11/05/2016

Jeoffrey godart , Maxime Millot

[nom de la société]

Documentation de conception

FileSecureStorage

Table des matières

[Historique du document 2](#_Toc450913079)

[Contexte 2](#_Toc450913080)

[Authentification : Communication entre le serveur et client 3](#_Toc450913081)

[Schéma 3](#_Toc450913082)

[Description 3](#_Toc450913083)

[Côté client 3](#_Toc450913084)

[Côté serveur 4](#_Toc450913085)

[Protection des données 4](#_Toc450913086)

[Communication côté Client 5](#_Toc450913087)

[Connexion 5](#_Toc450913088)

[Echange de fichier 6](#_Toc450913089)

[Communication côté Serveur 8](#_Toc450913090)

[Connection 8](#_Toc450913091)

[Echange de fichier 8](#_Toc450913092)

[Stratégie de surveillance des répertoires/fichiers 8](#_Toc450913093)

[Côté client 8](#_Toc450913094)

[Côté serveur 9](#_Toc450913095)

[Choix des technologies 10](#_Toc450913096)

[Partie mobile 10](#_Toc450913097)

[Partie Serveur 10](#_Toc450913098)

[Chiffrement 10](#_Toc450913099)

[Architecture 11](#_Toc450913100)

[Client 11](#_Toc450913101)

[Serveur 12](#_Toc450913102)

[Tests mises en place et description 12](#_Toc450913103)

[Changement dans nos choix initiaux de conception 12](#_Toc450913104)

[Procédure d’installation 13](#_Toc450913105)

[Serveur 13](#_Toc450913106)

[Application 13](#_Toc450913107)

[Utilisation 14](#_Toc450913108)

[Application 14](#_Toc450913109)

[Serveur 16](#_Toc450913110)

# Historique du document

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Auteur | Modification | Date |
| 0.0.1 | Godart Jeoffrey | Création | 11/05/2016 |
| 0.1.0 | Godart Jeoffrey | Modification | 12/05/2016 |
| 0.1.1 | Godart Jeoffrey | modification | 13/05/2016 |
|  |  |  |  |

# Contexte

Le but de ce projet est de réaliser une application mobile qui permet de sauvegarder régulièrement le contenu d'un ou plusieurs répertoires vers un serveur.

Cette application mobile devra être utilisable sur une plateforme Android et disposer d’un écran de connexion pour pouvoir s’authentifier.

Les informations devront transitées entre les deux parties de façon sécurisées et les fichiers devront être accessibles sur le serveur.

# Authentification : Communication entre le serveur et client

L’authentification entre l’application et le serveur est simple :

## Schéma

Chiffrement des données\* : Voir protection des données

## Description

### Côté client

L’Authentification du côté client est relativement simple :

* Démarrage de l’application
* Remplissage des champs identifiant et mot de passe et validation sur le bouton connexion
* Récupération des champs saisis et chiffrement
* Une fois que le mot de passe et l’identifiant sont chiffrés, ont les inserts dans un flux Json
* A partir de ce flux Json on créer une requête Post en direction du serveur qui nous retourne si la connexion est accepter ou non. En cas de connexion fausse l’utilisateur ne peut pas se connecter et ne peut donc pas accéder à la synchronisation.

### Côté serveur

Du côté du serveur plusieurs traitements sont effectués :

* En tout premier lieu on récupère le contenu de notre flux Json
* On déchiffre et récupère le mot de passe et l’identifiant on vérifie si l’utilisateur dispose d’un dossier

Première Connexion

* On créer un dossier en rapport avec cette utilisateur (Nom du dossier = identifiant)
* Ensuit on créer un dossier en lien avec cette utilisateur avec à l’intérieur un fichier config.txt contenant le mot de passe chiffré ainsi qu’un code de connexion chiffré
* Une fois le dossier créé on retourne l’autorisation de connexion avec un indicateur de première connexion

Connexion normal

* On Vérifie que le mot de passe et le code correspondent bien à cet utilisateur
* Si vérification on valide la connexion et on renvoi un indicateur de connexion normal

A chaque connexion le serveur renvoi une autorisation de connexion ainsi que le code contenu dans le fichier config.txt et l’IV.

# Protection des données

Le principal intérêt de cette application repose sur la protection des données qui transitent. Pour cela il faut assurer des échanges sécurisés entre les deux acteurs.

Dans cette optique chaque élément échangé se doit d’être chiffré.

Deux communications sont effectuées entre le client et le serveur :

* La connexion
* Echange de fichier

## Communication côté Client

Connexion

Elle doit envoyer :

* Identifiant chiffré
* Mot de passe chiffré
* IV

Tout d’abord on chiffre en AES128 les identifiants de connexion reçu (Identifiant et mot de passe) pour cela :

* On Créer le Cipher définissant la méthode de chiffrement (« AES/CBS/PKS5padding »)
* Ensuite les « IvParametersSpec » définissant le vecteur d’initialisation
* Puis on initialise le Cipher en mode « Decrypt » avec pour paramètre la clé Secret commune au serveur et client ainsi que le vecteur d’initialisation
* Puis on effectue un « doFinal » du Cipher prenant en paramètre les Bytes de notre fichier string qui renvoi un tableau de Byte contenant notre contenu chiffré.

Maintenant que notre fichier est chiffré correctement il faut l’encoder en base64 qui permet de renvoyer un string de notre élément.

Ensuite il faut créer notre flux Json à partir de nos strings.

Une fois se flux créé on peut l’envoyer par une requête http vers le serveur.

Le serveur renverra par la suite si l’authentification est bonne ou non.

### 

### Echange de fichier

L’échange de fichier nécessite plusieurs points clés :

* Récupération des fichiers
* Chiffrement du contenu
* Passage dans un flux Json
* Envoi

#### Récupération des fichiers

Pour récupérer les fichiers il faut tout d’abord récupérer le chemin du dossier contenant les fichiers.

sdCardDirectory = Environment.*getExternalStorageDirectory*().toString();

Ci-dessus nous récupérons la chemin de la Carte SD de l’appareil.

Ensuite on récupère le contenu par le biais d’un objet File :

File dirFileObj = new File(sdCardDirectory);

Une fois ce dossier récupérer on le parcours pour récupérer la liste des fichiers souhaités via la fonction :

List<File> files = getListFiles(dirFileObj);

Qui effectue un tri du dossier en récupérant fichier par fichier et si le fichier actuel est un dossier une récursive est mise en place pour rappeler cette même fonction.

#### Chiffrement

Une fois la liste des fichiers récupérer on la parcours. A chaque réitération on récupère le fichier et lui apposons plusieurs traitements :

* En premier lieu nous récupérons ces informations (Nom, contenu, chemin dans la carte, et la date de dernière mise à jour).
* Puis nous chiffrons chacune de ses informations en AES 128 (Comme précédemment dans la connexion côté client).
* Une fois ses champs chiffrés nous créons un objet de type FileText (Classe créée).

Une fois le fichier traité nous l’ajoutons dans une ArrayList d’objet FileText.

#### Passage dans un flux Json

Une fois toute la liste de fichier traité nous pouvons créer notre flux Json.

Pour cela nous faisons appel à la fonction suivante :

FilesTextTojson(filesText,id,code);

Elle prend en paramètre la liste de FileText, l’identifiant de l’utilisateur et le code unique situé dans son fichier de configuration et on lui ajout aussi son IV.

#### Envoi

Une fois l’objet Json récupérer nous pouvons l’envoyer via une requête http.

public String PostJsonToServer(JSONObject jsonObject,String url)  
{  
 InputStream inputStream;  
 String result = "";  
 try {  
 HttpClient httpclient = new DefaultHttpClient();  
 HttpPost httpPost = new HttpPost(url);  
 String json = "";  
 json = jsonObject.toString();  
 StringEntity se = new StringEntity(json);  
  
 httpPost.setEntity(se);  
  
 HttpResponse httpResponse = httpclient.execute(httpPost);  
 HttpEntity entity = httpResponse.getEntity();  
 inputStream = entity.getContent();  
  
 if(inputStream != null) {  
 result = *convertInputStreamToString*(inputStream);  
 }  
 else {  
 result = "false!";  
 }  
 } catch (Exception e) {  
 System.*out*.println("InputStream = " + e.getLocalizedMessage());  
 }  
  
 return result;  
}

Celle-ci prend en paramètre le flux Json ainsi que l’url du serveur et nous renvoi ensuite le résultat de l’envoi sous la forme d’un message.

## Communication côté Serveur

ConnexionNous créons un objet Json grâce aux flux de données que nous envoi le client.

Puis nous déchiffrons le flux et comparons si l’identifiant dispose d’un dossier.

Si il n’existe pas on le créer avec un fichier texte qui contient le mot de passe crypté du client. Si oui ont récupère le mot de passe du client  qu'on déchiffre et on compare avec ce qu'on a récupéré. Si le mot de passe est identique on valide la connexion et on envoie un code créé aléatoirement et chiffré. Sinon on envoi faux et un message d'erreur.

Echange de fichierNous créons un objet Json grâce aux flux de données que nous envoi le client. Puis nous déchiffrons l'objet.  Nous regardons ensuite si l'utilisateur possède un dossier.

S’il n'y a pas de dossier on lui indique qu'il y a une erreur. Puis on compare le code envoyé avec le code dans le fichier de configuration.

S’il n'est pas identique on lui renvoi une erreur sinon on compare la liste de fichier qui a été envoyée. Pour comparer les fichiers nous créons une variable contenant le chemin de base qui va être incrémenter par les dossiers qui vont être créé où récupéré.

# Stratégie de surveillance des répertoires/fichiers

La surveillance des répertoires et fichiers est divisés dans chaque structure.

## Côté client

Au moment de la demande d’envoi des fichiers l’application parcours le contenu de la carte SD du téléphone et explore chaque dossier pour récupérer les fichiers.

Associés à chaque fichier nous récupérons comme vu précédemment plusieurs informations utiles pour la surveillance :

* Le nom du fichier
* Contenu
* Sa date de mise à jour
* Son chemin dans la carte SD

Toutes ses informations sont envoyées au serveur qui aura pour tâche de trier les informations reçus avec les fichiers qu’ils possèdent déjà.

## Côté serveur

Une fois le flux des fichiers récupérer le serveur extrait les informations fichier par fichier.

Tout d’abord il traite le chemin de ce fichier, pour expliquer nous allons travailler avec l’exemple ci-dessous :

"/data/doc/fichier/"

On récupère le chemin, sépare les sous-dossiers du fichier et compare au contenu du serveur.

Une fois l’ensemble des sous-dossiers parcouru vérifie l’existence du fichier si il n’existe pas le créer sinon comparer la date de mise à jour si antérieur à la date du fichier mettre à jour.

# Choix des technologies

La solution apparait dans une architecture client-serveur.

Pour communiquer entre nos deux contextes nous utilisons des envois de requête en http, ce mode de communication nous apporte plusieurs avantages :

* un protocole fiable de type requête/réponse, le client et le serveur n'auront qu'à se mettre d'accord sur les données échangées.
* indépendance des plateformes et des langages

En effet si à l’avenir une version sur Ios ou bien Windows Phone de l’application serait à créer nous n’aurions pas besoin de modifier le contenu côté serveur étant données que les communications http sont disponibles sur ces formats.

## Partie mobile

La solution mobile est sur la plateforme Android comme demandé dans les prérequis.

## Partie Serveur

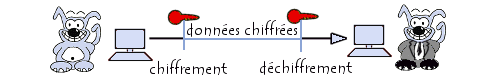
Nous avons opté pour un serveur de type Linux pour plusieurs raisons :

* Open-source donc qui n’entraine pas de frais d’achat
* Peu de maintenance
* Fiable
* Possibilités d’évolution (Gratuite)
* Mise à jour en permanence (Bénévolement)
* Moins sensible aux attaques que d’autre type de serveur
* Utilisable sur des systèmes dits « anciens »

## Chiffrement

Nous avons opté pour un système de chiffrement/déchiffrement en AES128.

Ce système de chiffrement est de type symétrique donc en blocs ce qui signifie que les deux parties voulant communiquer disposent d’une même clé appelé « Clé Secret ». En effet l’envoyeur chiffre ses données avec cette clé et le receveur les déchiffre avec, sans cette clé personne ne peut déchiffrer le message envoyer entre les deux parties.



Source : http://static.commentcamarche.net/www.commentcamarche.net/pictures/crypto-images-cleprivee.gif\*

Nous avons choisi ce système du faite qu’il permet un chiffrement/Déchiffrement plus rapide qu’un système asymétrique (Clé privée, clé publique).

# Architecture

## Client

Le client dispose d’une architecture composée de :

* Une Activity attribué au login (LoginActivity) qui permettra l’authentification entre client et serveur
* Une seconde Activity (SynchroActivity) qui permettra la synchronisation des fichiers
* Chacune de ses Activity dispose d’un fichier contenant ses méthodes (RecoverIds pour LoginActivity et GetFiles pour SynchroActivity)
* Une entity FileText, qui permet de créer des objets associés à ce que l’on récupère des fichiers
* Un fichier de configuration appelé AppConstants qui contient les constantes de l’application

## Serveur

Controller : Il contient tous les fichiers accessibles à l’utilisateur.

* Index.php pour la connexion.
* Link.php pour le transfère de fichier.

Model : Il contient tous les fichiers utilisé par le Controller et contient tous les fonctions.

* Chiffrer.php Les fonctions pour chiffrer/déchiffrer les données.
* Connection.php Celles pour créer une connection.
* Data.php Et enfin pour créer les dossiers et les fichiers.

# Tests mises en place et description

# Changement dans nos choix initiaux de conception

Il y a eu deux changements dans nos choix initiaux :

* Premièrement nous comptions gérer l’ensemble du système via une base de données, une solution abandonnée du faite qu’elle n’était pas nécessaire pour un projet de ce volume
* Ensuite une différence au niveau nommage des dossiers utilisateurs qui contenaient le nom, prénom et identifiant de l’utilisateur, cette solution également abandonnée du faite que nous ne mettions plus en place un fonctionnement en base de données

# Procédure d’installation

## Serveur

Tout d’abord il faut déployer sur votre serveur Linux L’ensemble des fichiers contenu dans le dossier Web.   
Il faudra ensuite modifier les urls du site dans les fichiers suivant:  
data.php :  
 les lignes 38, 6 et 4

Connection.php :   
Les lignes 9 et 23

## Application

Il existe deux moyens pour pouvoir installer l’application sur votre téléphone.

N’oublier pas avant toute installation de vérifier que votre projet dispose des bonnes url liées à votre serveur.

Via l’APK :

La procédure d’installation est assez simple :

* Vous devez tout d’abord autoriser l’installation d’application externe dans l’onglet sécurité du téléphone *Paramètres -> sécurité* cela va permettre étant donné que l’application n’est pas disponible sur le store d’autoriser l’accès
* Télécharger l’APK
* Dirigez-vous dans le dossier ou votre fichier a été télécharger
* Cliquer sur l’application et confirmer en cliquant sur *Installer*

Android Studio

Du côté Android studio la méthode de déploiement est toute aussi simple :

* Récupérer les sources du projet dans le dossier platform/Android
* Démarrer les sources dans Android Studio
* Connecter votre téléphone à votre ordinateur
* Sélectionner *run* dans le menu supérieur *flèche verte*
* Une fenêtre devrait apparaitre vous demandant l’appareil ou installer l’application
* Sélectionner votre téléphone
* Et voilà l’application démarrera normalement sur votre téléphone et sera accessible

# Utilisation

## Application

|  |  |
| --- | --- |
| C:\Users\ProtoConcept GJ\AppData\Local\Microsoft\Windows\INetCache\Content.Word\connection.png | Page de connexion :  L’utilisateur renseigne son identifiant et son mot de passe qui seront envoyé au serveur pour vérifier qu’il existe. |

|  |  |
| --- | --- |
| C:\Users\ProtoConcept GJ\AppData\Local\Microsoft\Windows\INetCache\Content.Word\ErrorConnection.png | En cas d’erreur dans la saisie du mot de passe ou bien de tentative de connexion hors-réseau,  Une fenêtre apparaitra pour avertir l’utilisateur. |

|  |  |
| --- | --- |
| C:\Users\ProtoConcept GJ\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Synchro.png | Page de Synchronisation :  Une fois connecté l’utilisateur est redirigé sur la page de synchronisation.  Pour effectuer la synchronisation il lui suffit de  Cliquer sur le bouton et les fichiers se synchroniserons. |

|  |  |
| --- | --- |
| C:\Users\ProtoConcept GJ\AppData\Local\Microsoft\Windows\INetCache\Content.Word\loading.png | Durant le temps ou l’application synchronise les fichiers avec le serveur une fenêtre de de chargement apparaitra.  Elle permet d’indiquer que le traitement est en cours. |

|  |  |
| --- | --- |
| C:\Users\ProtoConcept GJ\AppData\Local\Microsoft\Windows\INetCache\Content.Word\ErrorSynchro.png | En cas d’erreur de synchronisation une fenêtre d’avertissement apparaitra pour prévenir l’utilisateur. |

## Serveur