Rapport GS15 — A21

Thomas de Lachaux — Yohann Valo

Choix des librairies et spécificités

PyInquirer: Cette librairie nous a permis de faire de faire les menus de notre application.
Rich: Rich est une librairie d'outils graphique qui permet de formater le terminal. Cela permet notamment de faire des tableaux.

Nous avons fait le choix de stocker les données dans des fichiers JSON et non dans une base de données type SQLite car cela nous ajoutait des contraintes inutiles pour le projet. Cela nous permettait de plus de visualiser facilement les données présentes dans les fichiers afin de débugger le programme.

Nous avons utilisé des classes afin de représenter les serveurs afin de les séparer logiquement et que chaque serveur ait un champ de vision réduit sur seulement les données auquel il a accès.

Choix de fonctions mathématiques

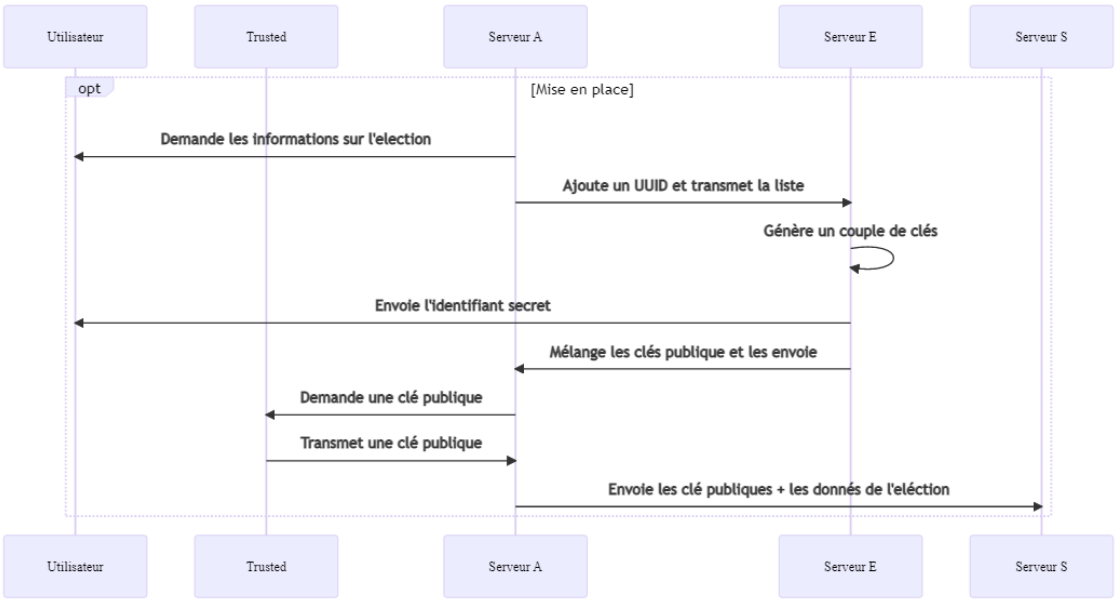
Pour la fonction de dérivation de clé, l'algorithme de Belenios choisit de prendre 14 caractères alphanumériques aléatoires et d'y ajouter un dernier caractère : un checksum des 14 premiers. Nous avons choisi dans le cadre de ce projet de ne pas réaliser le checksum et de directement tirer les 15 caractères. Nous utilisons la fonction *pbkdf2_hmac* pour générer cette dérivation.

Pour la fonction de hash nous avons utilisé la librairie *hashlib* avec SHA256 comme fonction de hachage, que ce soit pour le hash des messages chiffrés, pour la fonction de hash de la dérivation de clé ou la signature de certificats.

Nous avons fait le choix d'utiliser la librairie *pycryptodome* permettant de faire du chiffrement symétrique Blowfish car nous ne souhaitions pas manipuler des Sbox. Ce chiffrement est utilisé pour voter de façon sécurisée. La clé secrète est générée par le client et transmise au serveur de façon sécurisée grâce au chiffrement d'ElGamal.

En ce qui concerne la signature des messages utilisés dans les certificats, nous avons choisi d'utiliser la signature de El Gamal qui se base comme pour le chiffrement sur le problème du logarithme discret.

Problèmes rencontrés et résolution



Nous avons eu du mal au début à appréhender le sujet dans sa globalité et à comprendre comment tous les éléments étaient censés s'imbriquer ensemble. Pour remédier à cela nous avons effectué un diagramme de séquence afin de comprendre comment les fonctions devaient s'enchainer les unes après les autres.

Nous avons aussi eu du mal avec l'implémentation du chiffrement de El Gamal. Il s'agissait d'erreur de modulo sur les exponentiations du chiffrement.

Le plus gros problème à été au début pour générer des nombres premiers sécurisés très grands et trouver un générateur du groupe Zp. En effet, le temps de calcul pour trouver ces générateur nous a limité.
Nous avons donc fait le choix d'utiliser le nombre premier choisis par Belenios ainsi que le nombre générateur associé. Nous avons cependant codé les algorithmes de recherche de générateur.

Au début nous étions partis sur un dépouillement avec un seul "trusted user" car nous ne comprenions pas le système de dépouillement multi utilisateurs et cela nous posait problème. C'est seulement à la fin du projet que nous avons compris son fonctionnement et que nous l'avons implémenté.