0416005, 張彧豪

Network Security Project 1 Report

A. Introduction

In this project, chosen cipher attack is our attack method against to RSA crypto system. In this kind of attack against RSA crypto system, the attacker has the cipher text and also the access to the decrypt machine which means the attacker can feed any cipher text to the machine and get the decrypted result. Here is the detail procedure about chosen cipher attack against RSA crypto system.

Attack steps:

- choose X where X is relatively prime to n
- create $Y = C*X^e \bmod n$
- get Z = decrypted Y
- $Z = Y^d = (C*X^e)^d = C^d*X^{ed} = C^d*X = P^{ed}*X = P*X \bmod n$
- find out X^{-1}, the modular inverse of X
- $P = Z*X^{-1} \bmod n$

We may put the mathematical theory and proof behind and know to how to use these steps to complete our chosen cipher attack against RSA crypto system.

B. Attack Procedure

1. I choose 2 as my X, since the modulus of the keys “n” is shown below, and it is an odd number, 2 must be relatively prime to the “n”.

```
Public-Key: (1024 bit)
Modulus:
    00:c8:87:5e:92:6b:76:02:a6:98:9f:11:33:66:97:
    a8:0a:6b:f0:b2:e7:a0:13:fe:1c:4d:24:cc:7a:cd:
    09:90:b9:8b:e3:10:80:40:6c:1e:04:9d:6d:f8:e9:
    ec:32:d7:24:51:54:30:4a:55:8a:c9:4c:76:e6:c3:
    69:c8:1d:b4:06:3f:94:14:f0:60:db:46:88:0c:d3:
    94:42:92:8a:d2:66:30:74:dc:bf:19:86:9a:75:bb:
    b8:62:f3:f7:61:3e:72:9b:42:7a:c2:1e:b2:3e:4b:
    91:00:bb:83:8a:df:2c:3f:9e:1e:fa:2a:50:4e:6f:
    f1:b5:fa:9c:b4:d8:44:87:5b
```

The last bit in hexadecimal representation is b which is eleven in decimal representation and I am sure that the modulus is an odd number because of that.

2. I was stuck at this step when I was working on the project. First thing that bothered me is the cipher text has to be base64 decode first, and the second thing that bothered me is how to convert the decoded cipher text to a computable number properly.
   I. I used a method like BCD representation first, I convert the decoded cipher text to hex-presentation and treat each hex-char as a number, so the computable number of the cipher text will be all the number concatenate together, but when the number

needs to be converted to ASCII, it is a hard work so I give up this method.

II. This time I still convert the decoded cipher text to hexadecimal representation but treat the whole string as a big number in hexadecimal, all I have to do is converting it to decimal and convert it hexadecimal if ASCII is needed. All conversion can use help of built-in conversion function, it is more convenient.

```
# 2. Convert the ciphertext to hex-representation
originalCiphertextInHex = binascii.hexlify(base64.b64decode(openCiphertext()))

"""
3. Convert the hex-representation to decimal representation
   and compute the fakeCiphertext value
   # publicKeyObject.n is the modulus
"""
fakeCiphertextValue = long(originalCiphertextInHex, 16) * (2**long(publicKeyObject.e))

# 4. Get the remainder
remainder = fakeCiphertextValue % long(publicKeyObject.n) # remainder is a long number
```

3. Just feed the Y in base64 encoding to decrypt server and get result Z.

```
# 5. Convert the hex-representation of remainder to ASCII
fakeCiphertext = hexString2ASCII('%x' % remainder)

# 6. Feed the ASCII string in base64 encoding to the decrypt server and get the result in "out" variable
first = subprocess.Popen(["/bin/echo", base64.b64encode(fakeCiphertext)], stdout=subprocess.PIPE)
out = subprocess.check_output(["/bin/nc","140.113.194.66","8888"],stdin=first.stdout)
```

4. The processing flow for Z is pretty much as same as what I did to the cipher text, base64 decoding first and convert it to a huge computable number.

```
# 7. First convert the output to hex-representation and convert it to decimal
fakePlaintextValue = long(binascii.hexlify(base64.b64decode(out.split('\n')[1])), 16)
```

5. Divide the result from step 4. by X then I can get the plain text in number format, converting it to ASCII and the original plain text is shown.

```
# 9. Print the real plaintext out in ASCII
originalPlaintext = binascii.unhexlify('%x' % fakePlaintextValue)
print originalPlaintext
```

C. Python help

1. Public key extraction, use the “importkey” function in the RSA module in the Crypto.PublicKey module.[1]
2. ASCII to hexadecimal, use the “hexlify” function in the binascii module.[2]
3. Hexadecimal string to ASCII, use the format string syntax and the “unhexlify” function in the binascii module.[3]
4. Base64 encoding and decoding, use the base64 module.[4]

D. Script

- File name is “0416005.py”
- Written in python2.
- Pycrypto package need to be installed.
- Put the cipher text and this script under the same folder.
- Type the “python 0416005.py” to execute it and the original plain text will be printed out.

E. References

[1] Pycrypto Documents, “Module RSA”, Available at https://www.dlitz.net/software/pycrypto/api/2.6/Crypto.PublicKey.RSA-module.html

[2] Python 2.7.14 documentation, “binascii — Convert between binary and ASCII”, Available at https://docs.python.org/2/library/binascii.html#binascii.b2a_hex

[3] Python 2.7.14 documentation, “binascii — Convert between binary and ASCII”, Available at https://docs.python.org/2/library/binascii.html#binascii.unhexlify

[4] Python 2.7.14 documentation, “base64 — RFC 3548: Base16, Base32, Base64 Data Encodings”, Available at https://docs.python.org/2/library/base64.html