IUT Béziers
Département Réseaux et Télécoms
Bachelor R&T

2023/2024
R401
F.C.

R401 – Architectures sécurisées

Elements de crypto

TP 1

Consignes

Ce TP vous présente quelques bases d'utilisation des fonctions de crypto. On y verra notamment les fonctions de hachage, la génération de clefs symétriques et asymétriques pour le chiffrement des données et la génération de signatures. En vue de sécuriser un site web, on générera un certificat auto signé pour transformer un site http en https.

Attention !

Le travail est à réaliser en binôme pour tester le chiffrement des informations. Un seul compte rendu sera à rendre, par contre il est vivement conseillé à ce que chaque membre du binôme fasse les manipulations (génération de clef, etc...)

1 Fonction de hachage

Le condensé, de par ses propriétés mathématiques, est un outil de choix pour la vérification de l'intégrité des données. En effet, si le moindre octet est modifié dans le fichier de départ, son condensé change radicalement. En pratique, le condensé permet de vérifier qu'un fichier n'a pas été altéré lors d'une transmission. On s'en sert également pour le stockage des mots de passe sous Linux.

Rappel sur les condensés (Extrait de wikipedia)

A cryptographic hash function is a hash function which is considered practically impossible to invert, that is, to recreate the input data from its hash value alone. These one-way hash functions have been called "the workhorses of modern cryptography". The input data is often called the message, and the output the hash value (the hash) is often called the message digest or simply the digest. The ideal cryptographic hash function has four main properties :

- ▷ it is quick to compute the hash value for any given message
- ▷ it is infeasible to generate a message from its hash value
- ▷ it is infeasible to modify a message without changing the hash value
- ▷ it is infeasible to find two different messages with the same hash value.

1.1 Calcul d'un condensé à l'aide de MD5

A l'aide d'un éditeur quelconque, créez un fichier contenant un texte quelconque. Sauvez-le, puis calculez son condensé à l'aide de la fonction md5sum :

```
test@debian:~$ md5sum fichier.txt
```


Version du document
27 février 2023

Exercice 1 Quel hash obtenez vous ?

Exercice 2 Combien de condensés différents sont possibles ? (indice : c'est de l'hexadécimal !)

Exercice 3 La dernière propriété mathématique énoncée dans Wikipédia est-elle possible en pratique ?

1.2 Vérification des propriétés du condensé

Exercice 4 Reprenez le fichier précédent et renommez-le. Calculez son hash. Que remarquez vous ?

Exercice 5 Modifiez maintenant le contenu du fichier. Recalculez le hash. Que remarquez-vous ?

Exercice 6 Peut-on calculer le hash d'un fichier binaire (un executable par exemple) ? Vérifiez votre réponse.

2 Clefs de chiffrement

2.1 Génération des clefs

Nous allons commencer par créer un couple de clés publique/privée associées à votre Email. Pour ce faire, nous allons utiliser gnupg.

Attention !

Si les commandes ne sont pas disponibles mettez à jour votre système avec apt update suivi d'apt upgrade, puis installez la suite avec : apt instal gnupg.

Vous allez suivre la procédure suivante : Tapez la commande suivante pour générer la paire de clefs.

```
test@debian:~$ gpg --gen-key # version simplifiée
test@debian:~$ gpg --full-generate-key #version avec toutes les options
```

Répondez à toutes les questions interactives. Si vous ajoutez une phrase secrete dans la création de vos clefs elle sera demandée avant de l'utiliser.

Exercice 7 Vérifiez que votre trousseau numérique contient bien votre clef publique avec la commande suivante :

```
test@debian:~$ gpg --list-keys
```

Quels sont les différents champs ?

Exercice 8 Quelle commande permet de liste les clefs privées présentes sur votre machine ? Comment peut-on savoir comment relier la clef publique et la clef anglaise ?

2.2 Diffusion de la clef publique

Pour permettre aux gens de vous envoyer des messages chiffrés, il est nécessaire de leur distribuer votre clef publique. Cela pourrait se faire par Email, mais deux questions se poseraient :

- ▷ Êtes vous sur que l'expéditeur du mail est bien la bonne personne ?
- ▷ Ne peut-on pas faire cela de manière plus compliquée, donc plus rigolote ?

Des serveurs de clés sont mis gratuitement à disposition des internautes pour la distribution des clefs publiques. Nous allons donc exporter votre clef par ce biais :

```
test@debian:~$ gpg --keyserver pgp.mit.edu --send-keys <ID_clef>
gpg: envoi de la clef <ID_clef> au serveur hkp pgp.mit.edu
```

Votre clef publique est désormais accessible par tous vos interlocuteurs sur Internet. Vous pouvez vérifier que votre clé est bien accessible en vérifiant sur le site web suivant : http ://pgp.mit.edu

Attention !

Remarque : tous les serveurs de clés sont interconnectés : il vous suffit de publier votre clé publique sur l'un d'entre eux et elle sera transmise aux autres (cela peut prendre quelques minutes).

Il se peut que les serveurs soient très longs en terme de temps d'accès. Pour éviter d'être bloqué trop longtemps nous allons exporter la clef et l'envoyer à votre binôme. Pour l'exporter on va utiliser la commande suivante (l'option – armor permet de la sauvegarder au format 7 bits donc en hexadécimal) :

```
test@debian:~$ gpg --export --armor > NomFichierClefPublique.key
```

Exercice 9 Quel est le contenu de votre fichier ?

Exercice 10 Il est également possible d'exporter sa clef privée pour la mettre sur son ordinateur (si on ne l'a pas créée sur son ordinateur). Pour celà on utilise la commande suivante :

```
test@debian:~$ gpg --export-secret-keys --armor > NomFichierClefPrivée.key
```

3 Chiffrage d'un fichier

Exercice 11 Pour envoyer un message chiffré à votre binôme, quelle clef faut-il utiliser pour chiffrer le message ?

Pour cela soit on la récupère sur un site de dépôt de clef soit on l'importe à partir d'un fichier reçu.

```
test@debian:~$ gpg --keyserver pgp.mit.edu --recv-keys <ID_Clef>
```

Ou bien à partir d'un fichier :

```
test@debian:~$ gpg --import <NomFichierClefPublique.key>
```

Exercice 12 Vérifier que la clef est importée sur votre trousseau numérique.

Exercice 13 Générer un fichier texte "toto.txt" et chiffrez le avec la commande suivante :

```
test@debian:~$ gpg --armor --recipient <@mail_du_destinataire> --encrypt
    ↪ toto.txt
```

Quel est le fichier généré qui correspond à la version chiffrée de toto.txt ?

Le déchiffrage du fichier se fera avec la clef privée du destinataire. La commande utilisée pour faire cela est :

```
test@debian:~$ gpg --decrypt toto.txt.asc > toto.txt
```

Exercice 14 Vérifier que le fichier déchiffré est équivalent au fichier initial.

4 Signature numérique

Une signature électronique consiste à apposer sur un fichier une marque. Comme toujours en cryptographie, sa définition repose sur quelques propriétés mathématiques :

Signature

A digital signature is a mathematical scheme for demonstrating the authenticity of a digital message or documents. A valid digital signature gives a recipient reason to believe that the message was created by a known sender, that the sender cannot deny having sent the message (authentication and non-repudiation), and that the message was not altered in transit (integrity).

En pratique, vous connaissez tous les outils nécessaires à la signature électronique. Une signature est un condensé de la pièce à signer, chiffré à l'aide de la clef privée du signataire.

- ▷ Authentifiation : On peut vérifier à l'aide de votre clé publique que vous êtes bien le signataire
- ▷ Non-repudation : Seule votre clé privée a pu générer ce chiffrement
- ▷ Integrité : Le condensé garantit que la pièce n'a pas été modifiée depuis la signature

4.1 Signature d'un fichier

Les commandes pour signer un document sont les suivantes :

```
test@debian:~$ gpg --sign <Fichier> #signature et compression
test@debian:~$ gpg --clearsign <Fichier> #signature sans compression
test@debian:~$ gpg --detach-sign <Fichier> #si on ne souhaite que la
    ↪ signature sans le fichier
test@debian:~$ gpg [-u expéditeur] [-r destinataire] [--armor] --sign
    ↪ --encrypt <Fichier> #permet de chiffrer en plus de la signature.
```

Exercice 15 Tester les commandes ci-dessus et vérifier leurs résultat.

Pour vérifier la signature on peut utiliser la commande suivante :

```
test@debian:~$ gpg --verify <Fichier_avec_signature.asc>
```

Exercice 16 Vérifier la signature du fichier précédent.

4.2 Signature d'une clef publique

Il reste un dernier problème. Qui vous garantit qu'une clé publique téléchargée sur Internet est bien légitime ? Après tout, on pourrait imaginer créer une clé au nom de n'importe qui ! En pratique, la confiance repose sur la signature des clés publiques. Vous pouvez signer la clé publique de vos collègues pour signaler au monde que d'après vous, cette personne est bien celle qu'elle prétend être. On en arrive ainsi à la notion de chaine de confiance :

- ▷ Vous signez la clé de Paul
- ▷ Paul signe la clé de Marie
- ▷ Marie signe la clé de Bob

Donc Bob est à une distance de confiance de 3 par rapport à vous ! Plus une clé publique est signée, plus elle est proche de vous, plus vous pouvez penser qu'elle est légitime.

Exercice 17 Signer la clef de votre binôme avec la commande suivante :

```
test@debian:~$ gpg --keyserver pgp.mit.edu --recv-keys <ID_Clef>
    ↪ #récupération de la clef
test@debian:~$ gpg --keyserver pgp.mit.edu --recv-keys <ID_Clef>
    ↪ #récupération de la clef
test@debian:~$ gpg --sign-key <ID_clef>
test@debian:~$ gpg --keyserver pgp.mit.edu --send-key <ID_Clef>
```

Vérifier sur pgp.mit.edu que la clé a bien été signée. Indiquer l'identité du signataire.

5 Utilisation d'un certificat

Rappel : un certificat est une clef publique plus quelques informations signée (certifiée) par une autorité de certification (CA). Normalement cette opération est payante. Néanmoins Let's encrypt permet de générer des certificats gratuitement.

Certificat autosigné

Il est possible de signer soi même son certificat, cependant les navigateurs ne le reconnaitront pas et renverront un message d'alerte. C'est ce que nous ferons dans ce TP.

Dans un premier temps il faut installer openssl

```
test@debian:~$ apt install openssl
```

Il faut ensuite générer sa clef. Dans l'exemple on utilisera une clef de taille 4096 bits encrypté avec AES256.

```
test@debian:~$ openssl genrsa -aes256 -out MaClef.key 4096
```

Si on entre un mot de passe il faudra le saisir à chaque fois que le serveur apache sera redémarré. Pour éviter celà on peut généré un certificat "débloqué"

```
test@debian:~$ openssl rsa -in MaClef.key.lock -out MaClef.key
```

Générer le fichier de demande de signature de notre clef et renseigner tous les champs demandés :

```
test@debian:~$ openssl req -new -key MaClef.key.lock -out certificat.csr
```

Pour auto-signer son certificat X509 (RFC 5280) il faut exécuter la commamnde suivante.

```
test@debian:~$ openssl x509 -req -days 365 -in certificat.csr -signkey
    ↪ MaClef.key.lock -out certificat.crt
```

Installer ensuite apache2 pour le serveur web. Vous modifierez légèrement la page d'accueil pour la simplifier. Elle se trouve ici : */var/www/html/index.html*.

Ensuite il faut activer le module ssl pour le serveur apache avec la commande suivante :

```
test@debian:~$ e2enmode ssl
```

Ensuite on va rediriger le trafic du port 80 vers le port 443. Pour cela on créé un vHost sur le port 443 (port du https) en éditant un fichier de configuration inspiré de ceux situés ici : */etc/apache2/site − available s/*. Exemlpe de fichier */etc/apache2/site − available s/test.conf*

```
<VirtualHost *:80>
 ServerName      test.rt.local
 # On redirige le port HTTP vers le port HTTPS
 Redirect        / https://www.rt.local
irtualHost>
rtualHost *:443>
 ServerName      test.rt.local
 DocumentRoot    /var/www/test

 SSLEngine on
 SSLCertificateFile    /etc/ssl/www/certificat.crt
 SSLCertificateKeyFile /etc/ssl/www/MaClef.key
 # Facultatif, on accepte tout les protocoles SSL sauf SSLv2 et SSLv3 (donc
    ↪ on accepte que le TLS ici)
 SSLProtocol all -SSLv2 -SSLv3
 # Facultatif, c'est le serveur qui donne l'ordre des algorithmes de
    ↪ chiffrement pendant la négociation avec le client.
 SSLHonorCipherOrder on
 # Facultatif, liste des algorithmes de chiffrement disponibles.
 SSLCipherSuite
    ↪ ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:EC
irtualHost>
```

Ensuite on active le VHost avec :

```
test@debian:~$ a2ensite test.conf
test@debian:~$ service apache2 restart
```

Exercice 18 Effectuer la configuration sur le serveur. Faut-il prévoir quelque chose sur le client ? Vérifier si le trafic est chiffré lors de l'envoi de la page web.

Exercice 19 Quels sont d'après vous les risques pour un administrateur réseau d'avoir des utilisateurs qui utilisent principalement des sites en https ?

6 Client mail

Exercice 20 Installer et configurer sur votre machine le client Email Thunderbird. Envoyez à votre binôme un courrier à la fois chiffré et signé.