TD 12 : OpenSSL

christina.boura@uvsq.fr
d'après M. Petitot (Université Lille 1)

2 mai 2017

1 Présentation de OpenSSL

1.1 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.

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. une 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.

1.2 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).

Pour connaître toutes les fonctionnalités de openSSL : `man openssl`.

La syntaxe générale de la commande `openssl` est

```
$ openssl <commande> <options>
```

(le $ étant le prompt du shell)

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

2 Chiffrement symétrique avec openSSL

C'est la commande `enc` qui permet de chiffrer/déchiffrer avec `openssl` :

```
$ openssl enc<options>
```

Tapez `man enc` pour connaître les algorithmes symétriques disponibles.

Remarque : `base64` n'est pas un système de chiffrement, mais un codage des fichiers binaires avec 64 caractères ASCII. Ce codage est utilisé en particulier pour la transmission de fichiers binaires par courrier électronique.

2.1 Chiffrement avec mot de passe

Pour chiffrer le fichier *clair* avec le chiffrement par blocs `3DES` avec une clé générée par mot de passe, le chiffré étant stocké dans le fichier *chiffre*, on utilise la commande :

```
$ openssl enc -des3 -in clair -out chiffre
enter des-ede3-cbc encryption password:
Verifying - enter des-ede3-cbc encryption password:
```

Pour déchiffrer le message chiffré, on utilise la commande :

```
$ openssl enc -des3 -d -in chiffre -out chiffre.dechiffre
```

Vérification que tout s'est bien passé :

```
$ diff clair chiffre.dechiffre
```

Exercice 1

1. Chiffrez le fichier de votre choix avec le système de votre choix dans le mode de votre choix, puis déchiffrez-le.
2. Comparez les tailles des fichiers clairs et chiffrés. Donnez une explication sur la différence de ces tailles.
3. Tentez de déchiffrer un cryptogramme en utilisant un mauvais mot de passe. Comment réagit `openSSL` ?

Exercice 2 Le fichier `chiffre1` a été chiffré avec le système AES en mode CBC, la clé de 128 bits ayant été obtenue par mot de passe.

1. Le mot de passe codé en base 64 est `c3Ryb25ncGFzc3dvcmQK`. À l'aide de la commande openssl appropriée, décodez le mot de passe.
2. Déchiffrez ensuite le `chiffre1`.

3 RSA avec `openssl`

3.1 Génération d'une paire de clés

On peut générer une paire de clés RSA avec la commande `genrsa` de `openssl`, où *fichier* est un nom de fichier de sauvegarde de la clé, et *taille* est la taille souhaitée (exprimée en bits) du module de la clé.

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

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

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

Le fichier obtenu est un fichier au format PEM (Privacy Enhanced Mail, format en base 64), dont voici un exemple

```
$ cat maCle.pem
-----BEGIN RSA PRIVATE KEY-----
MIICWwIBAAKBgQDAPLwQeDq1LFydtmM7UBebrM6CaX2EZ5iihuHqxWfPFfgL6YAB
WPjoA0JxC87oNKxjV4jSfPv3cuP7kwAK6XEIY7zUttyQmAbAvllsCXU7MvA2EYT2
uiIs4k80jzOa1+F8sXebTpCNoJOHnh5doscr8MJgow+LPFY55KhgSuyl4QIDAQAB
AoGATzW3WCHyLszErMeGQ95Qm/isbtgAvBnjrrRdSltk1tN6jY39DJEYnqEZM3Nm
Ig/yKEzdN74AJInWnsaHcfeWduMO+IJzcZ5mfIWPmUc1L4D2awbz3q5AOuHCVY/c
ufN08pRBusoNwcBvMuOr+tfToNdHXOkMNRCchvGO+2kZXUECQQDxEEU7+0vhsptX
RLYVxITSg6XcmmngZLZ1C9LMT3+j/k2fiGXvI66OMGHyYdvKJpwIEh+64i60Hf1A
8iM/vfrFAkEAzCX8Qofju4XU+SYW2YB9cBrYWyZ8mb2t7FxNTSoqceXH9AzfB7+T
qzF6W4uuECgyXrHOdaWzUu0yYZk4YPNgbQJAXwzsT69Rh0e2ip10MncPbDYugyJ7
ltf/PX2Q+7BpAs+16a6NitKGA1SEel7tm/LHWUNUMYsXoTul7SLMlUiihQJATMtZ
Onm9zBPLCrIuVEFGbn5atuciZf75RvltxsI+1zuV3RNebp69YN+q5HcF0mQcloyS
WqUrcN1zX01w7N+AfQJAFYLySnHn/g9w/ZFN5TALdzsW7t8gDOkKfVJ4qbCGL/JE
VWrewxGS/nFtD84v+01/CDRQaG/3r6WoJIg94t2T3w==
-----END RSA PRIVATE KEY-----
```

3.2 Visualisation des clés RSA

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

```
$ openssl 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`.

Par exemple

```
openssl rsa -in maCle.pem -text -noout
Private-Key: (1024 bit)
modulus:
    00:c0:3c:bc:10:78:3a:b5:2c:5c:9d:b6:63:3b:50:
    17:9b:ac:ce:82:69:7d:84:67:98:a2:86:e1:ea:c5:
    67:cf:15:f8:0b:e9:80:01:58:f8:e8:03:42:71:0b:
    ce:e8:34:ac:63:57:88:d2:7c:fb:f7:72:e3:fb:93:
    00:0a:e9:71:08:63:bc:d4:b6:dc:90:98:06:c0:be:
    59:6c:09:75:3b:32:f0:36:11:84:f6:ba:22:2c:e2:
    4f:34:8f:33:9a:d7:e1:7c:b1:77:9b:4e:90:8d:a0:
    93:87:9e:1e:5d:a2:c7:2b:f0:c2:60:a3:0f:8b:3c:
    56:39:e4:a8:60:4a:ec:a5:e1
publicExponent: 65537 (0x10001)
privateExponent:
    4f:35:b7:58:21:f2:2e:cc:c4:ac:c7:86:43:de:50:
    9b:f8:ac:6e:d8:00:bc:19:e3:ae:b4:5d:4a:5b:64:
    d6:d3:7a:8d:8d:fd:0c:91:18:9e:a1:19:33:73:66:
    22:0f:f2:28:4c:dd:37:be:00:24:89:d6:9e:c6:87:
    71:f7:96:76:e3:0e:f8:82:73:71:9e:66:7c:85:8f:
    99:47:35:2f:80:f6:6b:06:f3:de:ae:40:3a:e1:c2:
    55:8f:dc:b9:f3:74:f2:94:41:ba:ca:0d:c1:c0:6f:
    32:e3:ab:fa:d7:d3:a0:d7:47:5c:e9:0c:35:10:9c:
    86:f1:8e:fb:69:19:5d:41
prime1:
    00:f1:10:45:3b:fb:4b:e1:b2:9b:57:44:b6:15:c4:
    84:d2:83:a5:dc:9a:69:e0:64:b6:75:0b:d2:cc:4f:
    7f:a3:fe:4d:9f:88:65:ef:23:ae:b4:30:61:f2:61:
    db:ca:26:9c:08:12:1f:ba:e2:2e:b4:1d:fd:40:f2:
    23:3f:bd:fa:c5
```

```
prime2:
    00:cc:25:fc:42:87:e3:bb:85:d4:f9:26:16:d9:80:
    7d:70:1a:d8:5b:26:7c:99:bd:ad:ec:5c:4d:4d:2a:
    2a:71:e5:c7:f4:0c:df:07:bf:93:ab:31:7a:5b:8b:
    ae:10:28:32:5e:b1:ce:75:a5:b3:52:ed:32:61:99:
    38:60:f3:60:6d
exponent1:
    5f:0c:ec:4f:af:51:87:47:b6:8a:9d:74:32:77:0f:
    6c:36:2e:83:22:7b:96:d7:ff:3d:7d:90:fb:b0:69:
    02:cf:b5:e9:ae:8d:8a:d2:86:03:54:84:7a:5e:ed:
    9b:f2:c7:59:43:54:31:8b:17:a1:3b:a5:ed:22:cc:
    95:48:a2:85
exponent2:
    4c:cb:59:3a:79:bd:cc:13:cb:0a:b2:2e:54:41:46:
    6e:7e:5a:b6:e7:22:65:fe:f9:46:f9:6d:c6:c2:3e:
    d7:3b:95:dd:13:5e:6e:9e:bd:60:df:aa:e4:77:05:
    d2:64:1c:96:8c:92:5a:a5:2b:70:dd:73:5f:4d:70:
    ec:df:80:7d
coefficient:
    15:82:f2:4a:71:e7:fe:0f:70:fd:91:4d:e5:30:0b:
    77:3b:16:ee:df:20:0c:e9:0a:7d:52:78:a9:b0:86:
    2f:f2:44:55:6a:de:c3:11:92:fe:71:6d:0f:ce:2f:
    fb:4d:7f:08:34:50:68:6f:f7:af:a5:a8:24:88:3d:
    e2:dd:93:df
```

Les différents éléments de la clé sont affichés en hexadécimal (hormis l'exposant public). On peut distinguer le module, l'exposant public (qui par défaut est toujours 65537), l'exposant privé, les nombres premiers facteurs du module, plus trois autres nombres.

Exercice 2. Générez vos clés RSA.

Exercice 3. Donnez une explication du choix de la valeur 65537 pour l'exposant public par défaut.

Exercice 4. À quoi servent les trois derniers paramètres ?

3.3 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`. (Il est possible de chiffrer le fichier lors de sa génération. Il suffit de mettre l'une des trois options `-des`, `-des3`, `-idea` dans la ligne de commande `genrsa`).

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

Un mot de passe est demandé deux fois pour générer une clé symétrique protégeant l'accès à la clé.

Exercice 5. Avec la commande `cat` observez le contenu du fichier `maCle.pem`. À quoi correspondent les informations affichées ? Utilisez à nouveau la commande `rsa` pour visualiser le contenu de la clé.

3.4 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.

```
$ openssl rsa -in maCle.pem -pubout -out maClePublique.pem
```

Exercice 6. Avec la commande rsa visualisez la clé publique. Attention vous devez préciser l'option `-pubin`, puisque seule la partie publique figure dans le fichier `maClePublique.pem`.

3.5 Chiffrement/déchiffrement de données avec RSA

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

```
$ openssl rsautl -encrypt -in <fichier ent.> -inkey <cle> -out <fichier sor.>
```

où
- fichier ent. 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).
- *cle* 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 sor.* est le fichier des données chiffré.

Pour déchiffrer on remplace l'option `-encrypt` par `-decrypt`. Le fichier contenant la clé doit obligatoirement contenir la partie privée.

Exercice 7. Chiffrez le fichier de votre choix avec le système symétrique de votre choix. Chiffrez le mot de passe utilisé(e) avec la clé publique de votre destinataire (demandez lui sa clé publique). Envoyez-lui le mot de passe chiffré ainsi que le fichier chiffré.

3.6 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.

```
$ openssl dgst <hachage> -out <empreinte> <fichier entrée>
```

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.

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

```
$ openssl rsautl -sign -in <empreinte> -inkey <cle> -out <signature>
```

et pour vérifier la signature

```
$ openssl 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 utilisée pour la signature.

Exercice 8. Signez le fichier de votre choix, puis vérifiez la signature.

Exercice 9. Récupérez le dossier `Signatures` qui contient deux fichiers accompagnés d'une signature
- fichier : `cigale.txt`, signature `signature1`
- fichier : `QuandLaMerMonte.txt`, signature `signature2`,

ainsi que la partie publique de la clé RSA ayant produit la signature : `uneClePublique.pem`.
De ces fichiers, lequel a été bien signé ?

4 Certificats

Vous allez maintenant élaborer un certificat pour votre clé publique.

4.0.1 Création d'une requête de certificats

Vous allez ici 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

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`.

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

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

```
$cat maRequete.pem
-----BEGIN CERTIFICATE REQUEST-----
MIIB/TCCAWYCAQAwgbwxCzAJBgNVBAYTAkZSMRswGQYDVQQIExJJbGUtZGUtRnJh
bmNlICg3NSkxDjAMBgNVBAcTBVBhcmlzMSEwHwYDVQQKExhVbml2ZXJzaXRlIGRl
IFZlcnNhaWxsZXMxFTATBgNVBAsTDExpY2VuY2UzIElORjEYMBYGA1UEAxMPQ2hy
aXN0aW5hIEJvdXJhMSwwKgYJKoZIhvcNAQkBFh1jaHJpc3RpbmEuYm91cmFAcHJp
c2OudXZzcS5mcjCBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEAwDy8EHg6tSxc
nbZjO1AXm6zOgml9hGeYoobh6sVnzxX4C+mAAVj46ANCcQvO6DSsY1eIOnz793Lj
+5MACulxCGO81LbckJgGwL5ZbAl1OzLwNhGE9roiLOJPNI8zmtfhfLF3m06QjaCT
h54eXaLHK/DCYKMPizxWOeSoYErspeECAwEAAaAAMA0GCSqGSIb3DQEBBQUAA4GB
AHeFLnh9/1tllf857HwA3y4lHI+QAZDgZO6ZoQJ7BApXXXXkZGiI3kIIXRQU86ES
FZHtYAkGliJq6tQdGqdJzianaiA21sGGlBOr405mP/ETFF0aeNbzR5XjWJhKWnHm
pZgVPsbLAybcCnLHVCehOm575xUkUs/pbYP3HuQWC5gh
-----END CERTIFICATE REQUEST-----
```

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

```
$openssl req -config req.cnf -in maRequete.pem -text -noout
Certificate Request:
    Data:
       Version: 0 (0x0)
       Subject: C=FR, ST=Ile-de-France (75), L=Paris, O=Universite de Versailles,
       OU=Licence3 INF, CN=Christina Boura/emailAddress=christina.boura@uvsq.fr
       Subject Public Key Info:
           Public Key Algorithm: rsaEncryption
           Public-Key: (1024 bit)
           Modulus:
               00:c0:3c:bc:10:78:3a:b5:2c:5c:9d:b6:63:3b:50:
               17:9b:ac:ce:82:69:7d:84:67:98:a2:86:e1:ea:c5:
               67:cf:15:f8:0b:e9:80:01:58:f8:e8:03:42:71:0b:
               ce:e8:34:ac:63:57:88:d2:7c:fb:f7:72:e3:fb:93:
               00:0a:e9:71:08:63:bc:d4:b6:dc:90:98:06:c0:be:
               59:6c:09:75:3b:32:f0:36:11:84:f6:ba:22:2c:e2:
               4f:34:8f:33:9a:d7:e1:7c:b1:77:9b:4e:90:8d:a0:
               93:87:9e:1e:5d:a2:c7:2b:f0:c2:60:a3:0f:8b:3c:
               56:39:e4:a8:60:4a:ec:a5:e1
           Exponent: 65537 (0x10001)
       Attributes:
           a0:00
```

```
Signature Algorithm: sha1WithRSAEncryption
       77:85:2e:78:7d:ff:5b:65:95:ff:39:ec:7c:00:df:2e:25:1c:
       8f:90:01:90:e0:64:ee:99:a1:02:7b:04:0a:57:5d:75:e4:64:
       68:88:de:42:08:5d:14:14:f3:a1:12:15:91:ed:60:09:06:96:
       22:6a:ea:d4:1d:1a:a7:49:ce:26:a7:6a:20:36:d6:c1:86:94:
       13:ab:e3:4e:66:3f:f1:13:14:5d:1a:78:d6:f3:47:95:e3:58:
       98:4a:5a:71:e6:a5:98:15:3e:c6:cb:03:26:dc:0a:72:c7:54:
       27:a1:3a:6e:7b:e7:15:24:52:cf:e9:6d:83:f7:1e:e4:16:0b:
       98:21
```

Exercice 10 Expliquez les différents éléments contenus dans cette requête. La clé privée du sujet y figure–t–elle ?

4.1 Demande de signature de certificat

Une fois que vous avez établi une requete de certificat, il vous reste à contacter une autorité de certification qui vous délivrera un certificat signé, après avoir procédé (normalement) à quelques vérifications vous concernant.

L'autorité de certification Père Noël

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 très notable autorité Père Noël dont vous trouverez dans l'archive le certificat et la paire de clés RSA.

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

```
$cat unCertif.pem
-----BEGIN CERTIFICATE-----

..........

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

Pour visualiser le contenu d'un certificat

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

Exercice 11 Après avoir récupéré le certificat de l'autorité, ainsi que sa paire de clés RSA, cherchez quelle est la date d'expiration du certificat et la taille de la clé.

Création d'un certificat

Pour créer et signer un certificat à partir d'une requête maRequete.pem, l'autorité invoque la commande x509

```
$ openssl x509 -days 10 -CAserial pereNoel.srl -CA pereNoelCertif.pem -CAkey
clePereNoel.pem -in maRequete.pem -req -out monCertif.pem
```

dans laquelle

- l'option `-days` détermine la durée de validité du certificat (ici 10 jours) ;
- `pereNoel.srl` est un fichier contenant le numéro de série du prochain certificat à signer (ce fichier est un fichier texte contenant une ligne donnant un nombre codé avec un nombre pair de chiffres hexadécimaux).

Exercice 12 Créez un certificat pour votre clé publique. (Lors de la signature du certificat, la commande `x509` invite l'autorité de certification à donner son mot de passe. Le voici codé en base 64 `Z2FyZ2FtZWw=`. Puis contrôlez le contenu du certificat obtenu avec les options appropriées de la commande x509.

Vérification de certificats

On peut vérifier la validité d'un certificat avec la commande `verify`. Pour vérifier la validité d'un certificat, il est nécessaire de disposer le certificat de l'autorité qui l'a émis.

```
$ openssl verify -CAfile pereNoelCertif.pem monCertif.pem
```