Faculté des Sciences & Techniques — Université de Limoges

Master 1ère année

Sécurité des usages TIC

TP n°3

Chiffrement asymétrique et identité

Identité sécurisée

1 – L'objet de cet exercice est de construire un document « Carte Crypto », permettant de lier une clé publique de chiffrement RSA à l'identité de son propriétaire.
Cette « carte crypto » doit être sécurisée afin d'éviter toute modification par une personne mal intentionnée.

a. Une proposition est la suivante :

Carte crypto

- ☐ identité du propriétaire de la clé (nom usuel) sur 20 octets ;
- ☐ adresse électronique sur 20 octets ;
- ☐ clé publique RSA générée comme indiqué dans le TP précédent ;
- ☐ empreinte des champs précédents chiffrée avec la clé privée RSA associée à la clé publique.

Écrivez un programme Python permettant de réaliser automatiquement la création de ce document.
Vous utiliserez la représentation PEM de la clé publique fournie par OpenSSL.

b. Écrivez le programme Python permettant de vérifier cette « carte crypto ».

c. Quelles sont les assurances que fournit cette « carte crypto » ?
Sur quoi repose sa sécurité ?

Document sécurisé et authentifié

2 – Un format de document chiffré et authentifié est construit de la manière suivante :

- ▷ le document est chiffré suivant la méthode « `aes-128-cbc` » ;
- ▷ une « passphrase » est utilisée pour construire la clé de chiffrement ;
- ▷ vous choisirez un algorithme de hachage robuste pour la dérivation de la clé à partir de la passphrase (SHA256) ;
- ▷ la passphrase est alors chiffrée avec la clé publique du destinataire du document ;
- ▷ le chiffrement résultat est transmis en même temps que le document.

Attention

Lors de l'utilisation d'OpenSSL, vous utiliserez l'option « `rsautl` » :

xterm

```
openssl rsautl -help
Usage: rsautl [options]
...
-inkey file     input key
-encrypt        encrypt with public key
-decrypt        decrypt with private key
...
```

Pour chiffrer **avec la clé publique**, vous fournirez la clé privée en entrée car OpenSSL regénère automatiquement la clé publique à partir de la clé privée (qui est en fait la totalité des paramètres RSA et que vous pouvez d'ailleurs afficher avec la commande « `openssl rsa -in private_key -text -noout` »).

1. Écrivez un programme Python réalisant ce travail.
2. Lors de l'utilisation de la « *passphrase* » un paramètre « SALT » peut être utilisé, à quoi sert-il ? Contre quels types d'attaques est-il efficace ?