Année Universitaire 2015-2016
3ième année IMAC

Mardi 19 janvier 2016

Cloud computing
Travaux pratiques

Objectif

Se donner une vision théorique et pratique de la sécurité de la correspondance par mail.

En particulier sont abordés :le chiffrement et l'authenticité des contenus.

Un peu de théorie

Chiffrement symétrique

Alice chiffre un message pour l'envoyer à Bob :

$Message_{chiffré} = Fonction\_Chiffrer(Clé_{symétrique}, Message_{initial})$

Bob déchiffre le message :

$Fonction\_Déchiffrer(Clé_{symétrique}, Message_{chiffré}) = Message_{initial}$

Le chiffrement et l'authenticité sont bien traités, mais si Alice possède 150 correspondants, elle devra partager 150 clés, ce qui n'est pas pratique.

Chiffrement asymétrique

Le chiffrement asymétrique permet d'avoir un couple de clés (clé privée, clé publique) unique pour l'ensemble de ses correspondants :

Pour envoyer un message chiffré à Bob, Alice utilise la clé publique de Bob

$Message_{chiffré} = Fonction\_Chiffrer(Clé_{publique\ de\ Bob}, Message_{initial})$

Bob reçoit le message chiffré avec sa clé publique et il est le seul à pouvoir déchiffrer le message :

$Message_{initial} = Fonction\_Déchiffrer(Clé_{privée\ de\ Bob}, Message_{chiffré})$

Ce message a pu être envoyé par n'importe qui et rien de prouve que Alice est bien l'expéditeur.

Signature numérique

Alice doit donc prouver qu'elle est bien l'expéditeur, elle utilise donc un mécanisme de signature numérique en suivant les étapes suivante :

1) création d'un condensat (empreinte) du message à l'aide d'une fonction de hachage :

   Condensat = Fonction_Hash (Message $_{initial}$)

2) chiffrer le condensat avec sa clé privée :

   Signature = Fonction_Chiffrer (Clé $_{privée\ de\ Alice}$, Condensat)

Finalement :

Signature = Fonction_Chiffrer (Clé $_{privée\ de\ Alice}$, Fonction_Hash (Message $_{initial}$))

Bob reçoit Message $_{initial}$ et Signature et pour vérifier la signature, il doit donc vérifier l'égalité :

Fonction_Hash (Message $_{initial}$) = Fonction_Déchiffrer (Clé $_{publique\ de\ Alice}$, Signature)

Chiffrement et signature

Alice souhaite envoyer un message à Bob

1) fabrication de la signature :

   Signature = Fonction_Chiffrer (Clé $_{privée\ de\ Alice}$, Fonction_Hash (Message $_{initial}$))

2) chiffrer le message :

   Message $_{chiffré}$ = Fonction_Chiffrer (Clé $_{publique\ de\ Bob}$, Message $_{initial}$)

3) envoie des deux éléments Message $_{chiffré}$ et Signature à Bob

4) Bob déchiffre le message :

   Message $_{initial}$= Fonction_Déchiffrer (Clé $_{privée\ de\ Bob}$, Message $_{chiffré}$)

5) Bob vérifie l'authenticité du message : il fabrique le condensat
   Fonction_Hash (Message $_{initial}$),
   il déchiffre la signature
   Fonction_Déchiffrer (Clé $_{publique\ de\ Alice}$, Signature)
   et le compare au condensat.

Définitions

Enigmail : extension pour client de messagerie qui utilise GnuPG

GnuPG ou Gnu Privacy Guard ou GPG : implémentation libre de PGP

PGP ou Pretty Good Privacy : PGP : logiciel créé par Phil Zimmermann en 1991, code source ouvert mais non libre ?

OpenPGP : standard (RFC 4880) qui décrit le protocole

Un peu de pratique

Installer un client de messagerie :

apt-get install icedove-l10n-fr

passer l'étape de la configuration automatique et configurer les éléments suivants où X est votre numéro d'étudiant :

nom et prénom : imacX où X est votre numéro d'étudiant

Adresse électronique : imacX@test.wargon.org où X est votre numéro d'étudiant

mot de passe : j7a2W4hX où X est votre numéro d'étudiant

la configuration trouvée par icedove doit être bonne, cliquer sur Terminé

accepter les certificats qui sont des certificats autosignés pour le serveur de lecture des mails d'une part et pour le serveur d'envoie de mail d'autre part.

Envoyer et recevoir des mails avec un autre compte imacX permettra de vérifier que les configurations sont bonnes.

installer l'extension Enigmail :

sur un système Debian : apt-get install enigmail

à partir des dépôts de modules Thunderbird :

Menu / Modules complémentaires / chercher enigmail / installer / redémarrer icedove

Utiliser l'assistant – Suivant

Ne pas chiffrer mes messages par défaut – Suivant

Ne pas signer mes messages par défaut – Suivant

Modifier les paramètres pour que Enigmail fonctionne mieux – Suivant

Créer une nouvelle paire de clés – Suivant

clé = « maclemonsecretX »

générer le certificat de révocation de la clé

ne pas déverrouiller automatiquement la clé, cela permet d'avoir une visibilité sur les mails dans leur format chiffré.

(B) envoie d'un mail chiffré à (A) :

(A) envoie sa clé publique à (B) : Menu / Enigmail / Gestion des clés / Fichier / Envoyer des clés publiques par courrier électronique

(B) importe la clé publique de (A) clic droit sur le fichier joint puis Importer une clef OpenPGP

(B) peut envoyer un mail chiffré à (A) la clé en bas à droite de la fenêtre doit être jaune.

(A) envoie un mail signé à (B) :

le crayon en bas à droite de la fenêtre doit être jaune.

(B) reçoit le mail, il peut vérifier la signature avec la clé publique de (A) et définir avec le menu

« Détail » le niveau de confiance de l'expéditeur, essayer la confiance absolue.

(B) envoie un mail chiffré et signé à (A)

le crayon et la clé en bas à droite de la fenêtre doivent être jaunes

(A) peut déchiffrer le mail, mais ne peut pas vérifier la signature, pourquoi ?

Trouver une solution.

Enfin envoyer un mail chiffré et signé de (A) vers (B)

Finalement, envoyer votre clé publique à laurent@wargon.org afin que je puisse vérifier votre signature, puis envoyer un mail chiffré et signé.

A refaire depuis votre email personnel pour le 2 février.

Chiffrer un fichier

Dans un terminal, fabriquer un fichier :

```
echo "je suis un contenu secret" > fichier_secret.txt
```

vérifier le contenu du fichier :

```
cat fichier_secret.txt
```

Chiffrer un fichier :

```
gpg -c fichier_secret.txt
```

vérifier que le fichier est bien chiffré

```
cat fichier_secret.txt.gpg
```

vous pouvez maintenant effacer le fichier initial :

```
rm fichier_secret.txt
```

pour déchiffrer le fichier :

```
gpg -d fichier_secret.txt.gpg
```

Créer une demande de certificat :

Expérimenter la création d'une demande de certificat :

si nécessaire :

```
apt-get install openssl
```

```
openssl req -nodes -newkey rsa:2048 -sha256 -keyout cle_privee.key -out demande_cert.csr
```

Références :

une introduction intéressante se trouve là :

http://www.controle-tes-donnees.net/outils/GnuPG.html

man gpg ; man openssl