Collisions MD5

Version	1.0
Date	2 Janvier 2014
Rédigé par	Yves Nouafo
Relu par	
Approuvé par	Magali Bardet

MISES À JOUR

Version	Date	Modifications réalisées
1.0	02/01/2014	Création

Table des matières

1 Introduction

Ces dernières années, le progrès en terme de sécurité informatique a été fulgurante. Plusieurs méthodes de cryptographie ont été développée. Ces méthodes moderne repose sur les fonctions de hachages cryptographiques. Ce sont des fonctions qui associent à un message de longueur arbitraire une chaine d'octets de longueur fixe appelé *hashé*.

Les fonctions de hachage doivent satisfaire les principes suivant :
- résistante au calcul de collision
- résistante au calcul de seconde pre-image
- résistante au calcul de pré-image

Une des fonctions de hachage les plus utilisés est MD5, inventé par Rivest en 1992. MD5 comme toute fonction de hachage prend en entrée un message de longueur variable et retourne une chaine de 128 bits.

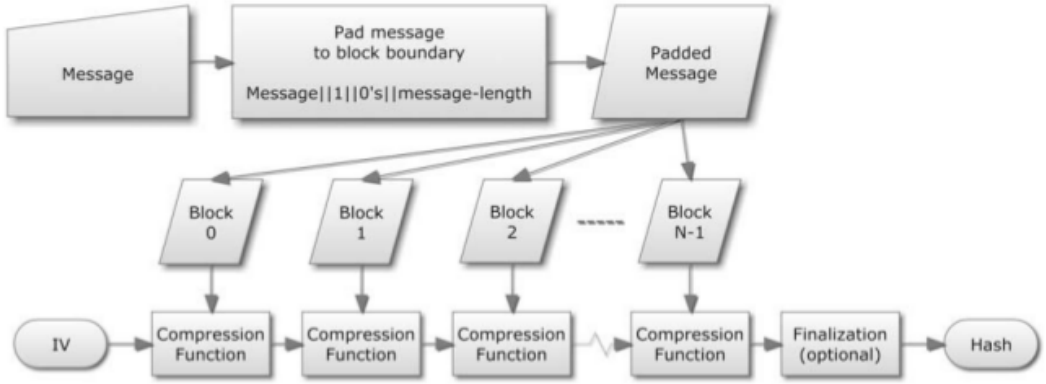
Le but de cette étude est de décrire le fonctionnement interne de la fonction de hachage MD5 et d'en réaliser un état de l'art sur sa résistance aux collisions. Plus précisément, comment générer des entrées différentes pour la fonction MD5 de telle sorte que ces deux entrées aient le même hashé MD5. Dans une étude plus approfondie, nous verrons comment il est possible d'appliquer l'étude à un cas pratique : La collision sur des fichiers.

2 Préliminaire

Comme la plupart des fonctions de hachage, MD5 est dotée d'une fonctionde compression qui s'appuie sur le fonctionnement de Merkle-Damgard, dont le principe est le suivant :

- La fonction de compression prend en entrée un bloc B de taille 512 bits et une IHV (valeur de hachage intermmédiaire) de 128 bits
- Chaque message passé en paramètre doit subir un padding si sa taille n'est pas multiple de 512.
- Le message (complété ou non) est ensuite découpé en N blocks de 512 bits.
- La fonction de hachage commence avec un IHV initial appelé IV (initial value).
- pour chaque bloc de message M_i la fonction de compression de est appelé avec la valeur courante IHV_i et calcul une nouvelle valeur IHV_{i+1}. Lorsque tout les blocs sont traité le *hashé* final IHV_N est construit.

Le schéma de fonctionnement est le suivant :



3 Fonctionnement de MD5

Comme nous l'avons vu dans la section précédente, on sait que MD5 suit le schéma de construction de Merkle-Damgard et fonctionne de la manière suivante :

1. *le padding.* Si la longueur du message n'est pas un multiple de 512, on ajoute un pad, dont le premier bit est à 1 suivi de 0 de telle sorte que la longueur résultante soit égale à 448 mod 512. Les bits restants sont utilisés pour ajouter la longueur du message initial ;
2. *le partitionnement.* MD5 découpe le message original résultant d'un padding ou non en N blocs M_i, ...,M_N de 512 bits ;
3. *le processus.* MD5 calcule des valeurs intermémdiaire de hash, *IHV*. Chaque IHV est diviser en 4 mots de 32 bits (a, b, c, d). (a_0, b_0, c_0, d_0) = (67452301_{16}, $EFCDAB89_{16}$, $98BADCFE_{16}$, 10325476_{16}) sont des valeurs publiques initiales fixées, chaque IHV calculée en utilisant la fonction de compression MD5Compress comme suit :
   $IHV_i = \text{MD5}(IHV_{i-1}, M_i)$;
4. *le résultat.* La valeur du hash résultant est la dernière valeur IHV calculée. Cet IHV est la concaténation en hexadécimal des dernières valeurs (a, b, c, d) calculés.

4 La fonction de compression MD5

La fonctionde compression de MD5 prend en paramère une IHV de 128 bits et un bloc de message de 512 bits. Elle s'effectue en 64 étapes découpés en 16 tours. Chaque étape est munies d'opérations telle que l'addition,
la rotation gauche RC telle que :

$$(RC_t,\ RC_{t+1},\ RC_{t+2},\ RC_{t+3}) = \begin{cases} (7, 12, 17, 22) & pour\ \ t = 0, 4, 8, 12 \\ (5, 9, 14, 20) & pour\ \ t = 16, 20, 24, 28 \\ (4, 11, 16, 23) & pour\ \ t = 32, 36, 40, 44 \\ (6, 10, 15, 21) & pour\ \ t = 48, 52, 56, 60 \end{cases}$$

une fonction non-linéaire f_t telle que :

$$f_t(\text{x, y, z}) = \begin{cases} F(x,y,z) = (x \bigwedge y) \bigoplus (\bar{x} \bigwedge z) \\ G(x,y,z) = (z \bigwedge x) \bigoplus (\bar{z} \bigwedge y) \\ H(x,y,z) = x \bigoplus y \bigoplus z \\ I(x,y,z) = y \bigoplus (x \bigvee \bar{z}) \end{cases}$$

Le bloc de message B est coupé en 16 mots consécutif M_0, ..., M_{15} et étendu à 64 mot W_t pour 0 <= t <= 64 de 32 bits chacun telle que :

$$W_t = \begin{cases} réecritrelesmt \end{cases}$$

En suivant la description de la fonction de hachage de MD5, pour chaque étape t, l'algorithme de fonction de compression sauvegarde un registre avec 4 états de mots Q_t, Q_{t-1}, Q_{t-2}, Q_{t-3} et calcul un nouveau mot Q_{t+1}. On a alors l'initialisation suivante :
$(Q_t, Q_{t-1}, Q_{t-2}, Q_{t-3})$ = (b, c, d, a)
for$t$ = 0, 1, ..., 63, Q_t est calculé comme suit :

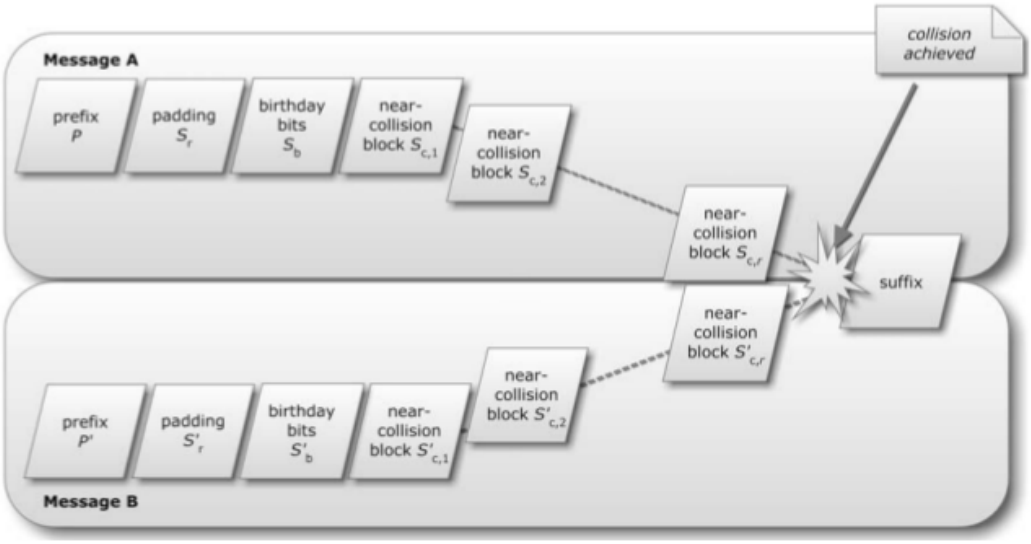
$$\begin{cases} F(t) = f(t)(Q(t), Q(t-1), Q(t-2) \\ T(t) = F(t) + Q(t-3) + AC(t) + W(t) \\ R(t) = RL(T(t), RC(t)) \\ Q(t+1) = Q(t) + R(t) \end{cases}$$

5 Présentation d'une collision à prefixe choisi

Cette partie s'inspire des travaux réalisés par Marc Stevens célèbre cryptologue. Marc Stevens reprend les travaux réalisés par Wang et Yu et améliore ainsi leur algorithme en y introduisant d'autre notions et montre comment à partir de deux messages arbitraires, il est possible de générer des collisions sous MD5. Les algorithmes implémentés sont ceux décrit par Stevens. Mais avant tout une brève explication de la méthodologie de Stevens.

5.1 Vue sur les préfixes choisis

Les deux messages initiaux peuvent avoir des longueur différentes. Dans ce cas, il faut appliquer un padding au plus court des deux, pour qu'elles aient des longueurs égales. En agissant de la sorte, lorsque les message seront passé à la fonction de compression de MD5, on s'assure que qu'ils auront le même padding.
//
////////////// COMPLETER /////////////////
//



5.2 Construction des préfixes choisis de collisions

Un préfixe choisi de collision est une paire de messages (M, M') qui consiste à choisir arbitrairement des préfixes P et P', telle que l'on ait des suffixe S et S' construit de telle sorte que :
M = P || S et M' = P' || S' et MD5(M) = MD5(M').

Les suffixes S et S' ont la même structure. Ils sont découpés en 3 parties.
- le padding S_r ;
- les bits d'anniversaire S_b ;
- bit de quasi-collision S_c.

5.2.1 Déterminer les bits d'anniversaires

Les bits d'anniversaires servent à réduire le nombre d'appel de la fonction de commpression de MD5 pour parvenir à trouver une collision. En introduisant cette chaine de bits on parvient ainsi d'un nombre d'appel de 2^{59} à 2^{39}. Ce qui reste largement en dessous du temps limite de calcul que l'on peut faire de nos jours.

//
//

6 Application d'exploitation de collision MD5

L'état de l'art de la collision MD5 faite dans les chapitres précédents peut être appliqué dans la vie réelle. On peut citer par exemple, la collision entre documents, la vérification de l'intégrité d'un logiciel ou encore, en ce qui nous concerne, la création de faux certificats d'autorité.

Dans ce chapitre, nous allons voir comment comment construire construire deux certificats X.509 avec des noms distinctifs mais ayant les mêmes signature électronique.

6.1 Rappel sur les certificats X.509

X.509 est une norme de cryptographie de l'Union internationale des télécommunications pour les infrastructures à clés publiques (PKI). Il établit entre autres les formats standard de certificats électroniques et un algorithme pour la validation de chemin de certification. X.509 à été crée en 1988 et repose sur un système hiérarchique d'autorités de certification, à l'inverse des réseaux de confiance (comme PGP), où n'importe qui peut signer (et donc valider) les certificats des autres.

Dans le système X.509, une autorité de certification attribue un certificat liant une clé publique à un nom distinctif (Distinguished Name), à une adresse électronique ou un enregistrement DNS.

Les certificats racines sont des clés publiques non signées, ou auto-signées, dans lesquels repose la confiance. Des autorités de certification commerciales détiennent des certificats racines présents dans de nombreux logiciels, par exemple les navigateurs Web. Quand le navigateur ouvre une connexion sécurisée (SSL) vers un site ayant acheté une certification auprès d'une autorité connue, il considère le site comme sûr dans la mesure où le chemin de certification est validé. Le passage en mode sécurisé est alors transparent.

Si le certificat est auto-signé (autorité de certification et créateur de la clé publique ne font qu'un), le navigateur propose de l'examiner, puis de l'accepter ou de le refuser selon la confiance qu'on lui accorde.

6.2 Structure d'un certificat

- Version
- Numéro de série
- Algorithme de signature du certificat
- Nom du signataire du certificat
- Validité (dates limite)
  - Pas avant
  - Pas après
- Détenteur du certificat
- Informations sur la clé publique :
  - Algorithme de la clé publique
  - Clé publique proprement dite
- Identifiant unique du signataire (optionnel, à partir de X.509 version 2)
- Identifiant unique du détenteur du certificat (optionnel, à partir de X.509 version 2)
- Extensions (optionnel, à partir de X.509 version 3)
  - Liste des extensions

6.3 Création d'un faux certificat

Lorque l'on génère deux certificats X.509 qui ont des signatures identiques mais des Distinguish Name différent on intervient directement sur le module de la clé publique.

6.3.1 Détails de construction du certificat

1. Il faut creer deux certificats de telle sorte que tous les champs soit rempli excepté, le module de la cle publique RSA et la signature. En se rassurant que :
   - les données doivent être de la forme X.509 ;
   - la longueur d'octets du module et de l'exposant publique doivent être fixé ;
   - contrôler la position où commence la partie le module RSA en rajoutant des informations au Distinguish Name ;
2. appliquer MD5 sur les champs des parties à être signés des certificats de façon à obtenir des IHVs que l'on utilisera pour l'étape qui suit ;
3. On utilise les IHVs et leur blocs correspondants en y ajoutant les bits d'anniversaires plus précisément 96 bits qui n'est autre que la satisfaction des 96 bits de différence entre les IHVs en sortie.
4. En utilisant la méthode de la section 4, il faut calculer la différence de chaine d'octets entre les blocks proche de collision S_c et S'_c de 4096 bits chacun. Comme vu dans la section 5.1 chaque blocs de quasi-collision est utilisé pour élimé la différence entre les IHVs de l'étape précédente de telle sorte que de la différence entre les IHVs soit nulle. À ce stade une collision MD5 à été accompli. S = S_b || S_c et S' = S'_b || S'_c sont alors de longueur 4192 bits sur le module RSA ;
5. en utilisant la methode ..., on construit de module RSA sécurisé de 8192 bits des chaine d'octets S et S' de longueur 4192 bits chacun en y ajoutant une chaine identique de 4000 bits ;
6. on complète le premier certificat en y inserant les informations de la clé publique. On calcule ensuite le hash MD5 de "la partie à être signé" et on l'utilise pour calculer la signature que l'on ajoutera au premier certificat et ainsi finir sa construction ;
7. ajouter les informations de la clé publique et la même signature dans le second certificat pour compléter sa construction.

Comme on peut le voir, les étapes 3 et 4 sont primordiales mais également les plus difficile, car c'est que les préfixes choisis sont construit comme montré dans la section 5.2.

ANNEXES

Given n-block $P\|S_r\|S_b$ and $P'\|S'_r\|S'_b$, the corresponding resulting IHV_n and IHV'_n, and a value for w, a pair of bitstrings S_c, S'_c is constructed consisting of sequences of near-collision blocks such that $M = P\|S_r\|S_b\|S_c$ and $M' = P'\|S'_r\|S'_b\|S'_c$ satisfy $\text{MD5}(M) = \text{MD5}(M')$. This is done by performing in succession steps 1, 2 and 3 below.

1. Let $j = 0$ and let S_c and S'_c be two bitstrings of length zero.
2. Let $\delta\text{IHV}_{n+j} = (0, \delta b, \delta c, \delta c)$. If $\delta c = 0$ then proceed to step 3. Let $(k_i)_{i=0}^{31} = \text{NAF}(\delta c)$ and $(l_i)_{i=0}^{31} = \text{NAF}(\delta b - \delta c)$. Choose any i for which $k_i \neq 0$ and let $w' = \min(w, 31 - i)$. Perform steps (a) through (f):
   (a) Increase j by 1.
   (b) Let $\delta S_{c,j} = (\delta m_0, \delta m_1, \ldots, \delta m_{15})$ with $\delta m_{11} = -k_i 2^{i-10 \bmod 32}$ and $\delta m_t = 0$ for $0 \leq t < 16$ and $t \neq 11$.
   (c) Given $\delta\text{IHV}_{n+j-1} = \text{IHV}'_{n+j-1} - \text{IHV}_{n+j-1}$ and $\delta S_{c,j}$, construct a few differential paths based on Table 2 with
   $$\delta Q_{61} = 0, \quad \delta Q_{64} = -k_i 2^i - \sum_{\lambda=i+21 \bmod 32}^{i+21+w' \bmod 32} l_\lambda 2^\lambda, \quad \delta Q_{63} = \delta Q_{62} = -k_i 2^i.$$
   How this is done is described in Sections 4.3 and 4.4.
   (d) Find message blocks $S_{c,j}$ and $S'_{c,j} = S_{c,j} + \delta S_{c,j}$ that satisfy one of the constructed differential paths. How this is done is described in Section 4.5. If proper message blocks cannot be found, back up to step (c) to find more differential paths.
   (e) Compute $\text{IHV}_{n+j} = \text{MD5Compress}(\text{IHV}_{n+j-1}, S_{c,j})$, $\text{IHV}'_{n+j} = \text{MD5Compress}(\text{IHV}'_{n+j-1}, S'_{c,j})$, and append $S_{c,j}$ and $S'_{c,j}$ to S_c and S'_c, respectively.
   (f) Repeat step 2
3. Let $\delta\text{IHV}_{n+j} = (0, \delta\hat{b}, 0, 0)$. If $\delta\hat{b} = 0$ then terminate. Let $(l_i)_{i=0}^{31} = \text{NAF}(\delta\hat{b})$. Choose i such that $l_i \neq 0$ and $i - 21 \bmod 32$ is minimal and let $w' = \min(w, 31 - (i - 21 \bmod 32))$. Perform steps (a) through (e) as above with $\delta m_{11} = 2^{i-31 \bmod 32}$ as opposed to $\delta m_{11} = -k_i 2^{i-10 \bmod 32}$ in step (b) and in steps (c) and (d) with
   $$\delta Q_{61} = 0, \quad \delta Q_{64} = 2^{i-21 \bmod 32} - \sum_{\lambda=i}^{i+w' \bmod 32} l_\lambda 2^\lambda, \quad \delta Q_{63} = \delta Q_{62} = 2^{i-21 \bmod 32}.$$
   Perform steps (a) through (e) again with $\delta m_{11} = -2^{i-31 \bmod 32}$ in step (b) and
   $$\delta Q_{61} = 0, \quad \delta Q_{64} = \delta Q_{63} = \delta Q_{62} = -2^{i-21 \bmod 32}$$
   in steps (c) and (d). Repeat step 3.