Hachage

Principes

Le hachage est un procédé de **chiffrement** destiné à remplacer une donnée de taille quelconque par une autre donnée de taille fixe, relativement réduite, le **hash-code**. Ceci quelle que soit la donnée initiale.

Il existe de très nombreux algorithmes de hachage qui se caractérisent notamment par leur rapidité et la taille (en nombre de bits) du hash-code produit.

Exemples :
Hachés avec l'algorithme CRC32

- l'entier **12** donne le hash-code **4f5344cd**, tandis que
- la chaîne “**Bonjour, ceci est un hash-code**” donne le hash-code **5ca9b6ed**

Intérêts du hachage

- Le hachage permet de masquer une information : remplacement de la donnée par son hash-code ;
- C'est un procédé « boite noire » : pour s'en servir, nul besoin de savoir comment le résultat est fabriqué ;
- L’usage est évolutif : choix de l’algorithme, choix de la taille du hash-code ;
- Il s'agit d'un procédé qui assimile le résultat à une **Signature** :
  - une information ⇒ un hash-code (toujours le même) ;
  - deux informations ⇒ deux hash-codes (en théorie);
  - l’algorithme est non-réversible : impossible de recalculer la donnée d'origine à partir du hash-code ;

Risques liés au hachage

- Les « **collisions** » : l'ensemble des hash-codes étant fini, il en résulte la possibilité que deux données distinctes produisent le même hash-code. Ce qui n'est pas nécessairement un problème mais, du point de vue d'un pirate, plutôt une qualité car une clé peut ouvrir plusieurs portes ;
- Une **exposition de l’information non-hachée** : toute donnée hachée connait un moment d'existence non-hachée. Ce moment est une fragilité et il doit être raccourci le plus possible en effaçant au plus vite l'information non-hachée de manière définitive ;
- **Vol de base de données** : lorsqu'une base de données contenant des informations hachées est volée, le hachage bien que non-reversible est fragilisé si :
  - il est appliqué de manière homogène sur toutes les données ;
  - il est appliqué au moyen d'un algorithme rapide.

Applications du hachage

- **Comparaison de fichiers** : quand on télécharge un fichier sur Internet, on n'est jamais certain qu'il n'a pas été modifié par rapport à l'original. Ce qui pourrait mettre en cause la sécurité du réseau. La publication des signatures de fichiers permet de s'assurer d'un fichier non modifié ;
- **Indexation de données** : le hash-code étant très compact, il peut être utilisé pour classer et indexer efficacement des données à partir de leur signature. C'est le principe qu'utilisent les dictionnaires et hashtables en programmation ;
- **Certificats numériques** : les certificats s'appuient sur le hachage pour délivrer des signatures authentifiant émetteur et organisme de certification ;
- La **BlockChain** : cette technologie s'appuie sur le hachage pour authentifier les transactions.
- etc.

Salage des mots de passe

Le hachage des mots de passe a beau être un procédé à haut niveau de sécurité lorsqu'il est convenablement mis en place (non-réversibilité, taille du hash-code élevée, algorithme lent, etc.), il n'en reste pas moins que l'ensemble comporte une fragilité particulière dans l'hypothèse du vol d'une base de données complète. Dans ce cas, un individu malveillant disposant de beaucoup de temps et d'un outillage spécialisé (**tables arc-en-ciel**) pourrait compromettre de nombreux mots de passe en s'appuyant sur l'homogénéité du hachage appliqué.

Pour contrer cette capacité, on appliquera un **salage** des mots de passe. Le salage consiste à ajouter une information aléatoire (le sel) au mot de passe avant son hachage. Le sel étant variable d'un hachage à l'autre, on aura un hash-code différent pour le hachage du même mot de passe à deux moments différents. Dans, ce cas, les tables arc-en-ciel deviennent inopérantes.

A titre d'exemple, PHP propose une fonction **password_hash** qui intègre un hachage BCrypt et le salage automatique.