Louis ALLAIN

CyberKey

Rapport de mi-projet

Le but de ce rapport de mi-projet est de faire un point sur l'avancement du projet CyberKey.
A cette date deux rendus ont été produits, l'un porte sur l'application web d'administration (rendu fin octobre 2020) et l'autre porte sur l'application mobile android utilisateur (rendu le 19 novembre 2020).

Premier rendu : Application web d'administration

Le premier rendu composé par de la documentation et d'un lien menant vers l'application web comporte toutes les spécifications qui étaient indiquées dans la proposition de projet.
En effet, l'administrateur peut se connecter à l'application grâce à un email et un mot de passe (indiqués dans la documentation du rendu). L'administrateur peut avoir connaissance des créneaux existants et en créer de nouveaux grâce à un calendrier. Il peut également visualiser les accès précédemment accordés.
L'administrateur peut de plus valider les accès pour certains créneaux ou les demandes d'accès VIP.

Les problèmes que j'ai pu rencontrer lors du développement de cette partie sont surtout liés à la gestion des données. En effet, en plus du fait qu'il faut en permanence que l'interface coïncide avec l'état de la base de données, il faut de plus que les données elles-mêmes soient cohérentes. Par exemple, imaginons qu'un utilisateur possède l'accès à un créneau. Entre-temps l'administrateur décide de supprimer ce créneau pour x raison, alors il faut que l'accès accordé à l'utilisateur pour ce créneau soit automatiquement supprimé. Voici précédemment un exemple parmi d'autres liés à la gestion des données que j'ai pu rencontrer.

Un autre problème, moins technique, que j'ai pu rencontrer se trouve dans l'interface de l'application. Les seuls retours d'expériences que j'ai pu avoir vis-à-vis de l'interface de l'application sont ceux de mon entourage proche, peut-être aurais-je dû trouver un meilleur moyen d'avoir plus de retour et plus pertinent. De plus, je voulais produire quelque chose qui fonctionne parfaitement relativement aux spécifications de la proposition de projet et qui soit rapide. C'est pour cela que parfois j'ai préféré concéder plus de temps au "moteur" de l'application plutôt qu'à son interface.

Second rendu : Application Android utilisateur

Le second rendu est composé du première version de l'application utilisateur sous forme d'un fichier au format APK (Android Package) et de documentation. En ce qui concerne l'application, elle répond partiellement aux spécifications de la proposition de projet étant donné que la partie "déverrouillage" ne peut être complète que si le système de déverrouillage à base d'un ESP32 est complètement développé.

Donc pour le moment, l'application permet à un utilisateur de se créer un compte avec une adresse mail du domaine de l'UBS (@univ-ubs.fr).

L'utilisateur peut se connecter à l'application après la création de son compte.

Une fois connecté, l'utilisateur accède directement à une liste de créneaux permettant de visualiser tous les créneaux définis par l'administrateur (à partir de la date d'aujourd'hui). Les créneaux sont séparés en semaine et triés par ordre décroissant. L'utilisateur peut sur cette même liste visualiser quels sont les créneaux auxquels il a demandé l'accès et quels sont les créneaux auxquels il possède l'accès.

Via une page de paramètre, l'utilisateur peut demander un accès VIP et peut aussi supprimer son compte.

Même si la fonctionnalité n'est pas totalement implémentée pour les raisons exprimées au début de cette partie, le problème majeur rencontré lors du développement de cette application a été la création d'un système de déverrouillage. En effet, il a fallu anticiper le développement du système de serrure. Il a donc fallu se renseigner et prendre connaissance d'un maximum d'informations sur les possibles techniques que je pourrai employer par la suite dans le développement de la serrure. Il a fallu par exemple que je me renseigne sur les techniques qui me permettront d'authentifier l'utilisateur souhaitant déverrouiller la porte. Pour cela, il fallait que l'application utilisateur soit capable de produire par exemple, un couple de clef RSA qui permettront par la suite de mettre en place un système de signature. Et surtout, produire une clef publique sous différents formats que je pourrai partager à la serrure le moment venu selon le format de clef qu'elle accepte.
J'ai dû aussi créer une sorte de petit protocole de déverrouillage entre l'application mobile et le système de serrure. Ce protocole devait être implanté de telle sorte qu'il permette de tolérer toutes les erreurs de communications qui peuvent survenir et prendre en compte les latences entre les communications (notamment liées à la signature / vérification par système de clef asymétrique). J'ai donc créé ce "protocole" mais j'ai choisi de ne pas le mettre en œuvre dans l'application pour le moment, car je préfère d'abord le tester en condition réelle une fois que le système de serrure sera développé.

Le protocole dont je parle ci-dessus est le suivant :

Prérequis :

. L'application possède deux caractéristiques d'un service BLE :
- rx_challenge : sert à recevoir une chaîne de caractère "challenge"
- rx_isAuth : sert à recevoir l'information indiquant si on est authentifié ou non

. Le système de serrure possède deux caractéristiques d'un service BLE :
- rx_user_id : sert à recevoir l'identifiant de l'utilisateur dans la base de données
- rx_user_sig : sert à recevoir la signature de la chaîne challenge

Etat initial : L'utilisateur souhaite déverrouiller la salle :

. L'application Android vérifie d'abord si un créneau qui a été défini par l'administrateur est en cours. Si oui, alors l'application vérifie si l'utilisateur a effectivement accès à ce créneau.

. Si l'utilisateur a accès à ce créneau, l'application scan les périphériques BLE alentour pendant 2 secondes et maintient une liste des périphériques découverts pendant ce scan. A la fin de scan si l'adresse MAC de l'ESP32 du système de serrure a été découverte alors l'application tente de se connecter à celle-ci et passe à l'état "Etat connecté" sinon retourne à l'état initial.

Etat connecté : L'application s'est connecté à l'ESP32 :

1. Demande à l'ESP32 un MTU de 400 octets pour les futurs communications (ordre de grandeur des informations qui seront transmises par la suite).

2. 1. Écrit sur la caractéristique "rx_user_id" l'identifiant en base de données de l'utilisateur.

   2. Attend une réponse pendant x ms, si au bout de cette durée rien n'est reçu sur la caractéristique "rx_challenge" alors recommence, si au bout de 3 essais il n'y a toujours pas de réponse alors retourne à l'état initial et se déconnecte de l'ESP32.

3. 1. Lit la caractéristique "rx_challenge", chiffre cette chaîne avec la clef privée de l'utilisateur et écrit la signature sur la caractéristique "rx_user_sig".

   2. Attend une réponse pendant x ms (temps pour que l'ESP32 vérifie la signature), si au bout de cette durée rien n'est reçu sur la caractéristique "rx_isAuth" alors recommence, si au bout de 3 essais il n'y a toujours pas de réponse alors retourne à l'état initial et se déconnecte de l'ESP32.

4. Lit la caractéristique "rx_isAuth" :
   . Si sa valeur est "V" alors l'utilisateur est authentifié (la porte doit s'ouvrir),
   . Sinon, alors l'utilisateur n'est pas authentifié.

5. Retourne à l'état initial et se déconnecte de l'ESP32.

Reste du projet

Le reste du projet consistera donc à implanter ce protocole dans l'application utilisateur et de mettre en place le système de serrure.
Ce qui veut dire, mettre en place un protocole sur l'ESP32 qui est capable de répondre au protocole d'authentification décrit ci-dessus. Et mettre en place lorsque l'utilisateur est authentifié un système de serrure électronique à base d'un relais, d'une serrure électrique et d'une alimentation externe.

.