




# TP2 : stéganographie, signature, etc

 Lien TP	<a href="https://www.lama.univ-savoie.fr/pagesmembres/hyvernats/Enseignement/2122/info002/tp2.html">https://www.lama.univ-savoie.fr/pagesmembres/hyvernats/Enseignement/2122/info002/tp2.html</a>
--	---

## Contenu du diplôme

L'académie de l'UNSC délivre un diplôme à un élève en mentionnant son nom et prénom ainsi que les résultats qu'il a eus. Elle appose sa signature numérique en la cachant.

## Protocole pour cacher les informations

Nous avons réalisé ce TP en Python en utilisant les bibliothèques :

- Pillow : dessin
  - Cryptodome
1. Dans notre programme, nous allons utiliser SHA-512 pour signer nos données et ensuite les crypter avec RSA (clef privée de 2048 bits) pour que des utilisateurs puissent la décrypter avec clef publique. Quand nous parlons plus bas de signature, cela concerne ce procédé.
  2. Chaque message est précédé de sa taille inscrit en clair.
  3. L'image contiendra, en (x,y), les informations ci-dessous cachées :
    - a. (0,0) : Taille du PRENOM en clair (2 octets)
    - b. (0,1) : PRENOM en clair (variable)
    - c. (0,10) : Taille de la signature du PRENOM en clair (2 octets)
    - d. (0,11) : Signature du PRENOM (256 octets)

- e. (0,20) : Taille du NOM en clair (2 octets)
  - f. (0,21) : NOM en clair (variable)
  - g. (0,30) : Taille de la signature du NOM en clair (2 octets)
  - h. (0,31) : Signature du NOM (256 octets)
  - i. (0,40) : Taille du SCORE en clair (2 octets)
  - j. (0,41) : SCORE en clair (variable)
  - k. (0,50) : Taille de la signature en clair (2 octets)
  - l. (0,51) : Signature du SCORE (256 octets)
  - m. (0,60) : Taille de l'EMPREINTE de l'UNSC (2 octets)
  - n. (0,61) : EMPREINTE en clair (23 octets)
  - o. (0,70) : Taille de la signature de l'EMPREINTE en clair (2 octets)
  - p. (0,71) : Signature de l'EMPREINTE (256 octets)
4. Ces informations sont nécessaires car elles permettent d'identifier l'étudiant.
5. Pour cacher une information d'une taille N octets, nous modifions le bit de poids faible d'un pixel,  $8 * N$  fois. Par exemple pour cacher une signature de 256 octets, nous modifions  $8 * 256 = 2048$  pixels de notre image. Une longue information est donc inscrite sur plusieurs lignes.
6. Nous allons générer une signature du document avec SHA-512 lors de la création du fichier. Cette signature sera accessible par le vérificateur de diplôme. Cette signature portera comme nom `<nom>_<prenom>.png.sig` et stocké dans le dossier `/sign`.
- Le certificat quant à lui sera stocké dans le dossier `/certificate`.

## Warning

Nous arrivons à écrire les données que n'on veut cacher mais nous ne sommes pas arrivés à vérifier si la donnée en clair correspond à la signature. Nous faisons face à un problème de taille (dans tous les sens du terme). Du fait de notre gestion des types (d'écriture et de lecture) qui n'est pas très bien construite.

Mais pour vérifier le document, la signature du document nous paraît suffisant, en effet, si des modifications ont été apportées au document, la signature du certificat que l'on calcule n'est plus la même que celle calculé au départ.

## **Vérifier le certificat d'un élève**

Lorsque l'on voudra vérifier un certificat, il faudra les différentes données qui sont cachées dans le certificat et les vérifier avec les signatures qui sont cachées.

Le but de la vérification est de hasher avec SHA-512 le texte clair inscrit dans le document et ensuite de décrypter avec la clef publique de l'académie la signature cachée et de les comparer. Si elles sont égales, alors la donnée est bonne sinon c'est une contrefaçon.

Rien n'empêche à un utilisateur tierce de crypter ces différentes données et délivrer des faux diplômes, mais un utilisateur pourra vite se rendre compte car l'empreinte de l'académie sera différente de l'officiel.

On peut rendre disponible cette signature aux utilisateurs afin qu'il puisse vérifier eux-même l'authenticité d'un diplôme.

Nous avons fait en sorte que le système possède un stockage des signatures pour permettre de vérifier ces valeurs.

Il aurait été possible de générer un QR Code avec les bons paramètres dans une requête GET pour faire vérifier automatiquement le certificat par un web service développé par l'académie.

Un utilisateur malicieux pourrait très bien prendre possession du serveur de vérification et ainsi pouvoir créer des faux à la volée (en changeant les signatures par des signatures générés par lui même) mais nous sortons du domaine de la cryptographie.



Fun fact : on peut voir les messages qu'on écrit, des lignes rouges apparaissent de façon ainsi distinctes