ENSIIE

Système d'information, Privacy by Design

Rapport du projet

Groupe 2 :
Increased security

Réalisé par :
Xiaoqi LIU
Lingxiao WANG
Xin LI
Xuesen WANG

18/05/2015

Table des matières

Introduction

L'objectif du projet est développer une application Privacy-by-Design (i.e., protégeant la vie privée des individus dès sa conception) en exploitant la technologie PlugDB.Le projet est proposé de développer une application d'échange de fichiers sécurisée,c'est à dire intégrant du contrôle d'usage.

1 Description générale

Dans notre partie,nous sommes responsable de la partie «Increased security».Elle consiste en deux fonctions générales,«Management de contact» et «Authentication d'utilisateur».

1.1 Management de contact

Dans cette partie,nous avons crée un API «buildCertifiedId»pour récupérer un identifiant certifié prouvant qu'il est bien utilisateur U et donnant ses informations de connexion.Ceci construit un message contenant (Id de U,adresse de la Tcell de U ,clé publique de U),après signe ce message en le hachant et en chiffrant le haché avec la clé privée de U.Pour éviter «Man in the middle attack», on chiffre aussi le haché avec la clé prevée de Tcell.Du coup ,le CertifiedId consiste trois parties :allInfo,signature de U,signature de Tcell. Nous avons aussi crée un API «addContact».Quand un utilisateur U2 recoit un CertifiedId de U1,il va verifier la singnature de U1 par comparant les trois parties de CertifiedId.Si le résultat de comparaision est correct ,U2 va ajouter les information de U1 dans le tableau «USER».

1.2 Authentification d'utilisateur

Dans cette partie,Nous avons réalisé la fonction de créer un utilisateur pour un Tcell.Nous conservons les information de ce utilisateur dans le tableau «MyInfo».

2 Objectif fonctionnel du composant

2.1 Classe AllInfo

Elle comporte 3 éléments : pubKey, userID et tcellIP. Dans cette classe, on crée les méthodes get et set pour récupérer ou attribuer la valeur de chaque élément d'un objet d'allInfo.En plus, on réécrit les méthodes hashCode et equals pour obtenir le hashCode d'un objet d'allInfo et comparer les deux objets d'allInfo.

2.2 Classe CertifiedId

Elle comporte 3 éléments : allInfo, encryptBySelf et encryptByTcell. AllInfo est la classe qu'on a déjà crée au dessus. EncryptBySelf et encryptByTcell sont les deux signatures en chaîne de byte du hashcode d'allInfo avec le

clé privée de Tcell et le clé privée d'utilisateur.Dans cette classe, on crée les méthodes get et set pour récupérer ou attribuer la valeur de chaque élément de un objet CertifiedId.

2.3 Classe Signature

Elle sert à obtenir les deux signatures du hashcode d'allInfo avec le clé privée de Tcell(encryptByTcell) et le clé privée d'utilisateur(encryptBySelf).

2.4 Classe CheckSignature

Elle sert à vérifier(déchiffrer) les deux signatures du hashcode d'allInfo avec le clé privée de Tcell(encryptByTcell) et le clé privée d'utilisateur(encryptBySelf).

2.5 Classe BuildCertifiedIdDB

Elle sert à récupérer les informations d'utilisateurvenant de la classe Tools : UserID, TcellIP, PublicKey et PrivateKey.En plus, On utilise un tableau du base de données MyInfos pour stocker tous ces informations. Enfin, on les relit du tableau pour encapsuler les données dans un objet d'allInfo et obtenir un objet de CertifiedId en utilisant ce objet d'allInfo.

2.6 Classe CreateUserDB

Elle sert à créer un tableau pour stocker les informations d'utilisateur et de la gestionnaire.

2.7 Classe AddContactDB

Elle sert à comparer les 3 parties : le hashcode d'allInfo de un utilisateur, le déchiffre de la signature du hashcode d'allInfo avec le clé privée de Tcell(encryptByTcell) et le déchiffre de la signature du hashcode d'allInfo avec le clé privée du utilisateur(encryptBySelf). Si les 3 partie sont égalité, on peut insérer les informations d'utilisateur(son UserID, TcellIP et PublicKey) dans le tableau du base de données.

3 Description technique du composant

3.1 BuildCertifiedIdDB.java

3.1.1 structure de données

AllInfo :Il est une classe nous avous crée. Il stocque PublicKey, UserId et TcellIp d'utilisateur.

CertifiedId :Il est une clqsse nous avous crée. Il contient 3 composants allInfo, encryptBySelf et encryptByTcell.

EncryptBySelf :Il est un variable de type de array de Byte. Il contient l'information qui est encrypté par PrivateKey d'utilisateur.

EncryptByTcell :Il est un variable de type de array de Byte. Il contient l'information qui est encrypté par PrivateKey de TCell.

3.1.2 algorithme

Pour récupérer l'information non-technical d'utilisateur(UserId, TcellIp) du fichier, nous avous utilisé les méthodes static fornisé par package Tools readDeviceInfo qui renvoye un variable de type DeviceInfo. Ensuit on peut utiliser les méthodes de deviceInfo.getUserID() et deviceInfo.getTCellIP() pour récupérer UserId de TcellIp. On supposer que le privateKey de TCell soit priv00.key. Pour le Publickey et PrivateKey d'user et PrivateKey de TCell, on a utilisé fonctions de Tools.getPublicKey et Tools.getPrivateKey.

Pour stocquer tous les User Infos dans la base de donnée, d'abord il faut Transférer PublicKey et PrivateKey à type String. On a utilisé les méthodes de PublicKeyToString et PrivateKeyToString qui sont fornisé par classe KeyGenerator. Ensuit, on a utilisé la méthode CreateUserDB.createUserDB pour le stoque dans le base de données ;

Pour obtenir l'information de User de base de données, on a utilisé la méthode database.MyTCellDB.getMyInfo() qui renvoyer un ArrayList<MyInfo>.

Pour transferer StringPublickey et StringPrivatekey qui sont récupéré de la base de donnée à type de pubKey et privKey, on a utilisé tools.Tools.stringToPublicKey(String stringPubKey) et tools.Tools.stringToPrivateKey(String stringPrivKey).

Enfin, on doit encrypter le hashcode de AllInfo par PrivateKey d'User et PrivateKey de Tcell. On a utilisé le méthode de cryptoTools.Signature.encryptBlockByBlock() qui est crée par nous pour générer le encryptBySelf et encryptByTcell. Finallement on obtient le CertifiedId.

3.2 AddContactDB.java

Dans cette classe, on doit d'abord vérifier le certifiedId, si il n'est pas correct, on génére une erreur, si non, on stocque l'information de cette user dans la base de donées.

Pour le vérifiation, on génére le hashcode de allinfo et le stocque dans hashCodeAllInfo. Ensuite, on decrypte les encryptBySelf et encryptByTcell par le méthode cryptoTools.CheckSignature.decryptBlockByBlock(byte[] encBytes,PublicKeypubKey) et on vérifie si les deux sont équals à hashCodeAllInfo. Si non, on génére une erreur et retourner. Si oui, on continue à stocquer l'information de cet user dans la base de donées.

Pour stocquer cet user, on doit utiliser le méthode void database.MyTCellDB.InsertUser(StringUserGID ,StringTCellIP ,String PubKey ,Statementstmt) qui est fourni par package MyTCellDB. Mais avant ça, il fault convertir le PublicKey au type string.

3.3 CreateUserDB.java

Cette classe sert à stocquer l'information principale de l'utilisateur quand l'initialiser le Tcell. Pour le réaliser, on a crée deux constructeurs. La première recoit privateKey et publicKey de Type string et la deuxième recoit privateKey et publicKey de type original.

3.4 Signature.java et CheckSignature.java

Pour réaliser le deux méthodes, il fault utiliser la méthode database.MyTCellDB.InsertMyInfo (String UserGID ,String myUserId , String password ,String TCellIP ,String PubKey , String PrivKey ,Statement stmt) qui est crée par nous dans le package de database. Cette méthode va ajouter les informations dans le tableux MyInfo.

4 Description du code

4.1 Signature.java

Utilisation :

```
//Construir un instance
Signature sig = new Signature();
//Signer message inbytes avec private key  prKey
// inbytes de type byte[],prKey de type PrivateKey
byte[] bytes = sig. encryptBlockByBlock(inbytes, prKey );
```

4.2 CheckSignature.java

Utilisation :

```
//Construir un instance
CheckSignature cs = new CheckSignature();
//check message encBytes avec public key pubKey
// encBytes de type byte[],pubKey de type PublicKey
byte[] bytes = cs.decryptBlockByBlock(encBytes, pubKey);
```

4.3 KeyGenerator.java

a. Une classe fournie par les professeurs
b. On a ajouté quelques fonctions
Utilisation :

```
//Construir un instance
KeyGenerator kg = new KeyGenerator()  ;
//convert PublicKey to String
pubKey_ = kg.PublicKeyToString(pubKey);
//convert PrivateKey to String
privKey_ = kg.PublicKeyToString(privKey);
```

4.4 BuildCertifiedIdDB.java

Utilisation :

```
//method static
CertifiedId cId = BuildCertifiedIdDB.buildCertifiedId();
```

4.5 CreateUserDB.java

Utilisation :

```
//method static
//PrivKey(PubKey) peuvent  tre  String ou PrivateKey(PublicKey)
//The other arguments are String
CreateUserDB. createUserDB(UserGid, MyUserId, Password,
    TCellIP, PrivKey, PubKey);
```

4.6 AddContactDB.java

Utilisation :

```
//method static
// certifiedId  est une CertifiedId
AddContactDB.addContact(certifiedId)  ;
```

4.7 CertifiedId.java

Utilisation :

```
// allInfo de type AllInfo, encryptBySelf et encryptByTcell
    sont de type byte[]
CertifiedId ci = new CertifiedId(allInfo, encryptBySelf,
    encryptByTcell);
```

4.8 AllInfo.java

```
Fields :public PublicKey Pubkey_;
        public String UserID_;
        public String TCellIP_;
Constructeur : AllInfo(PublicKey Pubkey, String UserID,
    String TCellIP);
Methods:getters et setters, public int hashCode()
```

Utilisation : constructeur et des méthodes au-dessus

Conclusion

Dans le cadre de notre projet, nous devions réaliser une application d'échanges de fichiers sécurisée, c'est à dire intégrant du contrôle d'accès et du contrôle d'usage. Nous sommes responsable de cette partie «Increased sécurité»,il faut utilister notre connaissance de cryptographie .Mais notre connaissance d'algorithme de la signature n'est pas suffisant pour notre projet .Grâce aux aides et les explications des profs ,nous avons amélioré notre algorithme et mieux compris le projet globalment.Nous avons aussi rencontré quelque difficultés du développement en Java et de la manipulation de base de données ,après les discussions et les recherche sur Internet,nous avons réussi à les résoudre . Au niveau de la gestion du projet en équipe, nous avons réussi à bien nous répartir les tâches afin de réaliser nos objectifs dans les temps et bien communiquer avec les autres équipes.