Projet de GS15

Etablir un protocole de communication sécurisé

# Chiffrement IDEA

La principale difficulté que nous avons rencontré dans l’implémentation du chiffrement IDEA fût l’absence d’implémentation et de documentation en ligne concernant cet algorithme. En effet, le fait que cet algorithme soit peu utilisé à travers le monde fait que très peu de ressources sont disponibles sur le sujet. Heureusement, le livre « Cryptographie appliquée » de Bruce Schneier, disponible au SCD, dispose de quelques pages sur le fonctionnement d’IDEA et de son déchiffrement (pages 341 et 342).

Le package chiffrement est organisé comme suit : encryption\_modes.py contient les différents modes de chiffrement implémenté (ECB, CBC et PCBC). file\_management.py contient les méthodes permettant de lire les fichiers à chiffrer et d’écrire le résultat de la méthode de chiffrement dans le fichier de sortie. [idea.py](https://github.com/cuckool/GS15/blob/master/chiffrement/idea.py) quand lui contient le chiffrement IDEA en lui-même, avec la méthode de chiffrement, la génération des clefs et sous-clefs pour le chiffrement / déchiffrement. Différents fichiers de test tels que [test\_encryption\_decryption\_file.py](https://github.com/cuckool/GS15/blob/master/chiffrement/test_encryption_decryption_file.py), [test\_encryption\_modes.py](https://github.com/cuckool/GS15/blob/master/chiffrement/test_encryption_modes.py) et [test\_idea.py](https://github.com/cuckool/GS15/blob/master/chiffrement/test_idea.py) contiennent les test nécessaire pour s’assurer du bon fonctionnement des fichiers précédents. Le framework utilisé pour ces tests étant Unittest. [main.py](https://github.com/cuckool/GS15/blob/master/chiffrement/idea.py) contient différentes méthodes englobant les méthodes des autres fichiers ainsi que des méthodes permettant de demander des variables à l’utilisateur.

# Certificat

L’algorithme utilisé pour signer les clefs publiques est RSA. Nous avons rencontré des problèmes pour la génération de grands nombre premiers, cependant la librairie [PyCryptodrome](https://pycryptodome.readthedocs.io/en/latest/src/util/util.html) fournit ce genre de services très efficacement. Un nombre impair est sélectionné, puis incrémenté de deux en deux jusqu’à ce qu’on le test de Rabbin-Miller déduise qu’il est premier. De même, une autre librairie, disponible de base dans Python cette fois (sympy) nous permet de trouver l’inverse de i modulo n.