Cégep de Trois-Rivières

Projet Scientifique

Les Fonctions de Hachage

Gabriel-Andrew Pollo-Guilbert

Travail remis à
Sylvain Marcoux

21 mars 2016

1 La Fonction de Hachage

Une fonction de hachage permet de calculer une empreinte mathématique à partir d'un ensemble de données. Soit S un ensemble de nombres entiers et S_n l'élément n dans cet ensemble, alors $h(S)$ est une fonction de hachage si elle retourne un nombre entier.

Selon cette définition, on pourrait définir une telle fonction $h(S)$ comme la somme des élèments d'un ensemble :

$$h(S) = \sum_{n=0}^{|S|} S_n, \tag{1}$$

où $|S|$ dénote la cardinalité de l'ensemble S.

1.1 La propagation d'erreurs

Selon un but précis, il est souvent préférable d'avoir une fonction de hachage ayant certaines propriétés. En général, une telle fonction doit avoir la propriété de propager une erreur. Il en résulte que l'empreinte générée diffère beaucoup si un élèment est altéré.

Soit une fonction de hachage $h(S)$ propageant une erreur, et A et B des ensembles identiques avec un seul élément différent, alors $h(A)$ doit être considérablement différent de $h(B)$.

La fonction en (1) n'a pas une telle propriété, car un chiffre mal transmis dans un élèment peut au maximum affecter deux positions décimales.

Tableau 1 – propagation peu efficace d'une erreur dans (1)

Ensemble	**Empreinte**
$\{1, 2, 3, ..., 37, ..., 100\}$	5050
$\{1, 2, 3, ..., \mathbf{47}, ..., 100\}$	50**6**0
$\{1, 2, 3, ..., \mathbf{36}, ..., 100\}$	50**49**

Les fonctions de hachage couramment utilisées ont tous cette propriété. Cela permet de confirmer si l'ensemble ou l'empreinte fut mal transmise. Si l'empreinte n'est pas bien reçu, alors l'empreinte va rester similaire. Si l'ensemble est mal reçu, alors l'empreinte va être totalement différente.

Tableau 2 – propagation efficace d'une erreur dans les algorithmes standards

Message	**Algorithme**		
	MD5	**CRC32**	**SHA512**
`Fonction`	`24A1B0 ... 237288`	`DE616B0B`	`E3545D ... 93C30E`
`Fonc`**`T`**`ion`	`56CA42 ... FE452B`	`B623E2D2`	`6E40C9 ... 69870C`
`Fon`**`d`**`tion`	`22BE03 ... 6AD563`	`36046ECB`	`6A94C9 ... 877189`

1.2 Les collisions

Un autre problème avec (1) est qu'il est facile de générer deux ensembles ayant une même empreinte. Une telle situation est appelée une collision. Soit A et B des ensembles, alors A est en collision avec B si $h(A) = h(B)$.

Il est possible de réduire la probabilité que deux ensembles entrent en collision en définissant une fonction plus complexe. Entre autres, l'ajout d'un opérateur modulo dans la somme tel que

$$h(S) = \sum_{n=0}^{|S|} S_n \bmod m, \tag{2}$$

où $m \in \mathbb{Z}$, peut diminuer le risque de collision. Malgré l'ajout de complexité, il encore possible de générer une collision dans (2), à l'aide de la théorie des nombres, dans un temps résonnable.

Une fonction de hachage où chaque ensemble est relié à une empreinte unique est dite parfaite. Soit F une famille d'ensemble et F_n l'ensemble n dans cet famille, alors I est l'ensemble d'empreinte créé par $h(F_n)$, $\forall F_n \in F$. Une fonction $h(F_n)$ est parfaite si et seulement si F est en injection avec I de sorte qu'il n'y ait aucune collision.

1.3 La résistance aux collisions

La majorité des fonctions de hachage utilisées retournent un résultat d'une longueur fixe. Donc, la quantité d'empreinte possible est limitée, tandis que l'ensemble peut être d'une cardinalité arbitraire. Alors, il est inévitable qu'il y ait une infinité de collisions. [1]

Hors, la problématique de la collision ne provient pas du fait qu'elles existent, mais du fait qu'elles peuvent être générées, ou trouvées. Alors, une fonction de hachage est dite résistante aux collision s'il est difficile de trouver une collision dans un temps résonnable.

1.4 Utilisation des fonctions de hachage

La sécurité digitale dépend de la résistance aux collisions. En effet, les bases de données enregistrent seulement l'empreinte des mots de passe. [2] Cela empêche l'administrateur, un employé ou a un pirate d'avoir accès aux mots de passe. Si la fonction utilisée ne résiste pas aux collisions, alors il est facile de recouvrir le mot de passe à partir de l'empreinte.

1. Cette conclusion provient du principe des tiroirs. Soit A et B des ensembles où $|A| > |B|$, et une fonction $h : A \mapsto B$. Alors, il existe au moins deux antécédents à un élèment de B.[4]

2. Il y a eu des incidents où une base de données enregistrait les mots de passe sous texte brute. Hors, c'est une pratique qui ne devrait jamais être utilisée dans le cas où un pirate accèdait au serveur.[1][2]

Les fonctions de hachage sont aussi utilisées pour vérifier la signature d'un fichier. Un pirate ayant accès à un serveur peut remplacer un fichier par un programme malveillant. Donc, l'utilisateur téléchargeant à partir de ce serveur est à risque. Afin de vérifier que le fichier provient d'une source fiable, comme le développeur d'un programme, il est souvent recommandé de vérifier l'empreinte du fichier avec celle fournit par le développeur. [3] Encore ici, la résistance aux collisions est une propriété requise à la fonction qui hachera le fichier.

Finalement, une fonction de hachage peut être utilisée pour simplement vérifier si un message ou un fichier fut bien transmis sans tenir compte de la validité de la source. Dans un tel cas, la résistance aux collisions n'est pas requise. Dans la majorité des cas, les fonctions de hachage devront avoir la propriété de propager une erreur.

2 L'algorithme MD5

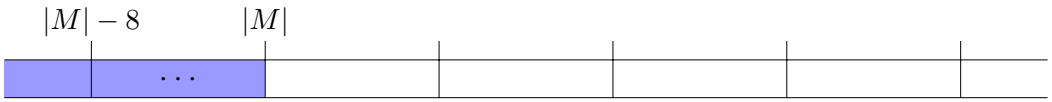
MD5 (*Message Digest 5*) est une fonction de hachage utilisée dans le domaine informatique afin d'obtenir l'empreinte d'un fichier ou d'un mot de passe. Elle a été publiée en 1992 par Ronald Riverst avec le but d'être le plus possible résistante aux collisions.[3]

Ce n'est que quelques années plus tard, en 1996, qu'une première vulnérabilité fut découverte. En 2004, une attaque complète fut développée afin de générer des collisions dans un temps résonnable.[5]

Aujourd'hui, l'algorithme n'est pas recommandé dans les situations où la sécurité est une priorité, comme dans les bases de données, mais elle reste très utilisée afin de vérifier un transfert d'information. Ses grandes puissances sont qu'elle est simple, rapide et optimisée pour les ordinateurs 32-bits.[3]

2.1 Algorithme

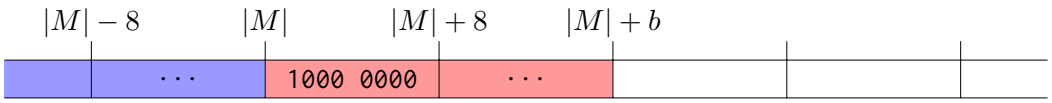
Soit un ensemble M binaire d'une taille $|M|$, où M peut être la représentation binaire d'un message. L'algorithme se divise en deux étapes : on étend le message et on transforme le message par bloc de 512-bits.



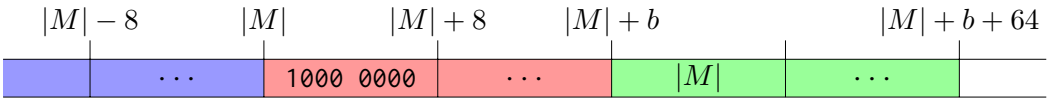
3. Cette vérification est automatique dans les systèmes d'exploitations ayant recours à un répertoire de logiciel signé tel que iOS (*App Store*), Android (*Google Play*) ou les multitudes de variantes UNIX (*apt-get*, *pacman*, *portage*, *pkg*, *yum*, *brew*, etc).

2.2 Extension du message

Puisque les transformations sont appliquées par bloc de 512-bits, il faut s'assurer que $|M|$ soit un multiple de 512. Un premier bit d'une valeur de 1 est ajouté à la fin du message. Ensuite, une quantité b de bits nulles sont ajouté afin que $|M|+b \equiv 448 \mod 512$. [4]



Après ces étapes, il manque 64-bits afin que $|M|+b$ soit un multiple de 512. Les 64-bits suivants sont comblés par l'ajout de $|M|$ au message. La plus grande taille possible à ajouter est donc de 2^{64}. Si $|M| > 2^{64}$, alors seulement les 64 premiers bits de $|M|$ sont ajoutés. Finalement, la taille résultante $|M|+b+64$ est un multiple de 512.



2.3 Transformation du message

L'empreinte retournée par l'algorithme est d'une longueur de 128-bits. Elle est divisée en 4 registres de 32-bits : A, B, C et D. Chaque régistre est initialisé à une valeur précise prédéfinie. L'algorithme spécifie 4 fonctions auxiliaires sur les registres :

$$\begin{aligned} F(B,C,D) &= (B \wedge C) \vee (\neg B \wedge D) \\ G(B,C,D) &= (B \wedge D) \vee (C \wedge \neg D) \\ H(B,C,D) &= B \oplus C \oplus D \\ I(B,C,D) &= C \oplus (B \vee \neg D) \end{aligned}$$

Pour chaque bloc M_{512n} de 512-bits, où $n \in \mathbb{N}$, 64 itérations sont effectuées sur les registres avec une fonction de la forme :

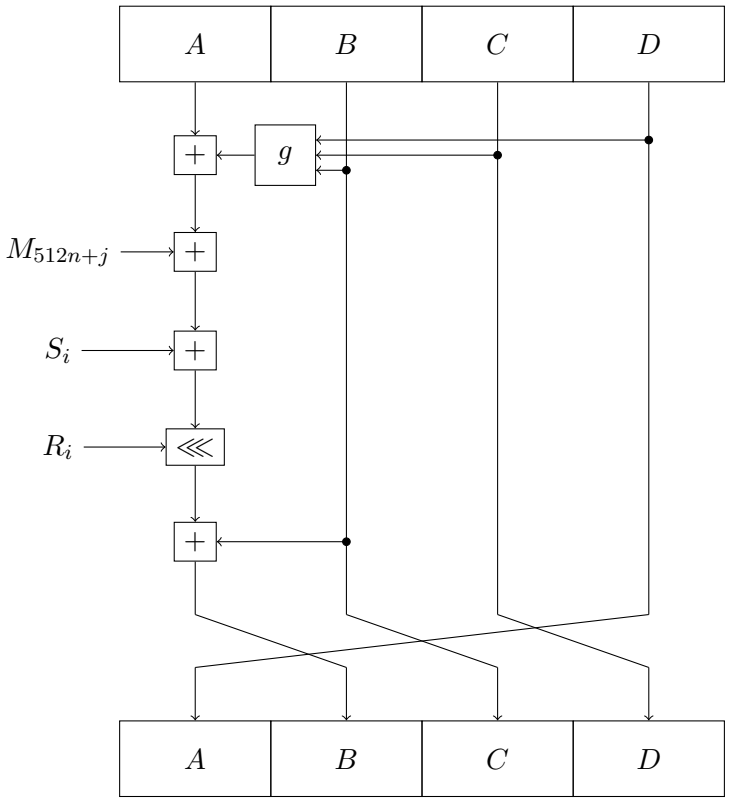
$$f(A,B,g(B,C,D),i,j) = B + [A + g(B,C,D) + S_i + M_{512n+j}] \lll R_i$$

où $i,j \in \mathbb{N}$ varient selon une séquence prédéterminée, $g(B,C,D)$ est une fonction auxiliaire, $\lll n$ dénote une rotation circulaire de n bits vers la gauche et, S et R sont des ensembles d'entier positif prédéterminés.

4. La plus petite valeur qu'un ordinateur 32-bits peut manipuler directement est 8-bits. Donc, $|M|$ est un multiple de 8. Il est plus efficace d'ajouter $128_{10} = 80_{16} = 1000\,0000_2$ que d'ajouter 1 et et ensuite les 0s.

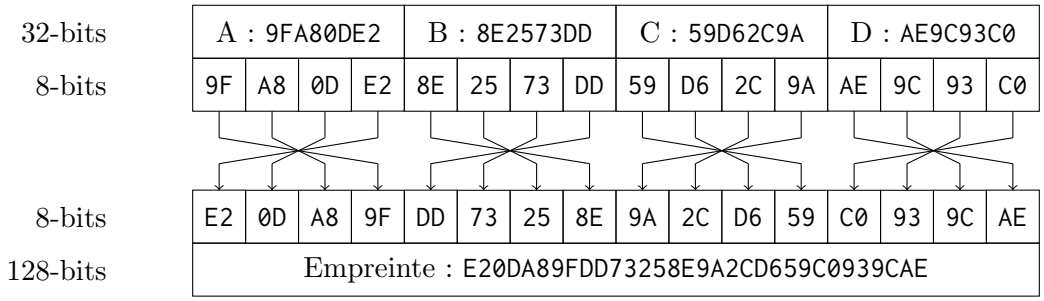
Une fois cette fonction calculée, on procède à une permutation des registres tel que $A = D$, $B = A$, $C = B$ et $D = C$. Finalement, on remplace B par le résultat de la fonction tel que $B = f(A, B, g(B, C, D), i, j)$.

FIGURE 1 – une itération d'une transformation MD5



Finalement, l'empreinte est générée en écrivant les registres dans l'ordre, mais les octets les moins significatifs de la gauche à droite comme dans la prochaine figure.

FIGURE 2 – manipulation des octets



Références

[1] Charles Arthur. *Google Chrome Security Flaw Offers Unrestricted Password Access*. 2013. URL : http://www.theguardian.com/technology/2013/aug/07/google-chrome-password-security-flaw (visité le 20/03/2016).

[2] Thomas Fox-Brewster. *13 Million Passwords Appear To Have Leaked From This Free Web Host*. 2015. URL : http://www.forbes.com/sites/thomasbrewster/2015/10/28/000webhost-database-leak (visité le 21/03/2016).

[3] Ronald Rivest. *The MD5 Message-Digest Algorithm*. RFC 1321. Avr. 1992. URL : https://www.ietf.org/rfc/rfc1321.txt.

[4] Wikipedia. *Principe des Tiroirs*. 2016. URL : https://fr.wikipedia.org/wiki/Principe_des_tiroirs (visité le 21/03/2016).

[5] Xiaoyun Wang, Hongbo Yu. *How to Break MD5 and Other Hash Functions*. Recherche. 2005. URL : http://merlot.usc.edu/csac-f06/papers/Wang05a.pdf.