

# ASSIGNMENT-3

## PART-2:IMPLEMENTING CUSTOMIZED TLS

Kashish jain

April 2021

### 1. README:

In first terminal/tab run,

```
python my_ttp.py
```

In second terminal/tab run,

```
python my_server.py
```

In third terminal/tab run,

```
python my_client.py
```

### 2. IMPLEMENTATION:

#### **my\_ttp.py:**

The Trusted Third Party, issues digital certificates to both the client and the server. I have used OpenSSL library for creating the certificates. For signing the certificates of client and server 'SHA384' hash algorithm is used and TTP used 'SHA256' for self-signing the certificate.

Initially TTP creates its RSA key pair then generates a X509 certificate which it signs using its key pair. After that it creates a socket-ttp\_socket to listen to server request and creates a signed x509 certificate using common name('server') of server received in message and public key of server(server\_key). Similarly for client also, TTP opens a new socket ttp\_socket and listens to client request and receives its message. Then creates a signed certificate for client using its common name('client') and public key(client\_key).

#### **my\_server.py:**

I have used ssl library for performing Handshake protocol. Server first creates its RSA key pair, then opens a socket `ttp_socket` to connect with TTP with common name-‘server’, and request for digital certificate for its public key.

And after receiving the certificate, server closes the socket and listens to client connection request.

After connecting to client, servers request for client digital certificate using `getpeercert()` command, also simultaneously send its digital certificate to client. After receiving client’s certificate, server checks for its validity/expiration and if its expired, then informs client about it, or if certificate is valid then connects to client.

After which, server uses secret key shared by client to send the authenticated message- “ The OTP for transferring Rs 1,00,000 to your friend’s account is 256345.” to client. Server has used sha hash function to create HMAC.

### **my\_client.py:**

Client first generates it’s RSA key pair and then open a socket `ttp_Socket` to connect with TTP and ask for certificate for its public keys. After receiving digital certificate from TTP , client open a new socket to connect with server and ask for its digital certificate. After receiving certificate from server , client checks the validation and expiration of certificate. After certificate is verified, client generates it’s secret key which is a random key and shares it with server. Server use this shared secret key to Generate a MAC of message, then sends this MAC to client. After receiving message from server, client verifies the MAC, and extracts the message. After which “Message received” is printed on terminal.

### **SECURITY:**

The TLS prototype is secured against Replay attack, Man in the middle attack, Downgrade service attack.

Since signature/HMAC is used for creating message authentication code, also public keys of server, clients are authenticated using their x.509 certificates, thus the prototype is secured against Man in middle attack.

Also for creating sockets, ssl sockets are used which uses session ID which is a randomly generated unique identifier for a session, also client\_hello, server\_hello messages uses random number nonce ,hence prototype is secured against Replay attacks.

Downgrade attacks can break the security of system by allowing the attacker to negotiate the use of lower version of TLS. I have set the ssl\_version as PROTOCOL\_TLSv1\_2 ie: TLS 1.2 , Hence downgrade attacks are not possible on this prototype.

### **COMMUNICATION COSTS:**

Only things that are communicated between server and client in this prototype involves,

- 1.server\_hello, client\_hello
- 2.sharing of certificates between client, server
- 3.key exchange protocol
- 4.Cipher spec protocol
- 4.secret key shared between server, client for exchanging message
- 5.Message that is delivered from server to client.

So communication costs involves Total cost of communicating above things.

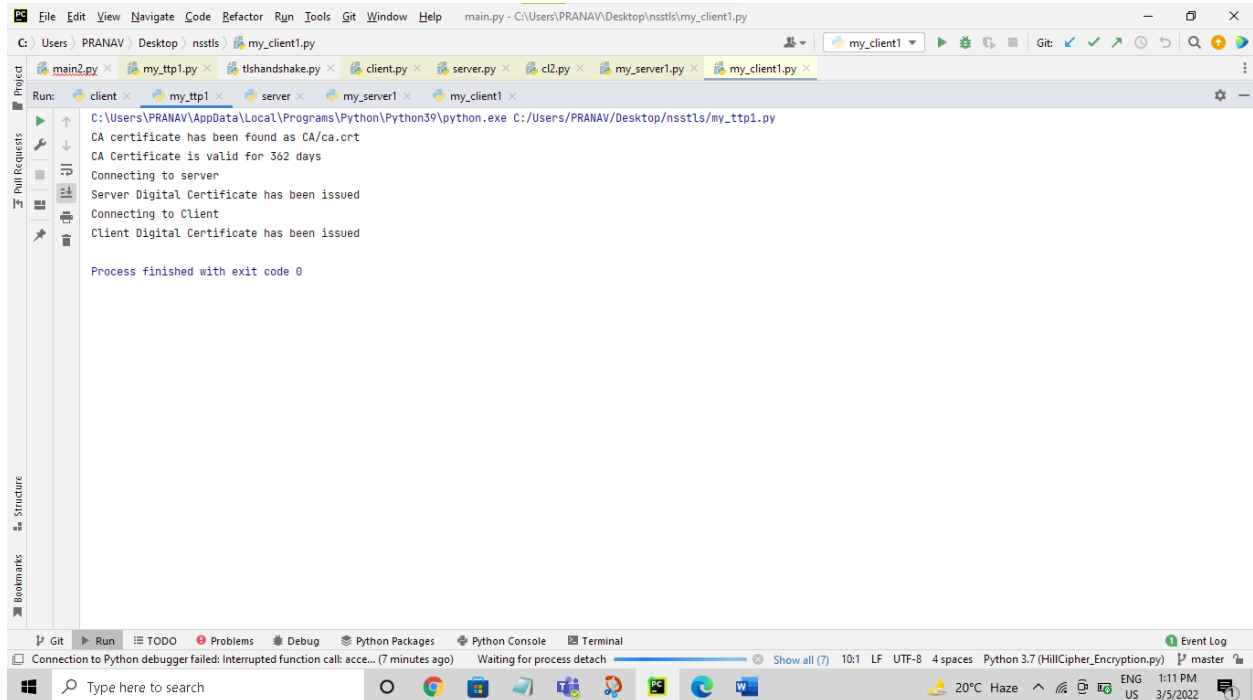
### **COMPUTATIONAL COSTS:**

Major number of computations are done using AES cipher such as it is used for encryption, decryption by server, client, also for exchanging key and sending message AES is used.

And RSA, asymmetric algorithm is used by TTP for creating certificates.

AES, symmetric algorithm is less computationally expensive then RSA, asymmetric algorithm.

# RESULTS:

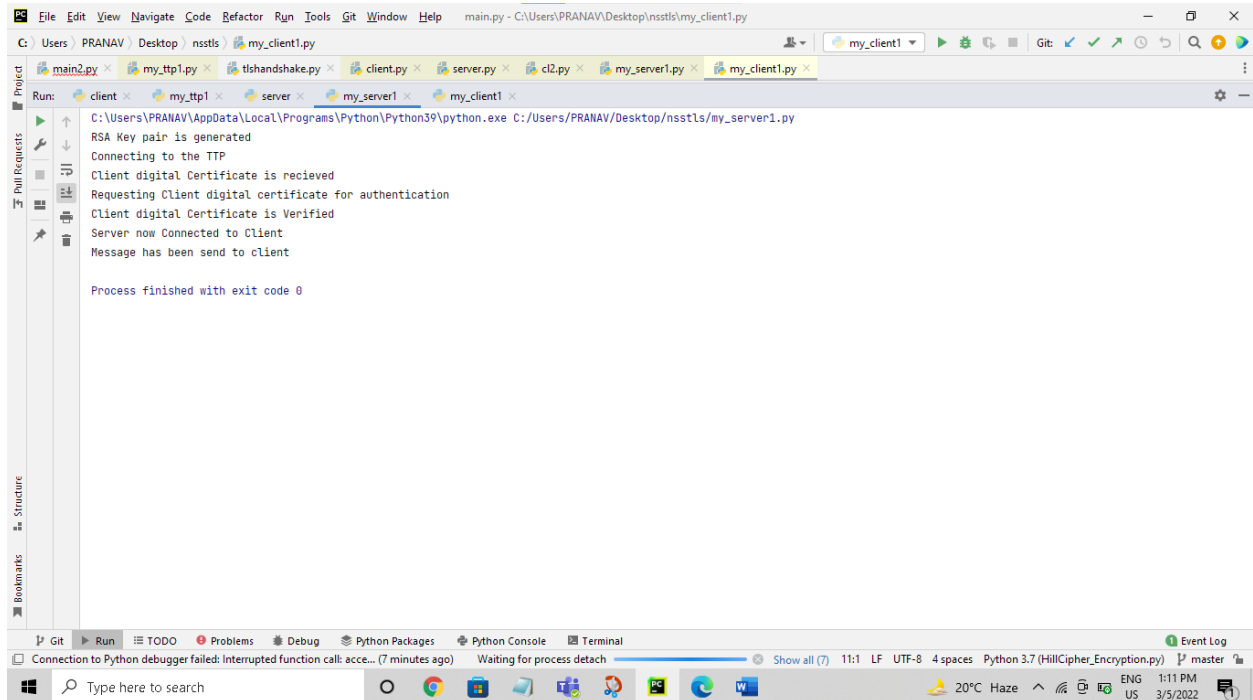


The screenshot shows the Visual Studio Code interface with the file explorer on the left displaying a project named 'nsstls' containing files like 'main2.py', 'my\_ttp1.py', 'tshandshake.py', 'client.py', 'server.py', 'cl2.py', 'my\_server1.py', and 'my\_client1.py'. The 'Run' panel on the right shows the execution of 'my\_ttp1.py' using the command 'C:\Users\PRANAV\AppData\Local\Programs\Python\Python39\python.exe C:/Users/PRANAV/Desktop/nsstls/my\_ttp1.py'. The output log displays the following messages: 'CA certificate has been found as CA/ca.crt', 'CA Certificate is valid for 362 days', 'Connecting to server', 'Server Digital Certificate has been issued', 'Connecting to Client', 'Client Digital Certificate has been issued', and 'Process finished with exit code 0'. The status bar at the bottom indicates the file is 'my\_client1.py' and the Python interpreter is 'Python 3.7 (HillCipher\_Encryption.py)'.

```
C:\Users\PRANAV\AppData\Local\Programs\Python\Python39\python.exe C:/Users/PRANAV/Desktop/nsstls/my_ttp1.py
CA certificate has been found as CA/ca.crt
CA Certificate is valid for 362 days
Connecting to server
Server Digital Certificate has been issued
Connecting to Client
Client Digital Certificate has been issued

Process finished with exit code 0
```

FIG1:my\_ttp.py running

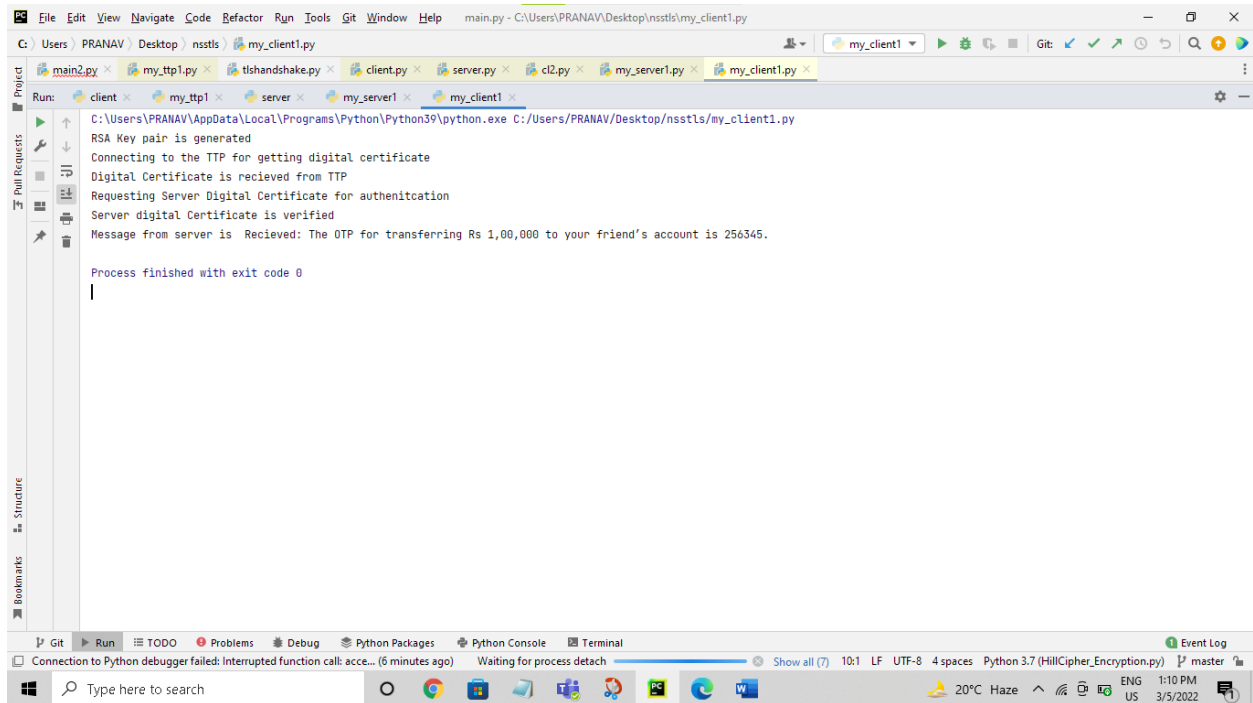


The screenshot shows the Visual Studio Code interface with the file explorer on the left displaying the same project. The 'Run' panel on the right shows the execution of 'my\_server1.py' using the command 'C:\Users\PRANAV\AppData\Local\Programs\Python\Python39\python.exe C:/Users/PRANAV/Desktop/nsstls/my\_server1.py'. The output log displays the following messages: 'RSA Key pair is generated', 'Connecting to the TTP', 'Client digital Certificate is recieved', 'Requesting Client digital certificate for authentication', 'Client digital Certificate is Verified', 'Server now Connected to Client', 'Message has been send to cClient', and 'Process finished with exit code 0'. The status bar at the bottom indicates the file is 'my\_client1.py' and the Python interpreter is 'Python 3.7 (HillCipher\_Encryption.py)'.

```
C:\Users\PRANAV\AppData\Local\Programs\Python\Python39\python.exe C:/Users/PRANAV/Desktop/nsstls/my_server1.py
RSA Key pair is generated
Connecting to the TTP
Client digital Certificate is recieved
Requesting Client digital certificate for authentication
Client digital Certificate is Verified
Server now Connected to Client
Message has been send to cClient

Process finished with exit code 0
```

FIG2 :my\_server.py running



**FIG3:my\_client.py running**