TP Cryptographie appliquée - Partie 2 RSA

Ilyas Bambrik

Table des matières

Introduction

Vous pouvez télécharger les codes dans cette série directement du lien suivant (*TP Crypto Partie 2 RSA.rar*):

https://drive.google.com/file/d/1BJdxfFYfDxRQ3RVyMamZWVxro3MZvJqT/view?usp=sharing

Avant de pourvoir entamer ce TP, vous devez avoir Anaconda 3 compatible avec votre système installé :

https://repo.anaconda.com/archive/Anaconda3-2020.02-Windows-x86_64.exe

https://repo.anaconda.com/archive/Anaconda3-2020.02-Windows-x86.exe

Génération de clé RSA et chiffrement /déchiffrement

- Contrairement à AES, RSA est un algorithme de chiffrement asymétrique.
- Celui-ci nécessite la génération d'une clé publique *(ligne 19)* et d'une clé privée *(ligne 8)*.
- La clé publique est sauvegardée (*ligne 23*) dans *`fichier_cle_publique.pem`*. La clé privée est sauvegardée (*ligne 11*) dans *`fichier_cle_prive.pem`* .
- Le programme suivant illustre comment génère des clés RSA.

```
1  # importe le module RSA
2  from Crypto.PublicKey import RSA
3
4  # genere une cle publique/prive avec modulos (N) de 2048 bits
5  cle= RSA.generate(2048)
6
7  # recupere la cle prive
8  cle_prive = cle.export_key()
9
10 #ouvre un fichier en mode d'ecriture en octets
11 Fichier = open("fichier_cle_prive.pem", "wb")
12
13 # ecrit le contenu de la cle prive sur fichier
14 Fichier.write(cle_prive)
15 # ferme le fichier
16 Fichier.close()
17
18 # recupere la cle publique
19 cle_publique = cle.publickey().export_key()
20
21
22 # ecrit le contenu de la cle publique sur fichier
23 Fichier = open("fichier_cle_publique.pem", "wb")
24 Fichier.write(cle_publique)
25 Fichier.close()
```

Listing 1 `Création d'une clé publique/privée`

- Le programme suivant permet d'importer les clés publique *(ligne 5)* et privée *(ligne 20)* ainsi que le chiffrement avec clé publique *(ligne 14)* et déchiffrement avec clé privée *(ligne 26)*.
- Un objet permettant de chiffrer avec la clé publique (*ligne 8*) est affecté à *objet_cle_rsa*.
- Un objet permettant de chiffrer avec la clé privée (*ligne 23*) est affecté à *objet_cle_rsa_prive*.

```python
import Crypto
from Crypto.PublicKey import RSA
from Crypto.Cipher import  PKCS1_OAEP
# importe la cle publique du fichier
contenu_clepublique = RSA.importKey(open("fichier_cle_publique.pem","rb").read())

# cree un objet a partir de la cle permetant de chiffre avec RSA
objet_cle_rsa = PKCS1_OAEP.new(contenu_clepublique)

# message claire
message=b"Message"

# chiffre le message avec la cle publique
message_chiffre=objet_cle_rsa.encrypt(message)

#imprime le resultat du chiffrement
print(message_chiffre)

# importe la cle prive du fichier
contenu_cleprive = RSA.importKey(open("fichier_cle_prive.pem","rb").read())

# cree un objet a partir de la cle permetant de chiffre avec RSA
objet_cle_rsa_prive = PKCS1_OAEP.new(contenu_cleprive)

#imprime le resultat du dechiffrement
print(objet_cle_rsa_prive.decrypt(message_chiffre))
```

Listing 2 `Chiffrement avec clé publique et déchiffrement avec la clé privée`

Question

Pour un exercice échauffement, créez un programme permettant de déchiffrer avec RSA le fichier [*1_crypte.txt*]situé dans le lien suivant (la clé privée permettant le déchiffrement du contenu est situé dans le même répertoire Google [*fichier_cle_prive.pem*], utilisez celle-ci pour déchiffrer) :

https://drive.google.com/drive/folders/1nR-WknZLy2PYOzlA6uptUCxw6ETFDaN3?usp=sharing

Communication avec serveur sécurisée

II

Pour commencer, vous devez générer les deux fichiers contenant la clé publique et la clé privée avec le programme expliqué dans *Listing 1*. La clé publique doit être placée dans le fichier *fichier_cle_publique.pem*, et la clé privé doit être placée dans le fichier *fichier_cle_prive.pem* .

Les deux programmes représentent un code du serveur muti-thread d'un client. Ces derniers doivent être placés dans le même répertoire que les fichiers des clés publique et privé (*fichier_cle_publique.pem* et *fichier_cle_prive.pem*).

- Le serveur muti-thread (*Listing 3*) est le même que le serveur multi-thread du 1 semestre mais transmet la clé publique au client initialement (*contenu_clepublique*) ;
- Par la suite les messages reçus du clients doivent être déchiffrés avec la clé privée (*objet_cle_rsa_prive*) ;

```python
import socket
import _thread as thread
from Crypto.PublicKey import RSA
from Crypto.Cipher import  PKCS1_OAEP


def Traiter_Connexion(connexion_avec_client,adresse_client):
   global objet_cle_rsa_prive,contenu_clepublique
   MessageRec=b""

   print ("Connexion de la machine = ", adresse_client)
   # Transmettre le contenu de la cle publique apres la connexion
   # Utilisez la methode send de connexion_avec_client pour
   # transmettre contenu_clepublique
   # (A FAIRE 1) ajoutez l'instruction dans cette ligne


   try:
      while True:
          MessageRec=connexion_avec_client.recv(1024)
          # Dechiffre le message recu (MessageRec)
          # Utilisez la methode decrypt de la cle prive (objet_cle_rsa_prive)
          # Affectez le resultat du dechiffrement a MessageRec
          # (A FAIRE 2) ajoutez l'instruction dans cette ligne



          if MessageRec==b"Fin":
                  break

          print("Client" ,adresse_client," a dit :",MessageRec.decode())
   except:
      print("Deconnexion")
   print("Deconnexion de :",adresse_client)
   try:
      connexion_avec_client.close()
   except:
      pass

SocketServeur = socket.socket()
host = socket.gethostname()
port = 9500
SocketServeur.bind(("127.0.0.1", port))

SocketServeur.listen(5)

contenu_clepublique = open("fichier_cle_publique.pem","rb").read()

contenu_cleprive = RSA.importKey(open("fichier_cle_prive.pem","rb").read())
# cree un objet a partir de la cle permetant de chiffre avec RSA
objet_cle_rsa_prive = PKCS1_OAEP.new(contenu_cleprive)
print("Lancement serveur")
while True:
   ConnexionAUnClient, addrclient = SocketServeur.accept()
   thread.start_new_thread(Traiter_Connexion,(ConnexionAUnClient,addrclient))

```

Listing 3. `Code Serveur multi-thread avec chiffrement`

- Après la connexion au serveur, le client (*Listing 4*) doit recevoir la clé publique (qui sera transmise automatiquement par le serveur). A partir de la clé publique reçue (*Cle_publique*), le client crée un objet de clé publique (*objet_cle_rsa_publique*) ;
- Par la suite chaque message saisie par le client, celui-ci est chiffré avec la clé publique (*objet_cle_rsa_publique*) et transmet le résultat (*resultat_chiffre*) ;

```
import socket
from Crypto.PublicKey import RSA
from Crypto.Cipher import  PKCS1_OAEP

SocketClient = socket.socket()
host = socket.gethostname()
port = 9500
SocketClient.connect(("127.0.0.1", port))

Cle_publique=b""
while True:
        recu=SocketClient.recv(1024)
        Cle_publique+=recu
        if b'-----END PUBLIC KEY-----' in Cle_publique:
                break

# Importez la cle a partir du contenu recu Cle_publique
# Utilisez la methode importKey de RSA pour importer la cle
# Mettez le resultat dans clepublique
# Creez un objet objet_cle_rsa_publique a partir de clepublique (objet de cle publique)
# Utilisez PKCS1_OAEP.new pour faire ceci
# (A FAIRE 1) ajoutez deux instructions dans les deux lignes suivantes


print("Lancement serveur")
while True:
   MessageATransmettre=input().encode()
   # Chiffre le message lu du clavier (MessageATransmettre)
   # Utilisez la methode encrypt de objet_cle_rsa_publique
   # Mettez le resultat dans resultat_chiffre
   # (A FAIRE 2) ajoutez l'instruction dans cette ligne



   SocketClient.send(resultat_chiffre)
   if(MessageATransmettre==b"Fin"):
      break
      print( "Deconnexion de :",addrclient)
ConnexionAUnClient.close()
```

Listing 4. `Code Client avec chiffrement`

Question

Complétez les instructions des programmes client/serveur là où il y a un commentaire # (A FAIRE). Vos objectifs sont les suivants :

1. *Programme Serveur :* Transmettre le contenu de la clé publique au client (A FAIRE1 - Ligne 16). *Une instruction suffirais pour ceci* ;
2. *Programme Serveur :* Déchiffrer le contenu reçu du client [*MessageRec*] avec la clé privée [*objet_cle_rsa_prive*](A FAIRE 2-Ligne 25). *Une instruction suffirais pour ceci* ;
3. *Programme Client :* Créer l'objet de la clé publique au niveau du client [*clepublique*] à partir de la clé reçue du serveur [*Cle_publique*](A FAIRE 1-Ligne 21). Voir l'exemple dans Exercice 1 (Listing 2- Ligne 5 pour importer la clé à partir du contenu reçu) ; (A FAIRE 1-Ligne 21) ;
4. *Programme Client :* Créer un objet de la clé publique à partir de *clepublique*, et mettre le résultat dans l'objet *objet_cle_rsa_publique*. Voir Listing 2- Ligne 8 pour comment créer un objet à partir du contenu de la clé .(A FAIRE 1-Ligne 22) ;
5. *Programme Client :* Chiffrer le contenu du message saisi (*MessageATransmettre*) par l'utilisateur avec la clé publique(*objet_cle_rsa_publique*). Mettez le résultat dans la variable *resultat_chiffre*. Voir Listing 2- Ligne 14 pour comment chiffrer. (A FAIRE 2-Ligne 32) ;

Remarque :

En réalité, le serveur transmet ça clé publique au client mais seulement afin que le client chiffre et transmet à son tour une clé symétrique (AES ou DES) au serveur. Par la suite, les messages transmis par le client et par le serveur sont chiffré avec la clé symétrique (et non pas par la clé publique RSA).

La difficulté de se programme réside dans comment encoder la langueur du message, calculer le nombre d'octets de bourrage ajoutés à la fin.