OS01 : TD2

Les fonctions de hachage cryptographiques

Une fonction de hachage est une fonction qui transforme une entrée de n'importe quelle taille en une sortie de taille fixe (par exemple 128, 256 ou 512 bits). Les fonctions de hachage non-cryptographiques sont utilisées pour la correction des erreurs (en transmission, au stockage, etc.).

Dans ce TD nous allons nous intéresser aux fonctions de hachage *cryptographiques*, comme par exemple SHA1, SHA256 ou la gagnante de la compétition SHA3, Keccak. Ces fonctions de hachage ont des propriétés spéciales :

1. la *résistance aux préimages* (étant données une fonction de hachage connue f et une valeur hachée f(x), il est difficile de trouver l'entrée x) ;
2. la *résistance à la seconde préimage* (étant données une fonction de hachage connue f et une valeur en entrée x, il est difficile à trouver une valeur $y \neq x$ tel que $f(x) = f(y)$) ;
3. la *résistance aux collisions* (il est difficile de trouver x et $y \neq x$ tel que $f(x) = f(y)$).

Exercice : le hachage et l'authentification

Supposons qu'on a un fournisseur de services en ligne, qui demande aux acheteurs d'enregistrer un compte avec un nom d'utilisateur $\mathrm{Nom}_{\mathrm{usager}}$ et un mot de passe mdp. Le fournisseur de services stocke, dans sa base de données des tuples $\left(\mathrm{Nom}_{\mathrm{usager}}, f(\mathrm{mdp})\right)$, où f est une fonction de hachage cryptographique. Lorsqu'un utilisateur veut demander un service, il doit s'enregistrer avec son nom d'utilisateur et son mot de passe (les deux valeurs sont envoyé sur un canal sécurisé).

Une fois les valeurs reçues, le fournisseur de service calcule l'hachée du mot de passe fourni par l'utilisateur et compare cette valeur avec la valeur stockée dans sa base de données. Si les deux valeurs coïncident, l'utilisateur est dirigé vers son compte. Sinon, l'usager ne peut pas accéder à son compte.

Un attaquant A connait les noms de tous les utilisateurs, mais pas leurs mots de passe. Son but est de gagner accès (illicitement) aux services proposés par le fournisseur de service.

Pour l'instant supposons que seulement le fournisseur de service a accès à sa base de données.

1. Pour chacune des trois propriétés des fonctions de hachage cryptographiques ci-dessus, discutez l'impact (s'il y en a un) de cette propriété sur la sécurité du schéma d'authentification contre notre attaquant.
2. Maintenant supposons que l'attaquant réussisse à trouver la base de données. Pour chacune des trois propriétés discutez quelles attaques possibles elles empêchent.

3. Comparez les bénéfices et inconvénients entre le stockage des mots de passe en clair et le stockage de leurs hachées uniquement.
4. Nous avons vu les schémas de chiffrement symétriques lors du TD précédent. Est-ce que le schéma de hachage utilisé ci-dessus nous donne une façon de chiffrer les mots de passe ? Justifiez vos réponses.

L'authenticité des messages

Nous avons déjà vu que le chiffrement garantit la confidentialité des messages : notamment, aucune entité qui n'est pas équipée d'une clé de déchiffrement ne peut pas lire le contenu des messages.

Par contre, le chiffrement ne garantit pas (en principe) l'**authenticité** des messages : nous n'avons aucune garantie sur l'entité qui a envoyé le message. Or, il est très important de pouvoir savoir si nous devons faire confiance à un message ou non -- et ce, avant de le traiter !

Pour les communications sécurisées, nous avons besoin que les messages échangés soient confidentiels et authentifiés. Toutefois, l'authenticité (sans confidentialité) peut être un but en soi. Pensons par exemple les mises-à-jour d'un programme, qui sont téléchargées par l'ordinateur et qui devront ensuite être installées. Mais, avant de les installer, nous devons vérifier leur provenance (et donc, leur authenticité).

En cryptographie nous utilisons deux outils cryptographiques pour authentifier des messages : les schémas d'authentification symétrique de messages (et en particulier une sous-catégorie de ces schémas, notamment les codes authentificateurs de messages -- *message authentication codes* -- MAC) et les signatures, qui sont à clé publique. Attention : l'utilisation d'un MAC ou d'une signature ne garantit pas la confidentialité du message signé ou authentifié, seulement son authenticité ! Les schémas de MAC sont à clé symétrique (les deux parties qui communiquent utilisent la même clef pour authentifier et pour vérifier l'authentification d'un message). Les schémas de signature sont à clef publique : on utilise la clé secrète pour signer et une clé publique pour vérifier.

Exercice : L'authentification basée sur les codes MAC

Disons qu'Alice veut envoyer un message (non-confidentiel) à Bob, tel que ce dernier puisse établir que le message vient d'Alice. Ils vont utiliser un schéma de MAC et une clé k. Avec son message m et la clé k qu'elle partage à priori avec Bob, Alice génère un *tag,* qui est envoyé avec le message à Bob et qui servira d'authentification. Elle envoie le tuple (m, tag) à Bob. Celui-ci utilise le message, le tag et la clé pour établir si le tag est valide pour le message reçu. Attention : si Bob ne reçoit pas le message, il ne pourra pas établir si un tag donné est valide.

1. Disons que le message arrive (par erreur) à Charles au lieu de Bob. Est-ce qu'il pourra lire le message ? Est-ce qu'il pourra établir si le message vient d'Alice ?

Disons maintenant qu'on a une fonction de hachage cryptographique, connue par tout le monde. Disons que le tag calculé par Alice pour un message m est obtenu selon la formule : $\text{tag} = H(k) \oplus m$, où H est la fonction de hachage, k est la clé partagée par Alice et Bob et m, le message choisi par Alice. Disons qu'un attaquant intercepte le tuple (m, tag) envoyé par Alice.

2. Est-ce que cet attaquant pourra trouver la clé k à partir de ces informations ?
3. Est-ce que l'attaquant pourra envoyer à Bob un message (malveillant) de son choix m^* et un tag t^* qui pourrait convaincre Bob que le message vient d'Alice ?

Un schéma basé sur des fonctions de hachage qui est utilisé en pratique c'est HMAC. Dans ce schéma les tags sont calculés selon la formule : $\text{tag} = H(k|m)$, où | dénote la concaténation de deux valeurs.

4. Discutez comment les différences entre HMAC et notre schéma antérieur de MAC peuvent impacter leur sécurité.

Le chiffrement et l'authentification des messages

La transmission sécurisée de messages doit garantir la confidentialité et l'authenticité de messages (ainsi que leur intégrité, c'est-à-dire, la garantie que le message envoyé n'a pas été modifié en transit). C'est pourquoi les messages envoyés sur des canaux sécurisés (en SSH ou TLS par exemple) sont chiffrés ET authentifiés en même temps.

Soit un schéma de chiffrement symétrique qui prendra en entrée une clé K_{Enc} et un schéma de MAC avec une clé K_{MAC}. Pour un message m, nous allons noter par $\text{Enc}(K_{\text{Enc}}; m)$ le chiffrement d'un message m avec la clé K_{Enc}, et par $\text{MAC}(K_{\text{MAC}}; m)$. Le déchiffrement d'un chiffré c est noté $\text{Dec}(K_{\text{Enc}}; c)$ – on se rappelle qu'on a considéré un schéma de chiffrement symétrique, donc on chiffre et on déchiffre avec la même clé. La vérification d'un tag t pour un message m est noté $\text{MAC.Vf}(K_{MAC}; m, t)$.

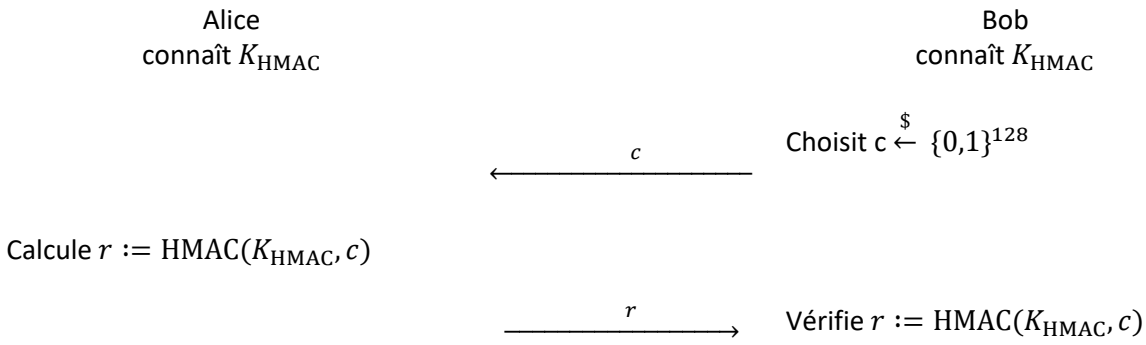
Idéalement lorsque Alice envoie des messages sécurisés à Bob, il faut que Bob soit sûr que le message vient d'Alice avant de le déchiffrer ou de le traiter.

1. Est-ce que cela sera garanti si Alice envoie ses messages dans le format $(\text{Enc}(K_{\text{Enc}}; m) \mid \text{MAC}(K_{\text{MAC}}; m))$ (méthode connue sous le nom MAC-and-Encrypt) ?
2. Et si Alice envoie des messages dans le format $\text{Enc}(K_{\text{Enc}};\ m|\text{MAC}(K_{\text{MAC}}; m))$ (méthode connue sous le nom MAC-then-Encrypt) ?
3. Et si Alice envoie des messages dans le format $(\text{Enc}(K_{\text{Enc}}; m), \text{MAC}(K_{\text{MAC}}; \text{Enc}(K_{\text{Enc}}; m)))$ (on appelle cette méthode également Encrypt-then-MAC) ?

Les MACs et l'authentification d'une entité

En dehors de l'authentification des messages, les MACs – et plus particulièrement HMAC – sont utilisés pour l'authentification des personnes (plutôt que de messages). Les protocoles d'authentification, qui ont lieu entre un prouveur et un vérificateur, peuvent être utilisés pour le contrôle d'accès dans un bâtiment (avec un badge ou une carte à puces), pour l'accès au transport en commun, dans la logistique, etc. Le protocole le plus simple d'authentification consiste en deux messages : une requête (challenge) et une réponse (response).

Dans ce protocole, Bob joue le rôle d'un vérificateur et Alice, le rôle d'un prouveur. Le but d'Alice, qui est un utilisateur légitime, c'est de prouver à Bob que c'est bien elle. Les deux partagent une clé de MAC K_{HMAC} tel que HMAC rend des valeurs à 128 bits. Bob commence en choisissant une valeur aléatoirement de 128 bits, qu'on appelle c. Alice utilise sa clé et cette valeur c pour calculer la réponse r, qu'elle envoie ensuite à Bob. Finalement Bob vérifie si c'est la bonne valeur et accepte l'authentification d'Alice si l'authentification marche.



1. Est-ce que Bob s'authentifie auprès d'Alice ? (est-ce qu'Alice peut vérifier qu'elle parle bien à Bob et à personne d'autre ?)

On suppose que l'attaquant est capable d'envoyer des message à Alice ou à Bob. Son propos est de se faire passer par Alice.

2. Disons que Bob ne choisit pas les valeurs de c aléatoirement, mais utilise c comme un compteur, qu'il incrémente de 1 à chaque fois qu'Alice essaie s'authentifier. Comment l'attaquant peut-il usurper l'identité Alice ?
3. Et si c'est trop petit ? (disons qu'il n'a que 10 bits mais qui sont aléatoirement choisis)
4. Quel est l'avantage du schéma ci-dessus si on a beaucoup de prouveurs et un seul vérificateur ? (indice : pensez aux clés).