Faculté des Sciences de Rabat, Master IPS 2021-2020

# Sécurité Informatique

TP2 : RSA et signature avec openSSL

1. Présentation de **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.

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, chi rées 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, Blow sh, )

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 (help sous windows) La syntaxe générale de la commande openssl est

$ openssl <commande> <options >

(le $ est le prompt du shell)

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

1. RSA et signature avec **openSSL**

## 2.1. Génération d'une paire de clés

On peut générer une paire de clés RSA avec la commande genrsa de openSSL.

$ openssl genrsa *−*out <f i c h i e r > < t a i l l e >

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

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

KXmAvFsHAkEA0qBDmjHMDPwcGaqbQ2lymYQIGlZ5TLQFA98Dey2uE+CB6pmS/ e /Z ilu1IaasuE3vBzXfB /JU7DUkV++JQ7TtvQJBAL2s5dUch2sXqlOhjhpDP/eE7CE6 9WLAsbm2Nmd4YJRZYtQLXPfLeeSapC9BCCMHsnfGQ3H9i4mFEQ6VUi7w1Q8CQAQa

pVaS09QI8Y86eM4GdvowzWud9b0d4N8jcFDtIfA3NrDYjzmte8KraMsgEUuCET9F uHPSL/9uRagE/dq44s0CQCMQU4PMqkMtwzCFsV8ZqLmkDPn1binIAwRLYFcsQRDt gTi6rycz3Pk1hCVzBfyMd8zwqpwKmR5FoOXuJEv+mVg=

*−−−−−*END RSA PRIVATE KEY*−−−−−*

MIICXAIBAAKBgQCveVjLltevTC5kSAiTYjHMVuAR80DHMLWCp3BOVZ49eXwraXxO

7AfKWpA5g0wFZgZNERIfFYaCnvaQDQA+9BRIfsSSr3oSw0My5SD6eg15v0VmJmvP d8LgBypJHbr6f5MXWqntvzp0Qvg6ddeNpUIrqkkh4uDfHFDWqyrkQUCvKwIDAQAB AoGANchUrfnq28DWy0fE0R+cscvC292Z8jN8vrIBWxEk8iSlKU0om6v+a0g8wlP6 3 gC6V66uxjY7xxdf7SD+/UykVl4PGFymhLtywSdGlgec3tLgBtV3ytJFilAVDBij LzQwUegCO4zt1JWYc6vvaVdNyQSaGIIeYGsNDWEYlOtDSlkCQQDVRn9JS15G8p+H 4Z0PbU9ZQg2L1u9/SD/kELVe3Kx1fdHulxH0v8V2AgPdXA29Nhi+TxUtC+V8CMc2

*−−−−−*BEGIN RSA PRIVATE KEY*−−−−−*

$ cat maCle.pem

## 2.2. 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

0 0 : a f : 7 9 : 5 8 : cb : 9 6 : d7 : a f : 4 c : 2 e : 6 4 : 4 8 : 0 8 : 9 3 : 6 2 :

3 1 : c c : 5 6 : e0 : 1 1 : f 3 : 4 0 : c7 : 3 0 : b5 : 8 2 : a7 : 7 0 : 4 e : 5 5 :

9 e : 3 d : 7 9 : 7 c : 2 b : 6 9 : 7 c : 4 e : e c : 0 7 : ca : 5 a : 9 0 : 3 9 : 8 3 :

4 c : 0 5 : 6 6 : 0 6 : 4 d : 1 1 : 1 2 : 1 f : 1 5 : 8 6 : 8 2 : 9 e : f 6 : 9 0 : 0 d :

0 0 : 3 e : f 4 : 1 4 : 4 8 : 7 e : c4 : 9 2 : a f : 7 a : 1 2 : c3 : 4 3 : 3 2 : e5 :

2 0 : f a : 7 a : 0 d : 7 9 : b f : 4 5 : 6 6 : 2 6 : 6 b : c f : 7 7 : c2 : e0 : 0 7 :

2 a : 4 9 : 1 d : ba : f a : 7 f : 9 3 : 1 7 : 5 a : a9 : ed : b f : 3 a : 7 4 : 4 2 :

f 8 : 3 a : 7 5 : d7 : 8 d : a5 : 4 2 : 2 b : aa : 4 9 : 2 1 : e2 : e0 : d f : 1 c :

5 0 : d6 : ab : 2 a : e4 : 4 1 : 4 0 : a f : 2 b

p u b lic E xp o n e n t : 65537 (0 x10001 ) p r iv ate E x p o n e n t :

3 5 : c8 : 5 4 : ad : f 9 : ea : db : c0 : d6 : cb : 4 7 : c4 : d1 : 1 f : 9 c :

b1 : cb : c2 : db : dd : 9 9 : f 2 : 3 3 : 7 c : be : b2 : 0 1 : 5 b : 1 1 : 2 4 :

f 2 : 2 4 : a5 : 2 9 : 4 d : 2 8 : 9 b : ab : f e : 6 b : 4 8 : 3 c : c2 : 5 3 : f a :

de : 0 0 : ba : 5 7 : ae : ae : c6 : 3 6 : 3 b : c7 : 1 7 : 5 f : ed : 2 0 : f e :

f d : 4 c : a4 : 5 6 : 5 e : 0 f : 1 8 : 5 c : a6 : 8 4 : bb : 7 2 : c1 : 2 7 : 4 6 :

9 6 : 0 7 : 9 c : de : d2 : e0 : 0 6 : d5 : 7 7 : ca : d2 : 4 5 : 8 a : 5 0 : 1 5 :

0 c : 1 8 : a3 : 2 f : 3 4 : 3 0 : 5 1 : e8 : 0 2 : 3 b : 8 c : ed : d4 : 9 5 : 9 8 :

7 3 : ab : e f : 6 9 : 5 7 : 4 d : c9 : 0 4 : 9 a : 1 8 : 8 2 : 1 e : 6 0 : 6 b : 0 d :

0d : 6 1 : 1 8 : 9 4 : eb : 4 3 : 4 a : 5 9

prime1 :

0 0 : d5 : 4 6 : 7 f : 4 9 : 4 b : 5 e : 4 6 : f 2 : 9 f : 8 7 : e1 : 9 d : 0 f : 6 d :

4 f : 5 9 : 4 2 : 0 d : 8 b : d6 : e f : 7 f : 4 8 : 3 f : e4 : 1 0 : b5 : 5 e : dc :

ac : 7 5 : 7 d : d1 : e e : 9 7 : 1 1 : f 4 : b f : c5 : 7 6 : 0 2 : 0 3 : dd : 5 c :

0d : bd : 3 6 : 1 8 : be : 4 f : 1 5 : 2 d : 0 b : e5 : 7 c : 0 8 : c7 : 3 6 : 2 9 :

7 9 : 8 0 : bc : 5 b : 0 7

P r i v ate *−*Key : (1 0 2 4 b i t )

modulus :

$ openssl rsa *−*i n maCle.pem *−*t e x t *−*noout

prime2 :

0 0 : d2 : a0 : 4 3 : 9 a : 3 1 : c c : 0 c : f c : 1 c : 1 9 : aa : 9 b : 4 3 : 6 9 :

7 2 : 9 9 : 8 4 : 0 8 : 1 a : 5 6 : 7 9 : 4 c : b4 : 0 5 : 0 3 : d f : 0 3 : 7 b : 2 d :

ae : 1 3 : e0 : 8 1 : ea : 9 9 : 9 2 : f d : e f : d9 : 8 a : 5 b : b5 : 2 1 : a6 :

ac : b8 : 4 d : e f : 0 7 : 3 5 : d f : 0 7 : f 2 : 5 4 : e c : 3 5 : 2 4 : 5 7 : e f :

8 9 : 4 3 : b4 : ed : bd exponent1 :

0 0 : bd : ac : e5 : d5 : 1 c : 8 7 : 6 b : 1 7 : aa : 5 3 : a1 : 8 e : 1 a : 4 3 :

3 f : f 7 : 8 4 : e c : 2 1 : 3 a : f 5 : 6 2 : c0 : b1 : b9 : b6 : 3 6 : 6 7 : 7 8 :

6 0 : 9 4 : 5 9 : 6 2 : d4 : 0 b : 5 c : f 7 : cb : 7 9 : e4 : 9 a : a4 : 2 f : 4 1 :

0 8 : 2 3 : 0 7 : b2 : 7 7 : c6 : 4 3 : 7 1 : f d : 8 b : 8 9 : 8 5 : 1 1 : 0 e : 9 5 :

5 2 : 2 e : f 0 : d5 : 0 f

exponent2 :

0 4 : 1 a : a5 : 5 6 : 9 2 : d3 : d4 : 0 8 : f 1 : 8 f : 3 a : 7 8 : c e : 0 6 : 7 6 :

f a : 3 0 : cd : 6 b : 9 d : f 5 : bd : 1 d : e0 : d f : 2 3 : 7 0 : 5 0 : ed : 2 1 :

f 0 : 3 7 : 3 6 : b0 : d8 : 8 f : 3 9 : ad : 7 b : c2 : ab : 6 8 : cb : 2 0 : 1 1 :

4b : 8 2 : 1 1 : 3 f : 4 5 : b8 : 7 3 : d2 : 2 f : f f : 6 e : 4 5 : a8 : 0 4 : f d :

da : b8 : e2 : cd c o e f f i c i e n t :

2 3 : 1 0 : 5 3 : 8 3 : c c : aa : 4 3 : 2 d : c3 : 3 0 : 8 5 : b1 : 5 f : 1 9 : a8 :

b9 : a4 : 0 c : f 9 : f 5 : 6 e : 2 9 : c8 : 0 3 : 0 4 : 4 b : 6 0 : 5 7 : 2 c : 4 1 :

1 0 : ed : 8 1 : 3 8 : ba : a f : 2 7 : 3 3 : dc : f 9 : 3 5 : 8 4 : 2 5 : 7 3 : 0 5 :

f c : 8 c : 7 7 : c c : f 0 : aa : 9 c : 0 a : 9 9 : 1 e : 4 5 : a0 : e5 : e e : 2 4 :

4b : f e : 9 9 : 5 8

Les différents éléments de la clé sont affichés en hexadécimal (hormis l'exposant public). On peut distinguer le modulos, l'exposant public (qui par défaut est toujours 65537 [1](#_bookmark0) ), l'exposant privé, les nombres premiers facteurs du modulos, plus trois autres nombres qui servent à optimiser l'algorithme de déchiffrement.

Exercice 1. Générez votre clé RSA et donnez une explication du choix de la valeur 65537 pour exposant public par défaut.

## 2.3. Chiffrement d'un fichier de clés RSA

Il n'est pas prudent de laisser une paire de clés en clair (surtout la partie privée). Avec la commande rsa, il est possible de chiffrer une paire de clés [2](#_bookmark1) . Pour cela trois options sont possibles qui précisent l'algorithme de chiffrement symétrique à utiliser : -des, -des3 et -idea.

$ openssl rsa in maCle.pem d e s 3 out maCle.pem w r i t i n g 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. Exercice 2. Avec la commande cat (type sous windows) observez le contenu du fichier maCle.pem. Utilisez à nouveau la commande rsa pour visualiser le contenu de la clé.

## 2.4. 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 3.

Question 1. Notez le contenu du fichier maClePublique.pem. Remarquez les marqueurs de début et de fin.

* + 1. openssl n'autorise que deux exposants publics : 65537 (valeur par défaut) ou 3 obtenu avec 'option -3
    2. 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.

Question 2. 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.

## 2.5. Chiffrement/déchiffrement de données avec RSA

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

*−*out <f i c h i e r \_ s o r t i e >

$ openssl rsautl *−*encrypt *−*in <f i c h i e r \_ e n t r e e > *−*i n k e y <cl e > \

Où

* fichier\_entree 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 parte 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 évidemment contenir la partie privée.

Exercice 4. Chiffrez le fichier de votre choix avec le système symétrique de votre choix. Chiffrez la clé ou le mot de passe utilisé(e) avec la clé publique de votre destinataire (demandez-lui sa clé publique si vous ne l'avez pas). Envoyez-lui le mot de passe chiffré ainsi que le fichier chiffré.

Exercice 5. Il s'agit de déchiffrer le fichier cryptogram14. Pour cela vous devez récupérer les fichiers suivants (attention la plupart de ces fichiers sont des fichiers binaires)

* Le fichier cryptogram14 (fichier binaire) a été obtenu en chiffrant un texte avec openssl et la commande enc. Le système de chiffrement symétrique utilisé est BlowFish en mode CBC avec un vecteur d'initialisation nul (voir TP1) ;
* La clé BlowFish a été dérivée à partir d'un mot de passe dont la version chiffrée est contenue dans le fichier motDePasse.rsa ;
* La clé privée RSA qui a servi à chiffrer le mot de passe est contenue dans le fichier uneCle.pem;
* Le mot de passe protégeant la clé RSA, codé en base 64, est ZnNyLXVtNQ== .

## 2.6. 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\_entree >

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

*−*in <empreinte> \

*−*out <signature >

*−*inkey <cle > \

$ openssl rsautl *−*sign

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 de la clé utilisée pour la signature.

Exercice 6. Signez le fichier de votre choix, puis vérifiez la signature.

Exercice 7. Récupérez l'archive signatures.rar qui contient deux fichiers accompagnés d'une signature

* fichier : cigale.txt, signature : signature1
* et fichier : QuandLaMerMonte.txt, signature : signature2,

ainsi que la partie publique de la clé RSA ayant produit la signature : uneClePublique.pem.

De ces deux fichiers, lequel a bien été signé ?