Bases des télécommunications 1

Labo Blockchain

Introduction

Le but de ce laboratoire est de mettre en pratique les concepts vus en classe de Bases des télécommunications. Les concepts suivants seront appliqués dans ce laboratoire : fonction de hachage, chiffrement et blockchain.

- Le rendu doit se faire sous la forme de programmes Python 3 sur Cyberlearn.
- Des rendus intermédiaires pour chaque partie seront demandés.
- **Les réponses aux questions du labo doivent être affichées directement sur le terminal lors de l'exécution du programme.**

Rappels

- '0x' indique que la suite est écrite de manière hexadécimale.
- Un fichier binaire est un fichier qui n'est pas un fichier texte.
  Par exemple des fichiers audios, vidéos, images, tableurs, programmes compilés… Ils peuvent donc avoir différentes extensions : .mp3, .jpg, .exe…
- L'extension .bin est souvent utilisée pour désigner qu'il s'agit d'un fichier binaire quelconque.
- Le caractère '0' n'est pas égale au bit '0'.
- 0x00 = 00000000 en bits.

Bases des télécommunications 1

I. Fonction de hachage

La manière la plus rependue pour vérifier l'intégrité des données reçues est d'envoyer le hash de ces données par un autre canal, pour vérifier qu'aucun changement n'a été apporté durant le transport.

Le but d'une fonction de hachage est de créer une empreinte (un hash) à partir d'une donnée, sans qu'il soit possible de retrouver cette donnée à partir de l'empreinte.

a) Créer un programme de chiffrement/déchiffrement de fichiers binaires, utilisant la librairie « cryptography.fernet » permettant de réaliser facilement un chiffrement symétrique AES. (`from cryptography.fernet import Fernet`)

b) Pour cet exercice la clé secrète de votre programme est imposée et doit être ajoutée dans le code :

```
KEY = b'HehUbvgn6GrbwyVEc2mgT8FoAXlrwGUHMOpeIu9R0jY='
```

(En temps normal cet clé ne doit jamais être révélée et doit être protégée par un mot de passe)

c) Afficher dans la console le hash du fichier non-chiffré. Utiliser la fonction de hachage **Sha3** sur **224** bits avec la librairie hashlib (`import hashlib`)

d) Faire en sorte que votre programme demande le hash en hexadécimal, du fichier déchiffré à l'utilisateur dans la console. Une fois le déchiffrement fait, vérifier l'intégrité du fichier déchiffré grâce au hash donné par l'utilisateur.

e) Le programme de chiffrement doit générer un fichier gardant le même nom que celui de base, mais en ajoutant à l'extension initiale « .en» pour encrypted.

f) Le programme de déchiffrement doit générer un fichier déchiffrer en reprenant le nom du fichier de base et en enlevant le « .en » rajouter précédemment.
Exemple : `maphoto.png.en -> maphoto.png`

g) Le programme doit être exécutable de la manière suivante :

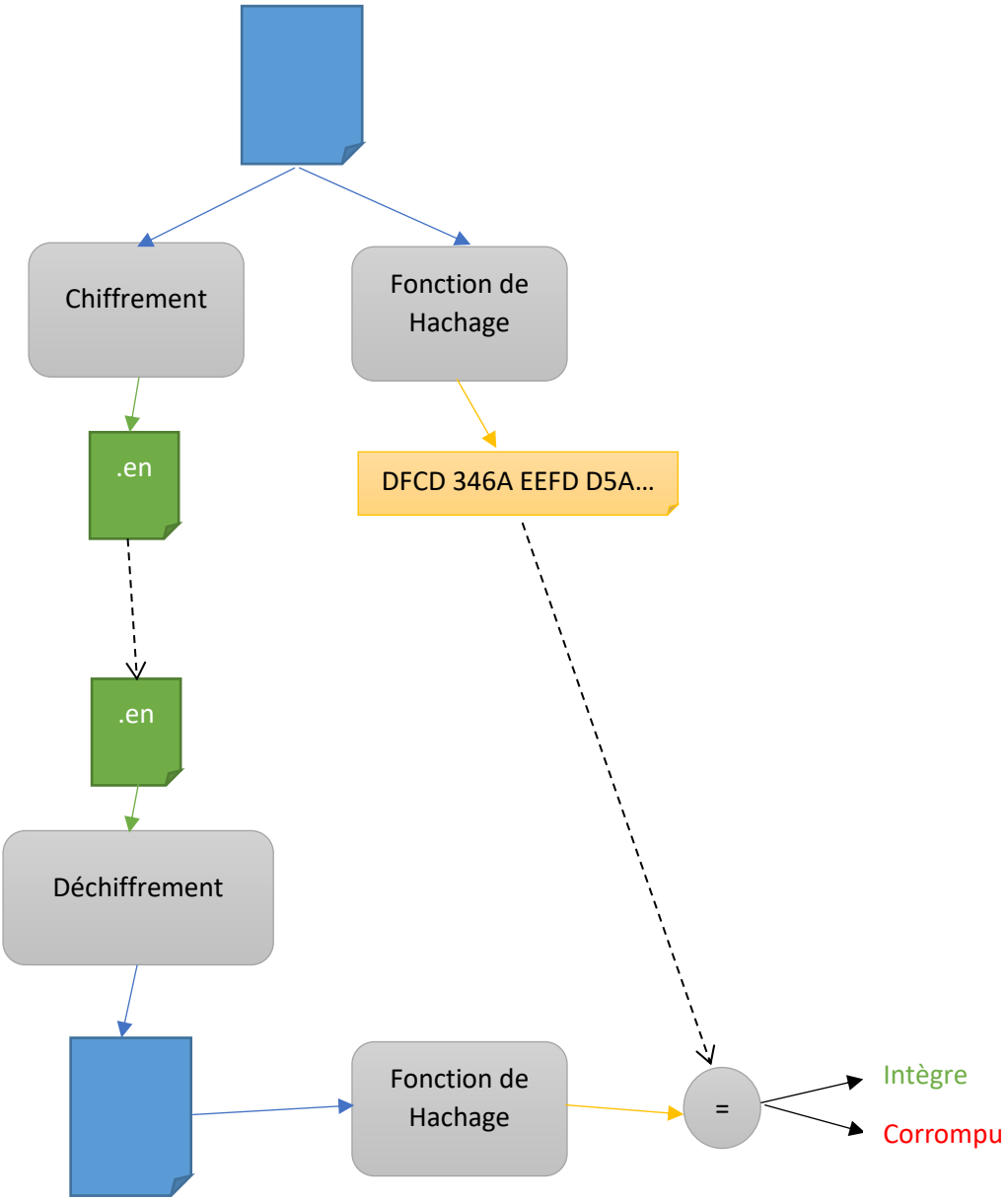
```
python myencry_[nom1]_[nom2].py -c in_file.bin
```

ou

```
python myencry_[nom1]_[nom2].py -d in_file.bin.en
```

Bases des télécommunications 1

Diagramme de fonctionnement partie I :



Bases des télécommunications 1

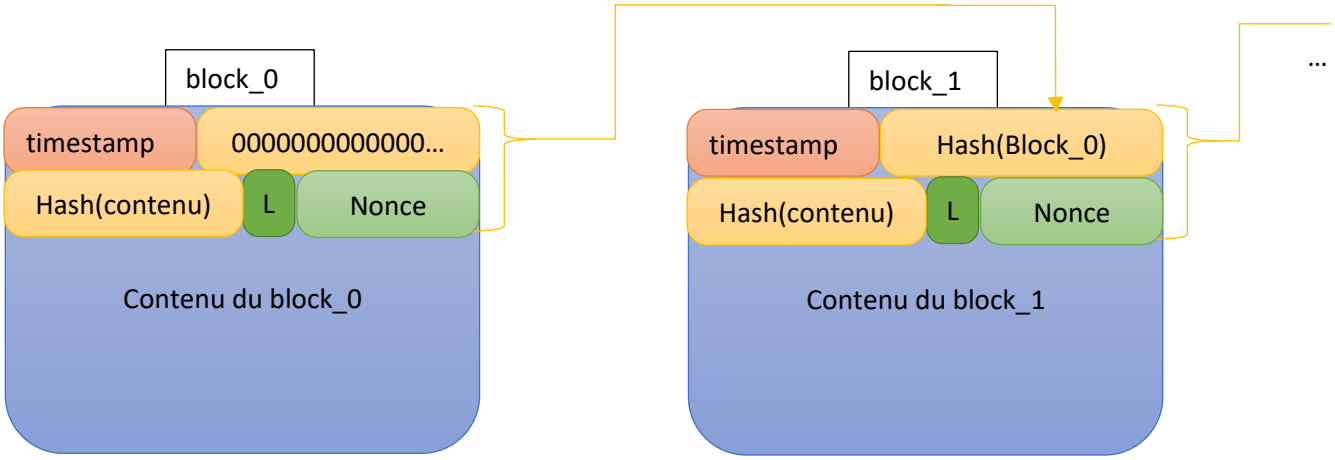
II. Blockchain

Le principe de blockchain est directement lié à l'utilisation de fonctions de hachage. On va ici réaliser un 2ème programme python permettant de vérifier l'intégrité d'une blockchain.

Cette blockchain sera sous la forme d'un dossier contenant des fichiers en tant que block. Ces blocks sont nommés de la manière suivante : block_1, block_2, block_3 … il peut y avoir une infinité de blocks.

Les blocks de cette blockchain sont constitués des éléments suivants :

1. Le timestamp de block sur 4 bytes en big-endian.
2. Le hash SHA-256, de l'en-tête du block précédent sur 32 bytes.
3. Le hash SHA-256, de son propre contenu sur 32 bytes.
4. La longueur L en bytes du Nonce sur 1 bytes.
5. Le Nonce sur L bytes en big-endian.
6. Le contenu du block.



a) Votre programme doit vérifier l'intégrité de la blockchain, pour ce faire il doit vérifier que :
   a. Les blocks ont été créés les uns après les autres d'un point de vue temporel.
   b. Chaque block contient bien le hash de l'entête du block précédent.
   c. Chaque block contient bien le hash de son contenu.
   d. Le hash de chaque en-tête de block commence bien par X bits à 0.
      i. Le nombre X de bits à 0 doit pouvoir être changé dans une variable au début de votre code (13 par défaut).

b) Votre programme doit prendre le chemin vers le block_0 comme argument à l'exécution. Ce fichier est le début de la blockchain à vérifier.
`blockchain_[nom1]_[nom2].py block_0`

c) Votre programme affichera ensuite dans la console si la chaine est valide ou non. Si elle ne l'est pas, votre programme indiquera une cause qui rend cette blockchain invalide, en précisant le ou les blocks mis en cause.
Exemples : « Invalide : le block 235 ne contient pas le hash du block 234. »
ou
« Invalide : le hash du block 12 ne commence pas par 13 bits à 0. »

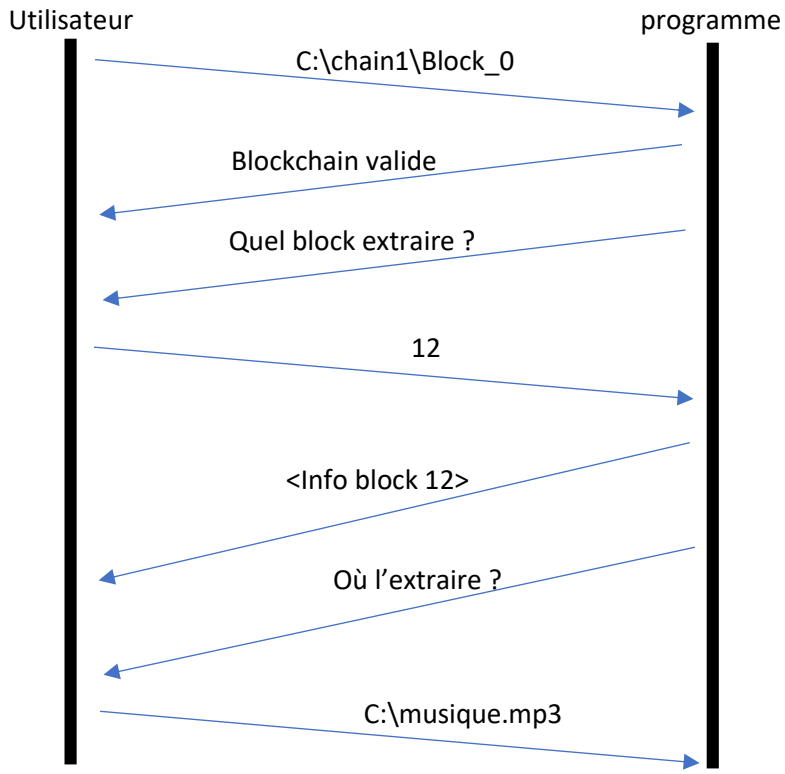
Bases des télécommunications 1

III. Lecture de block

Votre programme actuel permet de valider l'intégrité d'une blockchain. Il est maintenant tant de rajouter la possibilité d'extraire le contenu des blocks.

a) Si la blockchain est valide, votre programme doit demander à l'utilisateur d'entrer un numéro de block à lire.
b) Une fois le numéro entré, votre programme doit afficher les informations du block de manière lisible pour l'utilisateur.
    a. Le timestamp au format date et heure.
    b. Le hash de l'entête du block précédent en hexadécimal.
    c. Le hash du contenu du block actuel en hexadécimal.
    d. Le Nonce en décimal et le nombre de bytes sur lesquels il est écrit.
    e. Le hash de l'entête du block actuel.
    f. Les 128 premières bytes des données du block actuel.
c) Votre programme demandera ensuite à l'utilisateur d'entrer le chemin vers lequel il faut extraire le contenu du block en tant que fichier.

Exemple en diagramme temps séquence de l'utilisation du programme :



Dans cet exemple l'utilisateur sait que le block 12 contient un fichier .mp3, à la fin de l'exécution le fichier extrait du block 12 de la blockchain est lisible. Les blocks peuvent contenir n'importe quels types de fichier (.txt, .mp3, .exe, .jpg …) c'est à l'utilisateur de renseigner la bonne extension en donnant le chemin vers le fichier de sortie.