Name: Eduard Antonovic Occhipinti

UGA - Master 2 MOSIG Information Security
Practical work with Open SSL

Goal: become familiar with OpenSSL, its use and its modification. OpenSSL supports a large number of ciphers: encryption methods, secret and public keys, hash functions, ...

1 Preamble

Practical work to be done alone or in pairs, preferably on the machine pcserveur.ensimag.fr.

Otherwise, check that OpenSSL is installed: `$ openssl version -a`
Otherwise, download from https://github.com/openssl/openssl.git, compile and install (**locally**):
`$ ./config ; make ; make install`
For help: `$ man openssl`
Then upload the result files to pcserveur.ensimag.fr.

2 RSA key generation

To get the list of command arguments, add `-h`. For instance:
`$ man openssl genrsa` or `$ man openssl-genrsa`

of course you can use different options for the passphrase

Question 1) Which command generates a private 2048-bit RSA key ?
Answer: openssl genrsa

Question 2) Generate a private 2048-bit RSA key, and store it encrypted with AES 256 bits (with your passphrase as secret key); your key must be stored in the privkey.pem file. What command are you using ?
Answer: openssl genrsa -out privkey.pem -aes256 -passout pass:<passphrase>

Question 3) Use the command openssl rsa to display the modulo public : what is the number of bits of this modulo, what is the associated public exponent?
Answer: The number of bits of the modulo is 2048, while the associated public exponent is 17 bits long (it's 0x10001)

Question 4) Use the -text option to retrieve private key information (private exponent, prime factors). What is the number of bits of each of these two factors ? Why are these factors stored in the private key?
Answer: The two prime factors are 1024 bits long while the privateExponent is 2048 bits long.
- The two prime factors are stored because RSA relies on the difficulty of factoring the product of these two primes
- The privateExponent is stored to decrypt messages encoded with the public key

Question 5) Generate the associated public key in a file named pubkey-rsa-<LOGIN name>.pem replacing <LOGIN name> with your login name:

```
openssl rsa -in privkey.pem -pubout -out pubkey-rsa-‘whoami‘.pem
```

and copy this public key in the directory /tmp/4MMCRY-signargh/ on machine pcserver:

```
scp pubkey-rsa-‘whoami‘.pem pcserveur.ensimag.fr:/tmp/4MMCRY-signargh/
```

or else on a local directory named 4MMCRY-signargh-Alice/ that is assumed owned by another user, let say Alice:

```
mkdir 4MMCRY-signargh-Alice; cp pubkey-rsa-‘whoami‘.pem 4MMCRY-signargh-Alice/
```

first we need to generate a message file, then we can encrypt it and save it (after inserting our passphrase)

When this is done in pcserveur.ensimag.fr:/tmp/4MMCRY-signargh/ , we can send you an encrypted message with your public key. Or else you will encrypt alone a message with your private key.

```
echo "hello :)" > message.txt && openssl pkeyutl -sign -inkey privkey.pem -in message.txt -out encrypted.bin
```

3 Encrypted file exchange

If you can read the directory /tmp/4MMCRY-signargh/ : retrieve from this directory the file containing our encrypted message: message_cipher.bin. This file is encrypted by AES 256 in CBC PBKDF2 mode whose symmetric key is stored encrypted with your previous public key in the file: sessionkey-<LOGIN name>.bin

Else, in the directory 4MMCRY-signargh-Alice/

Name: Eduard Antonovic Ocahipinti

1. Alice writes `message-plain.txt` and want to secretly send it to you (ie Bob) :
   `cat "Hi Bob, this is my secret message..." > 4MMCRY-signargh-Alice/message-plain.txt`

2. Alice chooses a secret session key to communicate with Bob and put it in a file `sessionkey.txt`
   `cat "ThisIsMyUltraSecretSessionKey4com-with-Bob" > 4MMCRY-signargh-Alice/sessionkey.txt`

3. With which command Alice encrypts `message-plain.txt` with AES-256 CTR mode and symmetric key `sessionkey.txt` to generate `message-cypher.bin`?
   Answer: openssl enc -aes-256-ctr -pbkdf2 -in message-plain.txt \
   -out message-cypher.bin -pass file:./sessionKey.txt

4. Alice encrypts `sessionkey.txt` with your public RSA key `pubkey-rsa-‘whoami‘.pem` to generate `sessionkey-<LOGIN name>.bin`. Which command does Alice enter for this?
   Answer: openssl pKeyutl -encrypt -inKey pubKey-rsa-eduard.pem \
   -pubin -in sessionKey.txt -out sessionKey-eduard.bin

Question 6) From the two files `sessionkey-<LOGIN name>.bin` and `message-cypher.bin`, decrypt this message in the file `clear-<Name of LOGIN>`: give the commands to calculate this decryption.
Answer: openssl pKeyutl -decrypt -inKey privKey.pem -in sessionKey-eduard.bin \
-out sessionKey-decrypted.txt && openssl enc -d -aes-256-ctr -pbkdf2 \
-in message-cypher.bin -out clear-eduard -pass file:./session-decrypted.txt

4 Encrypted file exchange

↖ first we decrypt the session key, then the message

Question 7) Add a line with only your login name at the end of this file with the command:
`echo ‘whoami‘ >> clear-‘whoami‘`
then sign this file with your rsa key and with the function of signature (digest) SHA-256, the signature must be called `signature-‘whoami‘.bin`. Then, if you can, copy this file to `/tmp/4MMCRY-signargh/`, :
`scp signature-‘whoami‘.bin pcserveur.ensimag.fr:/tmp/4MMCRY-signargh/`
else, if you can’t, in `4MMCRY-signargh-Alice/`:
`cp signature-‘whoami‘.bin 4MMCRY-signargh-Alice/`
Which command generates the signature.
Answer: openssl dgst -sha256 -sign privKey.pem -out signature-eduard.bin clear-eduard
- you also have to insert the passphrase when prompted

When this is done, we can verify your signature with your public key.

Question 8) With which command Alice verfiies the signature ?
Answer:
openssl dgst -sha256 -verify pubKey-rsa-eduard.pem \
-signature signature-eduard.bin clear-eduard