OS01-- TP 3+4

Contexte

Le but principal de ce TP sera de comprendre le fonctionnement des fonctions de hachage et de la dérivation de clé. Le langage utilisé sera Java.

Exercice 1 : Messages hachés

En Java, une bibliothèque de base pour la sécurité est java.security :

https://docs.oracle.com/javase/7/docs/api/java/security/package-summary.html

Nous allons utiliser cette bibliothèque pour expérimenter avec les fonctions de hachage. Pour ces TPs, nous allons utiliser par défaut la fonction de hachage SHA1. Ceci étant dit, il faut mentionner qu'un groupe de chercheurs a trouvé l'année dernière une première collision avec SHA1, ce qui indique une première faille de sécurité. En conséquence on remplace de plus en plus SHA1 par SHA256 dans les applications pratiques.

Rappelez-vous l'exemple du TD, où un utilisateur devait donner son nom d'utilisateur et son mot de passe pour s'authentifier. Les mots de passe étaient stockés en forme hachée.

1. Utilisez la bibliothèque java.security pour hacher un mot de passe de votre choix en utilisant la fonction de hachage SHA1 (qui s'appelle juste SHA dans la terminologie java.security). Vous pouvez utiliser la documentation de la classe MessageDigest (un texte haché est un digest) :

   https://docs.oracle.com/javase/7/docs/api/java/security/MessageDigest.html

2. Nous allons maintenant programmer une attaque de force brute sur des petits hachés. Disons que le mot de passe est haché à un digest de 5 bits (disons les premiers 5 bits du résultat du calcul SHA(mot_de_passe)).

   Utilisez le fichier http://www.ottawascrabbleclub.com/Lists/ods5.txt comme dictionnaire pour trouver une collision (deux mots qui donnent le même haché). Combien de mots avez-vous essayé ?

   Indice : Pour extraire N bits d'une valeur entière X on peut utiliser la commande (X % (1 << N)). Par exemple pour X = 13 = [1101] et N = 3, (13 % (1 << 3)) donne 13 % 8, alors 5 = [101]. Ceci n'est pas la méthode la plus rapide pour extraire des bits ; pouvez-vous en trouver une autre ?

3. Essayez la même chose pour une taille du digest de 6 bits, puis 7, 8, 9 et 10 bits. Combien de mots avez-vous dû essayer ?
4. Quel est la propriété des fonctions de hachage que vous êtes en train de casser ?
5. Maintenant vous allez essayer un autre type d'attaque. Hachez le mot « poisson » et prenez juste les premiers 5 bits du digest. Trouvez un mot de passe qui donne le même haché. Combien de mots avez-vous dû essayer ?
6. Répétez l'attaque pour une taille de digest de 6, 7, 8, 9 ou 10 bits.

Exercice 2 : HMAC en Java

Une fonction de hachage habituelle n'utilise pas des clés. Par contre, si on veut utiliser HMAC, il faut d'abord générer une clef symétrique, et puis utiliser l'algorithme de HMAC avec SHA1. Nous allons alors découvrir le package https://docs.oracle.com/javase/7/docs/api/javax/crypto/package-summary.html . Dans ce package on a déjà une implémentation de l'algorithme (non-simplifié de HMAC).

1. Regardez les spécifications de la classe Mac, qui implémente les algorithmes de Mac :
   https://docs.oracle.com/javase/7/docs/api/javax/crypto/Mac.html

   Nous allons avoir besoin des méthodes getInstance (qui définit quelle fonction de hachage on utilise, dans notre cas SHA1), init (qui va décider avec quelle clé on exécute l'algorithme) et doFinal (qui fera exécuter le Mac).

2. Pour initialiser l'algorithme avec une clé il nous faudra un autre objet Java, qui s'appelle SecretKeySpec. Ce type d'objet peut être utilisé pour n'importe quel type d'algorithme cryptographique qui a besoin d'une clé symétrique du type chaine d'octets. Il n'est pas possible, par exemple, de générer une paire de points sur une courbe elliptique, ou dans un groupe finit. Pour une clé de MAC, de HMAC ou de chiffrement DES ou 3DES, cela suffira.

   https://docs.oracle.com/javase/7/docs/api/javax/crypto/spec/SecretKeySpec.html

   Le deuxième constructeur de cette classe peut transformer une clé qui est une chaine d'octets dans un objet du type SecretKeySpec. Cet objet sera donné en entrée à la méthode init de la classe Mac.

3. Nous allons utiliser une clé très simple, par exemple un mot (comme Exercise2, monAnimalFavorit, etc.). Cette clé commencera en tant que chaine de caractères. Puis, elle sera transformée dans une chaine d'octets (byte[]). Pour transformer une chaine de caractères dans une chaine d'octets nous utiliserons la méthode getBytes("UTF-8"). Cette chaine d'octets sera mise en entrée du constructeur SecretKeySpec, pour que cette clé puisse être utilisée par l'algorithme de Mac.

4. Ecrivez un programme très court qui calculera le HMAC-SHA1, avec la clé que vous avez choisie, du texte "Ceci est mon premier HMAC SHA1." Affichez le résultat sur l'écran.

5. Le résultat du Mac sera une chaine d'octets. Pouvez-vous la changer à une chaine de caractères en hexadécimal.
6. Changez le programme pour calculer le HMAC-SHA512 du même texte avec la même clé. Qu'est-ce que vous observez ?