Crypto Homework 5b—RSA Encryption

# Required reading

Crypto 5 slides

We will use the RSA key pair you generated in Lab 5a to encrypt a short (< 2048 bits or < 256 bytes) message and exchange it with your partner. With RSA we can have a secure key exchange. We can post our public key where everyone can see it. Anyone who wants to send an encrypted message to us can encrypt the message with our public key. The private key, which only we have, is used to decrypt the message.

Note: This method is not failsafe; we must be certain that the public key we are using is really that person’s key. If an attacker can trick us into using their public key instead of the recipient’s key, they can break our encryption.

The lab follows the example, Encrypt data with RSA in the Pycryptodome documentation. <https://pycryptodome.readthedocs.io/en/latest/src/examples.html>

You will use the RSA public/private key pair you created in the last lab and give the public key to your partner. Your partner will encrypt the message with your public key and give the encrypted message to you. You should be able to decrypt the session key with your RSA private key.

Note: If your course is entirely online, you can send the message to yourself instead of to a partner; your choice.

# Overview

This is the procedure we will follow.  
Application

Description automatically generated

# RSA key pair

You each should have your own RSA public/private key pair that you generated in the last lab. The first step will be to get your public key to your partner. Public key encryption makes secure key exchange possible. It will not affect security if the attacker sees your public keys as you exchange them; the public key is public, after all.

However, you must be certain that the public key you have from your partner really did come from your partner. If an attacker can substitute their key for your partner’s, the attacker can conduct a Man in the Middle (MitM) attack against you. We will talk about certificates as a means of exchanging public keys (more or less) securely in a later lesson. For the time being you can choose the method you want to use to get your public key to your partner.

* Sneakernet. Trade public keys using a flash drive. If you are working by yourself, just give yourself the files you need ;-)
* Email your public keys to each other.
* Optional fun—use netcat or a simple Python web server as shown in the appendix.

It may be wise to rename your partner’s public key (Bob\_public.pem, or something) so you do not get it confused with your own. Put your partner’s public key in your python directory.

# Encrypt a message

In this example we will encrypt a message to send to Bob

from Crypto.PublicKey import RSA

from Crypto.Cipher import PKCS1\_OAEP

# read and import Bob's RSA public key

with open('Bob\_public.pem') as fh:

Bob\_pub = fh.read()

Bob\_pub\_key = RSA.import\_key(Bob\_pub)

plaintext = 'This is the message I want to send to Bob'.encode()

# create an RSA cipher (encryption) object using OAEP

cipher\_rsa = PKCS1\_OAEP.new(Bob\_pub\_key)

# encrypt the message

ciphertext = cipher\_rsa.encrypt(plaintext)

# save the message in a file

with open('message\_to\_bob.bin', 'wb') as fh:

fh.write(ciphertext)

# print ciphertext just so we can see it  
print(ciphertext)

Although the message is short, you will see that the encrypted message is much longer. This is because PKCS1 OAEP pads the message so that it fills an entire block. In RSA encryption the block size is the same as the key length, which is 256 bytes or 2048 bits in our case.

### Windows

Graphical user interface, text, email

Description automatically generated

Text

Description automatically generated

### Linux

Graphical user interface, text, application, email

Description automatically generated

Text

Description automatically generated

# Decrypt the message.

In this example we will decrypt a message Bob sent to us. He should have encrypted it with our public key, so we will decrypt with our private key.

from Crypto.PublicKey import RSA

from Crypto.Cipher import PKCS1\_OAEP

#import the private key

with open('private.pem') as fh:

private\_key = RSA.import\_key(fh.read())

#import the encrypted message

with open('message\_to\_bob.bin', 'rb') as fh:

ciphertext = fh.read()

# Decrypt the message with the RSA private key

cipher\_rsa = PKCS1\_OAEP.new(private\_key)

plaintext = cipher\_rsa.decrypt(ciphertext)

print(plaintext.decode())

## Windows

Text

Description automatically generated



## Linux

Text

Description automatically generated

Text

Description automatically generated

# Hand in, Part 1

Hand in screenshots of your encryption and decryption.

# Hand in, Part 2

Your foolish instructor has posted a private key, saving\_private.pem, on Canvas along with an encrypted file. The file (also on Canvas), saying.bin was created using the procedure we just followed (Bob in the overview.) Use the private key to decrypt saying.bin. What is the saying?

You can use the same decryption script you used before, except that you will need to read the private key saving\_private.pem and read saying.bin instead of encrypted\_data.bin.

# Appendix—Fun ways to exchange keys with your partner.

Flash drives and email might be too boring for an uber hacker to use to exchange keys. If that is the case, netcat or a simple python web server may be more fun.

## Netcat

If you have Nmap installed on your computer, you have their version of netcat, called ncat, installed. In Windows, it is usually in C:\Program Files (x86)\Nmap, so change directories  
cd C:\Program Files (x86)\Nmap. If you use Linux, their (original) version of netcat is called nc, and usually installed so you can run it from any directory.

If you use a VM, the VM Network Adapter must be in bridged mode so that it has its own IP address.

1. Make sure you can ping the other members of your group and fix the problems if you can't. You may have to turn off your firewalls to allow people to ping your hosts. You can use an elevated command prompt and NetSh Advfirewall set allprofiles state off. You can use the GUI if you wish—use your favorite search engine to learn how.
2. One person needs to set their netcat to listen on a TCP port (this is what a server does; listen for connections.) For the port number, choose a number between 2000 and 65,000. If you don’t choose a port, ncat defaults to 31337. If I chose port 3000, I would enter this. (The lower case “L” means listen.)  
   ncat -l 3000 (Windows)  
   nc -l 3000 (Linux)
3. The other person needs to use ncat to connect to the first person’s listener (this is what a client does; connect to a server.) You need to know the other person’s IP address and the port number they are listening on. For example, if the other person’s IP is 10.0.0.4 and they are listening on port 3000, you would enter this.  
   ncat 10.0.0.4 3000 (Windows)  
   nc 10.0.0.4 3000 (Linux)
4. Now that you and your partner are connected, you should be able to paste your public keys into the terminal to exchange them.

Simple Web Server.

Python has a built-in module that contains a very simple web server, but there are limitations. First, you must know how to open a port on your firewall (or temporarily disable it) so that your partner can connect. Second, if you use a VM, the VM Network Adapter must be in bridged mode so that the VM has its own IP address. The server will publish any files in the directory you run the command from. If you can handle all that, the command is simple.  
Windows Python 3  
python -m http.server 8000  
Ubuntu  
python3 -m http.server 8000  
where 8000 is the port number to use, your choice. Your partner will need to point their web browser to your IP address and port, 192.168.86.130:8000, as an example.

The simple Python web server logs everything it hears, so you don’t have to write a web page to receive data. You can netcat your data to the server or use a browser. Here I just pasted the public key after a “/”.  
Graphical user interface, text, application, email

Description automatically generated

Text

Description automatically generated

You can see the public key in the server’s log.