

Travail pratique 1

Thème

Services de sécurité et mécanismes cryptographiques

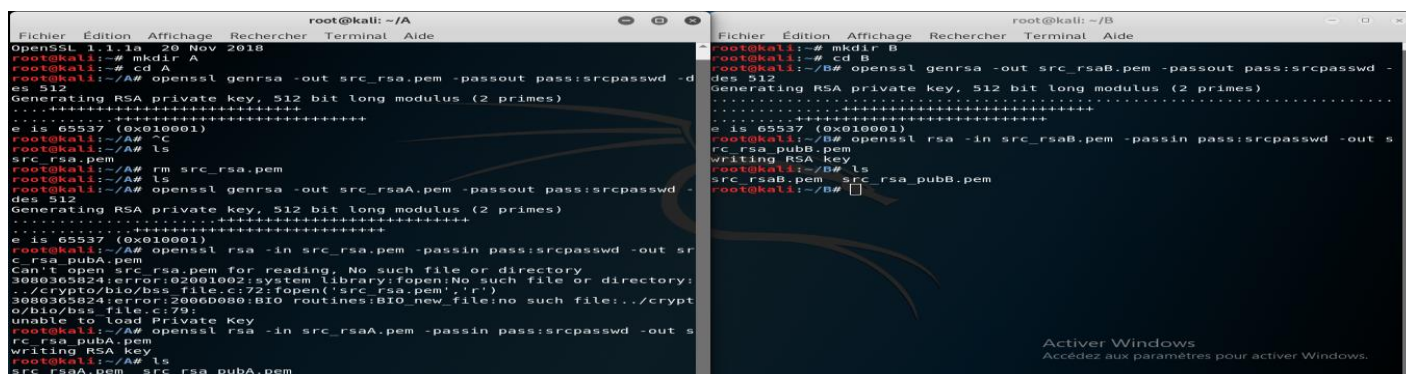
Réalisé par

- ADLA ilyes chiheb eddine

Implémentation d'un protocole d'échange sécurisé avec OpenSSL :

Ce tp concerne l'implémentation du protocole qu'on avait conçu dans le dernier exercice.

La partie réalisation consiste à ouvrir deux lignes de commande : une pour le fichier A et l'autre pour le fichier B. Chaque fichier représente un utilisateur, l'un joue le rôle d'un émetteur et l'autre joue le rôle d'un récepteur. Dans notre cas le A est l'émetteur et B est le récepteur.



```
root@kali: ~/A
OpenSSL 1.1.1a 20 Nov 2018
root@kali:~# mkdir A
root@kali:~# cd A
root@kali:~/A# openssl genrsa -out src_rsa.pem -passout pass:srcpasswd -des 512
Generating RSA private key, 512 bit long modulus (2 primes)
.....+++++
e is 65537 (0x010001)
root@kali:~/A# ^C
root@kali:~/A# ls
src_rsa.pem
root@kali:~/A# rm src_rsa.pem
root@kali:~/A# ls
root@kali:~/A# openssl genrsa -out src_rsaA.pem -passout pass:srcpasswd -des 512
Generating RSA private key, 512 bit long modulus (2 primes)
.....+++++
e is 65537 (0x010001)
root@kali:~/A# openssl rsa -in src_rsaA.pem -passin pass:srcpasswd -out src_rsaA.pem
Can't open src_rsaA.pem for reading: No such file or directory
3080365824:error:02001002:system library:fopen:No such file or directory:/crypto/bio/bss_file.c:72:fopen('src_rsaA.pem','r')
3080365824:error:2006D080:BIO routines:BIO_new_file:no such file:../crypto/bio/bss_file.c:79:
unable to load Private Key
root@kali:~/A# openssl rsa -in src_rsaA.pem -passin pass:srcpasswd -out src_rsaA.pem
writing RSA key
root@kali:~/A# ls
src_rsaA.pem src_rsaA.pem
```

```
root@kali: ~/B
root@kali:~# mkdir B
root@kali:~# cd B
root@kali:~/B# openssl genrsa -out src_rsaB.pem -passout pass:srcpasswd -des 512
Generating RSA private key, 512 bit long modulus (2 primes)
.....+++++
e is 65537 (0x010001)
root@kali:~/B# openssl rsa -in src_rsaB.pem -passin pass:srcpasswd -out src_rsaB.pem
writing RSA key
root@kali:~/B# ls
src_rsaB.pem src_rsaB.pem
```

Au debut on doit créer les clés publique et privée de chaque utilisateur par les commandes suivantes :

Pour la clé privée, on utilise la commande :

Openssl genrsa -out src_rsa.pem -passout pass :srcpasswd -des 512

Puis on extraira la clé publique de la clé privée avec la commande :

Openssl rsa -in src_rsa.pem -passin pass :srcpasswd -out src_rsa_pub.pem -pubout

Le resultat de ce passage est l'obtention d'une cle public et privé dans le fichier A et B

L'utilisateur A dispose d'une clé symétrique qui souhaite la partager avec l'utilisateur B donc on crée cette clé symétrique (Kab) dans le fichier [secret.text](#).

A la fin de la première étape, l'utilisateur B doit disposer de ses propres clés (privée et publique).de même pour l'utilisateur A avec la clé symétrique Kab (secret.text) comme un document en plus.

Comme les clés publiques sont connues par tout le monde, l'utilisateur A et B doivent disposer des clés publiques des autres utilisateurs pour permettre l'échange des données.

```
root@kali: ~/A
Fichier Édition Affichage Rechercher Terminal Aide
root@kali:~/A# openssl genrsa -out src_rsaA.pem -passout pass:srcpasswd -des 512
Generating RSA private key, 512 bit long modulus (2 primes)
.....+++++
e is 65537 (0x010001)
root@kali:~/A# openssl rsa -in src_rsaA.pem -out src_rsa_pubA.pem -pubout
Enter pass phrase for src_rsaA.pem:
writing RSA key
root@kali:~/A# cat > secret.txt
c est une cle symetrique
^C
root@kali:~/A# ls
secret.txt src_rsaA.pem src_rsa_pubA.pem src_rsa_pubB.pem
```

```
root@kali: ~/B
Fichier Édition Affichage Rechercher Terminal Aide
root@kali:~/B# openssl genrsa -out src_rsaB.pem -passout pass:srcpasswd -des 512
Generating RSA private key, 512 bit long modulus (2 primes)
.....+++++
e is 65537 (0x010001)
root@kali:~/B# ls
src_rsaB.pem
root@kali:~/B# openssl rsa -in src_rsaB.pem -out src_rsa_pubB.pem -pubout
Enter pass phrase for src_rsaB.pem:
writing RSA key
root@kali:~/B# ls
src_rsaB.pem src_rsa_pubA.pem src_rsa_pubB.pem
```

La 2ème étape consiste à envoyer la clé symétrique Kab chiffrée avec la clé publique de B et le hash de cette dernière signé par la clé privé de A.

Remarque : d'après les commandes données dans la fiche TD, on a signé le chiffrement de la clé symétrique (hashage + chiffrement avec clé privé de {Kab}Pkb) au lieu de signer la clé symétrique.

Les commandes utilisées dans cette étape sont :

- Chiffrement du secret (clé symétrique) avec la clé publique de B :

Openssl rsautl -in secret.txt -out secret.crypt -inkey src_rsa_pubB.pem -pubin -encrypt

- Calculer le condensat avec le MD5 et le signer avec la clé privée de A :

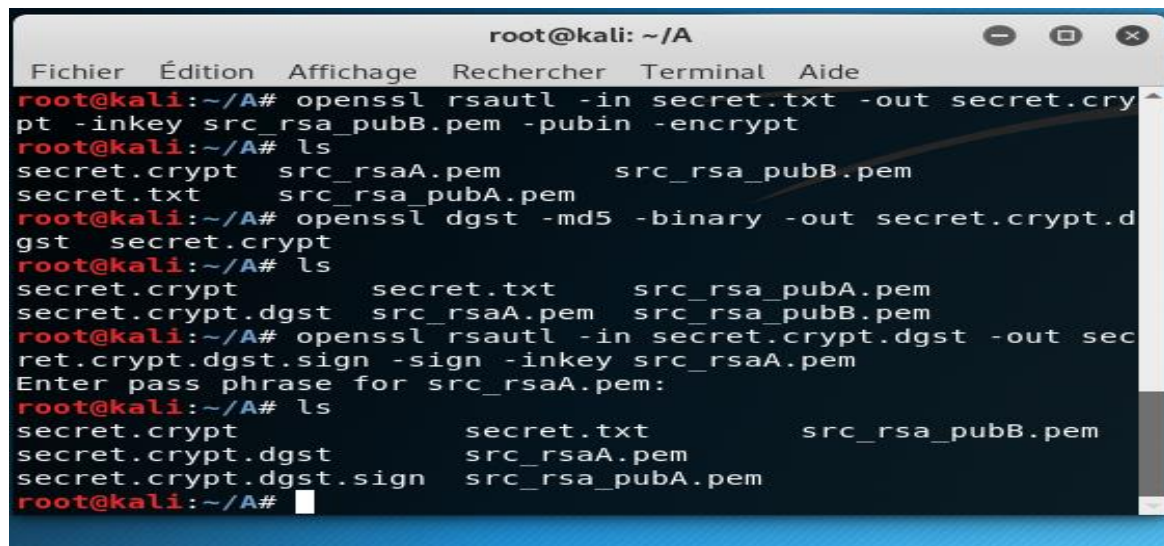
Openssl dgst -md5 -binary -out secret.crypt.dgst secret.crypt

Openssl rsautl -in secret.crypt.dgst -out secret.crypt.dgst.sign -sign -inkey src_rsaA.pem

- Envoyer les deux résultats à B par la commande (cette commande a été exécutée dans une autre ligne de commande):

Cp /root/A/secret.crypt.dgst.sign /root/B/secret.crypt.dgst.sign

Cp /root/A/secret.crypt /root/B/secret.crypt



```
root@kali: ~/A
Fichier  Édition  Affichage  Rechercher  Terminal  Aide
root@kali:~/A# openssl rsautl -in secret.txt -out secret.crypt -inkey src_rsa_pubB.pem -pubin -encrypt
root@kali:~/A# ls
secret.crypt  src_rsaA.pem  src_rsa_pubB.pem
secret.txt    src_rsa_pubA.pem
root@kali:~/A# openssl dgst -md5 -binary -out secret.crypt.dgst secret.crypt
root@kali:~/A# ls
secret.crypt  secret.txt  src_rsa_pubA.pem
secret.crypt.dgst  src_rsaA.pem  src_rsa_pubB.pem
root@kali:~/A# openssl rsautl -in secret.crypt.dgst.sign -out secret.crypt.dgst -inkey src_rsaA.pem
Enter pass phrase for src_rsaA.pem:
root@kali:~/A# ls
secret.crypt  secret.txt  src_rsa_pubB.pem
secret.crypt.dgst  src_rsaA.pem
secret.crypt.dgst.sign  src_rsa_pubA.pem
root@kali:~/A#
```

A la reception du secret chiffré (secret.crypt) et la signature (secret.crypt.dgst.sign), l'utilisateur B déchiffre la signature et hash le secret chiffré (secret.crypt) pour vérifier la signature de A.

Les commandes utilisées dans cette étape :

-déchiffrement de la signature avec la clé publique de A :

openssl rsautl -in secret.crypt.dgst.sign -out dgst1 -pubin -inkey src_rsa_pubA.pem

- hachage du secret chiffré (secret.crypt)

openssl dgst -md5 -binary -out dgst2 secret.crypt

-vérification de la différence entre les deux hash dgst1 et dgst2 par la commande

diff dgst1 dgst2

```

root@kali:~/B# ls
src_rsaB.pem src_rsa_pubA.pem src_rsa_pubB.pem
root@kali:~/B# ls
secret.crypt.dgst.sign src_rsaB.pem src_rsa_pubA.pem src_rsa_pubB.pem
root@kali:~/B# openssl rsautl -in secret.crypt.dgst.sign -out dgst1 -pubin -inkey src_rsa_pubA.pem
root@kali:~/B# ls
dgst1 secret.crypt.dgst.sign src_rsa_pubA.pem
secret.crypt src_rsaB.pem src_rsa_pubA.pem src_rsa_pubB.pem
root@kali:~/B# openssl dgst -md5 -binary -out dgst2 secret.crypt
root@kali:~/B# ls
dgst1 secret.crypt src_rsaB.pem src_rsa_pubB.pem
dgst2 secret.crypt.dgst.sign src_rsa_pubA.pem
root@kali:~/B# diff dgst1 dgst2
root@kali:~/B#

```

Après la récupération de la clé symétrique avec la clé privée de B on utilisant la commande :

openssl rsautl -decrypt -in secret.crypt -out secret.txt -inkey src_rsaB.pem

L'utilisateur B peut l'utiliser pour envoyer le message à A

```

root@kali:~/B# ls
dgst1 secret.crypt src_rsaB.pem src_rsa_pubB.pem
dgst2 secret.crypt.dgst.sign src_rsa_pubA.pem
root@kali:~/B# openssl rsautl -decrypt -in secret.crypt -out secret.txt -inkey src_rsaB.pem
Enter pass phrase for src_rsaB.pem:
root@kali:~/B# ls
dgst1 secret.crypt secret.txt src_rsa_pubA.pem
dgst2 secret.crypt.dgst.sign src_rsaB.pem src_rsa_pubB.pem
root@kali:~/B#

```

Les étapes suivantes sont :

- Création et remplissage du message **cat > message.txt**

- Hachage du message avec la commande :

openssl dgst -md5 -binary -out message.crypt.dgst message.txt

- Cryptage du message haché avec la clé privée de B

openssl rsautl -in message.crypt.dgst -out message.crypt.dgst.sign -sign -inkey src_rsaB.pem

- Cryptage du message avec la clé symétrique (secret.txt)

openssl enc -des-cbc -in message.txt -out message.crypt -pass file :secret.txt

```

root@kali:~/B# cat > message.txt
message vide
^C
root@kali:~/B# openssl dgst -md5 -binary -out message.crypt.dgst message.txt
root@kali:~/B# openssl rsautl -in message.crypt.dgst -out message.crypt.dgst.sign -sign -inkey src_rsaB.pem
Enter pass phrase for src_rsaB.pem:
root@kali:~/B# openssl enc -des-cbc -in message.txt -out message.crypt -pass file :secret.txt
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
root@kali:~/B#

```


L'utilisateur B envoie le message crypté avec la clé symétrique (secret.txt) et la signature de son message par sa clé privée. Après la récupération, l'utilisateur A déchiffre le message avec la clé symétrique (secret.txt) pour voir le message (message.txt)

```

root@kali: ~/A
Fichier  Édition  Affichage  Rechercher  Terminal  Aide
secret.crypt.dgst      src_rsaA.pem
secret.crypt.dgst.sign src_rsa_pubA.pem
root@kali:~/A# ls
message.crypt
message.crypt.dgst.sign secret.crypt.dgst.sign src_rsa_pubA.pem
secret.crypt            secret.txt             src_rsa_pubB.pem
root@kali:~/A# openssl enc -in message.crypt -out message.txt -pass file:secret.txt -d -des-cbc
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
root@kali:~/A# ls
message.crypt
message.crypt.dgst.sign secret.crypt.dgst.sign src_rsa_pubA.pem
message.txt             secret.crypt.dgst.sign src_rsa_pubB.pem
secret.crypt            secret.txt             src_rsaA.pem
root@kali:~/A# cat message.txt
message vide
root@kali:~/A#

```

On remarque que c'est le même message envoyé par l'utilisateur B dans la partie **cat > message.txt** de B (photo précédente).

Pour que l'utilisateur A puisse vérifier la signature de B, il peut exécuter les commandes dans la figure suivante :

```

root@kali:~/A# openssl rsautl -in message.crypt.dgst.sign -out dgstvir -pubin -inkey src_rsa_pubB.pem
root@kali:~/A# ls
dgstvir          messagevir.crypt.dgst secret.txt
message.crypt    secret.crypt          src_rsaA.pem
message.crypt.dgst.sign secret.crypt.dgst      src_rsa_pubA.pem
message.txt      secret.crypt.dgst.sign src_rsa_pubB.pem
root@kali:~/A# openssl dgst -md5 -binary -out messagevir.crypt.dgst.txt
root@kali:~/A# ls
dgstvir          messagevir.crypt.dgst secret.txt
message.crypt    secret.crypt          src_rsaA.pem
message.crypt.dgst.sign secret.crypt.dgst      src_rsa_pubA.pem
message.txt      secret.crypt.dgst.sign src_rsa_pubB.pem
root@kali:~/A# diff dgstvir messagevir.crypt.dgst.txt
root@kali:~/A#
root@kali:~/A#

```