Network Security

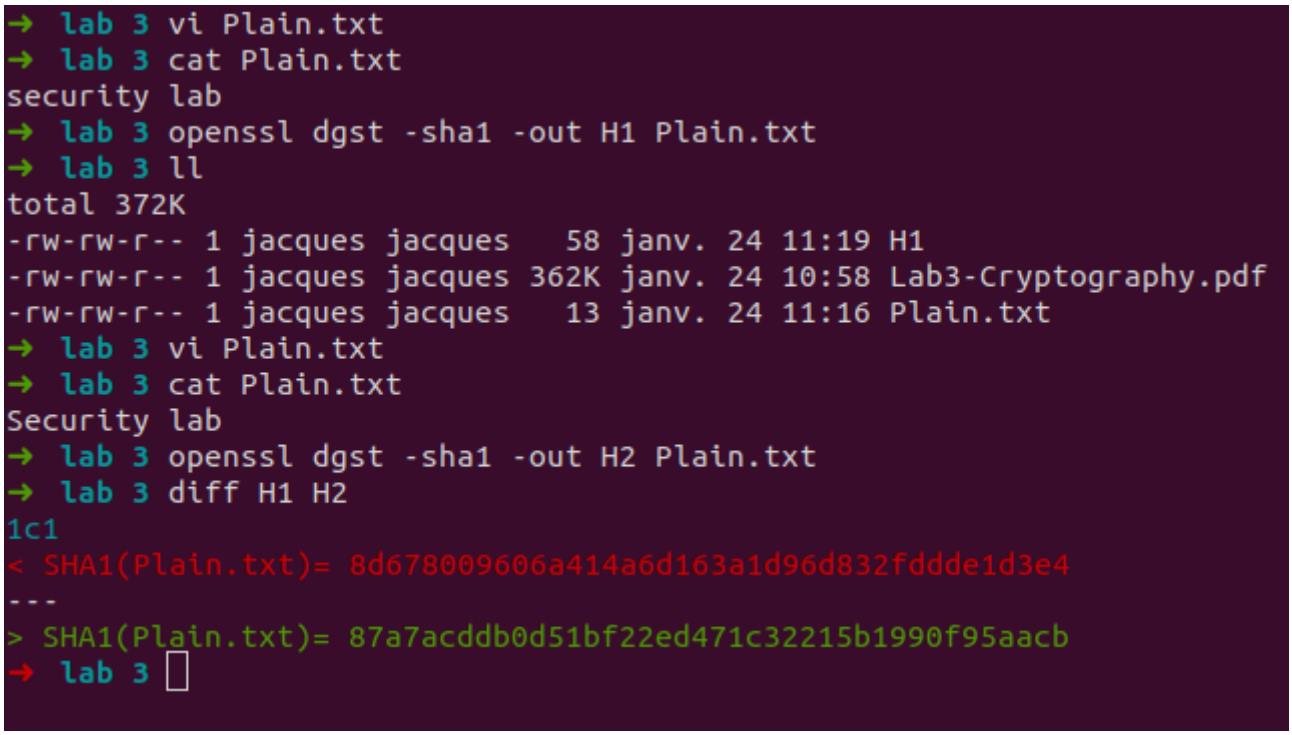
LAB3 - Cryptography

Jacques Polart
ing4 – SI – group 4

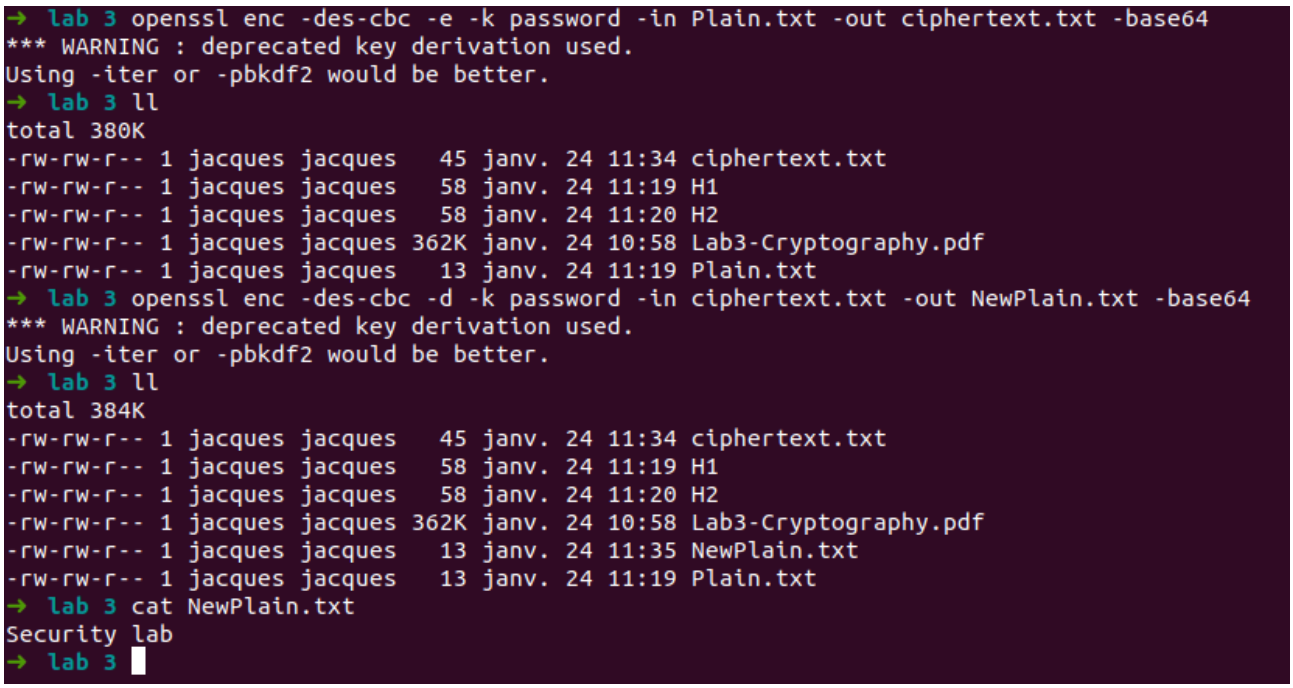
26/01/2022

Task 1: Hash function

A hash function is any function that can be used to map data of arbitrary size to fixed-size values.



Task 2: Symmetric Encryption



With symetric en cryption we use the same key to encrypt and decrept.

Task 3: Asymmetric Encryption

With assymetric encryption, we use to different key, one to encrypt and the other to decrypt. One will be private will the other will be public. Generaly the private key is the one that decrypt while the public key is used to encrypt.

```
→ lab 3 openssl genrsa -out privJacques.key 2048
Generating RSA private key, 2048 bit long modulus (2 primes)
......+++++
..........+++++
e is 65537 (0x010001)
→ lab 3 ll
total 372K
-rw-rw-r-- 1 jacques jacques 362K janv. 24 10:58 Lab3-Cryptography.pdf
-rw-rw-r-- 1 jacques jacques   13 janv. 24 11:19 Plain.txt
-rw------- 1 jacques jacques 1,7K janv. 24 11:38 privJacques.key
→ lab 3 openssl rsa -in privJacques.key -pubout -out pubJacques.key
writing RSA key
→ lab 3 ll
total 376K
-rw-rw-r-- 1 jacques jacques 362K janv. 24 10:58 Lab3-Cryptography.pdf
-rw-rw-r-- 1 jacques jacques   13 janv. 24 11:19 Plain.txt
-rw------- 1 jacques jacques 1,7K janv. 24 11:38 privJacques.key
-rw-rw-r-- 1 jacques jacques  451 janv. 24 11:40 pubJacques.key
→ lab 3 openssl rsautl -encrypt -in Plain.txt -pubin -inkey pubJacques.key -out cipherRsa.txt
→ lab 3 cat cipherRsa.txt
Q◆f◆9◆H ◆9=◆◆◆◆◆TG#
◆s◆M◆◆◆jp~◆◆◆>◆as◆◆,◆Y-◆t0◆◆◆k◆,#f◆Ч◆◆◆F◆@◆V&◆9 ◆◆Qa(◆i◆◆◆þ◆◆?◆◆@◆◆Ž◆b◆PV◆a◆◆/◆◆+◆a◆◆◆◆Y◆l]|_◆"◆◆Y◆2◆x◆◆◆
I◆õ◆◆◆◆◆F◆◆◆,◆◆◆<◆◆◆◆UU]◆◆◆T:;◆◆1J
                ◆◆P+◆&MgC◆?◆-$◆/!◆7◆v◆1W◆◆Dn◆◆.Gvz◆◆c◆C%
→ lab 3 openssl rsautl -decrypt -in cipherRsa.txt -inkey privJacques.key -out NewPlain.txt
→ lab 3 diff Plain.txt NewPlain.txt
→ lab 3
```

The diff command return nothing becfause both Plain.txt and NewPlain.txt are identical.

Task 4: Asymmetric/Symmetric Encryption, Digital signature

a. Explain the above scenario

-First Alice encrypt a message with an symetric key, and send it to Bob.
-Alice now need to send the symetric key to Bob.
-Alice and Bob both generate a pair of assymetric keys (KprivA / KpubA, KprivB / KpubB) and share with each other there public key (KpubA and KpubB)
-Then Alice send to Bob : the symetric key encrypted with Bob public key (the data), and the symetric key hashed and signed with Alice private key (the signature).
-Bob can decrypt the the data with his private key. He obtain the symetric key.
-To verify the data, Bob can verify the signature with Alice public key. He obtain a hash of the symetric key. He generate a second hash from the symetric key he has obtained and compare both hash (they should be identical)
-now that Bob has ensure data integrity and non-repudiation, he can use the symetric key to decrypt the message Alice sent him.

b. Generate the symmetric key sym.key with length of 128 bits

```
openssl rand -out sym.key -hex 16
```

a. Create Plaintext.txt

```
vi Plain.txt
```

b. Generate RSA keys with length of 2048 bits

```
➜ Alice openssl genrsa -out privA.key
Generating RSA private key, 2048 bit long modulus (2 primes)
...+++++
.....................................................+++++
e is 65537 (0x010001)
➜ Alice openssl rsa -in privA.key -pubout -out pubA.key
writing RSA key
➜ Alice cd ../Bob
➜ Bob openssl genrsa -out privB.key
Generating RSA private key, 2048 bit long modulus (2 primes)
.............................+++++
.......+++++
e is 65537 (0x010001)
➜ Bob openssl rsa -in privB.key -pubout -out pubB.key
writing RSA key
➜ Bob ll
total 8,0K
-rw------- 1 jacques jacques 1,7K janv. 24 13:25 privB.key
-rw-rw-r-- 1 jacques jacques  451 janv. 24 13:25 pubB.key
➜ Bob ll ../Alice
total 16K
-rw-rw-r-- 1 jacques jacques   43 janv. 24 13:23 Plain.txt
-rw------- 1 jacques jacques 1,7K janv. 24 13:24 privA.key
-rw-rw-r-- 1 jacques jacques  451 janv. 24 13:24 pubA.key
-rw-rw-r-- 1 jacques jacques   33 janv. 24 13:23 sym.key
➜ Bob
```

c. Exchange the public keys (use sftp or USB Flash)

```
➜ Bob ll
total 12K
-rw------- 1 jacques jacques 1,7K janv. 24 13:25 privB.key
-rw-rw-r-- 1 jacques jacques  451 janv. 24 13:24 pubA.key
-rw-rw-r-- 1 jacques jacques  451 janv. 24 13:25 pubB.key
➜ Bob ll ../Alice
total 20K
-rw-rw-r-- 1 jacques jacques   43 janv. 24 13:23 Plain.txt
-rw------- 1 jacques jacques 1,7K janv. 24 13:24 privA.key
-rw-rw-r-- 1 jacques jacques  451 janv. 24 13:24 pubA.key
-rw-rw-r-- 1 jacques jacques  451 janv. 24 13:25 pubB.key
-rw-rw-r-- 1 jacques jacques   33 janv. 24 13:23 sym.key
➜ Bob
```

Bob has Alice public key (pubA,key)
and Alice has Bob public key (pubB,key)

d. Encrypt PlaintextM.txt using symmetric algorithm

```
➜ Alice openssl enc -aes-128-cbc -e -in Plain.txt -out cipher.txt -kfile sym.key
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
➜ Alice ll
total 24K
-rw-rw-r-- 1 jacques jacques   64 janv. 24 13:28 cipher.txt
-rw-rw-r-- 1 jacques jacques   43 janv. 24 13:23 Plain.txt
-rw------- 1 jacques jacques 1,7K janv. 24 13:24 privA.key
-rw-rw-r-- 1 jacques jacques  451 janv. 24 13:24 pubA.key
-rw-rw-r-- 1 jacques jacques  451 janv. 24 13:25 pubB.key
-rw-rw-r-- 1 jacques jacques   33 janv. 24 13:23 sym.key
➜ Alice
```

The encrypted text in stored in the file cipher.txt

e. Encrypt the symmetric key using asymmetric algorithm

```
➜ Alice openssl rsautl -encrypt -pubin -inkey pubB.key -in sym.key -out secret.key
➜ Alice ll
total 28K
-rw-rw-r-- 1 jacques jacques   64 janv. 24 13:28 cipher.txt
-rw-rw-r-- 1 jacques jacques   43 janv. 24 13:23 Plain.txt
-rw------- 1 jacques jacques 1,7K janv. 24 13:24 privA.key
-rw-rw-r-- 1 jacques jacques  451 janv. 24 13:24 pubA.key
-rw-rw-r-- 1 jacques jacques  451 janv. 24 13:25 pubB.key
-rw-rw-r-- 1 jacques jacques  256 janv. 26 11:18 secret.key
-rw-rw-r-- 1 jacques jacques   33 janv. 24 13:23 sym.key
➜ Alice
```

Alice use bob public key (pubB,key) to encrypt the symetric key (sym,key) and store the crypted key in the file secret,key

f. Generate the hash value of symmetric key using SHA1

```
➜ Alice openssl dgst -sha1 -out sym.sha1 sym.key
➜ Alice ll
total 32K
-rw-rw-r-- 1 jacques jacques   64 janv. 24 13:28 cipher.txt
-rw-rw-r-- 1 jacques jacques   43 janv. 24 13:23 Plain.txt
-rw------- 1 jacques jacques 1,7K janv. 24 13:24 privA.key
-rw-rw-r-- 1 jacques jacques  451 janv. 24 13:24 pubA.key
-rw-rw-r-- 1 jacques jacques  451 janv. 24 13:25 pubB.key
-rw-rw-r-- 1 jacques jacques  256 janv. 26 11:18 secret.key
-rw-rw-r-- 1 jacques jacques   33 janv. 24 13:23 sym.key
-rw-rw-r-- 1 jacques jacques   56 janv. 26 11:19 sym.sha1
➜ Alice
```

The hash of the key is store in the file sym.sha1. It will be used by Bob to check that the data integrity.

g. Sign sym.sha1

```
➜  Alice openssl rsautl -sign -inkey privA.key -in sym.sha1 -out sym.sig
➜  Alice ll
total 36K
-rw-rw-r-- 1 jacques jacques   64 janv. 24 13:28 cipher.txt
-rw-rw-r-- 1 jacques jacques   43 janv. 24 13:23 Plain.txt
-rw------- 1 jacques jacques 1,7K janv. 24 13:24 privA.key
-rw-rw-r-- 1 jacques jacques  451 janv. 24 13:24 pubA.key
-rw-rw-r-- 1 jacques jacques  451 janv. 24 13:25 pubB.key
-rw-rw-r-- 1 jacques jacques  256 janv. 26 11:18 secret.key
-rw-rw-r-- 1 jacques jacques   33 janv. 24 13:23 sym.key
-rw-rw-r-- 1 jacques jacques   56 janv. 26 11:19 sym.sha1
-rw-rw-r-- 1 jacques jacques  256 janv. 26 11:22 sym.sig
➜  Alice
```

Alice sign the hash of the key (sym.sha1) with her private key (privA,key) and generate a signature (sym.sig)
It will be used by Bob to ensure non-repudiation.

h. Send the necessary files to your colleague that allows decrypting your message, and verify your signature.

Bob need the following files :
- the encrypted message (cipher.txt)
- the encrypted symetric key (secret.key)
- the signature (sym.sig)
- alice public key (pubA.key)

```
➜  Bob ll
total 24K
-rw-rw-r-- 1 jacques jacques   64 janv. 24 13:28 cipher.txt
-rw------- 1 jacques jacques 1,7K janv. 24 13:25 privB.key
-rw-rw-r-- 1 jacques jacques  451 janv. 24 13:24 pubA.key
-rw-rw-r-- 1 jacques jacques  451 janv. 24 13:25 pubB.key
-rw-rw-r-- 1 jacques jacques  256 janv. 26 11:18 secret.key
-rw-rw-r-- 1 jacques jacques  256 janv. 26 11:22 sym.sig
➜  Bob
```

let decrypt the message :

```
→  Bob openssl rsautl -decrypt -inkey privB.key -in secret.key -out sym.key
→  Bob openssl dgst -sha1 -out sym.sha1 sym.key
→  Bob openssl rsautl -verify -pubin -inkey pubA.key -in sym.sig -out sym2.sha1
→  Bob diff sym.sha1 sym2.sha1
→  Bob openssl enc -aes-128-cbc -kfile sym.key -in cipher.txt -d -out Plain.txt
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
→  Bob cat Plain.txt
My Security LAB: My Name is Jacques Polart
→  Bob
```

first we decrypt secret.key with Bob private key (privB.key) to get the symetric key (sym.key)

```
openssl rsautl -decrypt -inkey privB.key -in secret.key -out sym.key
```

then we need to check the signature :
- We hash the symetric key (sym.key) to get sym.sha1

```
openssl dgst -sha1 -out sym.sha1 sym.key
```

- We verify the signature (sym.sig) wich gives us another hash of the symetric key (sym2.sha1)

```
openssl rsautl -verify -pubin -inkey pubA.key -in sym.sig -out sym2.sha1
```

- we compare both hash to check for any differences.

```
diff sym.sha1 sym2.sha1
```

- the diff command dont print anything so both hash are identical.

We just ensure the data integrity and non-repudiation.

Now lets just decrypt the message sent by Alice (cipher.txt) with the symetric key (sym.key), and output the plain text (Plain.txt)

```
openssl enc -aes-128-cbc -kfile sym.key -in cipher.txt -d -out Plain.txt
```

if we check the content of Plain.txt, we see that the message has been successfully decrypted !