Sécurité des systèmes d'information

LAB 1
Introduction to Cybersecurity

Jacques Polart
Thomas Grisez
ing4 – SI – groupe 4

20/01/2022

1. We want to do for Bob the same steps we did for Alice

```
→ Bob openssl genrsa -out BobKeyPair
Generating RSA private key, 2048 bit long modulus (2 primes)
.....................................+++++
.............................+++++
e is 65537 (0x010001)
→ Bob openssl rsa -in BobKeyPair -pubout -out BobPublicKey
writing RSA key
→ Bob mv BobKeyPair BobPrivateKey
→ Bob ls
BobPrivateKey  BobPublicKey
→ Bob
```

2. Alice wants to encrypt AliceDocument thanks to Bob's public key

• Go back to the parent folder LAB1 and access Alice's folder.
• Check that BobPublicKey has been copied correctly to Alice's folder

```
→ Bob cd ..
→ tp1 cd Alice
→ Alice ls
AliceDocument  AlicePrivateKey  AlicePublicKey  BobPublicKey
→ Alice
```

• Check that AliceDocumentEncrypted has been created correctly

```
→ Alice openssl rsautl -encrypt -in AliceDocument -pubin -inkey BobPub
licKey -out AliceDocumentEncrypted
→ Alice ls
AliceDocument           AlicePrivateKey  BobPublicKey
AliceDocumentEncrypted  AlicePublicKey
→ Alice
```

• Check the content of the file AliceDocumentEncrypted. What do you notice?

```
<9f>¶
^0TííQYQúì^F^\J<93>     <9a><89><99>lR¿W»Ð6ÿ2»V:å~µx) Ü<82>ø<84>/<86>K<
8c>Þ^_^^º]<91>^Y.^O^H^Yçx/g#$àvÙx8^Ga>ã»]Í<95>¸^WG?VÃb?æçqjïyé^UB¹È¤c/é
¿9,A4&<81>^P<8d>f>&ñ<85>í^^^EqßÎ<ÿTæÓw0R^@<97><90>^[å<9d>d:Jj#ò<84>Õ^Uu
·<82>ðÞÌ<98>Ú ZâCgs^?<8e>i²D<97>Æoj^]d<92>@¸ÍY3dÿZ&´è=^VVbgs·:^]^U^\C<8
1>áòx^H<92>°^PÔç¥÷F¦^C<89>i]<85>Vù^M1Gdý^^ S<80>ÙÛW¼Ìq)¸^N=ê8BE^\e\Ù%³ç
^Nñ¸bòã<9d>_Fÿ©iÞ
```

The document is unreadable, and the encrypted text is biger than the clear text.

3. The objective in this step is to decrypt AliceDocumentEncrypted thanks to BobPrivateKey

• Make sure you are in Alice's folder. We ask you to copy AliceDocumentEncrypted to Bob's folder.
• Go back to the parent folder LAB1 and access Bob's folder → (0.25 pt).
• Check that AliceDocumentEncrypted has been copied correctly to Bob's folder

```
→ Alice cp AliceDocumentEncrypted ../Bob
→ Alice cd ..
→ tp1 cd Bob
→ Bob ls
AliceDocumentEncrypted  BobPrivateKey  BobPublicKey
→ Bob
```

• You will now proceed to decrypt AliceDocumentEncrypted thanks to BobPrivateKey by naming the decrypted document AliceDocumentDecrypted.
• Check that AliceDocumentDecrypted has been created correctly

```
→ Bob openssl rsautl -decrypt -in AliceDocumentEncrypted -inkey BobPri
vateKey -out AliceDocumentDecrypted
→ Bob ls
AliceDocumentDecrypted  BobPrivateKey
AliceDocumentEncrypted  BobPublicKey
→ Bob
```

• Check the content of the file AliceDocumentDecrypted. What do you notice?

```
Hello Bob, i'm Alice
```

The text has been successfully decrypted by bob private key

4. The objective of this step is to show you that asymmetric encryption cannot be applied to large files

• Go back to the parent folder LAB1 and access Alice's folder

```
→ Bob cd ../Alice
→ Alice
```

• Try now to encrypt LargeFile by using BobPublicKey and by naming the encrypted file LargeFileEncrypted

```
→ Alice openssl rsautl -encrypt -in LargeFile -pubin -inkey BobPublicK
ey -out LargeFileEncrypted
RSA operation error
140048261076288:error:0406D06E:rsa routines:RSA_padding_add_PKCS1_type_
2:data too large for key size:../crypto/rsa/rsa_pk1.c:124:
→ Alice
```

5. The objective of this step is to show you how Alice can generate an electronic signature in order to authenticate herself to Bob, ensure the non-repudiation for herself and guarantee the integrity of the signed data.

• Make sure you are in Alice's folder. Create a file named AuthData and write a text of your choice.
• Check that AuthData has been created correctly.
• Copy AlicePublicKey to Bob's folder.

```
➜ Alice pwd
/home/jacques/Documents/ece/inge4/S2/securite_des_si/tp1/Alice
➜ Alice vi AuthData
➜ Alice cat AuthData
hello world
➜ Alice cp AlicePublicKey ../Bob
➜ Alice ls ../Bob
AliceDocumentDecrypted  AlicePublicKey  BobPublicKey
AliceDocumentEncrypted  BobPrivateKey
➜ Alice
```

• Check that HashAuthData has been created correctly.
• Check the content of HashAuthData.

```
➜ Alice openssl dgst -sha256 -out HashAuthData AuthData
➜ Alice ls
AliceDocument           AlicePublicKey  HashAuthData
AliceDocumentEncrypted  AuthData        LargeFile
AlicePrivateKey         BobPublicKey    LargeFileEncrypted
➜ Alice cat HashAuthData
SHA256(AuthData)= a948904f2f0f479b8f8197694b30184b0d2ed1c1cd2a1ec0fb85d
299a192a447
```

• You will now proceed to sign HashAuthData thanks to AlicePrivateKey by naming the signature AliceSignature.

```
➜ Alice openssl rsautl -sign -in HashAuthData -inkey AlicePrivateKey -
out AliceSignature
➜ Alice ls
AliceDocument           AliceSignature  LargeFile
AliceDocumentEncrypted  AuthData        LargeFileEncrypted
AlicePrivateKey         BobPublicKey
AlicePublicKey          HashAuthData
➜ Alice
```

• You will now proceed to verify AliceSignature thanks to AlicePublicKey.

Bob has now the AuthData, and AliceSignature witch is the AuthData hashed then signed with alice private key.
Bob can retrive the HashAuthData by verifying AliceSignature with alice public key.
He can also create his own hash of the AuthData, we call it HashBob.
Bob has now two version of the AuthData hash (HashAuthData and HashBob).
If nothing gone wrong, they sould be identical.
If the AuthData has been signed by another key than alice private key, or AuthData has been modifyed since alice signed it, hash sould be different.

So we have acomplish identification and non repudiation of data.

In the following exemple, no fraud were commited. Hash are identical.

```
➜  Bob rm -rf HashBob
➜  Bob vi AuthData
➜  Bob openssl dgst -sha256 -out HashBob AuthData
➜  Bob diff HashBob HashAuthData
➜  Bob
```

In the following exemple, AuthData has been modified :

```
➜  Bob vi AuthData
➜  Bob cat AuthData
hello world (edited)
➜  Bob openssl dgst -sha256 -out HashBob AuthData
➜  Bob diff HashBob HashAuthData
1c1
< SHA256(AuthData)= 6b1542de62c4bb18b36fa8f7c31a6baa386de820dcb22f495b1
1df9b73ee7334
---
> SHA256(AuthData)= a948904f2f0f479b8f8197694b30184b0d2ed1c1cd2a1ec0fb8
5d299a192a447
➜  Bob
```