Lab 7 - RSA Implementation

Quick Overview of RSA:

RSA (Rivest–Shamir–Adleman) is a public-key cryptosystem that is widely used for secure data transmission. RSA uses an asymmetric encryption algorithm to encrypt data. The RSA algorithm involves four steps: key generation, key distribution, encryption, and decryption.

Import Important Libraries

```
# To Use Crypto, we need to install pycrypto first.

!pip install pycrypto
```

```
from Crypto.PublicKey import RSA
from Crypto import Random
from Crypto.Cipher import PKCS1_OAEP
from Crypto.Random import get_random_bytes
import time
```

```
# RSA Program

time.clock = time.time

class RSA_Scheme:
  # RSA encryption method
  def RSA_Encryption(self, key, msg):
    public_key = key.publickey().exportKey('PEM')
    rsa_public_key = RSA.importKey(public_key)
    rsa_public_key = PKCS1_OAEP.new(rsa_public_key)
    encrypted_text = rsa_public_key.encrypt(msg.encode())
    return encrypted_text

  # RSA decryption method
  def RSA_Decryption(self, key, encrypted_text):
    private_key = key.exportKey('PEM')
    rsa_private_key = RSA.importKey(private_key)
    rsa_private_key = PKCS1_OAEP.new(rsa_private_key)
    decrypted_text = rsa_private_key.decrypt(encrypted_text)
    return decrypted_text.decode()
```

Here we test our RSA program.

```
# Correct example test

key = RSA.generate(2048) # Generate key for public and private, and e is random.
message = "We shall attack on March 5th at noon!" # Message to be encrypted.

RSA_Crypto = RSA_Scheme()

ciphertext = RSA_Crypto.RSA_Encryption(key, message)
msg = RSA_Crypto.RSA_Decryption(key, ciphertext)

print("Your encrypted message is: ", ciphertext)
print()
print("Your decrypted message is: ", msg)
```

```
Your encrypted message is:  b"T\xf7\xb4\x1c\x9e\x90\xee\xfd\xbb'/\xcd\xa7\xe7T\xc7\xc6$O\x1b\xf97[\xae\xdf\xb4eX7\x95\r\xa1

Your decrypted message is:  We shall attack on March 5th at noon!
```

```
# Wrong key usage test

key = RSA.generate(2048) # Generate key for public and private, and e is random.
fake_key = RSA.generate(2048)
message = "We shall attack on March 5th at noon!" # Message to be encrypted.

RSA_Crypto = RSA_Scheme()

ciphertext = RSA_Crypto.RSA_Encryption(key, message)
msg = RSA_Crypto.RSA_Decryption(fake_key, ciphertext)

print("Your encrypted message is: ", ciphertext)
print()
print("Your decrypted message is: ", msg)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-17-1d922a1d0b37> in <module>
      8
      9 ciphertext = RSA_Crypto.RSA_Encryption(key, message)
---> 10 msg = RSA_Crypto.RSA_Decryption(fake_key, ciphertext)
     11
     12 print("Your encrypted message is: ", ciphertext)

 ---------------------------------------- 1 frames ----------------------------------------
/usr/local/lib/python3.8/dist-packages/Crypto/Cipher/PKCS1_OAEP.py in
decrypt(self, ct)
    225             valid = 0
    226         if not valid:
--> 227             raise ValueError("Incorrect decryption.")
    228         # Step 4
    229         return db[hLen+one+1:]

ValueError: Incorrect decryption.

SEARCH STACK OVERFLOW
```

Observations:

In the first example, we use a right encryption key and the RSA cryptosystem program does do its job and encrypts and decrypts correctly. In the second example, we use the wrong key (fake_key) to see what would happen and if our program will handle it correctly instead of decrypting it. As we see on the second example, it terminated the program and the error message was printed by Crypto; hence, this program is doing its job correctly.

References:

https://stackoverflow.com/questions/30056762/rsa-encryption-and-decryption-in-python

https://pythonhosted.org/pycrypto/Crypto.Cipher.PKCS1_OAEP-module.html

---