By-Varun Singhal(MT20099)
Sachin Kotiyal(MT20093)

CSE 540 Network Security
Assignment 4
Project 0

Overall Program Structure:

- Generate Public and private keys of Client server.
- Generate Public and private keys of Server.
- Send doc encrypted data from client to server.
- Then add gmt date and time after that data.
- And also add hash function value for digital signature.
- Send all data from Server to client and finally decrypt that data with gmt time and data at client site and then save that data into a new doc file.

Generate Keys for Client and Server:

- Generate Public and private key of Client server

```
public_key_client =  (74594497642439042403421, 79292053079056066600697)
private_key_client =  (37315076390064917859001, 79292053079056066600697)
public_key_server =  (11431981512385889769490 9, 31711093763837212435585 1)
```

- Generate Public and private keys of Server.

```
public_key_server =  (11431981512385889769490 9, 31711093763837212435585 1)
private_key_server =  (27388701301087999674238 9, 31711093763837212435585 1)
public_key_client =  (74594497642439042403421, 79292053079056066600697)
```

Program Architecture: Program has been divided into 2 parts : One server and clients.

- **Server :**There are 2 main components of a.
  - Receive Request : It is used to receive documents from the client and add GMT date and time stamp into the document.
  - **Digital signature**: Calculate the hash of the document using SHA for the document is digitally signed by the server(by encryption using the private key of the server) and sent to the client.

```
""" Receiving the file data from the client. """
data = conn.recv(SIZE).decode(FORMAT)
print(f"[RECV] Receiving the file data.")
#file.write(data)

gmt_time = datetime.datetime.now(datetime.timezone.utc).strftime("%Y-%m-%dT%H:%M:%S.%f%Z")
# gmt_time = str(gmt_time)
gmt_time = encrypt(gmt_time, public_key_client)

data = data + " " +str(gmt_time[0])
for i in range(1, len(gmt_time)):
    data += " " + str(gmt_time[i])
```

- **Client:**There are 5 main functions in the client.
  - **Send message** : Main driver function of the client used to send messages to the server. The message(document) is encrypted using the public key of the client.
  - **Receive message** : This Function gets the message from the server which included the document, added GMT date and time and hash value. Hash value is appended by the separator $.

```
hash_object = hashlib.sha256(str(data).encode('utf-8'))
hashvalue = hash_object.hexdigest()

hashvalue_encrpyt = encrypt(hashvalue, private_key_server)

hash_added = str(hashvalue_encrpyt[0])
for i in range(1, len(hashvalue_encrpyt)):
    hash_added += " " + str(hashvalue_encrpyt[i])

actual_data = data + "$" + hash_added

print(actual_data)
conn.send(actual_data.encode(FORMAT))
```

- **Check Authentication** : This function first splits the message by $ function and separates the hash value and message. This hash function is compared by the decryption of a message by the public key of the server and calculates the hash value. If it is similar means the document is not modified, otherwise the document is modified.

```
gmt_data_decrpyt = decrypt(new_gmt_data, private_key_client)
hash_data_decrpyt = decrypt(new_hash_data, public_key_server)

hash_object = hashlib.sha256(str(gmt_data).encode('utf-8'))
hashvalue = hash_object.hexdigest()

if hashvalue == hash_data_decrpyt:
    print("digital signature Verify , Authenticated")
    print(gmt_data_decrpyt)

    text_file = open("Output.txt", "w")
    text_file.write(gmt_data_decrpyt)
    text_file.close()


else:
    print("Not Authenticated")
```

Summary of Model Working: client wants to send a message(document) to

the server.

- And we all know the public key of client and server is known to each other.
- Client encrypts the message using his/her public key and sends it to the server.
- Now the server adds the GMT Time stamp and encrypts this message using a private key for digital signature and appends a hash value by the separator $.
- Client receives the document and checks the similarity of hash value for validation.