Rapport projet GS15 : FileFort

Ce projet vise à concevoir un système sécurisé permettant de stocker et gérer des informations sensibles sous forme numérique. Dans ce rapport, nous synthétiserons les problèmes rencontrés, les solutions apportées, les choix technologiques effectués, les bibliothèques python et les aides utilisées.

Problèmes rencontrés et solutions

1. Génération des S-boxes conformes (COBRA)

Problème : Transformation des S-boxes de DES pour Cobra, en respectant les propriétés différentielles et linéaires des S-boxes de Serpent (trouvées sur Wikipédia).

Solution : Création d'un algorithme Python pour rechercher des S-boxes répondant aux conditions souhaitées, en testant différentes stratégies d'échantillonnage aléatoire.

Problème : Générer des S-boxes inversibles sans doublons.

Solution : Modification de l'algorithme pour éliminer les doublons tout en respectant les contraintes mathématiques.

2. Protocole Guillou-Quisquater (pour ZPK)

Problème : J'ai eu du mal à comprendre le fonctionnement de Guillou-Quisquater. Je pensais que lorsqu'on modifiait la clé privée de l'utilisateur avant qu'il se connecte, le protocole échouait. Je n'avais pas compris le rôle de la valeur du vérificateur qui doit donc être stockée de manière publique au préalable.

Solution : Créer la valeur v pour le vérificateur et le stocker de manière publique.

3. Fonctions de hachage et KDF

Choix : Utilisation de deux fonctions de hachage différentes :

- "Sponge" pour la génération de clés (KDF).
- Méthode Merkle-Damgård avec la compression Davies-Meyer pour les signatures et HMAC.

4. Création de compte utilisateur

Problème : Différenciation des clés pour des comptes choisissant le même mot de passe.

Solution : Prendre en compte le timestamp de création du compte utilisateur lors de la génération des clés.

5. Vecteurs de permutation (COBRA)

Choix : Lorsqu'il est nécessaire d'utiliser des vecteurs de permutation dans COBRA, ceux-ci ne sont pas définis en dur dans le code mais générés pseudo-aléatoirement en utilisant une des clés de tour.

6. Interface utilisateur

Choix : J'ai choisi de développer une interface web à l'aide de ReactJS pour l'utilisateur et le test des différentes fonctionnalités demandées. Mais celle-ci simule seulement le fonctionnement de la plateforme car la communication entre l'interface web et le serveur python qui gère les calculs n'est pas sécurisée.

Bibliothèques Python utilisées

Voici la liste des bibliothèques Python utilisées dans ce projet :

os, io, datetime, time, json sys, struct, base64, random, ast, multiprocessing, flask, flask_cors, bitarray, numpy

Sources extérieures utilisées

- https://www.oreilly.com/library/view/computer-security-and/9780471947837/sec9.3.html (S-boxes AES)
- https://asecuritysite.com/encryption/gq (Protocole Gouillou-Quisquater)
- https://www.maxicours.com/se/cours/comprendre-le-chiffrement-asymetrique/ (Chiffrement asymétrique)
- https://www.simplilearn.com/tutorials/cryptography-tutorial/rsa-algorithm (Génération du couple de clés RSA)
- https://www.geeksforgeeks.org/merkle-damgard-scheme-in-cryptography/ (Merkle-Damgård)
- https://www.geeksforgeeks.org/davies-meyer-hash-function/ (Hash Davies-Meyer)
- https://en.wikipedia.org/wiki/HMAC (Pseudo-code HMAC)
- https://www.geeksforgeeks.org/multiplicative-inverse-under-modulo-m/ (Calcul d'inverse modulaire)