LAB Cryptographie

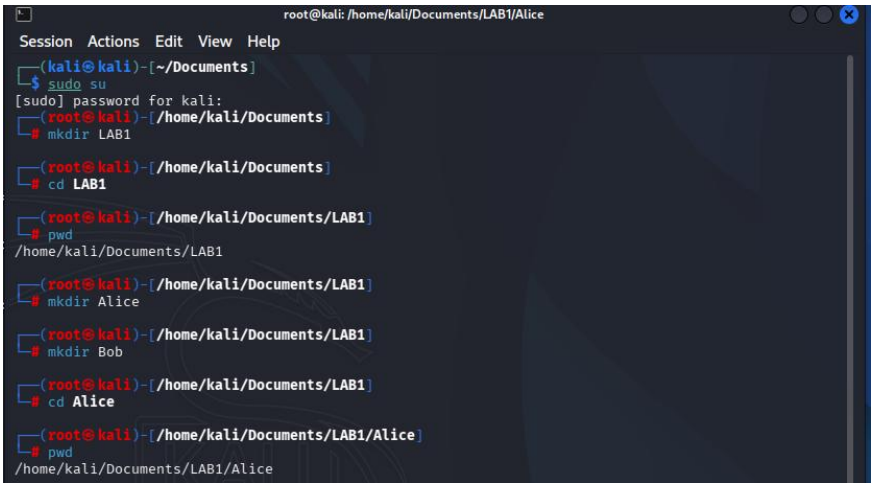
U7 CYBER

SPINELLI Dylan
BTS SIO SISR | PARIS YNOV CAMPUS

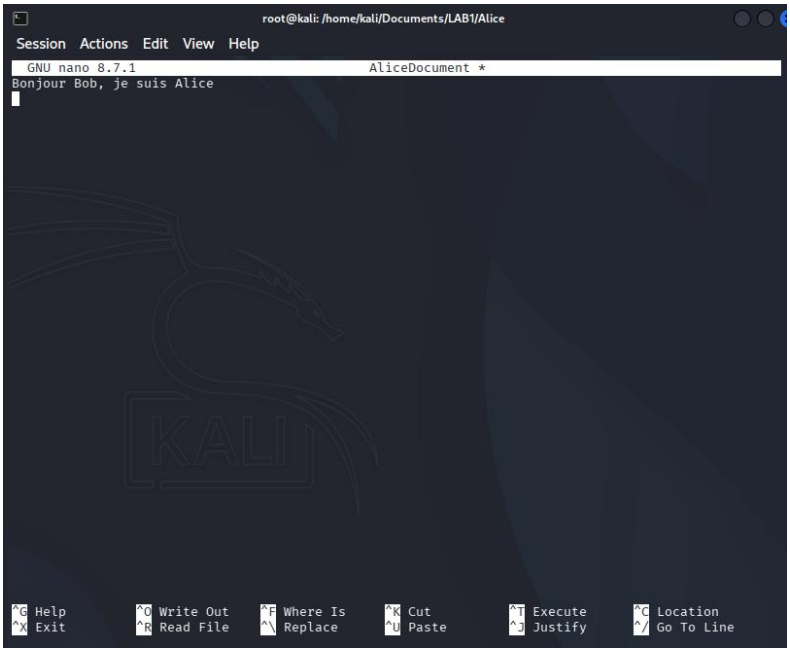
Table des matières

1-Introduction à OpenSSL

On commence par créer le dossier LAB1, puis on crée un dossier Alice dans ce dossier :



On ajoute un fichier texte AliceDocument dans le dossier Alice :



Nous allons désormais générer une paire de clés de cryptage pour Alice :

```
┌──(root㉿kali)-[/home/kali/Documents/LAB1/Alice]
└─# openssl genrsa -out AliceKeyPair

┌──(root㉿kali)-[/home/kali/Documents/LAB1/Alice]
└─# ls
AliceDocument  AliceKeyPair

┌──(root㉿kali)-[/home/kali/Documents/LAB1/Alice]
└─#
```

On sépare la clé publique en la renommant AlicePublicKey :

```
┌──(root㉿kali)-[/home/kali/Documents/LAB1/Alice]
└─#  openssl rsa -in AliceKeyPair -pubout -out AlicePublicKey
writing RSA key

┌──(root㉿kali)-[/home/kali/Documents/LAB1/Alice]
└─# ls
AliceDocument  AliceKeyPair  AlicePublicKey

┌──(root㉿kali)-[/home/kali/Documents/LAB1/Alice]
└─#
```

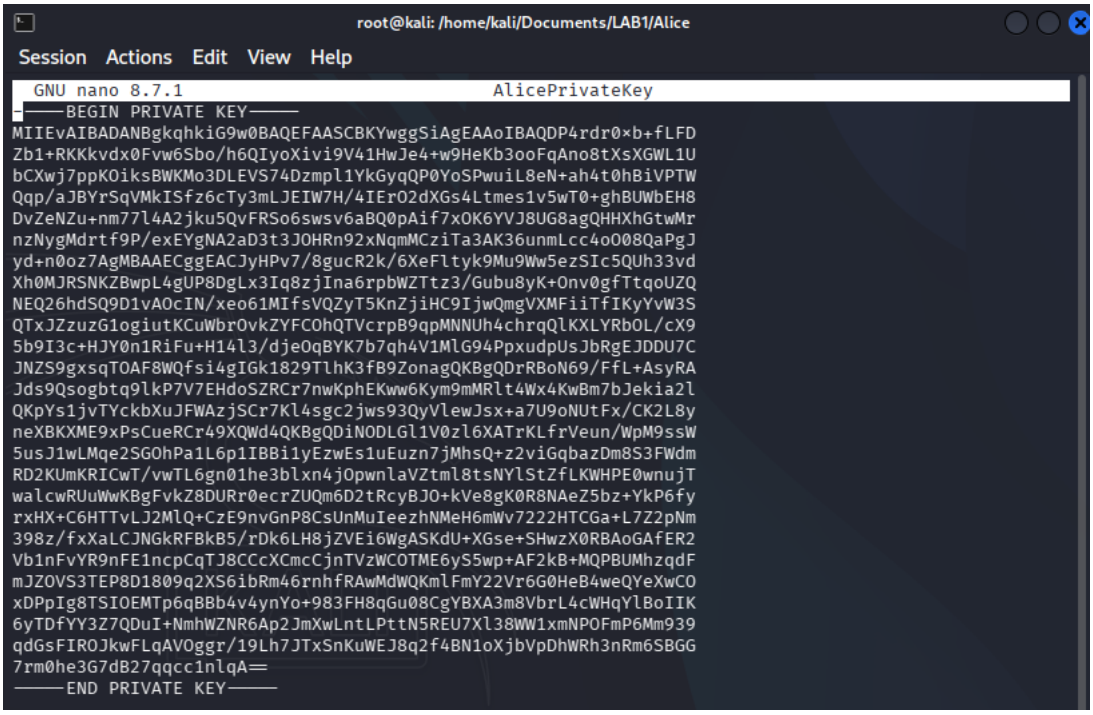
On renomme la clé privée AlicePrivateKey :

```
┌──(root㉿kali)-[/home/kali/Documents/LAB1/Alice]
└─# mv AliceKeyPair AlicePrivateKey
```

Voici la clé publique de Alice :

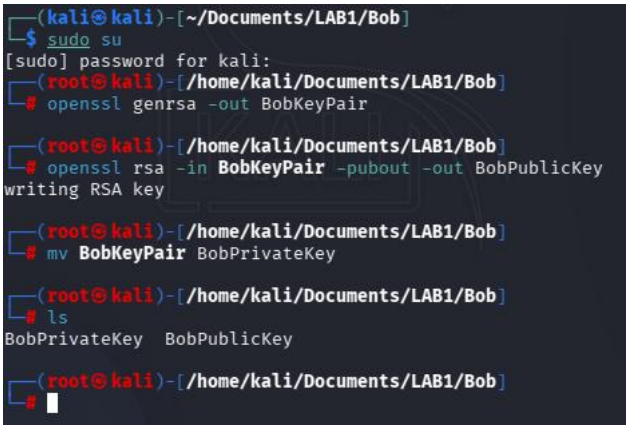
```
root@kali: /home/kali/Documents/LAB1/Alice
Session  Actions  Edit  View  Help
  GNU nano 8.7.1                          AlicePublicKey
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAz+K3a9MW/nyxQ2W9fkSi
pL3cdBb8Okm6P4ekCMqF4r4vVeNR8CXuPsPR3im96KBagJ6PLV7Fxli9VGwl8I+6
aSjopLAVijKNwyxFUu+A85qZdWJBsqkD9GKEj8Loi/HjfmoeLdIQYlT01kKqf2iQ
WK0qlTJCEn8+nE8t5iyRCFux/+CBKztnVxrOC7ZnrNb+cE9PoIQVFmxB/A72XjWb
vp5u+5eANo5LuULxUUqOrMLL+mgUNKQIn+8TiumFSfFBvGoEBx14RrcDK58zcoDH
a7X/T/3sRGIDQNmg97dyTh0Z/dsTapjAs4k2twCt+rp5i3HOKDtPEGj4Ccnfp9KM
+wIDAQAB
-----END PUBLIC KEY-----
```

Voici la clé privée de Alice :



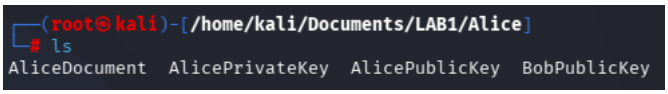
2-Cryptographie Asymétrique

Nous allons répéter les mêmes étpaes pour générer une paire de clés Publique et Privée pour Bob :

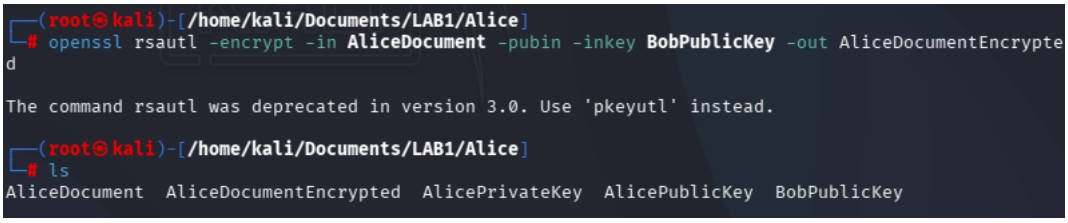


On copie la clé publique de Bob dans le dossier de Alice :

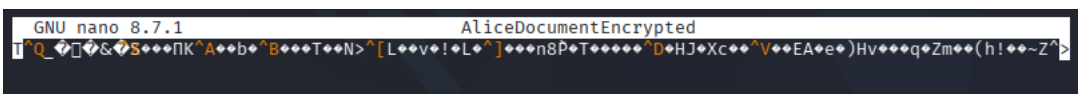
On retrouve bien la clé publique de Bob dans le dossier de Alice :



Nous allons désormais chiffrer le document AliceDocument grâce à BobPublicKey en nommant le document chiffré AliceDocumentEncrypted :

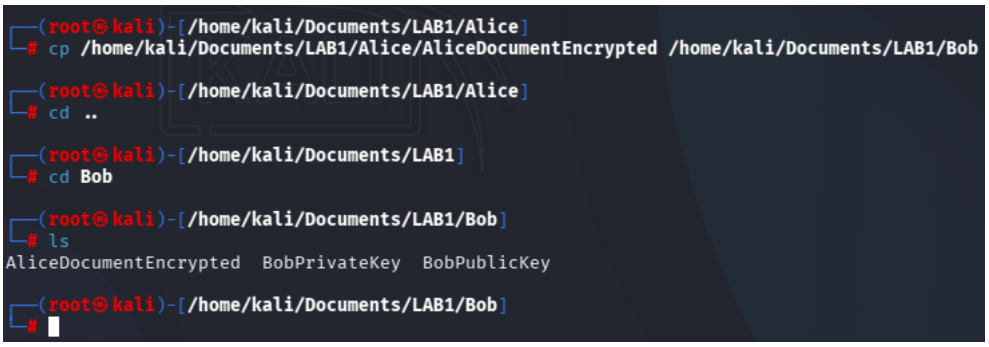


Vérifions maintenant le contenu du fichier AliceDocumentEncrypted :

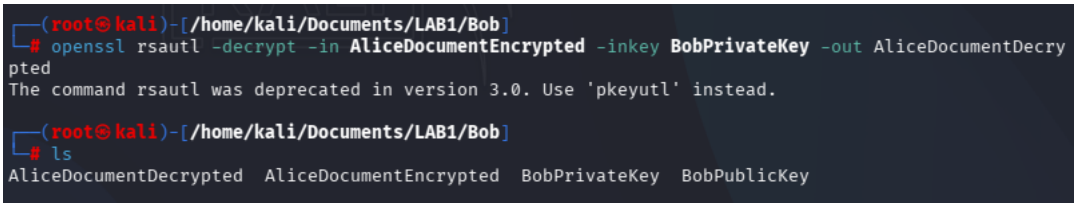


On voit que le fichier à bien été crypté, ses données sont illisibles.

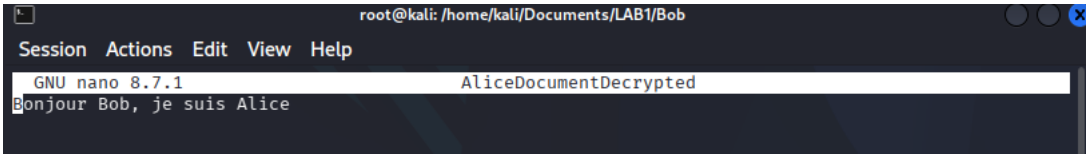
Nous allons désormais décrypter AliceDocumentEncrypted grâce à la clé Privée de Bob. Pour cela, on copie AliceDocumentEncrypted dans le dossier de Bob :



Désormais, nous allons décrypter AliceDocumentEncrypted avec BobPrivateKey :



Vérifions le contenu du fichier décrypté :



On retrouve bien le texte d'origine.

3-Chiffrement d’un fichier volumineux

Nous allons désormais chiffrer un fichier volumineux. On commence par créer le fichier LargeFile dans le dossier Alice :

```
┌──(root㉿kali)-[/home/kali/Documents/LAB1/Alice]
└─# openssl rand -out LargeFile -base64 $((2**30 * 3/4))
```

On encrypte ce fichier avec la clé publique de Bob :

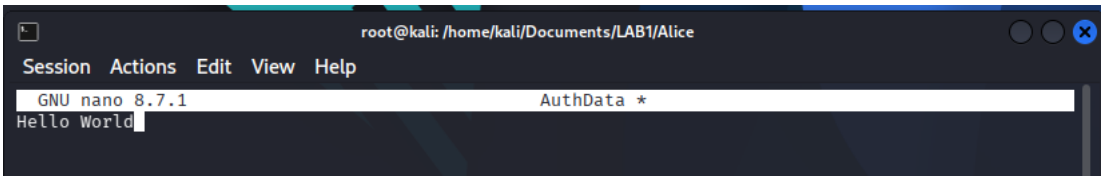
```
┌──(root㉿kali)-[/home/kali/Documents/LAB1/Alice]
└─# openssl rand -out LargeFile -base64 $((2**30 * 3/4))

┌──(root㉿kali)-[/home/kali/Documents/LAB1/Alice]
└─# openssl rsautl -encrypt -in LargeFile -inkey BobPublicKey -out LargeFileEncrypted
The command rsautl was deprecated in version 3.0. Use 'pkeyutl' instead.
Could not find private key from BobPublicKey
40478EE48C7F0000:error:1608010C:STORE routines:ossl_store_handle_load_result:unsupported:../crypto/
store/store_result.c:160:provider=default

┌──(root㉿kali)-[/home/kali/Documents/LAB1/Alice]
└─# ls
AliceDocument  AliceDocumentEncrypted  AlicePrivateKey  AlicePublicKey  BobPublicKey  LargeFile
```

Nous obtenons un message d’erreur. Cela s’explique par le fait que le chiffrement asymétrique n’est pas possible sur les grands fichiers. Pour des fichiers volumineux, on préfèrera utiliser le chiffrement symétrique.

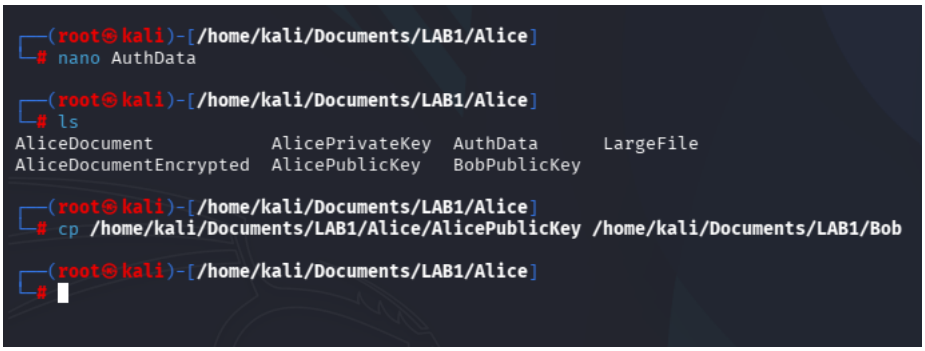
4-Signature électronique

Nous allons désormais générer une signature électronique pour Alice. La signature permet à Alice de s'authentifier auprès de Bob et de s'assurer de l'intégrité des données.

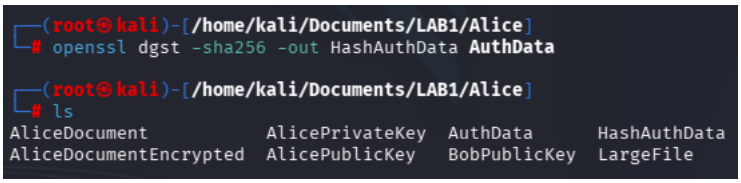
On commence par créer un fichier texte AuthData dans le dossier Alice :



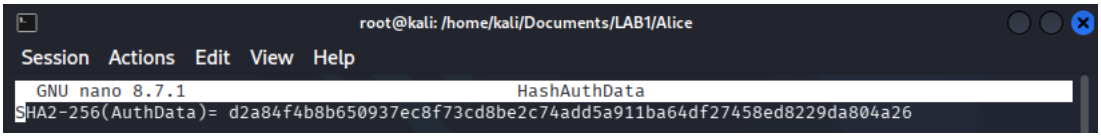
On copie également la clé publique de Alice dans le dossier de Bob :



On applique désormais la fonction de hachage SHA256 sur le fichier AuthData pour générer son hachage HashAuthData :



Vérifions le contenu de HashAuthData :



HashAuthData contient bien le hachage de AuthData

Nous allons maintenant procéder a la signature de HashAuthData avec la clé Privée de Alice en la nommant AliceSignature :

```
┌──(root㉿kali)-[/home/kali/Documents/LAB1/Alice]
└─# openssl rsautl -sign -in HashAuthData -inkey AlicePrivateKey -out AliceSignature
The command rsautl was deprecated in version 3.0. Use 'pkeyutl' instead.

┌──(root㉿kali)-[/home/kali/Documents/LAB1/Alice]
└─# ls
AliceDocument           AlicePrivateKey  AliceSignature  BobPublicKey  LargeFile
AliceDocumentEncrypted  AlicePublicKey   AuthData        HashAuthData
```

Vérifions le contenu de AliceSignature :

On copie ensuite AliceSignature et AuthData dans le dossier de Bob :

```
┌──(root㉿kali)-[/home/kali/Documents/LAB1/Alice]
└─# cp /home/kali/Documents/LAB1/Alice/AliceSignature /home/kali/Documents/LAB1/Bob

┌──(root㉿kali)-[/home/kali/Documents/LAB1/Alice]
└─# cp /home/kali/Documents/LAB1/Alice/AuthData /home/kali/Documents/LAB1/Bob
```

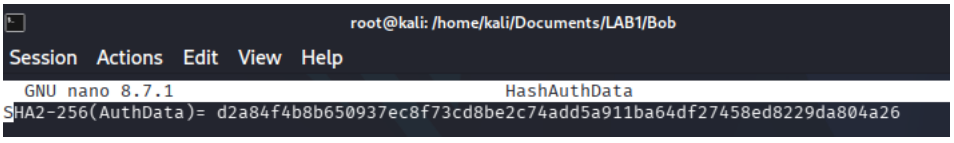
```
┌──(root㉿kali)-[/home/kali/Documents/LAB1/Bob]
└─# ls
AliceDocumentDecrypted  AlicePublicKey  AuthData       BobPublicKey
AliceDocumentEncrypted  AliceSignature  BobPrivateKey
```

Ensuite, nous allons vérifier AliceSignature avec la clé Publique de Alice.

Pour cela, nous allons récupérer le HashAuthData de AliceSignature :

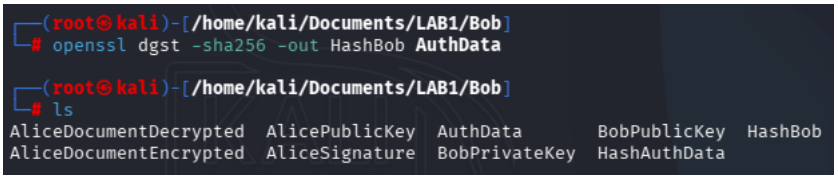
```
┌──(root㉿kali)-[/home/kali/Documents/LAB1/Bob]
└─# openssl rsautl -verify -in AliceSignature -pubin -inkey AlicePublicKey -out HashAuthData
The command rsautl was deprecated in version 3.0. Use 'pkeyutl' instead.

┌──(root㉿kali)-[/home/kali/Documents/LAB1/Bob]
└─# ls
AliceDocumentDecrypted  AlicePublicKey  AuthData       BobPublicKey
AliceDocumentEncrypted  AliceSignature  BobPrivateKey  HashAuthData
```

On vérifie le contenu de HashAuthData :

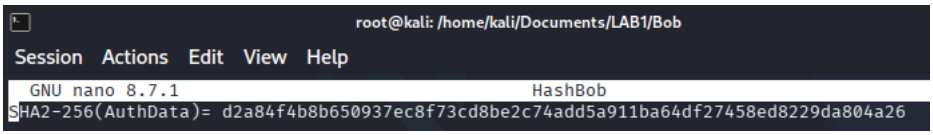


Nous allons maintenant créer un nouveau hachage HashBob sur le fichier AuthData et le comparer avec HashAuthData pour vérifier que les données n'ont pas été modifiées.

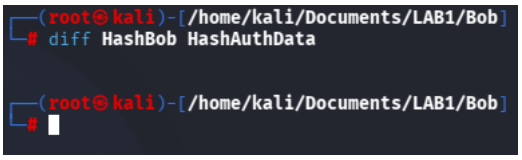
Calculons HashBob sur AuthData :



Vérifions le contenu de HashBob :

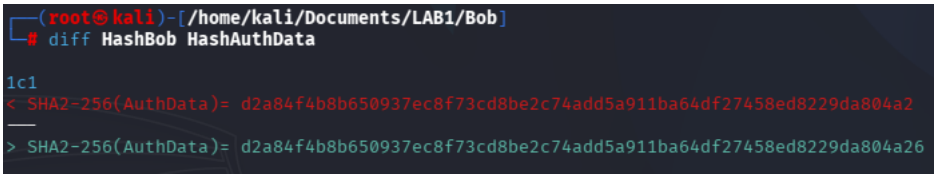


Comparons désormais HashBob et HashAuthData afin de vérifier que le fichier reçu par Bob n'a pas été modifié :



La commande ne renvoie aucun résultat, ce qui signifie que les deux hash sont identiques. Le fichier AuthData reçu par Bob est donc identique à celui de Alice

Si Modifions HashBob, la fonction « diff » devrait nous indiquer une différence entre les deux hash :



Nous voyons ici que les deux hash sont différents.

Dans cette situation, si les deux hash sont différents, on peut supposer que le fichier a été modifié lors de l'envoi par un acteur malveillant.