HMAC et gestion de mots de passe

Louiza Khati

1 Prise en main : HMAC

Dans cette section, nous allons manipuler quelques fonctions vues en cours.

1.1 Fonctions de hachage

1. Soient deux messages `m_1` et `m_2` tels que : `m_1 = b'tito'` et `m_2 = b'titn'`. Afficher le type de ces deux variables. Utiliser la méthode `.hex()` pour afficher les valeurs hexadécimales contenues dans les variables `m_1` et `m_2`.
2. Ecrire une fonction `xor()` qui prend en entrées deux variables de type *bytearray* de même longueur et qui renvoie le xor de ces deux entrées (variable de type *bytearray*). Si les longueurs des entrées sont différentes, renvoyer un message d'erreur.
3. Afficher la valeur hexadécimale de la variable v telle que `v = xor(m_1,m_2)`. Que constatez-vous ?
4. Utiliser la fonction `SHA256`[1] du package `Crypto.Hash` pour calculer les empreintes SHA2-256 notées `h_1` et `h_2` des messages `m_1` et `m_2`. Afficher les valeurs hexadécimales de ces empreintes et vérifier que vous retrouvez bien les valeurs ci-dessous :
   `h_1 = 504db5aeeabd937a18a36b62d6e948d236b889b33c84e9e0561d7fda1c145bce`
   `h_2 = 9734a6725a8cfd2f438aa871f4c1bf083e3bfc4b1d97bbb1a9e43220390abbdc`
5. Quelle est la taille en bits de ces empreintes ? (trouver le moyen d'afficher la taille en bits de ces valeurs)
6. Quelle est la différence entre `h_1` et `h_2` ? Quelle propriété vue en cours avez-vous mis en évidence ?

1.2 HMAC

1. Rappeler la taille minimale d'un tag pour avoir un bon niveau de sécurité.
2. Utiliser la fonction HMAC avec les fonctions de hachage suivantes :`SHA-224`,`SHA-256`, `SHA-384` `SHA-512` et une clé de 128 bits pour obtenir les tags du même message `m= b'message'`. Regarder les différentes sorties obtenues. Que constatez-vous ?
3. Calculer les empreintes HMAC avec la fonction de hachage `SHA-256` et du message `m= b'message'` avec des clés de tailles différentes. Que constatez-vous ?
4. Générer une clé secrète `K` de taille correcte et la partager avec votre voisin. Envoyer un couple (message, tag) à votre voisin et vérifier l'intégrité du message.

1. https ://pycryptodome.readthedocs.io/en/latest/src/hash/hash.html

5. Utiliser la nouvelle fonction de hachage SHA3 pour générer des tags. Quelles sont les différentes tailles de tag possibles avec cette fonction ?

2 Gestion de mots de passe

Sylvia met en place un blog pour décrire les merveilleux endroits qu'elle visite chaque année. Pour améliorer son site, elle demande à ses lecteurs de créer un compte avec un identifiant `id` et un mot de passe `mdp` et les invite à laisser des commentaires. Elle se rappelle de ses cours de cryptographie suivis à l'ESILV et elle se souvient que l'OTP ("One-Time Password") ou "masque jetable" est un bon moyen pour préserver la confidentialité d'un message. Elle sait aussi qu'un mot de passe robuste fait au moins 20 caractères donc elle force ses utilisateurs à choisir un mot de passe de 20 caractères. Elle génère donc une valeur aléatoire et secrète de 20 octets pour masquer les mots de passe de ses utilisateurs. Sa base de données $B1$ est publique puisque les mots de passe de ses lecteurs sont chiffrés (OTP). La ligne i de sa base de données notée $B1[i]$ est composée du couple $(id, mdp \oplus mask)$ qui représente les données d'authentification d'un utilisateur. L'opérateur $\oplus$ représente l'opération xor entre deux mots binaires.

1. [*Ecrit*] Pourquoi 20 caractères pour un mot de passe devrait être suffisant ? Il est rappelé qu'un caractères ASCII est codé sur 7 bits.
2. [*Ecrit*] Analyser le script v1 de Sylvia (fichier `mdp_Sylvia_v1.py`). Quel "détail" important a oublié Sylvia dans l'implémentation de son système de gestion de mots de passe et qui met en péril la sécurité de son système de gestion de mots de passe.
3. Marco est passionné par les voyages et suit Sylvia depuis quelques mois. Il étudie aussi la cybersécurité dans son master et s'intéresse au système de gestion de mots de passe de Sylvia. Il s'est enregistré sous l'identifiant `id = "mrc"` et il a choisi le mot de passe suivant `mdp = b'HDY64!MJGFDT3ZSGzert'`. Il lit, lui aussi, le script `mdp_Sylvia_v1.py`.
    a. [*Ecrit*] Déduire de la question précédente, comment Marco peut récupérer les mots de passe des autres utilisateurs.
    b. [*Implémentation*] Écrire un script qui permet de récupérer les mots de passe de tous les utilisateurs. Vous trouverez la base de données de mots de passe B1 dans le fichier `B1.txt`. Écrire ces mots de passe dans votre script en commentaire (précédé du symbole "#").

Questions bonus

4. Sylvia a retenu la leçon. Elle modifie son script en conséquence et remplace son script par `mdp_Sylvia_v2.py`. Marco se réinscrit et il garde le même identifiant et son nouveau mot de passe est `mdp = b'78tfJf34mtDGHvtiGkl4'`. La nouvelle base B2 de mots de passe est dans le fichier `B2.txt`. Marco a tout de même réussi à récupérer les mots de passe même avec cette nouvelle méthode.

   a. [*Ecrit*] Expliquer rapidement comment Marco peut récupérer les mots de passe de cette nouvelle base de données.

   b. [*Implémentation*] Écrire un script pour récupérer les nouveaux mots de passe. Écrire ces mots de passe dans votre script en commentaire (précédé du symbole "#").

5. [*Ecrit*] Marco aimerait aider Sylvia et il voudrait lui proposer un système plus résistant et très simple à mettre en place. Proposer un mécanisme simple en utilisant ce que vous avez vu en cours.