Week 7 Practical: Cryptography with *PyCrypto*

Dr. Qublai Ali Mirza

University of Gloucestershire

qalimirza@glos.ac.uk

Overview

1. Recap
2. PyCrypto
3. RSA
4. DES
5. AES
6. Bringing it all together
7. Post-sessional work

Recap

- Last week, we looked at using Hashing
- We also looked at incorporating it with different algorithms that we have looked thus far
- This week, we will be looking at using `PyCrypto` to implement:
    - RSA
    - DES
    - AES

PyCrypto

- Is a `Python` module for cryptography
- Consists of a variety of:
  - Symmetric and asymmetric encryption algorithms
  - Hash algorithms
- Can be set up via the Command Line Interface (*CLI*)
- On your Windows computer, open up the Command Prompt
- To install `PyCrypto`, type in:
  - `conda create -n Envpycrypto python=3.5`
  - `activate Envpycrypto`
  - `conda install pycrypto`

RSA

- The first algorithm we will be looking at is the RSA algorithm
- Public-key encryption
- Uses large prime numbers
- We will be looking at:
  - Key Generation
  - Encryption
  - Decryption

```
#!/usr/bin/python3

from Crypto.PublicKey import RSA

from Crypto import Random

#  Get a random number from the RNG

random_generator = Random.new().read

# Generate RSA key pair using the new RNG value

key = RSA.generate(1024, random_generator)

print(key)
```

```
...
...
# Represent the plaintext as a byte-based string

strPlaintext = b'RSA encryption is pretty straightforward'

# First get the public key from the key-pair

key_public = key.publickey()

# The encrypt the plaintext using it

strCipherText = key_public.encrypt(strPlaintext, 32)

print(strCipherText)
```

```
...
...
...
# Now let's decrypt the ciphertext

strDecryptedText = key.decrypt(strCipherText)

print(strDecryptedText)
```

Exercise

Download `plaintext.txt` file from Moodle, and then from `Python`:

- Open up the file and read line by line
- Encrypt *each* line using RSA
- Write the resulting ciphertext into a file called `ciphertext.txt`

DES

- Block cipher
- Block size: 64 bits
- Key size: 64 bits, with only 56-bits used
- In the context of `Pycrypto`, it means that we need to use a key the length of which is a multiple of 8.

```
#!/usr/bin/python3


from Crypto.Cipher import DES

from Crypto import Random

# First create a key which needs to be a multiple of 8
# in length

strKey = b'helloall'

# Create a DES object. See next slide.
```

```
obj_des = DES.new(strKey, DES.MODE_ECB)

# The plaintext must also be a multiple of 8 in length
# as well

strPlainText = b'Roses are red. Violets are blue!'

# Encrypt it using DES

strCipherText = obj_des.encrypt(strPlainText)

print(strCipherText)
```

AES

- Designed to replace the aging DES encryption algorithm
- Number of rounds varies depending on the key length
- For the purposes of our demonstration, we will be using 128-bit key

```
#!/usr/bin/python3

from Crypto.Cipher import AES

from Crypto import Random

# First define key

strKey = b'Hello world all!'

# Then define the target plaintext

strPlainText = b'Roses are red. Violets are blue!'
```

```
iv = Random.new().read(AES.block_size)

mode = AES.MODE_CBC

# Create an AES object

encryptor = AES.new(strKey, mode, iv)

# Then encrypt the plaintext

strCipherText = encryptor.encrypt(strPlainText)

print(strCipherText)
```

```
# Now decrypt the ciphertext

decryptor = AES.new(strKey, mode, iv)

strDecryptedText = decryptor.decrypt(strCipherText)

print(strDecryptedText)
```

Bringing it all together

- We looked at `PyCrypto`
- We also looked at how we can implement different cryptography approaching using it
- Next week: *Hashing & Elliptical Curve Cryptography*

Post-sessional work

- Using the in-lab exercise at starting point, perform encryption on `plaintext.txt` and measure the amount of time in *both* encryption and decryption.
- **Hint:** you might want to use the `timeit.timeit` function

Q & A