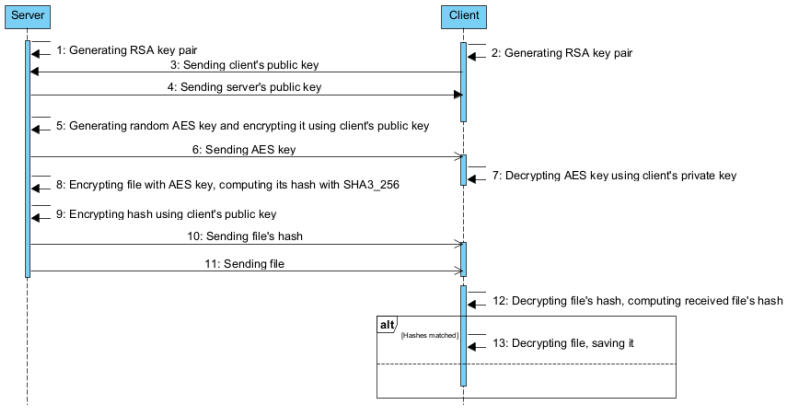
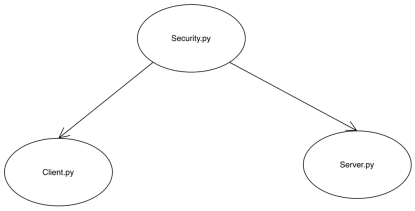
OUERGHEMMI Mongi

Cryptographie

EXIGENCE



STRUCTURE DU CODE ET IMPORTATION



Security.py : Tout les fonctions de cryptage et hachage

Client.py : Partie Client (Importe les fonctions de security.py)

Server.py : Partie Serveur (Importe les fonctions de security.py)

Pour le fonctionnement du script veuillez importer les librairies suivantes :

RSA -→ pip install rsa

PYAESCRYPT -→ pip install pyaescrypt

HASHLIB -→ déjà intégré sur python

GENERATION ET CRYPTAGE RSA

Pour le cryptage **RSA** Python offre plusieurs librairies :

Cryptodome RSA :

L'algorithme RSA peut être utilisé à la fois pour la confidentialité (chiffrement) et l'authentification (signature numérique). La signature et le déchiffrement sont significativement plus lents que la vérification et le chiffrement.

Cryptography RSA :

L'algorithme RSA peut être utilisé à la fois pour la confidentialité (chiffrement) et l'authentification (signature numérique). Possibilité d'importer et exporter les clés en format PEM

Python-RSA :

Python-RSA est une implémentation RSA pure en Python. Il prend en charge le chiffrement et le

déchiffrement, la signature et la vérification des signatures, ainsi que la génération de clés selon la norme PKCS#1 version 1.5

Pour ce travail le choix s'est porté sur Python-RSA pour les différentes raisons :

Pour ce travail le choix s'est porté sur Python-RSA pour les différentes raisons :

1) Facilité d'utilisation : Python-RSA offre une interface simple et intuitive pour effectuer des opérations cryptographiques RSA, ce qui facilite son utilisation même pour les utilisateurs débutants.
2) Flexibilité : Étant écrit en Python, Python-RSA est facilement intégrable dans divers projets et peut être utilisé sur différentes plateformes sans nécessiter de dépendances externes complexes.

3) Documentation complète : Python-RSA est bien documenté, ce qui facilite l'apprentissage de son utilisation et permet aux développeurs de comprendre rapidement comment effectuer des opérations cryptographiques avec RSA.

4) Code source ouvert : Python-RSA est un projet open source, ce qui signifie que son code source est disponible gratuitement et peut être vérifié et amélioré par la communauté des développeurs.

5) Support complet des fonctionnalités RSA : Python-RSA prend en charge toutes les fonctionnalités essentielles de RSA, y compris le chiffrement, le déchiffrement, la signature et la vérification des signatures, ainsi que la génération de clés conformément à la norme PKCS#1 version 1.5.

6) Performance raisonnable : Bien que Python ne soit pas aussi rapide que les langages compilés, Python-RSA offre des performances raisonnables pour la plupart des cas d'utilisation courants de la cryptographie RSA.

Une clé de la taille de 1024 bits a été choisi car représentait un bon équilibre entre sécurité et rapidité.

Keysize (bits)	single process	eight processes
128	0.01 sec.	0.01 sec.
256	0.03 sec.	0.02 sec.
384	0.09 sec.	0.04 sec.
512	0.11 sec.	0.07 sec.
1024	0.79 sec.	0.30 sec.
2048	6.55 sec.	1.60 sec.
3072	23.4 sec.	7.14 sec.
4096	72.0 sec.	24.4 sec.

GENERATION CLE ET CRYPTAGE AES

Pour la génération de la clé pour AES, la librairie Os a été utilisée avec la fonction os.urandom(), dans cet exercice la fichier ciblé n'était pas lourd donc la taille de la clé était de 256 bits pour mieux sécuriser la transaction, aussi la fonction os.urandom() est optimisée pour fournir des données aléatoires rapidement, ce qui la rend adaptée à une utilisation dans des applications nécessitant des performances élevées, telles que la génération de clés de chiffrement ou la création de jetons de session.

Pour le cryptage **AES** Python offre plusieurs librairies :

Cryptodome AES :

Cryptodome prend en charge différents modes de chiffrement pour AES tels que ECB, CBC, CFB, OFB, CTR, etc., offrant ainsi une flexibilité dans le choix du mode qui convient le mieux à vos besoins.

Cryptography AES :

Bien que la sécurité soit une priorité, Cryptography offre également des performances optimisées pour les opérations AES, ce qui en fait une bibliothèque appropriée pour les applications nécessitant à la fois sécurité et efficacité.

PyAESCrypt:

La bibliothèque est légère et ne nécessite pas l'installation de dépendances externes, ce qui facilite son intégration dans des projets Python existants.

Pour ce travail le choix s'est porté sur PyAesCrypt pour les différentes raisons :

Facilité d'utilisation : PyAESCrypt propose une interface simple et intuitive pour chiffrer et déchiffrer des fichiers, ce qui le rend adapté aux utilisateurs débutants en cryptographie.

Implémentation légère : La bibliothèque est légère et ne nécessite pas l'installation de dépendances externes, ce qui facilite son intégration dans des projets Python existants.

Rapidité d'exécution : PyAESCrypt est conçu pour être rapide et efficace dans le traitement des opérations de chiffrement et de déchiffrement, ce qui le rend adapté à une utilisation dans des applications nécessitant des performances élevées.

HASHAGE

Pour ce travail le choix s'est porté sur Hashlib, une librairie de la bibliothèque standard de Python, de plus cette librairie offre différentes variétés d'algorithme de hachage mais pour ce cas la le SHA-3 256 a été mis en place et elle offre différents avantages

Résistance aux attaques cryptographiques connues : SHA-3 a été conçu comme une alternative à SHA-2 pour fournir une meilleure résistance aux attaques cryptographiques connues, offrant ainsi une sécurité accrue.

Robustesse : SHA-3-256 est capable de produire des hachages de 256 bits, offrant une robustesse suffisante pour la plupart des applications de sécurité informatique.

Performances : Bien que les performances varient selon les implémentations spécifiques, SHA-3 est généralement efficace en termes de vitesse de traitement, ce qui en fait un choix attractif pour les applications nécessitant des opérations de hachage rapides.

DIFFÉRENTS POINTS À AMÉLIORER

Pour cet exercice différents choix auraient pu être misent en place mais pour une contrainte de temps n'ont pas étaient retenus.

Cryptodome est une meilleure solution car elle propose les différents chiffrements tous dans une seule librairie qui est donc très intéressante du point de vue des fonctions à employer.