Module EA4 – Éléments d'Algorithmique

Dominique Poulalhon
dominique.poulalhon@liafa.univ-paris-diderot.fr

Université Paris Diderot
L2 Informatique, Math-Info et EIDD
Année universitaire 2013-2014

Contrôle continu

Interrogation n° 3 *facultative* lundi 5 mai

Règles du jeu :

- la note obtenue compte quoi qu'il arrive pour le calcul de la note de CC
- inscription obligatoire (via Didel, mail à suivre)
- modalités précises (amphi, durée...) : voir Didel

Construction de fonctions de hachage

une fonction de hachage doit

- être facile à calculer
- remplir la table *uniformément*, donc *disperser les données similaires*

Construction de fonctions de hachage

une fonction de hachage doit

- être facile à calculer
- remplir la table *uniformément*, donc *disperser les données similaires*

deux étapes

- transformer toute donnée en valeur numérique (entière)
- hacher les nombres

Construction de fonctions de hachage

une fonction de hachage doit

- être facile à calculer
- remplir la table *uniformément*, donc *disperser les données similaires*

deux étapes

- transformer toute donnée en valeur numérique (entière)
- hacher les nombres

pour du texte, le plus simple : remplacer chaque caractère par son code ASCII, et considérer le texte $t_0 \dots t_m$ comme l'entier

$$h(t_0 t_1 ... t_m) = t_0 b^m + t_1 b^{m-1} + \cdots + t_{m-1} b + t_m$$

(en Java : $b = 31$)

Construction de fonctions de hachage

une fonction de hachage doit

- être facile à calculer
- remplir la table *uniformément*, donc *disperser les données similaires*

méthode *par division*

$$h(x) = x \mod m$$

Construction de fonctions de hachage

une fonction de hachage doit

- être facile à calculer
- remplir la table *uniformément*, donc *disperser les données similaires*

méthode *par division*

$$h(x) = x \mod m$$

méthode *par multiplication*

$$h(x) = \lfloor m \times \{Ax\} \rfloor$$

avec $\{x\} = x - \lfloor x \rfloor$

- m a peu d'importance, par exemple une puissance de 2
- une bonne valeur pour A est $\frac{\sqrt{5}-1}{2}$ (ou une approximation fractionnaire)

Hachage cryptographique

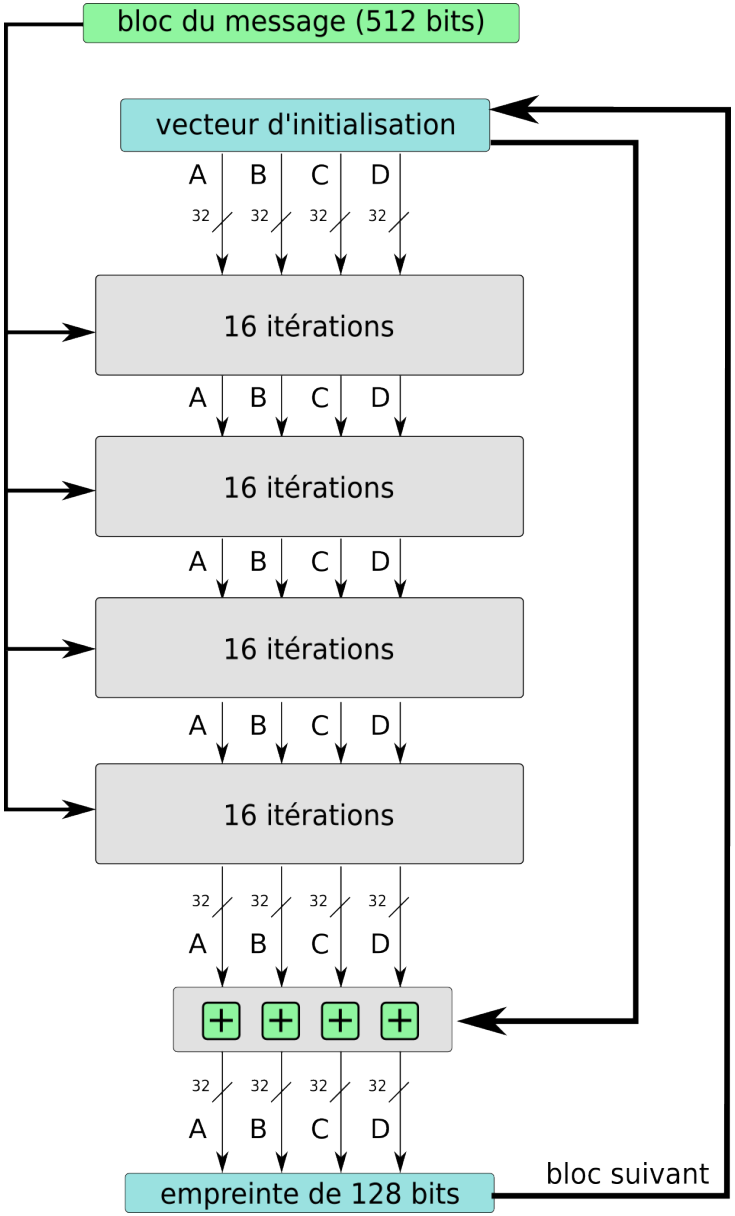
idée : utiliser une fonction qui disperse les données similaires pour coder des données secrètes

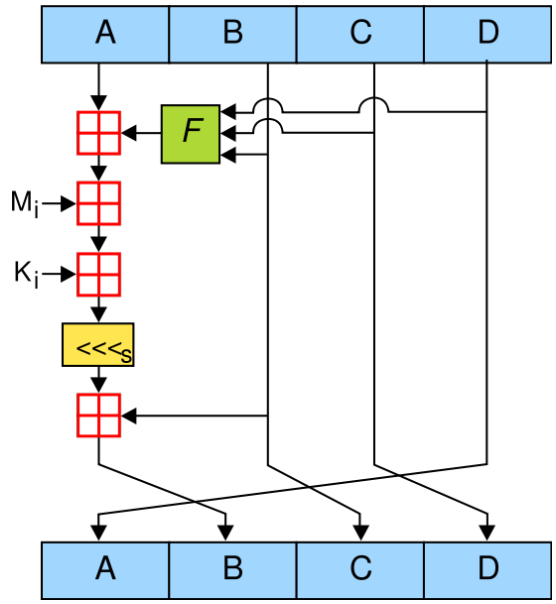
Exemple

- gestion des mots de passe et stockage dans `/etc/shadow`
- vérification de l'intégrité d'un fichier (après un transfert notamment)

$\Longrightarrow$ impératif de sécurité : il doit être difficile d'inverser la fonction – que ce soit pour retrouver la donnée d'origine ou « seulement » trouver une collision

Exemple : MD5





source : Wikipedia

Sécurité de MD5

depuis 2004, on sait engendrer des collisions pour MD5, qui n'est plus considérée comme sûre, même si la falsification de documents n'est pas encore (connue comme) faisable

Solution 1

utiliser des fonctions plus compliquées, produisant un haché plus long (famille SHA)

Solution 2

utiliser la technique du salage

Exemple

Linux utilise MD5 avec un *sel* de 8 bits pour gérer les mots de passe

Contexte

cryptage par une fonction de hachage trop faible, permettant une attaque grâce à une table précalculée

Contexte

cryptage par une fonction de hachage trop faible, permettant une attaque grâce à une table précalculée

Principe

- ajouter [a] un grain de *sel* à la donnée *avant* de la hacher :
  $h(x + sel)$ est très différent de $h(x)$
- stocker $h(x + sel)$ et sel

---

a. *i.e.* concaténer : il ne faut pas que l'opération soit inversible

Contexte

cryptage par une fonction de hachage trop faible, permettant une attaque grâce à une table précalculée

Principe

- ajouter [a] un grain de *sel* à la donnée *avant* de la hacher :
  $h(x + sel)$ est très différent de $h(x)$
- stocker $h(x + sel)$ et sel

---

a. *i.e.* concaténer : il ne faut pas que l'opération soit inversible

même si l'attaquant obtient toute l'information (haché et sel), les attaques doivent être recalculées – à condition que le sel soit différent à chaque fois (*i.e.* pour chaque utilisateur, dans le contexte de la gestion des mots de passe)