University Marie and Louis Pasteur, ISIFC, Cryptographie. TP sur les fonctions de hachage

Jean-François Couchot

September, 7th 2026

1 Sensibilité des fonctions de hachage (effet d'avalanche)

l'objectif est de vérifier expérimentalement la propriété d'effet d'avalanche d'une fonction de hachage cryptographique (comme SHA-256) sur un texte long. Même si deux textes ne diffèrent que d'un unique caractère (ou bit), leurs empreintes doivent être radicalement différentes.

1.1 Travail demandé

1. Choisir ou générer un texte long (par exemple un paragraphe de 500 à 1 000 caractères).
2. Créer une copie de ce texte en modifiant un seul caractère (par exemple remplacer une lettre minuscule par une majuscule, ou changer une lettre au milieu du texte).
3. À l'aide de la bibliothèque standard `hashlib` par exemple:
    1. Calculer l'empreinte SHA-256 du texte original et du texte altéré au format hexadécimal.
    2. Convertir ces deux empreintes en chaînes binaires (256 bits).
    3. Calculer le pourcentage de bits modifiés.
4. Conclure : le résultat observé correspond-il aux attentes théoriques pour une bonne fonction de hachage ?

2 Choix d'une fonction de hachage pour le stockage sécurisé de mots de passe

Une règle d'or en sécurité est qu'un mot de passe ne doit **jamais être stocké en clair**. En général, une fonction de hachage cryptographique classique (comme SHA-256) est destinée à être très rapide pour traiter de grandes quantités de données.

Cependant, pour limiter la portée d'une attaque (comme le bruteforce ou l'attaque par dictionnaire), il est primordial que la fonction de hachage appliquée aux mots de passe soit **la plus lente possible**. C'est pourquoi on utilise des fonctions spécifiques de dérivation de clés, comme **Argon2**, qui permettent de régler le temps d'exécution et l'espace mémoire utilisé.

2.1 Objectifs

- Mesurer et comparer la vitesse d'exécution d'une fonction rapide (SHA-256) face à une fonction lente dédiée (Argon2).
- Simuler l'impact de cette différence de vitesse lors d'une attaque par dictionnaire locale.

2.2 Travail demandé

1. Évaluation du débit (Benchmark)
    1. Écrivez une fonction `hash_sha256(password)` utilisant `hashlib.sha256`.
    2. Instanciez un objet `PasswordHasher` de la bibliothèque `argon2` (regarder les paramètres).
    3. Concevez une boucle permettant de mesurer le nombre de hachages réalisés en une seconde par SHA-256, puis par Argon2.
2. Simulation d'attaque par dictionnaire
    1. Créez une liste (dictionnaire) contenant 500 mots de passe génériques (ex: `motdepasse_1`, `motdepasse_2`, etc.).
    2. Insérez un mot de passe "secret" à la fin de cette liste.
    3. Calculez l'empreinte cible de ce secret avec SHA-256, puis avec Argon2.
    4. Mesurez le temps nécessaire à une boucle `for` pour retrouver le secret en testant chaque mot du dictionnaire contre les deux empreintes cibles.
3. Analyse
    1. Rédigez une courte conclusion expliquant pourquoi Argon2 est plus sûr que SHA-256 pour cet usage précis.