Rapport TP Diplôme
INFO 002 - Cryptologie

Huseyin YURTSEVEN
Amaury Durand-Noël
Rémi Chavance

Introduction

L'objectif principal de ce TP était de concevoir un système robuste pour la création de diplômes, permettant aux étudiants de les imprimer de manière sécurisée et vérifiable afin de prévenir toutes fraudes. De plus, il nous a été demandé de proposer des améliorations potentielles pour renforcer l'authenticité du diplôme et faciliter sa vérification par l'étudiant.

Comment avons-nous mis en place notre système ?

Pour la réalisation de notre système de création sécurisée de diplômes, nous avons adopté une approche basée sur la cryptographie et la stéganographie, utilisant le langage de programmation Python. Les principales techniques mises en œuvre incluent la signature numérique avec une paire de clés RSA, ainsi que l'utilisation de la stéganographie pour dissimuler des informations importantes dans les images des diplômes.

Techniques de Cryptographie :

Au cœur de notre système se trouve l'utilisation de la cryptographie asymétrique avec la paire de clés RSA. Lors de la génération d'un diplôme, notre programme génère une paire de clés RSA (clé privée et clé publique). La clé privée est utilisée pour signer numériquement les données du diplôme, tandis que la clé publique permet à quiconque de vérifier l'authenticité de la signature.

Voici un extrait du code illustrant la génération de la paire de clés RSA et la signature des données du diplôme :

```python
def generate_key_pair(private_key_filename='private_key.pem', public_key_filename='public_key.pem'):
    try:
        # Essayer de charger les clés à partir des fichiers existants
        with open(private_key_filename, 'r') as private_file, open(public_key_filename, 'r') as public_file:
            private_key = private_file.read()
            public_key = public_file.read()
    except FileNotFoundError:
        # Si les fichiers n'existent pas, générer une nouvelle paire de clés
        key = RSA.generate(2048)
        private_key = key.export_key()
        public_key = key.publickey().export_key()

        # Écrire les clés dans les fichiers
        with open(private_key_filename, 'wb') as private_file, open(public_key_filename, 'wb') as public_file:
            private_file.write(private_key)
            public_file.write(public_key)

    return private_key, public_key
```

Stéganographie :

La stéganographie est utilisée pour cacher des informations spécifiques dans les images des diplômes. Notre programme prend le message du diplôme, le signe numériquement, puis cache à la fois le message et la signature dans les composantes rouge et bleue de l'image, respectivement.

Voici un extrait du code illustrant comment le message est caché dans l'image :

```python
def hideData(img_input_file_path, img_output_file_path,
             input_message, composante):
    img = Image.open(img_input_file_path)
    write_message(img, input_message, composante)
    img.save(img_output_file_path)
```

Utilisation de la Paire de Clés RSA pour la Vérification :

La vérification de l'authenticité d'un diplôme se fait en utilisant la clé publique associée. Le programme extrait le message caché et la signature de l'image, puis vérifie la signature avec la clé publique pour assurer l'intégrité des données.

Voici un extrait du code illustrant la vérification de la signature du diplôme :

```python
def verifyDiploma(img_path):
    img = Image.open(img_path)
    print("\n INFORMATION DE L'ELEVE : \n")
    message = read_message(img, 'r')
    print(message)
    signature = read_message(img, 'b')
    # trasnform signature in bytes
    signature = bytes.fromhex(signature)
    private_key, public_key = generate_key_pair()
    if verify_signature(public_key, message, signature):
        print("La signature est valide")
    else :
        print("La signature n'est pas valide")
```

Ces extraits de code démontrent comment notre système Python intègre la cryptographie et la stéganographie pour créer des diplômes sécurisés et vérifiables. Dans la section suivante, nous explorerons la possibilité de développer une application web pour faciliter l'utilisation de notre système Python.

Comment mettre en place une application web ?

La mise en place d'une application web pour faciliter l'utilisation de notre système Python nécessite la sélection appropriée d'une pile technologique, la configuration d'un front-end interactif, la mise en place d'un back-end robuste, l'intégration d'une base de données pour le stockage, et la possibilité de considérer une version mobile future. Nous explorerons également la notion de microservices pour une gestion modulaire, et enfin, nous discuterons des aspects de déploiement, y compris l'utilisation de Docker et le déploiement continu avec Vercel.

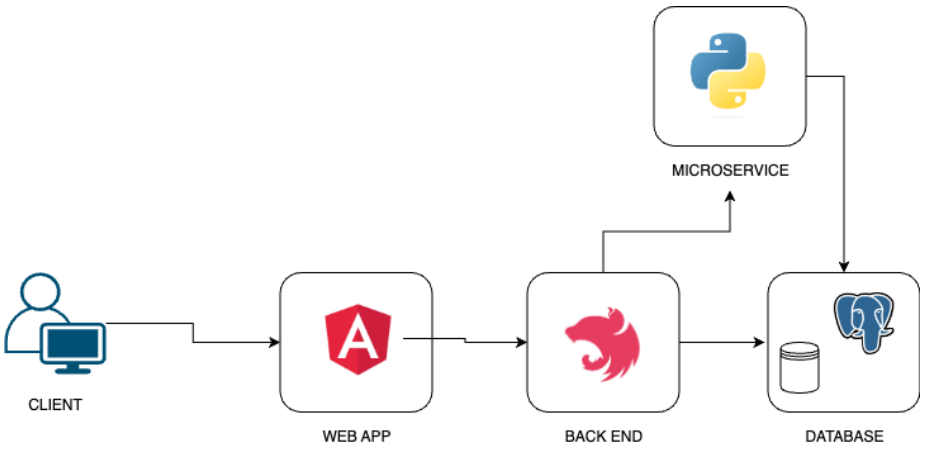
Pile Technologique :

- **Front-End** : Pour le développement du front-end, un choix courant est Angular, un framework JavaScript populaire permettant la création d'interfaces utilisateur interactives. Il offre une gestion efficace des composants réutilisables, favorise la création d'expériences utilisateur dynamiques, et est bien soutenu par une large communauté.
- **Back-End Principal** : Nous optons pour l'utilisation de NestJS pour la gestion centralisée des données utilisateur. Nous établirons une connexion entre cette composante et le micro service responsable de la manipulation des diplômes.
- **Microservices python**: Flask, un microframework web Python, peut être un excellent choix pour le back-end. Il offre une approche légère tout en étant puissant, et il est bien adapté pour la construction de microservices.
- **Base de données** : PostgreSQL, une base de données relationnelle, peut être utilisée pour stocker des informations relatives aux diplômes et aux utilisateurs.
- **Version Mobile** : Pour envisager une version mobile future, React Native ou Flutter peut être choisi. Il permet le développement d'applications mobiles à l'aide de JavaScript et de React, partageant le même code entre les plateformes Android et iOS.

Déploiement avec Docker et Vercel :

Docker : Docker peut être utilisé pour créer des conteneurs légers et portables, encapsulant notre application et ses dépendances, assurant ainsi la cohérence entre les environnements de développement et de production.

Déploiement Continu avec Vercel : Vercel peut être utilisé pour un déploiement continu facile et rapide du front-end. Il permet une intégration transparente avec les dépôts Git, déclenchant automatiquement des déploiements lors des mises à jour du code source.



En considérant ces choix technologiques et exemples de code, notre système Python pourrait être transformé en un microservice robuste intégré dans une application web moderne.

Comment les utilisateurs pourront utiliser notre système ?

Notre objectif est de permettre aux étudiants d'obtenir un diplôme valide pouvant être authentifié par d'autres établissements scolaires.

Cette authentification doit être simple et efficace. Pour cela, nous allons fournir aux établissements scolaires un outil informatique permettant, à partir de l'image originale, de récupérer automatiquement les informations cachées et de vérifier la clef afin d'assurer l'authenticité du document.

Ces informations sont aussi inscrites sur le document afin de permettre aux établissements ne disposant pas encore de l'outil de vérifier manuellement l'authenticité du document.

Conclusion

Pour conclure, notre application pourrait être grandement améliorée, notamment via une application web, comme expliqué précédemment. Cette application permettrait aux étudiants et aux établissements scolaires de vérifier simplement la validité de leur diplôme en intégrant notre outil de vérification. De plus, elle pourrait aussi être utilisée par les établissements scolaires afin de générer les diplômes de leurs étudiants. Enfin, les diplômes générés pourraient être équipés d’un QR code permettant de rediriger directement sur le site afin de vérifier le diplôme en question.