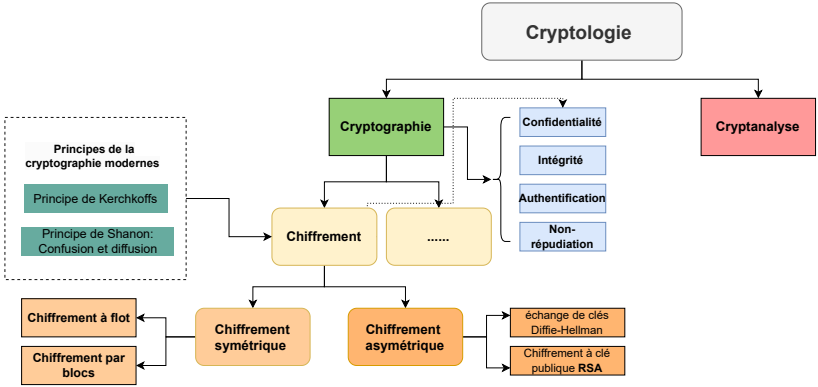
Initiation à la Cryptographie

Fonctions de hachages cryptographique et intégrité des données

May 20, 2025

Awaleh HOUSSEIN

Recap de posts précédents [1]



Jusqu'ici, nous nous sommes concentrés sur la *confidentialité* des données. Intéressons-nous maintenant à un autre pilier fondamental de la cryptographie : comment garantir **l'intégrité** des données ?

[1] Quatrième post: `https://www.linkedin.com/feed/update/urn:li:activity:7319752681365950465/`

Rappel et quelques idées sur comment garantir l'integrité

Rappel : integrité

l'**intégrité** consiste à s'assurer que le message n'a pas été modifié ou alteré durant sa transmission.

Alors, une question clé se pose : comment concevoir des mécanismes qui garantissent l'intégrité des données ?

Quelques idées simples à explorer :

1. *Ajouter un résumé du message* (ex. : somme des caractères) pour vérifier à l'arrivée.
2. *Envoyer plusieurs copies* et comparer leur cohérence.
3. *Intégrer un mot-clé ou motif connu* : s'il est modifié, le message l'est aussi.
4. *Demander un accusé de réception* avec le contenu reçu pour le valider.

Fonction de hachage cryptographique

Définition

Une **fonction de hachage cryptographique** est une fonction à sens unique, notée *h*, qui prend une entrée de taille arbitraire et produit un *haché* de taille fixe, servant d'empreinte unique du message.

$$h : \mathcal{E}^* \longrightarrow \mathcal{F}^t$$

- $\mathcal{E}^*$: l'ensemble des chaînes binaires de longueur quelconque (entrée).
- $\mathcal{F}^t$: l'ensemble des chaînes binaires de longueur fixe t (sortie).

Propriétés clés d'une fonction de hachage :

- Déterministe : même entrée $\longrightarrow$ même haché;
- Rapide à calculer;
- Petite modification du message $\longrightarrow$ haché très différent (voir slide 12).

Propriétés des fonctions de hachage

Une fonction de hachage cryptographique doit satisfaire trois propriétés clés :

- **Préimage difficile à inverser** :
  Trouver x tel que $h(x) = y$ est difficile pour un y donné.
- **Seconde préimage résistante** :
  Pour un x donné, il est difficile de trouver $x' \neq x$ tel que $h(x') = h(x)$.
- **Résistance aux collisions** :
  Il est difficile de trouver deux $x \neq x'$ tels que $h(x) = h(x')$.

Ces propriétés assurent la base de la sécurité des fonctions de hachage utilisées en cryptographie.

Familles historiques de fonctions de hachage

MD5 (Message Digest 5)

- Taille hash : 128 bits (RFC 1321, 1992).
- Déconseillé : collisions pratiques découvertes (Wang et al., 2004) : [a].

[a] X. Wang et al., "How to Break MD5 and Other Hash Functions", CRYPTO 2005

SHA-1 (Secure Hash Algorithm 1)

- Taille hash : 160 bits (NIST FIPS 180-4, 1995).
- Risqué : Collision en 2017 (SHAttered attack) [a].

[a] Stevens et al., "The first collision for full SHA-1", 2017

SHA-2 (Famille sécurisée)

- Variantes : SHA-256, SHA-512 (NIST FIPS 180-4).
- **Sécurisé** : Aucune collision pratique connue (2024).
- Utilisation : Bitcoin (SHA-256), TLS 1.2/1.3.

SHA-3 : Le nouveau standard (Keccak)

Conception révolutionnaire

- Basé sur une **éponge** (sponge construction) [a].
- Standardisé en 2015 (NIST FIPS 202) [b].
- Variantes principales:
  1. SHA3-224/256/384/512 bits de tailles fixes.
  2. SHAKE 128/256 : Sortie extensible (XOF).

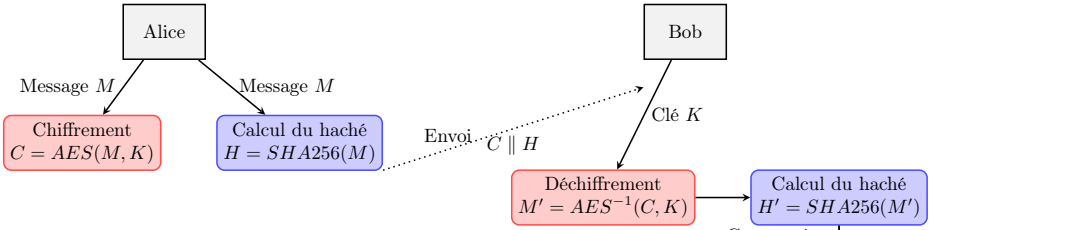
[a] https://keccak.team/sponge_duplex.html

[b] https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.202.pdf

Pourquoi SHA-3 ?

- Alternative à SHA-2 (non remplaçant !).
- Résilience contre futures attaques quantiques.

Fonction de hachage et integrité (cas illustrative)



Légende :

- Opérations avec clé (confidentialité)
- Calcul de haché (intégrité)
- Vérification réussie
- Échec de vérification

Message Authentification Code (MAC)

Définition

$$MAC(K, M) = T$$

Le code d'authentification de message (MAC) est un outil cryptographique symétrique garantissant l'intégrité : il génère un tag T à partir d'un message M et d'une clé secrète K, assurant que le message n'a pas été modifié.

Propriété : Toute modification de M invalide T.

HMAC - Keyed-Hashing for Message Authentication

$$\text{HMAC}(k, m) = h\Big((k \oplus \text{opad}) \parallel h((k \oplus \text{ipad}) \parallel m)\Big)$$

Symboles:

- k : clé secrète
- m : message
- h : fonction de hachage (SHA-256)
- $\oplus$: XOR bit à bit
- $\parallel$: concaténation

Constantes:

- ipad = `0x36` répété
- opad = `0x5C` répété

Étapes:

1. Créer la clé interne: $k \oplus$ ipad
2. Hacher avec message: h(clé_int $\parallel$ m)
3. Créer la clé externe: $k \oplus$ opad
4. Hacher le tout: h(clé_ext $\parallel$ resultat_étape2)

But:

- Double protection cryptographique
- Résiste même aux faiblesses de h

HMAC : Combiner hachage et clé

Avantages clés

- Résiste aux attaques (extensions, collisions)
- Rapide et standardisé (SHA-256/3, RFC 2104)
- Plus léger que les signatures numériques (dont nous allons voir prochainement)

Applications typiques

- Sécurité API (JWT, Webhooks)
- TLS 1.2/1.3 et VPNs
- Vérification de fichiers

Pourquoi utiliser HMAC ?

Garantit l'origine ET l'intégrité des données avec une simple clé secrète

[1] RFC 2104 https://datatracker.ietf.org/doc/html/rfc2104
FIPS 198-1 https://csrc.nist.gov/pubs/fips/198-1/final

Empreinte numérique (haché) : le moindre changement compte

Une simple lettre modifiée change complètement le haché (SHA-256)

```
$ echo "Bonjour"  | sha256sum
8dc2a6966f1be1644ec6b1f7223f47e53de5ad05e1c976736d948e7977a13dd3

$ echo "Bonjour!"  | sha256sum
91300e33125e8b4a3ad93b5f21ebda4f355c1db00f1db09e5ff3fa7685fdaaba

$ echo "bonjour!"  | sha256sum
1c30055b0f9a6fed0c4bc575992523cf38a74ad202edec435d88d91f501bf133
```

- Changement : `!` ajouté $\longrightarrow$ hash totalement différent
- Changement : `B` remplacé par `b` $\longrightarrow$ nouveau hash radicalement différent

Toute altération, même infime, est immédiatement détectable avec la fonction de hachage cryptographique

À retenir : Hachage et Intégrité

- Le **chiffrement** garantit la confidentialité, mais pas l'intégrité.
  1. Sans intégrité, un message peut être modifié à l'insu du destinataire.
  2. **Solutions** : fonctions de hachage cryptographiques, MAC, HMAC.
- **Fonction de hachage** :
  - Transforme une donnée en une empreinte de taille fixe (ex. : 256 bits pour SHA-256).
  - **Propriétés essentielles** :
    - Déterminisme : même entrée $\Rightarrow$ même sortie.
    - Diffusion : 1 bit modifié $\Rightarrow$ empreinte complètement différente.
    - Irréversibilité : on ne peut retrouver l'entrée à partir du haché.
- **MAC (Message Authentication Code)** : assure l'intégrité grâce à une clé secrète partagée.
- **HMAC (Hashed MAC)** : version renforcée utilisant une fonction de hachage et deux clés dérivées (`ipad`, `opad`) pour plus de sécurité.
- **Algorithmes standards** :
  - Fiables : SHA-256, SHA-512 (SHA-2), SHA-3
  - Obsolètes : MD5, SHA-1 (vulnérables aux collisions)