Fonctions de hachage

Fonction de hachage

Toute donnée informatique peut être représentée par une suite finie de 0 et de 1. On note $\{0,1\}^* = \bigcup_{k\geq 0}\{0,1\}^k$ l'ensemble de ces suites.

On fixe un entier N.

Une *fonction de hachage* (en: *hash function*) de longueur N est une fonction $h : \{0,1\}^* \to \{0,1\}^N$ sans collision.

Une *collision* est une paire de mots $x \neq y$ tels que $h(x) = h(y)$.

Tiroirs

Telle que "définie" plus haut, une fonction de hachage n'existe pas puisque $\{0,1\}^*$ est infini et que $\{0,1\}^N$ est fini.

L'objectif est d'éviter les collisions.

Paradoxe des anniversaires

On peut se rabattre sur une approche statistique.

Si la fonction h "mélange" suffisamment, alors si on teste k entrées différentes et que k est très petit devant $\sqrt{2^N}$, alors on a très peu de chances de tomber sur une collision.

Remarque : $\sqrt{2^N} = 2^{N/2}$

Limite calculatoire

Une fonction de hachage est dite *cryptographique* s'il est en pratique impossible de construire une collision en temps raisonnable avec les moyens de calcul actuels.

Les fonctions `md5sum` et `sha1sum` ne sont plus réputées sures.

Les fonctions `sha256sum` et `sha512sum` sont encore réputées sures.

S'il est impossible en pratique de construire des collisions, alors pour une chaîne $x \in \{0,1\}^*$ donnée, il n'est pas non-plus possible de construire une autre chaîne $y \neq x$ qui a le même hash que x.

Application : table de hachage

Voir le cours "structures de données et algorithmes". Les fonctions de hachages utilisées pour de telles tables ne sont en général pas cryptographiques et doivent pouvoir être évaluées rapidement.

Application : intégrité

Exemple : le CRC dans les trames ethernet (voir le cours de système et réseaux).

Exemple : Quand vous téléchargez un gros fichier, il est possible que des octets aient été modifiés sur la route. Vérifier le hash permet de s'assurer que ça n'est pas le cas. Si vous craignez que le hash soit lui aussi corrompu, vérifiez sa signature. Consulter la démo "Téléchargement et vérification d'une image ISO".

Application : signature de gros fichiers

La signature (cryptographie asymétrique) coûte très cher en ressources sur des gros fichiers. Ainsi, la signature d'un gros fichier est en fait la signature d'un hash du gros fichier.

Application : fingerprint

Le fingerprint d'une clef publique est son hash (et le nom de la fonction de hachage utilisée précède en général le hash, par exemple : `SHA256:9j1WU56gjFkSACA3zILrVv2EdTZ2Aphs2sxuas/P004`).

Il est plus facile de vérifier des fingerprints en les alignant dans un fichier que de vérifier les clefs publiques, car le fingerprint est plus court que la clef.

Application : stockage de mots de passe

Voir `/etc/shadow`, `man 5 shadow` puis `man 3 crypt`

Attention : collision vs préimage.

Rainbow table, salting.

Application : attribution des adresses IPv6 sur les conteneurs

La machine hôte dispose du préfixe IPv6 `2001:910:1410:523`. Pour attribuer une adresse IP à chaque conteneur, on pourrait maintenir une table dans un fichier en numérotant les étudiant·es et en leur affectant une adresse IP, dans l'ordre `2001:910:1410:523::1`, `2001:910:1410:523::2`, `2001:910:1410:523::3`, ...

C'est possible, mais ça demande de maintenir ce fichier à jour pour ne pas attribuer deux fois la même adresse IP.

Au lieu de ça, on attribue une adresse IP de la façon suivante : chaque étudiant·e se voit attribuer l'adresse `2001:910:1410:523::fada:****:****` où `********` sont les 8 premiers caractères hexadécimaux du `sha256sum` de votre `prenom.nom` apparaissant dans votre adresse mail.

Vous pouvez tester sur l'adresse IP de votre propre conteneur.

Il y a bien sur un risque qu'une même adresse IP soit attribuée à deux conteneurs. Néanmoins, ce risque est très faible : chaque caractère hexadécimal peut prendre 16 valeurs différentes, se sorte qu'il y a 16^8 adresse IP différentes. Si on considère que le nombre d'étudiant·es est

Application : dépendance, DAG, chaîne de blocs

Si un objet A dépend d'un autre objet B immutable, la dépendance peut s'inscrire en ajoutant dans le contenu de A, le hash de B.

Application : vérification d'une information partielle

Documentez-vous sur les arbres de Merkle.

Application : preuve de travail

Si $s \in \{0,1\}^*$ est donné, trouver un $t \in \{0,1\}^*$ tel que $h(st)$ commence par k zéros demande de calculer de l'ordre de 2^k hashes.

Application des 3 applications précédentes : blockchains

Pour les plus motié·es, consulter directement l'article
https://bitcoin.org/bitcoin.pdf