M1 MEEF NSI — Sécurité des communications informatiques

Authentification

Pierre Karpman
pierre.karpman@univ-grenoble-alpes.fr
https://membres-ljk.imag.fr/Pierre.Karpman/tea.html

2024–02–09

Objectif : authentification

Contexte du moment :

- Deux personnes $\mathscr{A}$ & $\mathscr{B}$ souhaitent communiquer (éventuellement *publiquement*) sur un canal fiable
- Face à des adversaires actifs

$\rightsquigarrow$ une possibilité : utiliser des *message authentication codes* (MACs) à évaluer v. sécurité UP ou PRF

Comment faire si $\mathscr{A}$ & $\mathscr{B}$:

- Possèdent un petit *secret partagé* ?

Authentification ; petit secret partagé

Fonction de hachage (cryptographique)

Authentification : l'idée

Un adversaire actif sur un canal peut notamment (en toute généralité) :

1. Bloquer des messages
2. Émettre des messages
   - (Et donc modifier des messages)

Combattre cet adversaire avec de la cryptographie :

1. Impossible ?
2. *Détecter* les messages de l'adversaire (pour mieux les rejeter)
   - L'ensemble des messages non-rejetés « émulent » un canal avec adversaire seulement passif
   - $\rightsquigarrow$ Preuve d'identité (il suffit de le dire)
   - $\rightsquigarrow$ Confidentialité en présence d'adversaire actif (les solutions passives suffisent)

Authentification avec des MACs

Avec un (petit) secret partagé :

1. $\mathscr{A}$ et $\mathscr{B}$ se mettent *publiquement* d'accord sur un MAC $\mathsf{F} : \{0,1\}^\kappa \times \{0,1\}^* \to \{0,1\}^\tau$
2. $\mathscr{A}$ tire uniformément une clef $K \in \{0,1\}^\kappa$ et la partage avec $\mathscr{B}$
3. Chaque fois que $\mathscr{A}$ (resp. $\mathscr{B}$) souhaite envoyer un message m, elle envoie $(m, t := \mathsf{F}(K, m))$ à $\mathscr{B}$ (resp. $\mathscr{A}$)
4. Sur réception d'un message (m, t), $\mathscr{A}$ (resp. $\mathscr{B}$) calcule $t' := \mathsf{F}(k, m)$ et rejette le message si $t' \neq t$

Vocabulaire : t est un *tag*

Remarques :

- Système *non* nécessairement probabiliste (mais attention aux interactions, cf. TD)
- Exemples de paramètres : $\kappa = \tau = 128$ (peuvent varier)

MACs : sécurité

Quelles propriétés de sécurité nécessaires pour F ?

Informellement on souhaite que :

- L'adversaire ne doit pas être capable d'émettre un message passant la vérification
- Même après avoir vu passer de nombreuses paires (m, t) valides

Mais :

- Pas nécessaire pour F de « cacher » quoi que ce soit (si nécessaire dans le système, doit être fait en amont sur m)
- Pas nécessaire de détecter un « rejeu » (si nécessaire —)

MACs : sécurité (bis)

Usuellement, F est analysé relativement à :

- L'*universal unforgeability* (UUF) : pour un challenge m arbitraire, l'adversaire gagne en produisant une paire (m, t) passant la vérification
- L'*existential unforgeability* (EUF) = UP : l'adversaire gagne en produisant une nouvelle paire (m, t) passant la vérification
- La sécurité PRF

On peut montrer (cf. TD ?) que attaque UUF $\Rightarrow$ attaque EUF $\Rightarrow$ attaque PRF

$\rightsquigarrow$ *hypothèse* de sécurité la plus forte (la plus exigeante) : F est une PRF

MACs : quelles constructions ?

Beaucoup d'options existent pour construire un MAC, par exemple :

- Avec un mode pour chiffre par bloc (PRP)
- — (UP)
- À partir d'une famille de fonction de hachage statistique + PRP/PRF
- Avec un mode pour *fonction de hachage cryptographique* ← voyons ça de plus près

Authentification ; petit secret partagé

Fonction de hachage (cryptographique)

Fonction de hachage cryptographique : définition

Fonction de hachage

Une fonction de hachage est une application $\mathcal{H} : \mathcal{M} \to \mathcal{D}$

C'est donc juste une fonction...

Paramètres typiques :

- $\mathcal{M} = \bigcup_{\ell<N}\{0,1\}^\ell$, $\mathcal{D} = \{0,1\}^n$, $N \gg n$
- Avec $N \geq 2^{64}$, $n \in \{128, 160, 224, 256, 384, 512\}$

Quand $\mathcal{D}$ est de taille variable, on parle généralement d'*extendable-output functions* (XOFs) : $\mathcal{D} = \bigcup_{\ell<N'}\{0,1\}^\ell$

— cryptographique : on s'attend à ce que la fonction ait une bonne sécurité relativement à certaines définitions

Sécurité d'une fonction de hachage

Modèle idéal (non standard) :

Oracle aléatoire

Un *oracle aléatoire* est une fonction $\mathcal{H} : \mathcal{M} \to \mathcal{D}$ t.q. :

$$\forall\, x \in \mathcal{M},\ \mathcal{H}(x) \twoheadleftarrow \mathcal{D}$$

(Autrement dit, $\mathcal{H} \twoheadleftarrow \mathsf{Funcs}(\mathcal{M}, \mathcal{D})$)

- Difficile/impossible à obtenir en pratique
- Permet (entre autres) de conceptualiser la sécurité d'une fonction de hachage : ce qui est « difficile à faire » pour un oracle aléatoire devrait (idéalement) être « difficile à faire » pour une fonction concrète

Sécurité d'une fonction de hachage : définitions classiques

Fonction à sens-unique

- *Première préimage* : étant donné $t \twoheadleftarrow \mathcal{M}$, trouver m t.q. $\mathcal{H}(m) = t$
- *Seconde préimage* : étant donné m, trouver $m' \neq m$ t.q. $\mathcal{H}(m) = \mathcal{H}(m')$

Fonction résistante aux collisions

- *Collision* : étant donné $\varnothing$, trouver $(m, m' \neq m)$ t.q. $\mathcal{H}(m) = \mathcal{H}(m')$

Probabilité de succès *pour un algorithme générique* faisant $q \leq N$ appels à son oracle aléatoire de domaine de taille N :
préimages : $\approx q/N$;
collisions : $\approx q^2/N \leftarrow$ paradoxe des anniversaires (encore lui)

Définitions classiques (bis)

Réductions entre ces définitions :

- (résister à) 2PRE $\Rightarrow$ (résister à) PRE (Q : comment le prouver ?)
- (résister à) COL $\Rightarrow$ (résister à) (2)PRE (**ATTENTION :** réduction de coût exponentiel !)
- La plupart (mais pas toutes !) des utilisations de fonctions de hachage (cf. plus loin) reposent sur la sécurité v. l'une ou plusieurs de ces définitions

Paradoxe des anniversaires

Paradoxe des anniversaires : les collisions « arrivent rapidement »
Un énoncé :

Espérance du nombre de collisions dans une liste aléatoire

Soit L une liste de q éléments $x_i \twoheadleftarrow \mathcal{S}$, $\#\mathcal{S} =: N$,
$C := \#\{(i, j > i) : x_i = x_j\}$ la variable aléatoire désignant le nombre de *collisions* dans L, alors $\mathbb{E}[C] = q(q-1)/2N$

Dém. : Par linéarité de l'espérance appliquée aux variables intermédiaires $C_{i,j}$

- $q \approx \sqrt{N}$ pour que $\mathbb{E}[C] = 1$ (on peut aussi m.q. suffisant pour obtenir une collision avec prob. constante)

Construction de fonctions de hachage

Approche usuelle similaire aux constructions de chiffrement :

1. Construire une « petite » fonction (souvent une *fonction de compression*) d'entrée de taille fixe (« comme » un chiffre par bloc)
2. Concevoir un « mode opératoire » pour traiter des entrées de taille quelconque
3. Réduire la sécurité de la fonction ainsi obtenue à celle de la petite fonction

- Les petites fonctions peuvent elle-même être obtenues en appliquant des « modes » à des chiffres par bloc, mais **ATTENTION** aux détails (pas de réduction standard possible ici)

Quelques exemples de fonctions de hachage

- 1992 : MD5 (Rivest) ; attaques concrètes (collisions) 2005 (Wang et al.)
- 1992 : RIPEMD (RIPE) ; attaques concrètes (collisions) 2005 (Wang et al.)
- 1993 : SHA-0 (NSA) ; attaques « théoriques » (collisions) 1998 (Chabaud and Joux) ; — concrètes 2005 (Biham et al.)
- 1995 : SHA-1 (NSA) ; attaques « théoriques » (collisions) 2005 (Wang et al.) ; — concrètes 2017 (Stevens et al. (et moi !))
- 1996 : RIPEMD-128 (Dobbertin et al.) ; attaques « théoriques » (collisions) 2013 (Landelle and Peyrin)
- 1996 : RIPEMD-160 (Dobbertin et al.) ; pas d'attaques connues
- 2001 : SHA-2 (NSA) ; —
- 2008 : SHA-3 (Keccak team) ; —

Quelques remarques

- Difficile de construire des fonctions de hachage résistantes aux collisions !
- Les attaques « théoriques » (trop coûteuses pour être implémentées) sont toujours de mauvais signes
- $\rightsquigarrow$ attention à ne pas utiliser de fonction de faible sécurité
- (Beaucoup plus « facile » de résister aux préimages, mais la résistance aux collisions est souvent nécessaire)

Fonctions de hachage : application à l'authentification

Retour à l'objectif : authentification
Soit $\mathcal{H} : \mathcal{M} \to \mathcal{D}$, on peut définir :

- $\texttt{PrefixMAC}_{\mathcal{H}} : \{0,1\}^\kappa \times \mathcal{M} \to \mathcal{D}$ comme $\texttt{PrefixMAC}_{\mathcal{H}}(k, m) = \mathcal{H}(k||m)$
- $\texttt{SuffixMAC}_{\mathcal{H}} : \{0,1\}^\kappa \times \mathcal{M} \to \mathcal{D}$ comme $\texttt{SuffixMAC}_{\mathcal{H}}(k, m) = \mathcal{H}(m||k)$

Remarque : $\texttt{PrefixMAC}_{\mathcal{H}} \approx \texttt{SuffixMAC}_{\mathcal{H}^\triangleleft}$ avec $\mathcal{H}^\triangleleft$ pris comme $\mathcal{H}$ « à l'envers »

- Constructions sûres *génériquement* (pour un oracle aléatoire), mais **sans réduction** aux propriétés de sécurité standard « traditionnelles »
- $\rightsquigarrow$ délicat à instancier
- (Par exemple, okay avec SHA-512/256, mais pas avec SHA-512 ou SHA-256 !)

Application à l'authentification (bis)

Plusieurs constructions de MACs à partir de fonctions de hachage réduisent leur sécurité à des propriétés standard :

- $\texttt{SandwichMAC}_{\mathcal{H}} : \{0,1\}^{\kappa} \times \mathcal{M} \to \mathcal{D} \approx$ $\texttt{SandwichMAC}_{\mathcal{H}}(k, m) = \mathcal{H}(k||p||m||p'||k)$ (pour certains *paddings* p et p') réduit sa sécurité PRF à la sécurité PRF de fonctions de compression utilisées (d'une certaine façon) dans $\mathcal{H}$
  - Toujours pas une réduction à « juste » des propriétés de $\mathcal{H}$, mais hypothèses raisonnables
- Autres MACs populaires avec le même type de sécurité : NMAC, HMAC

Fonctions de hachage : autres applications

Les fonctions de hachage servent dans de nombreuses constructions / applications, par ex. :

- Dérivation de clef (par ex. dans un protocole d'échange de clef)
- *Padding* RSA (OAEP, PSS...)
- Transformations diverses (par ex. « Fiat-Shamir », cf. prochain cours)
- Hachage de mot de passe

ATTENTION : Ces applications reposent sur des propriétés des fonctions de hachage variées, parfois systématiquement invalides pour les fonctions « classiques » (par ex. hachage de mot de passe)
$\rightsquigarrow$ parfois difficile à naviguer