Compte Rendu TP2 stéganographie, signature, etc

Antoine DEPOISIER & Jules FINCK

Lien Github : https://github.com/Tann-dev/INFO002-TP2

Choix fonctionnels

Nous cachons dans le diplôme un hash qui contient le nom de l'étudiant, sa note et la date d'obtention du diplôme. Une signature de ce hash est ensuite créée, par la clé privée de l'université.

De ce fait, la signature du diplôme n'est pas falsifiable, et si une entreprise a le diplôme d'un candidat, elle pourrait vérifier si les données du diplôme n'ont pas été falsifiées et sont les mêmes que l'université a délivrées.

L'entreprise utiliserait alors la clé publique de l'université pour déchiffrer le message, ce qui leur donnerait un hash. Suite à ça, ils auraient juste à hasher les données du diplôme, et si les deux hash sont les mêmes, ça veut dire que les données du diplôme ne sont pas falsifiées.

Choix technologiques

Nous avons choisi d'utiliser le langage python pour réaliser ce TP.

L'algorithme de la signature est RSA, comme ça la clé privée connue uniquement par l'université permet de cacher la signature dans le diplôme. De ce fait, en utilisant la clé publique, on est sûr que les données contenues dans le diplôme ont été signées par l'université.

Pour hasher le message dans la signature, on utilise SHA256 parce que c'est sécurisé et rapide.

Au niveau de la bibliothèque, nous utilisons pycryptodome, qui utilise et implémente de manière plus efficace que nous pourrions le faire RSA et SHA256.

Pour le produit final, on recommanderait la même bibliothèque, tout est déjà implémenté, et pourquoi pas un autre algorithme que SHA256, qui est plus sécurisé.

Description de notre solution

Nous aurions pour idée d'ajouter un QR-code dans le diplôme, qui redirige vers un site qui permet en donnant l'image du diplôme de vérifier le message caché à l'intérieur du diplôme. Ce message caché étant la signature, le site analyserait si les données contenues dans le diplôme (date d'émission, note et nom de l'étudiant) sont conformes.

Mais si nous n'avons pas les documents d'identité de la personne, ou bien si la personne a le même nom qu'une autre, il serait facile d'usurper son identité. Pour ce faire, une autre solution serait de cacher l'image de l'étudiant dans le diplôme et d'avoir une IA qui compare la tête d'une personne avec celle cachée dans le diplôme. Ou bien, une empreinte digitale ou rétinienne au lieu de la photo. Dans ce cas, pour que ces données ne soient pas modifiées, l'université possèderait dans une base de données les photos et empreintes digitales des étudiants.

Documentation

Question 1

Voici la commande utilisée pour encoder un message dans une image :

```
python main.py hide <input image> <message> <output image>
```

Avec un exemple d'utilisation

```
python main.py hide img\chablais-orig.png coucou img\chablais-coucou.png
```

Voici ensuite la commande permettant de déchiffrer le message contenu dans l'image.

```
python main.py unhide <input image> <size>
```

Test avec l'image précédemment crée

```
python main.py unhide img\chablais-coucou.png 6
```

Question 2

Pour générer notre couple de clé publique/clé privée, il faut utiliser la commande :

```
python main.py generate_key
```

Les clés seront toutes deux stockées dans les fichiers `private_key.pem` et `public_key.pem`.

Pour créer la signature d'un message, il faut utiliser la commande :

```
python main.py sign <message>
```

La signature sera contenue dans le fichier `signature.bin`

Pour créer la signature d'un message, il faut utiliser la commande :

```
python main.py verify <msg>
```

En lançant cette commande, le programme va vérifier si le contenu de la signature dans `signature.bin` est le même que le message donné en paramètre.

Question 3

On peut générer un diplôme sans données cachées pour le moment, avec la commande :

```
python main.py make_degree <input image> <output image> <student_name>
<mean_student>
```

Par exemple, avec cette commande, le diplôme s'appelant `diplome-test.png` est généré.

```
python main.py make_degree img/diplome-BG.png diplome-test.png "Jules Finck" 18.5
```

Question 4

Dans le diplôme, nous cachons une signature pour le nom de l'étudiant, sa note et la date d'émission du diplôme.

Cette signature sert à être sûr que les données du diplôme n'ont pas été trafiquées.

```
python main.py make_degree <input image> <output image> <student_name>
<mean_student>
```

Question 5

Voici la commande permettant de vérifier le diplôme :

```
python main.py verify_degree <input image> <message> <hash_size>
```

Si on prend l'exemple de :

```
python main.py make_degree img/diplome-BG.png diplome.png "Jules Finck" 18.5
```

et que l'on vérifie le diplôme avec la commande :

```
python main.py verify_degree diplome.png "2024-01-17Jules Finck18.5" 512
```

Le programme va nous dire si les données du diplôme sont valides.

Le message que l'on donne en entrée est de la former `<date><student_name><mean_student>`, avec une date au format `YYYY-MM-DD`.

Conclusion

Le service que nous avons imaginé est tout à fait réalisable, on pourrait même imaginer que l'on intègre la clé publique dans le diplôme, mais en clair, pour permettre de l'avoir facilement, et ce même si l'université n'existe plus.

Il y aurait tout de même une faille : si la clé privée de l'université fuit, il serait facile de falsifier son diplôme, ou bien un diplôme ayant été délivré pendant que la clé privée était valide.

Pour pallier ce problème, on pourrait intégrer un QR-Code sur chaque diplôme, qui serait propre à ce diplôme, qui permettrait d'avoir les vraies données du diplôme, même si la clé privée a fuité.

Nous pensons qu'il y a un intérêt à ce service : dans le cas où la clé privée n'a pas fuité, le temps des échanges entre une entreprise et l'université ayant délivré le diplôme n'existerait plus, car l'entreprise elle-même pourrait attester de la certification du diplôme.