MasterI RSS

Tavaux Pratiques
SSL

M. Moustapha Kaba

Plan

- Présentation de SSL
- RSA avec OpenSSL
- Certificats

Présentation de SSL

- Protocole SSL
- OpenSSL

Protocole SSL

- Le protocole SSL (Secure Socket Layer) a été développé par la société Netscape Communications Corporation pour permettre aux applications client/serveur de communiquer de façon sécurisée. TLS (Transport Layer Security) est une évolution de SSL réalisée par l'IETF. La version 3 de SSL est utilisée par les navigateurs tels Netscape et Microsoft Internet Explorer depuis leur version 4.

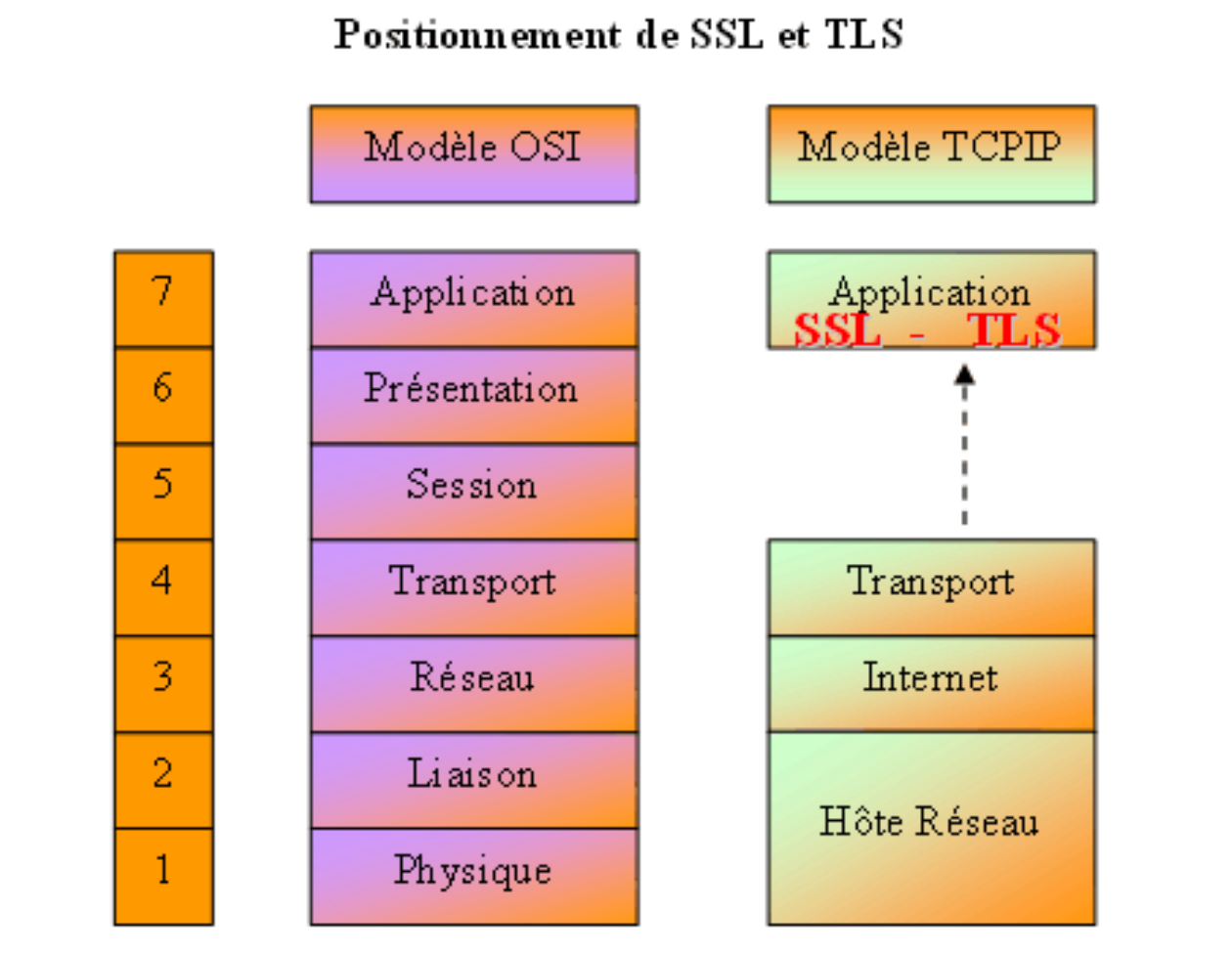
Protocole SSL

- SSL est un protocole qui s'intercale entre TCP/IP et les applications qui s'appuient sur TCP. Une session SSL se déroule en deux temps
  - 1. une phase de poignée de mains (handshake) durant laquelle le client et le serveur s'identifient, conviennent du système de chiffrement et d'une clé qu'ils utiliseront par la suite.
  - 2. la phase de communication proprement dite durant laquelle les données échangées sont compressées, chiffrées et signées.
- L'identification durant la poignée de mains est assurée à l'aide de certificats X509.

OpenSSL

- OpenSSL est une boîte à outils cryptographiques implémentant les protocoles SSL et TLS qui offre
  - 1. une bibliothèque de programmation en C permettant de réaliser des applications client/serveur sécurisées s'appuyant sur SSL/TLS.
  - 2. une commande en ligne (openssl) permettant
    - la création de clés RSA, DSA (signature)
    - la création de certificats X509
    - le calcul d'empreintes (MD5, SHA, RIPEMD160, ...)
    - le chiffrement et déchiffrement (DES, IDEA, RC2, RC4, Blowfish, ...)
    - la réalisation de tests de clients et serveurs SSL/TLS
    - la signature et le chiffrement de courriers (S/MIME)

Position de SSL/TLS



OpenSSL

- Pour connaître toutes les fonctionnalités de OpenSSL : *man openssl*.
- La syntaxe générale de la commande openssl est

<commande> <options>

- Dans le texte qui suit, les commandes invoquant openssl supposent que cette commande est dans votre PATH.

RSA avec OpenSSL

- Génération d'une paire de clés
- Visualisation des clés RSA
- Chiffrement d'un fichier de clés RSA
- Exportation de la partie publique
- Chiffrement/déchiffrement de données avec RSA
- Signature de fichiers

Génération d'une paire de clés

- On peut générer une paire de clés RSA avec la commande **genrsa** de OpenSSL.

```
genrsa -out <fichier> <taille>
```

où fichier est un nom de fichier de sauvegarde de la clé, et taille et la taille souhaitée (exprimée en bits) du modulo de la clé.

- Par exemple, pour générer une paire de clés de 1024 bits, stockée dans le fichier maCle.pem, on tape la commande

```
genrsa -out maCle.pem 1024
```

- Le fichier obtenu est un fichier au format PEM (Privacy Enhanced Mail, format en base 64).

Visualisation des clés RSA

- La commande rsa permet de visualiser le contenu d'un fichier au format PEM contenant une paire de clés RSA.

```
rsa –in <fichier> –text –noout
```

- L'option *-text* demande l'affichage décodé de la paire de clés. L'option *-noout* supprime la sortie normalement produite par la commande *rsa*.

Visualisation des clés RSA

- Exemple **rsa –in maCle.pem –text –noout** donne

```
Private-Key: (1024 bit)
modulus:
    00:af:79:58:cb:96:d7:af:4c:2e:64:48:08:93:62:
    31:cc:56:e0:11:f3:40:c7:30:b5:82:a7:70:4e:55:
    9e:3d:79:7c:2b:69:7c:4e:ec:07:ca:5a:90:39:83:
    4c:05:66:06:4d:11:12:1f:15:86:82:9e:f6:90:0d:
    00:3e:f4:14:48:7e:c4:92:af:7a:12:c3:43:32:e5:
    20:fa:7a:0d:79:bf:45:66:26:6b:cf:77:c2:e0:07:
    2a:49:1d:ba:fa:7f:93:17:5a:a9:ed:bf:3a:74:42:
    f8:3a:75:d7:8d:a5:42:2b:aa:49:21:e2:e0:df:1c:
    50:d6:ab:2a:e4:41:40:af:2b
public Exponent: 65537 (0x10001)
private Exponent:
    35:c8:54:ad:f9:ea:db:c0:d6:cb:47:c4:d1:1f:9c:
    b1:cb:c2:db:dd:99:f2:33:7c:be:b2:01:5b:11:24:
    f2:24:a5:29:4d:28:9b:ab:fe:6b:48:3c:c2:53:fa:
    de:00:ba:57:ae:ae:c6:36:3b:c7:17:5f:ed:20:fe:
    fd:4c:a4:56:5e:0f:18:5c:a6:84:bb:72:c1:27:46:
    96:07:9c:de:d2:e0:06:d5:77:ca:d2:45:8a:50:15:
    0c:18:a3:2f:34:30:51:e8:02:3b:8c:ed:d4:95:98:
    73:ab:ef:69:57:4d:c9:04:9a:18:82:1e:60:6b:0d:
    0d:61:18:94:eb:43:4a:59
```

Visualisation des clés RSA

prime1 :

00:d5:46:7f:49:4b:5e:46:f2:9f:87:e1:9d:0f:6d:
4f:59:42:0d:8b:d6:ef:7f:48:3f:e4:10:b5:5e:dc:
ac:75:7d:d1:ee:97:11:f4:bf:c5:76:02:03:dd:5c:
od:bd:36:18:be:4f:15:2d:0b:e5:7c:08:c7:36:29:
79:80:bc:5b:07

prime2 :

00:d2:ao:43:9a:31:cc:0c:fc:1c:19:aa:9b:43:69:
72:99:84:08:1a:56:79:4c:b4:05:03:df:03:7b:2d:
ae:13:eo:81:ea:99:92:fd:ef:d9:8a:5b:b5:21:a6:
ac:b8:4d:ef:07:35:df:07:f2:54:ec:35:24:57:ef:
89:43:b4:ed:bd

exponent1 :

00:bd:ac:e5:d5:1c:87:6b:17:aa:53:a1:8e:1a:43:
3f:f7:84:ec:21:3a:f5:62:co:b1:b9:b6:36:67:78:
60:94:59:62:d4:0b:5c:f7:cb:79:e4:9a:a4:2f:41:
08:23:07:b2:77:c6:43:71:fd:8b:89:85:11:0e:95:
52:2e:fo:d5:0f

exponent2 :

04:1a:a5:56:92:d3:d4:08:f1:8f:3a:78:ce:06:76:
fa:30:cd:6b:9d:f5:bd:1d:eo:df:23:70:50:ed:21:
fo:37:36:bo:d8:8f:39:ad:7b:c2:ab:68:cb:20:11:
4b:82:11:3f:45:b8:73:d2:2f:ff:6e:45:a8:04:fd:
da:b8:e2:cd

coef icient :

23:10:53:83:cc:aa:43:2d:c3:30:85:b1:5f:19:a8:
b9:a4:0c:f9:f5:6e:29:c8:03:04:4b:60:57:2c:41:
10:ed:81:38:ba:af:27:33:dc:f9:35:84:25:73:05:
fc:8c:77:cc:fo:aa:9c:0a:99:1e:45:ao:e5:ee:24:
4b:fe:99:58

Visualisation des clés RSA

- Les différents éléments de la clé sont affichés en hexadécimal (hormis l'exposant public). On peut distinguer :
  - le modulus;
  - l'exposant public (qui par défaut est toujours 655371);
  - l'exposant privé;
  - les nombres premiers facteurs du modulus;
  - plus trois autres nombres qui servent à optimiser l'algorithme de déchiffrement.

Chiffrement d'un fichier de clés RSA

- Il n'est pas prudent de laisser une paire de clé en clair (surtout la partie privée). Avec la commande rsa, il est possible de chiffrer une paire de clés . Pour cela trois options sont possibles qui précisent l'algorithme de chiffrement symétrique à utiliser : -des, -des3 et -idea.

```
rsa –in  maCle . pem  –des3  –out   maCle . pem
writing RSA key
Enter PEM pass phrase :
Verifying – Enter  PEM  pass phrase :
```

- Une phrase de passe est demandée deux fois pour générer une clé symétrique protégeant l'accès à la clé.

Exportation de la partie publique

- La partie publique d'une paire de clés RSA est publique, et à ce titre peut être communiquée à n'importe qui. Le fichier maCle.pem contient la partie privée de la clé, et ne peut donc pas être communiqué tel quel (même s'il est chiffré). Avec l'option ***-pubout*** on peut exporter la partie publique d'une clé.

rsa –in maCle.pem –pubout –out maClePublique . pem

Chiffrement/déchiffrement de données avec RSA

- On peut chiffrer des données avec une clé RSA. Pour cela on utilise la commande ***rsautl***

```
rsautl –encrypt –in <fichierentree> –inkey <cle> –out <fichiersortie>
```

- où
  - – fichier entrée est le fichier des données à chiffrer. Attention, le fichier des données à chiffrer ne doit pas avoir une taille excessive (ne doit pas dépasser 116 octets pour une clé de 1024 bits).
  - – clé est le fichier contenant la clé RSA. Si ce fichier ne contient que la partie publique de la clé, il faut rajouter l'option ***-pubin***.
  - – fichier sortie est le fichier de données chiffré.
- Pour déchiffrer on remplace l'option ***-encrypt*** par ***-decrypt***. Le fichier contenant la clé doit obligatoirement contenir la partie privée.

Signature de fichiers

- Il n'est possible de signer que de petits documents. Pour signer un gros document on calcule d'abord une empreinte de ce document. La commande ***dgst*** permet de le faire.

dgs t <hachage> –out <empreinte> <f i c h i e r e n t r e e >

- où hachage est une fonction de hachage. Avec openssl, plusieurs fonctions de hachage sont proposées dont
  - –MD5 (option -md5), qui calcule des empreintes de 128 bits,
  - – SHA1 (option -sha1), qui calcule des empreintes de 160 bits,
  - – RIPEMD160 (option -ripemd160), qui calcule des empreintes de 160 bits.

Signature de fichiers

- Signer un document revient à signer son empreinte. Pour cela, on utilise l'option ***-sign*** de la commande ***rsautl***

```
rsautl –sign –in <empreinte> –inkey <cle> –out <signature>
```

- et pour vérifier la signature

```
rsautl –verify –in <signature> –pubin –inkey <cle> –out
<empreinte>
```

- il reste ensuite à vérifier que l'empreinte ainsi produite est la même que celle que l'on peut calculer.
- L'option ***-pubin*** indique que la clé utilisée pour la vérification est la partie publique de la clé utilisée pour la signature.

Certificats

- Génération de la paire de clés
- Création d'une requête de certificats
- Demande de signature de certificat
- Signature et chiffrement de courriers électroniques

Certificats

- Vous allez maintenant élaborer un certificat pour votre clé publique. Puis, vous verrez comment utiliser les clés certifiées pour signer et/ou chiffrer des courriers électroniques.

Génération de la paire de clés

- Exercice 1 : Générez votre paire de clés RSA d'une taille de 1024 bits, protégée par un mot de passe (cf génération d'une paire de clés). Dans la suite, on suppose nommé maCle.pem le fichier contenant votre paire de clés RSA. Ce fichier est protégé par le mot de passe fourni lors de la génération.
- Exercice 2 : Créez un fichier ne contenant que la partie publique de votre clé RSA (cf Exportation de la clé publique). Dans la suite, on suppose ce fichier nommé maClePublique.pem.

Création d'une requête de certificats

- Maintenant que vous disposez d'une clé RSA, vous allez établir une requête pour obtenir un certificat. Outre la clé publique du sujet, un certificat contient un certain nombre d'informations concernant l'identité de son propriétaire :
  - – Pays (C),
  - – État ou province (ST)
  - – Ville ou localité (L)
  - – Organisation (O)
  - – Unité (OU)
  - – Nom (CN)
  - – Email

Création d'une requête de certificats

- Toutes ces informations et d'autres encore sont demandées lors de la création de la requête. Un fichier de configuration (***req.cnf***) peut-être défini qui propose les informations à apporter dans le certificat avec des valeurs par défaut.
- On établit une requête avec la commande ***req*** de openssl.

```
req  -config  req.cnf  -new  -key maCle.pem  -out  maRequete.pem
```

- Le fichier produit maRequete.pem est aussi au format PEM.

Création d'une requête de certificats

- Le contenu du fichier maRequete.pem

−−−−−BEGIN CERTIFICATE REQUEST−−−−−
MIIB/zCCAWgCAQAwgb4xCzAJBgNVBAYTAkZSMRIwEAYDVQQIEwlOb3JkICg1OSkx
GjAYBgNVBAcTEVZpbGxlbmV1dmUgZCdBc2NxMR4wHAYDVQQKExVVbml2ZXJzaXRl
IGRlIExpbGxlIDExGTAXBgNVBAsTEExpY2VuY2UgUHJvIERBMkkxGjAYBgNVBAMT
EUVyaWMgV2Vncnp5bm93c2tpMSgwJgYJKoZIhvcNAQkBFhlFcmljLldlZ3J6eW5v
d3NraUBsaWZsLmZyMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCveVjLltev
TC5kSAiTYjHMVuAR80DHMLWCp3BOVZ49eXwraXxO7AfKWpA5g0wFZgZNERIfFYaC
nvaQDQA+9BRIfsSSr3oSw0My5SD6eg15v0VmJmvPd8LgBypJHbr6f5MXWqntvzp0
Qvg6ddeNpUIrqkkh4uDfHFDWqyrkQUCvKwIDAQABoAAwDQYJKoZIhvcNAQEEBQAD
gYEAHOwGNN6A8d4EhjfXCRvC2fhIjt7i6jxfkHKBkHpm2yNBBDHQwiv+O/Y0MeNh
I r a+y8KUMjeImsSiH4731sfgA6ycm+6JoDV7n6z8tzN5QMGsw7V3ErduskayKP4T
j a+BMImEcDwlr+KuRO704rGeiAG7pvtDGcDcj2Mex68ki94=
−−−−−END CERTIFICATE REQUEST−−−−−

Création d'une requête de certificats

- On peut consulter les informations contenues dans la requête avec la commande

```
req -config req.cnf -in maRequete.pem -text -noout
```

```
Certificate Request:
    Data:Version: 0 (0x0)
    Subject: C=FR, ST=Nord (59), L=Villeneuve d'Ascq, O=Universite de Lille 1, OU=Li Subject Public Key Info:
    Public Key Algorithm: rsaEncryption
    RSA Public Key: (1024 bit)
        Modulus (1024 bit):
        00:af:79:58:cb:96:d7:af:4c:2e:64:48:08:93:62:
                31:cc:56:e0:11:f3:40:c7:30:b5:82:a7:70:4e:55:
                9e:3d:79:7c:2b:69:7c:4e:ec:07:ca:5a:90:39:83:
                4c:05:66:06:4d:11:12:1f:15:86:82:9e:f6:90:0d:
                00:3e:f4:14:48:7e:c4:92:af:7a:12:c3:43:32:e5:
                20:fa:7a:0d:79:bf:45:66:26:6b:cf:77:c2:e0:07:
                2a:49:1d:ba:fa:7f:93:17:5a:a9:ed:bf:3a:74:42:
                f8:3a:75:d7:8d:a5:42:2b:aa:49:21:e2:e0:df:1c:
                50:d6:ab:2a:e4:41:40:af:2b
        Exponent: 65537 (0x10001)
    Attributes:
      a0:00
Signature Algorithm: md5WithRSAEncryption
        1c:ec:06:34:de:80:f1:de:04:86:37:d7:09:1b:c2:d9:f8:48:
        8e:de:e2:ea:3c:5f:90:72:81:90:7a:66:db:23:41:04:31:d0:
        c2:2b:fe:3b:f6:34:31:e3:61:22:b6:be:cb:c2:94:32:37:88:
        9a:c4:a2:1f:8e:f7:d6:c7:e0:03:ac:9c:9b:ee:89:a0:35:7b:
        9f:ac:fc:b7:33:79:40:c1:ac:c3:b5:77:12:b7:6e:b2:46:b2:
        28:fe:13:8d:af:81:30:89:84:70:3c:25:af:e2:ae:44:ee:f4:
        e2:b1:9e:88:01:bb:a6:fb:43:19:c0:dc:8f:63:1e:c7:af:24:
        8b:de
```

Demande de signature de certificat

- Une fois que nous avons établi une requête de certificat, il nous reste à contacter une autorité de certification qui nous délivrera un certificat signé, après avoir procédé (normalement) à quelques vérifications nous concernant.
  - L'autorité de certification Création d'un certificat
  - Création d'un certificat
  - Vérification de certificats

Demande de signature de certificat

- L'autorité de certification :
  - Vous jouerez dans ce TP le rôle de l'autorité de certification. Pour cela il vous faut un certificat d'autorité de certification, ainsi qu'une paire de clés. Afin de ne pas multiplier les autorités, vous utiliserez tous la trés notable autorité Père Ubu.

Demande de signature de certificat

- Un certificat produit par openssl est un fichier au format PEM.

```
-----BEGIN CERTIFICATE-----

. . . .


-----END CERTIFICATE-----
```

- Pour visualiser le contenu d'un certificat

```
x509  -in  unCertif.pem  -text  -noout
```