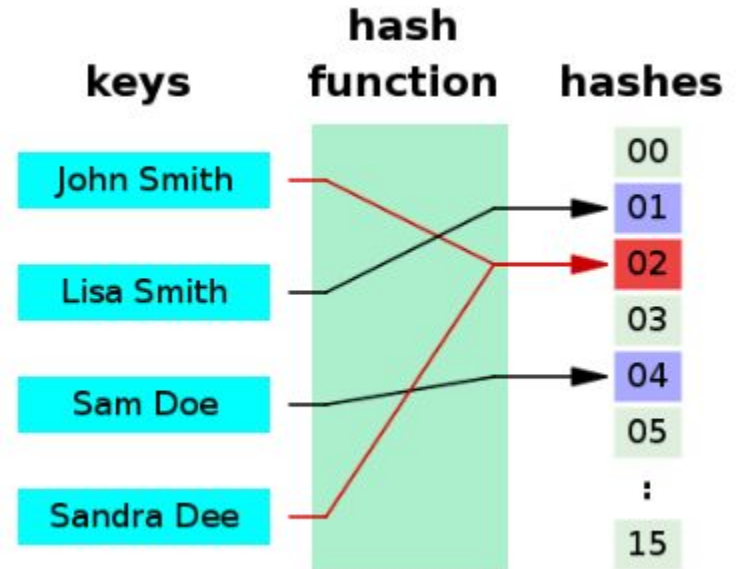




Les fonctions de hachage

1, 2, Tous - (1mn, 1mn, 3mn)

Recherchez et proposez une fonction de hachage qui remplisse (au mieux) les propriétés demandées.





MD5 (Message Digest) - déprécié

- Inventé par Ronald Rivest en 1991
- Empreintes de 128 bits, avec une forte probabilité que deux haches soient différents
- En 1996, une faille permet de créer des collisions à la demande

Fonctionnement général.

1. Découpe du message en blocs de 512 bits
2. Effectue 64 séries d'opérations AND, XOR, OR, ROT, ADD sur chaque blocs
3. Produit une empreinte condensée de 128 bits



SHA-1 (Secure Hash Algorithm) - déprécié

Caractéristiques:

- Inventé par la NSA (National Security Agency) en 1995
- Empreintes de 160 bits

Fonctionnement général.

1. Découpe du message en blocs de 512 bits
2. Effectue 80 séries d'opérations AND, XOR, OR, ROT, ADD sur chaque blocs
3. Produit une empreinte condensée de 160 bits



SHA-3

Caractéristiques:

- Inventé en 2015
- Empreintes de 224 à 512 bits selon la version de SHA-3
- SHA-3 se distingue des versions précédentes par une nouvelle méthode de hache

Fonctionnement général.

1. Découpe du message en blocs de 512 bits
2. Effectue 24 séries d'opérations AND, XOR, ROT sur chaque blocs
3. Produit une empreinte condensée de 1600 bits



Les hash tables

Les hash tables:

- Calculer l'empreinte des éléments à ranger dans le tableau
- Ranger et trier les éléments en fonction de leur empreinte

Exemple. L'empreinte de chaque élément est l'index de sa première lettre dans l'alphabet (a = 0, b = 1, etc.)

Bulbe	Chat	Clé	Fleur	Gauffre
0	1	2	3	4



Vérifier les fichiers

Comment vérifier qu'un fichier n'a pas été modifié par un tiers ?

1. Le créateur du fichier calcule son empreinte
2. Le destinataire télécharge le fichier
3. Le destinataire re-calcule l'empreinte du fichier et la compare à l'empreinte calculée par le créateur



En résumé

2 familles de fonctions de hachage:

- MD (Message Digest):
 - Plus rapide que les algorithmes SHA
 - Peu sécurisé
- SHA (Secure Hash Algorithm):
 - Fonctionnement plus complexe et plus lent
 - Plus sécurisé que MD5



Identifier les apprentissages

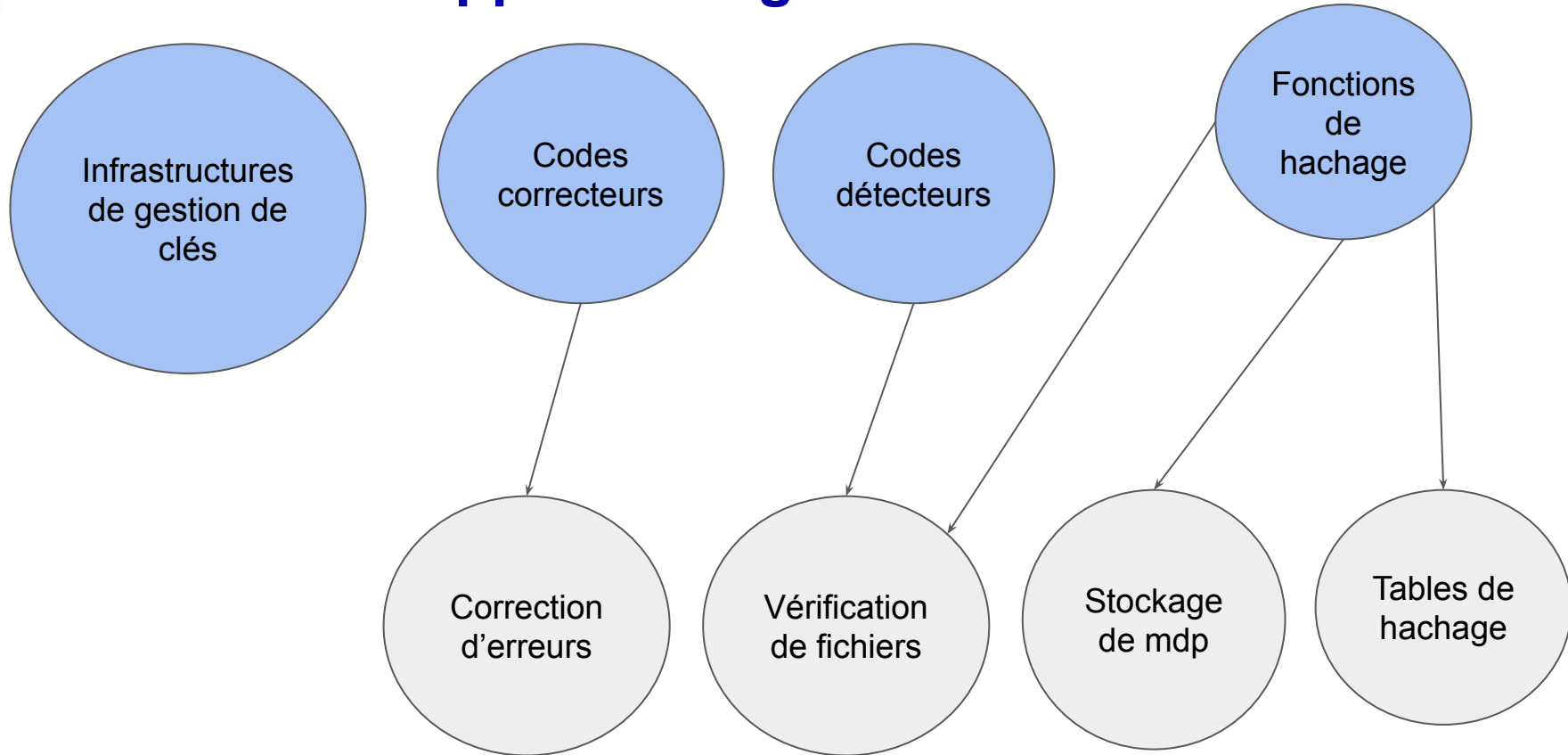
Quels sont les concepts à retenir de ce cours ?

Activité 1, 2, Tous:

- Réfléchissez individuellement à cette question pendant - **3mn**
- Comparez vos idées avec l'un de vos voisins, convainquez-le que vous avez raison ! - **3mn**
- Mise en commun des réponses, des binômes sont interrogés - **2mn**



Identifier les apprentissages





Quelques dernières questions sur le quizz



Ressources complémentaires

- https://en.wikipedia.org/wiki/Hash_function
- https://www.youtube.com/watch?v=KyUTuwz_b7Q&ab_channel=ComputerScience
- Distributed Systems, Marteen Van Steen, 2017 (chapitre 9: cybersécurité)

