

## Lab 8-6-2023

1. Do some research on the following signature algorithms

- a. RSA
- b. DSA

then answer the questions at this link

<https://forms.gle/TSU3pG4LDDixfDzdA>

2. Write a program that does the following

- a. Takes a plain text message
- b. Generates the public and private key pair for RSA
- c. Encrypts and decrypts the message using RSA
- d. Generates a public and private key pair for DSA
- e. Generates a signature for the message and verifies it using DSA

### Instructions:

1. Note: For this hands on exercise you need to install the “cryptography” library. You can install it by “**pip install cryptography**”.

2. To start, you need to import necessary modules from the cryptography library.

```
# Import the necessary modules from the cryptography library
from cryptography.hazmat.primitives.asymmetric import rsa, dsa
from cryptography.hazmat.primitives import serialization, hashes
from cryptography.hazmat.primitives.asymmetric.padding import PKCS1v15
from cryptography.hazmat.primitives import padding
```

3. For the rest of the exercise please consult the following pseudo code.

### **Function generateRSAKeyPair():**

1. Generate a private key using `rsa.generate_private_key()` function with `public_exponent` and `key_size` as parameters. Save the result in `privateKey`.
  - a. Set `public_exponent` to 65537
  - b. Set `key_size` to 2048
2. Obtain the corresponding public key by calling `privateKey.public_key()`. Save the result in `publicKey`.
3. Return `privateKey` and `publicKey`.

### **Function RSAEncrypt(publicKey, plainText):**

1. Encrypt `plainText` using the `encrypt(plainText, padding technique)` method of `publicKey`. Use `PKCS1v15()` padding. Save the result in `cipherText`.
2. Return the resulting `cipherText`.

### **Function RSADecrypt(privateKey, cipherText):**

1. Decrypt `cipherText` using the `decrypt()` method of `privateKey`. Use `PKCS1v15()` padding.
2. Return the resulting `plainText`.

**Function generateDSAKeyPair():**

1. Generate a private key using `dsa.generate_private_key()` function with `key_size` as a parameter. Save the result in `privateKey`.
  - a. Set `key_size` to 1024
2. Obtain the corresponding public key by calling `privateKey.public_key()`. Save the result in `publicKey`.
3. Return `privateKey` and `publicKey`.

**Function DSASign(privateKey, message):**

1. Generate a signature by calling the `sign(message, hashing algorithm)` method of `privateKey`. Use `hashes.SHA256()` as the hashing algorithm.
2. Return the resulting signature.

**Function DSAVerify(publicKey, message, signature):**

1. Try to verify the signature by calling the `verify(signature, message, hashing algorithm)` method of `publicKey`. Use `hashes.SHA256()` as the hashing algorithm.
2. If the verification succeeds, return `True`.
3. If an exception occurs (doesn't verify) during verification, return `False`.

**Function main():**

1. Generate an RSA key pair by calling `generateRSAKeyPair()`. Store the returned keys as `RSAPrivateKey` and `RSAPublicKey`.
2. Convert the message e.g "Message for RSA algorithm" to bytes and store it in `plainText`.

3. Encrypt plainText using RSAEncrypt(RSAPublicKey, plainText). Store the result in cipherText.
4. Decrypt cipherText using RSADecrypt(RSAPrivateKey, cipherText) function with privateKey. Store the result in decryptedText.
5. Print RSA details:
  - a. RSA Public Key: RSAPublicKey
  - b. RSA Private Key: RSAPrivateKey
  - c. Plaintext: Decode plainText
  - d. Ciphertext: cipherText
  - e. Decrypted Text: decryptedText
6. Generate a DSA key pair by calling generateDSAKeyPair(). Store the returned keys as DSAPrivateKey and DSAPublicKey.
7. Convert the message e.g "Message for DSA algorithm" to bytes and store it in a message.
8. Generate a signature by calling the DSASign(DSAPrivateKey, message) function with privateKey and message as parameters. Store the result in signature.
9. Verify the signature by calling DSASign(DSAPublicKey, message, signature) function with publicKey, message, and signature as parameters. Store the result in verified.
  - a. Print DSA details:
  - b. DSA Public Key: DSAPublicKey
  - c. DSA Private Key: DSAPrivateKey
  - d. Message: Decode message
  - e. Signature: signature
  - f. Verification: verified

10. Call the `main()` function.