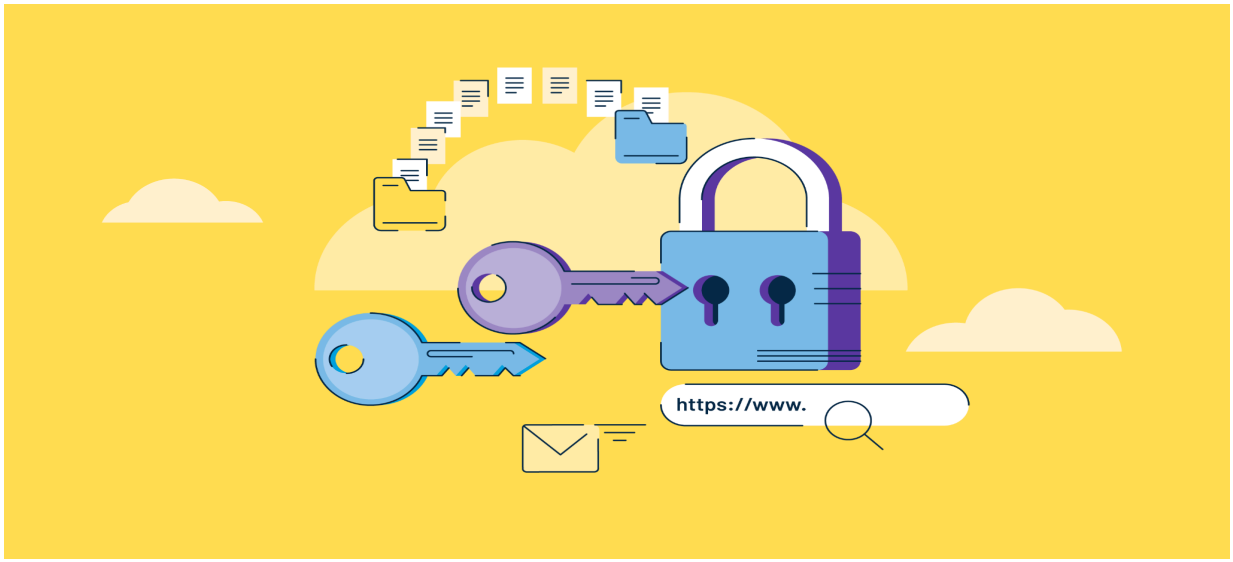
Cryptographie avancé

Compte Rendu

TTH (Top Tiger Hash)



Année Universitaire: 2023/2024

LEKOUAGHET imad eddine

Introduction à TTH

Le Top Tiger Hash (TTH) est un algorithme de hachage utilisé pour garantir l'intégrité des données lors de leur transfert sur des réseaux pair à pair, notamment dans les réseaux de partage de fichiers comme Direct Connect (DC++). Conçu pour être rapide et efficace, le TTH génère une empreinte unique pour chaque fichier en fonction de son contenu. Cette empreinte est utilisée pour vérifier que le fichier téléchargé est identique à celui qui a été partagé initialement, assurant ainsi l'intégrité des données sans avoir besoin de vérifier chaque octet du fichier. Le TTH utilise des techniques de hachage cryptographique, notamment SHA-1 et SHA-256, pour générer ces empreintes. Grâce à sa capacité à détecter les altérations accidentelles ou malveillantes des fichiers, le TTH est un outil précieux dans les environnements où la confiance dans l'intégrité des données est primordiale.

Implémentation du TTH sous python :

1- Son fonctionnement:

L'algorithme TTH fonctionne comme suit :

Diviser le fichier en blocs : Le fichier à partager est divisé en blocs de taille fixe, généralement 1024 octets.

Hacher chaque bloc : Pour chaque bloc de données, un algorithme de hachage, tel que SHA-1 ou SHA-256, est appliqué pour calculer un hachage unique.

Concaténer les hachages de blocs : Les hachages de chaque bloc sont concaténés pour former une chaîne de hachage.

Appliquer un dernier hachage : Un dernier hachage est effectué sur la chaîne de hachage résultante des blocs pour produire l'empreinte finale du fichier.

Comparaison des empreintes : Lorsque les utilisateurs téléchargent le fichier, ils calculent également son empreinte TTH. Ils comparent ensuite cette empreinte à celle fournie par l'hôte initial pour vérifier l'intégrité du fichier.

Détection des altérations : Si les empreintes ne correspondent pas, cela indique une altération du fichier pendant le transfert ou un téléchargement incorrect. Les utilisateurs peuvent alors demander une nouvelle copie ou prendre d'autres mesures pour assurer l'intégrité des données.

2-Code source du TTH:

```python
import hashlib

def TTH(message):
    block_size = 1024
    block_hashes = []

    for i in range(0, len(message), block_size):
        block = message[i:i+block_size]
        block_hashes.append(hashlib.sha256(block.encode()).hexdigest())

    concatenated_hashes = ''.join(block_hashes)

    # Calculer le hachage final avec TTH
    final_hash = hashlib.sha256(concatenated_hashes.encode()).hexdigest()

    return final_hash

message = "LE HACHAGE AVEC TTH EST FACILE"

empreinte = TTH(message)
print("Empreinte du message avec TTH :", empreinte)
```

3- Résultat de empreinte TTH:

Le resultat de l'empreinte du message `LE HACHAGE AVEC TTH EST FACILE` est donné comme suit : e753c86bd9174939280dba7fc22e263a0a6e1448e325b9745c9f45e1803cefd5

4- Paradoxe des anniversaire:

Le paradoxe des anniversaires est souvent utilisé pour estimer la probabilité de trouver des collisions dans des fonctions de hachage. Dans le cas de la fonction TTH, qui utilise SHA-256 pour les hachages de blocs individuels, nous devons considérer le fait que SHA-256 produit une sortie de 256 bits (ou 32 octets).Le paradoxe des anniversaires nous dit que dans un ensemble de **n** éléments, il y a de fortes chances (probabilité supérieure à 50%) qu'au

moins deux d'entre eux partagent la même valeur si **n** est environ $\sqrt{2^{b}}$ où **b** est la longueur des valeurs de sortie en bits (dans notre cas, 256 bits).

donc $n \approx 2^{128} \approx 3.4 \times 10^{36}$

Cela met en évidence la puissance des fonctions de hachage cryptographique comme SHA-256, qui fournissent une sécurité robuste en générant des empreintes numériques uniques pour chaque fichier, rendant pratiquement impossible de trouver deux fichiers différents avec la même empreinte (collision) même avec des quantités massives de données.