Smart Social Network
Rapport de projet SSI

25 février 2013

Table des matières

1 Introduction

De nos jours, les réseaux sociaux ont pris une grande ampleur sur internet et accueillent chaque jour de plus en plus d'adhérents. Le principe consiste à y créer un profil, y insérer des données que l'utilisateur désire partagées,

telles que des photos, des vidéos, des messages, etc. Sur ce réseau, des "amis" seront ajoutés : ils pourront alors accéder à ces informations.

La problématique soulevée était de limiter la diffusion des informations à certaines personnes dans nos "amis", mais surtout limiter la diffusion d'informations vis à vis du réseau social lui-même. En effet, lorsque nous partageons une donnée, celui-ci détient cette information qu'elle soit définie comme privée ou non.

L'idée de ce projet était alors de limiter cette fuite d'information, afin de garantir la confidentialité des données des utilisateurs et ce, renforcé par une authentification forte. Nous nous concentrerons sur le réseau social Facebook puisqu'il est le plus utilisé.

La concrétisation du projet s'est traduite par le développement d'une extension pour le logiciel Mozilla Firefox permettant à l'utilisateur de gérer le chiffrement et le déchiffrement de ses données sur le réseau social Facebook, les traitements lourds étant confiés à une application Java.

Concernant l'authentification forte, nous avons utilisé des cartes à puce de type Java Card J3A (marque NXP) avec 40 Kilo-octets (Ko) d'EEPROM, via des lecteurs Omnikey 3121.

L'intérêt du projet était également d'analyser la sécurité de ces cartes à puce, à savoir la génération de nombres aléatoires, de clefs (sysmétriques et asymétriques), chiffrement, déchiffrement et signature. C'est cette même carte à puce qui contiendra à posteriori les données sensibles de l'utilisateur comme son identifiant, son mot de passe, sa clef privée... Le dialogue avec la carte se fait par l'intermédiaire d'un client Java : "SoftCard".

Initialement prévu comme deux projets différents, un par groupe, il s'est avéré que nous travaillerions conjointement pour se concentrer sur un unique projet regroupant :

- l'étude et la mise en œuvre de solutions d'authentifications et de signatures par cartes à puce, proposé par Magali Bardet ;
- les solutions cryptographiques pour les réseaux sociaux, proposé par Ayoub Otmani ;

2 SmartCard

Aujourd'hui nous sommes tous menés à utiliser les cartes à puce comme les cartes bancaires, les cartes vitales. Elles sont notamment utilisées pour effectuer de l'authentification forte et pour contenir des informations confidentielles.

Dans cette partie du projet, nous détaillerons notre étude des solutions cryptographiques pouvant permettre l'authentification ou la signature, puis de mettre à profit ces caractéristiques pour la confidentialité liée à Facebook.

2.1 Génération de nombres aléatoires

La carte à puce permet de générer des nombres aléatoires, utiles dans la création d'IV (Initialization Vector), de mots de passe, de clefs, etc. Ce générateur peut donc être considéré comme un point crucial dans la sécurité de l'application. C'est pour cette raison qu'il a fallu nous assurer que les résultats suivent une distribution uniforme.

La librairie Javacard d'Oracle met à notre disposition deux moteur de génération de nombres aléatoires :

- ALG_PSEUDO_RANDOM : algorithme pseudo aléatoire.
- ALG_SECURE_RANDOM : algorithme cryptographiquement sûr.

Bien entendu, pour notre projet, nous avons utilisé l'algorithme décrit comme "cryptographiquement sûr". Cependant, par acquis de conscience, nous avons voulu vérifier le niveau de l'aléatoire du générateur dit sûr à l'aide d'un outil permettant de réaliser une analyse stastistique dont le compte rendu apparaît ci-dessous :

```
************************************************
*                                              *
*                  CR de Yicheng               *
*                                              *
************************************************
```

2.2 Chiffrement/Déchiffrement

Dans l'étude des cartes à puce, nous avons également utilisé des méthodes de chiffrement et de déchiffrement sysmétriques et asymétriques. Pour notre cas d'utilisation, nous avons choisi des clefs RSA de 1024 bits pour l'algorithme de chiffrement asymétrique RSA-PKCS1. Quant aux opérations cryptographiques symétriques, des clefs AES de 128 bits ont été générées et utilisées. Ces clefs et algorithmes ont été choisis pour leur sûreté.

- Asymétrique : ALG_RSA_PKCS1
- Symétrique : ALG_AES_BLOCK_128_CBC_NOPAD

Nous avons testé ces algorithmes en chiffrant et en déchiffrant, à l'aide de clefs préalablement générées par la carte, divers messages.

Ici, nous avons utilisé ces algorithmes dans différents cas. Concernant le chiffrement par clef publique, bien qu'implanté, nous n'en avons pas eu besoin : c'est en effet Facecrypt, de la seconde partie du projet, qui s'en chargera, la clef publique lui ayant été fournie. Quant au cryptosystème sysmétrique, nous l'avons utilisé lors de la communication avec SoftCard via un "tunnel" sécurisé par AES-128. Ce dernier a pour objectif d'apporter confidentialité, intégrité et authentification aux données échangées entre la carte et SoftCard.

2.3 Signature/Vérification

Nous avons étudié un autre cas où le cryptosystème asymétrique peut être utilisé via la carte à puce : la signature et la vérification de données. Grâce à la signature, nous pouvons obtenir authentification et non-répudiation, puisque la signature se fait via la clef privée de l'émetteur. Une méthode de vérification existe également permettant de vérifier l'authenticité des données via la clef publique du destinataire. Un booléen nous est alors retourné selon la réponse.

2.4 Code PIN/PUK

Une carte à puce étant associée à un utilisateur, il va de soi que ce dernier dispose d'un secret pour pouvoir la carte protéger. Il existe pour cela un code PIN, affecté à chaque carte, et dont seul l'utilisateur a connaissance.

Le code PIN est défini sur deux octets, ce qui représente $2^{16} = 65536$ solutions. Ceci est relativement faible, notamment contre une attaque de type "bruteforce", mais elle est contrée ici via à un nombre d'essais limité. Dans notre cas nous limitons le nombre d'essais à trois. En cas de blocage de la carte, dû à un nombre de tentatives trop élevé, seul le code PUK pourra débloquer la carte. Tout comme le code PIN, le nombre d'essais pour entrer le code PUK est de trois. Si ce nombre est dépassé la carte devient inutilisable et la réinstallation des applets est alors la seule option pour la rendre de nouveau opérationnelle.

Durant l'exécution de l'application, dès qu'une fonctionnalité sensible de la carte sera sollicitée, le code PIN de l'utilisateur lui sera demandé, bloquant ainsi temporairement l'accès à la carte. Au final, un déchiffrement, une signature et une modification des identifiants engendreront une telle interrogation.

4 Terminologie et sigles utilisés

CdR : Cahier de Recettes ;

AdR : Analyse des Risques ;

DAL : Document d'Architecture Logicielle ;

PdD : Plan de développement ;

STB : Spécification Technique de Besoins ;

SC : *SmartCard*, relatif au sous-projet sur les cartes à puce ;

SSN : *Secure Social Network*, relatif au sous-projet sur Facebook ;

FaceCrypt : Application Java gérant les traitements lourds (chiffrement/-déchiffrement) de l'extension et étant en relation avec la carte à puce ;

IHM : Interface Homme-Machine, (interface graphique) ;

Utilisateur : entité (humaine ou programme) interagissant avec ce sous-projet ;

Système : ce sous-projet ;

Sécurisé : ce terme sous-entend un chiffrement des données et une vérification de leur intégrité ;

SoftCard : application effectuant le relais entre la carte et FaceCrypt ;

Extension : programme incorporé dans le navigateur ;

Aléatoire : les résultats suivent une distribution de probabilité uniforme ;

Pseudo-aléatoire : les résultats sont indistingables en temps polynomial d'une distribution de probabilité uniforme ;

PRNG : (Pseudo Random Number Generator) générateur de nombres pseudo-aléatoires ;

RNG : (Random Number Generator) générateur de nombres aléatoires ;

PIN : (Personal Identification Number) code servant à authentifier l'utilisateur auprès de la carte ;

PUK : (Personal Unlock Key) code servant à débloquer la carte quand trop de codes PIN erronés ont été entrés.

5 Annexe A : Fonctionnement détaillé du tunnel entre la carte

5.1 Objectifs

Le tunnel entre la carte et le logiciel qui contrôle le lecteur sert à protéger les communications avec la carte. Contrairement aux autres liens entre logiciels, il n'a pas été possible d'utiliser un protocole de sécurisation tel que TLS car la carte ne dispose pas de pile TCP/IP. En implanter une étant hors de notre domaine de compétences, nous avons utilisé les connaissances acquises en cours en cryptographie pour ajouter une couche de sécurité sur la liaison de données déjà présente.

Les objectifs cryptographiques réalisés par le tunnel sont les suivants :

Confidentialité : Les données ne peuvent pas être lues par une personne non autorisée.

Intégrité : Les données n'ont pas été modifiées durant leur transport.

Authentification : Les données ont été envoyées par une entité qui connait le secret.

Les deux derniers objectifs sont réalisables conjointement sans diminuer la sécurité du système alors que le premier nécessite une clé séparée. Dans la suite, le second objectif désigne l'intégrité et l'authentification.

5.2 Choix techniques

Nous avons utilisé AES avec des clés (différentes) de 128 bits pour les deux objectifs. Le mode CBC permettant de faire du chiffrement et de l'authentification-intégrité [1], nous l'avons utilisé pour les deux. Pour conjuguer performance et sécurité, nous avons implanté une version modifiée de CBC-MAC qui intègre la taille du message en début de message. Cette modification garantit la sécurité lorsque les messages ont une taille variable comme indiqué dans [The Security of the Cipher Block Chaining Message Authentication Code]. De plus, l'utilisation conjointe de CBC en mode chiffrement et de CBC-MAC avec une clé identique rend les attaques particulièrement triviales.

Le choix d'AES nous a paru être le plus pertinent car il correspond bien aux besoins de sécurité du projet. Il a été approuvé par la NSA avec des

1. L'un ou l'autre, jamais les deux en même temps

clés de 128 bits pour protéger des données classifiées au niveau SECRET. L'annexe B1 du RGS publié par l'ANSSI affirme également que cette longueur est satisfaisante.

L'établissement du tunnel se fait selon un protocole challenge-réponse réciproque pour assurer l'authentification mutuelle de la carte et du logiciel avec laquelle elle communique. Une clé de session est envoyée chiffrée par la carte pour assurer la confidentialité des données.

5.3 Etablissement du tunnel

L'authentification mutuelle peut être résumée comme ceci :

- génération d'un nonce client ;
- envoi du nonce client à la carte ;
- la carte récupère le nonce client, génère un nonce carte, un IV et une clé de session ;
- la carte envoie la clé, le nonce client et le nonce carte chiffrés avec la cle partagée et l'IV ;
- le client extrait l'IV et déchiffre le message avec la clé partagée ;
- le client vérifie si le nonce client est présent pour authentifier la carte, extrait la clé de session et récupère le nonce carte ;
- le client génère un IV et renvoie le nonce carte avec la clé de session établie ;
- la carte récupère le nonce carte et vérifie s'il est identique à celui envoyé pour authentifier le client.

C'est le constructeur de l'objet Java "Tunnel" qui s'occupe de tout, côté client.

5.4 Communications dans le tunnel

Une fois les entités mutuellement authentifiées et la clé de session échangée, les communications dans le tunnel peuvent se faire.

L'envoi dans le tunnel d'un fragment se fait comme ceci :

- génération d'un IV de transmission ;
- chiffrement du fragment avec la clé de session et l'IV ;
- concaténation de l'IV et des données chiffrées ;
- calcul du CBC-MAC avec la taille totale ajoutée au début ;
- concaténation de l'IV, des données chiffrées et du MAC ;
- envoi du message final.