




Cours : Hachage et Intégrité

☐ 1. Définitions clés

- **Fonction de hachage** : algorithme qui transforme une entrée (texte, fichier, message) de **longueur arbitraire** en une **empreinte** (hash) de **longueur fixe**.
 - **Intégrité** : garantie que les données **n'ont pas été modifiées** entre leur envoi et leur réception.
-

2. Caractéristiques d'une bonne fonction de hachage cryptographique

1.  **Déterministe** : même entrée → même sortie.
 2.  **Longueur fixe** : peu importe la taille du message, le hash a une longueur constante (ex : 256 bits).
 3. ☐ **Rapide à calculer**.
 4.  **Résistance à la préimage** : difficile de retrouver l'entrée à partir du hash.
 5. ☐ **Résistance à la seconde préimage** : impossible de trouver une autre entrée donnant le même hash.
 6.  **Résistance aux collisions** : très difficile de trouver **deux entrées différentes** avec le **même hash**.
-

3. Algorithmes de hachage célèbres

| Algorithme | Longueur du hash | Statut | Utilisation courante |
|------------|------------------|---------------|---------------------------------------|
| MD5 | 128 bits | Obsolète | Empreintes de fichiers non sécurisées |
| SHA-1 | 160 bits | Obsolète | Anciennement utilisé dans SSL |
| SHA-256 | 256 bits | Sûr (2024) | SSL/TLS, Blockchain, etc. |
| SHA-3 | Variable | Très sécurisé | Cryptographie moderne |
| BLAKE2 | Variable | Très rapide | Hachage rapide et sécurisé |

🔗 4. Applications concrètes

- **Vérification de fichiers** : comparer le hash attendu et obtenu après téléchargement.
 - **Signatures numériques** : on signe un **hash**, pas le document complet.
 - **Blockchain** : les blocs sont chaînés par des fonctions de hachage.
 - **Authentification** : les mots de passe sont stockés sous forme hachée.
 - **Détection de modification** : un changement même minime dans un fichier → hash complètement différent (effet avalanche).
-

📦 5. Exemple : SHA-256

Entrée :

Bonjour123

Hash SHA-256 :

a9efcf52c3054c91f87534b42711c4a0d41a4e9d6d162b6a99dd5ad6f1a0bcd9

Une seule lettre changée (ex: `bonjour123`) produit un hash **totale**ment différent.

🔒 6. Hachage ≠ Chiffrement

| Aspect | Hachage | Chiffrement |
|-----------------|----------------------|-------------------------|
| Sens inverse | Impossible | Possible avec une clé |
| Usage principal | Intégrité, empreinte | Confidentialité |
| Longueur sortie | Fixe | Variable selon l'entrée |
| Clé nécessaire | Non | Oui |

📁 7. Stockage sécurisé des mots de passe

- On ne stocke jamais un mot de passe en clair.
- On stocke :
 - le **hash** du mot de passe (avec un **sel** pour éviter les rainbow tables),
 - et on compare le hash au moment de la connexion.

Exemple : avec *bcrypt* / *scrypt* / *Argon2*

- Fonction de hachage **lente et sécurisée**.
- Protège contre les attaques par force brute.

🛡️ 8. Menaces et contre-mesures

| Menace | Contre-mesure |
|----------------------|--|
| Rainbow table | Utiliser des sels aléatoires |
| Collision volontaire | Utiliser SHA-256 / SHA-3 |
| Brute-force | Utiliser bcrypt, scrypt, Argon2 |
| Hashing rapide (MD5) | Éviter MD5 / SHA-1 |

✅ 9. Résumé visuel

- Une **fonction de hachage** est comme une **empreinte digitale** du fichier.
 - Elle permet de **vérifier l'intégrité**, pas de cacher le contenu.
 - Un bon hash doit être : **unique, non réversible et résistant aux attaques**.
-