Chat Application Report

- First we set the socket for client and server

CLIENT SOCKET

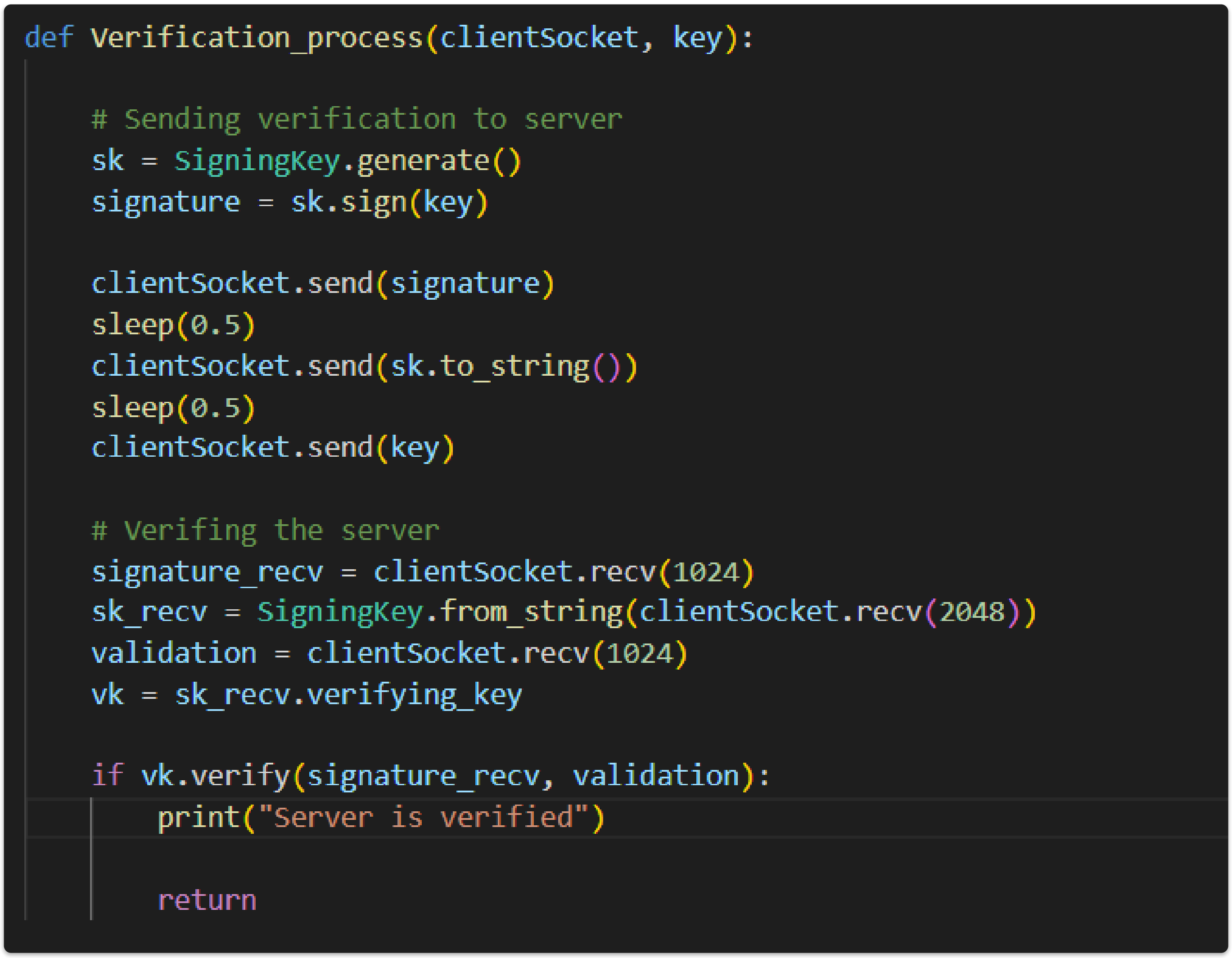
```
f __name__ == "__main__":
    Target_IP = "127.0.0.1"
    Port = 12345
    clientSocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    clientSocket.connect((Target_IP, Port))
```

SERVER SOCKET

```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
ip = '127.0.0.1'
port = 12345
s.bind((ip, port))
s.listen()
print ("socket is listening")

client, addr = s.accept()
print ('Got connection from', addr )

key = Verification_process(client)
f = Fernet(key)
```

- Second we use the ECDSA to verify the client and server
- We send the certificate from client to server for verification and then send the certificate from server to client for verification

CLIENT VERIFICATION

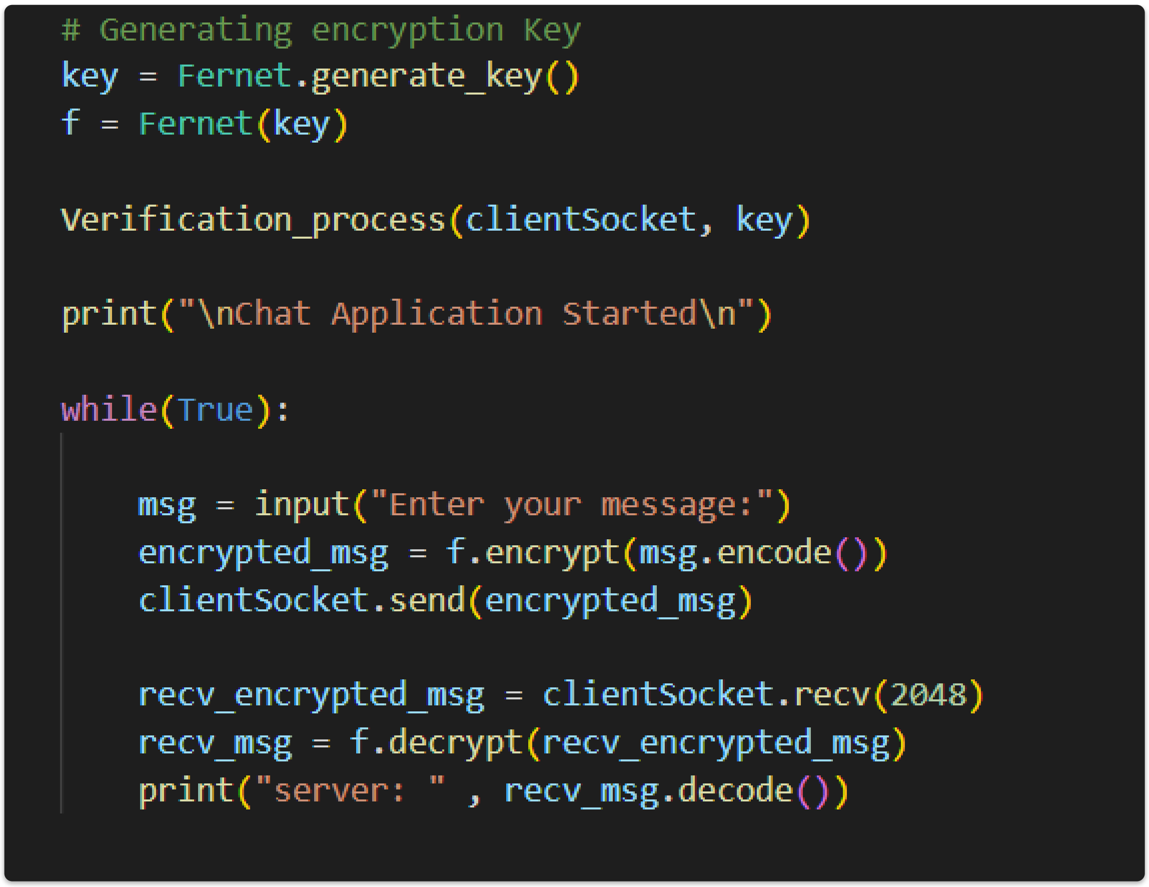


```python
def Verification_process(clientSocket, key):

    # Sending verification to server
    sk = SigningKey.generate()
    signature = sk.sign(key)

    clientSocket.send(signature)
    sleep(0.5)
    clientSocket.send(sk.to_string())
    sleep(0.5)
    clientSocket.send(key)

    # Verifing the server
    signature_recv = clientSocket.recv(1024)
    sk_recv = SigningKey.from_string(clientSocket.recv(2048))
    validation = clientSocket.recv(1024)
    vk = sk_recv.verifying_key

    if vk.verify(signature_recv, validation):
        print("Server is verified")

        return
```

SERVER VERIFICATION

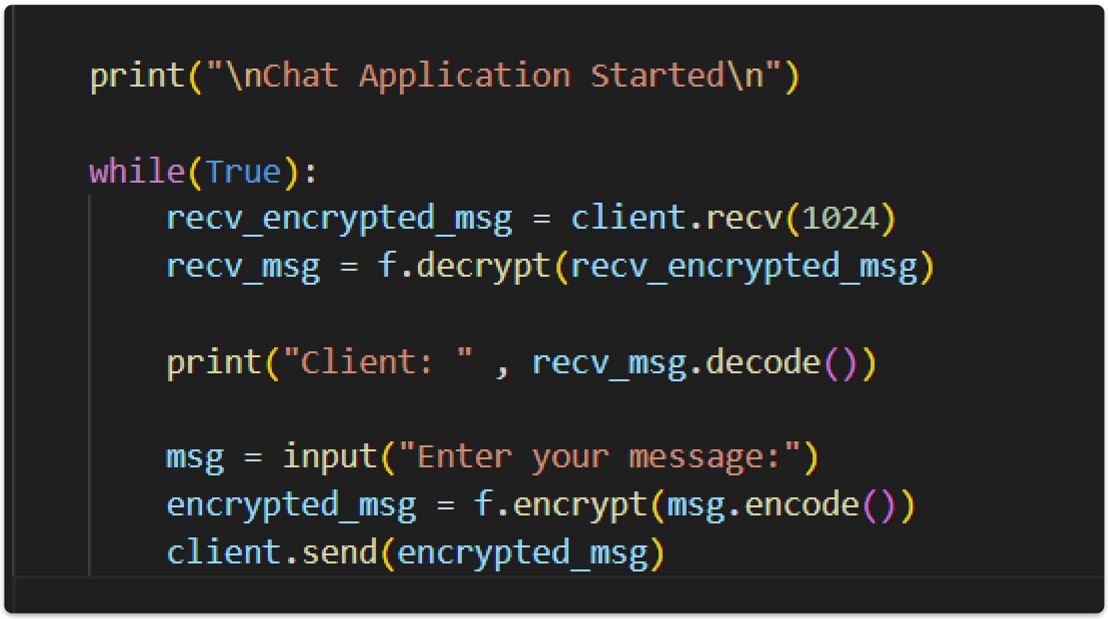
```python
def Verification_process(client):
    signature_recv = client.recv(1024)
    sk_recv = SigningKey.from_string(client.recv(2048))
    key = client.recv(1024)
    vk = sk_recv.verifying_key

    if vk.verify(signature_recv, key):
        print("Client is verified")

        # Sending verification to Client
        sk = SigningKey.generate()
        signature = sk.sign(b"Received")

        client.send(signature)
        sleep(0.5)
        client.send(sk.to_string())
        sleep(0.5)
        client.send(b"Received")

        return key
```

- Third we will share the key and encrypt the message and send to server
- If server send the message, client will decrypt the message and display it.



```python
# Generating encryption Key
key = Fernet.generate_key()
f = Fernet(key)

Verification_process(clientSocket, key)

print("\nChat Application Started\n")

while(True):

    msg = input("Enter your message:")
    encrypted_msg = f.encrypt(msg.encode())
    clientSocket.send(encrypted_msg)

    recv_encrypted_msg = clientSocket.recv(2048)
    recv_msg = f.decrypt(recv_encrypted_msg)
    print("server: " , recv_msg.decode())
```

- Fourth Encrypt the message and send to client
- If client send the message, server will decrypt the message and display it.



```python
print("\nChat Application Started\n")

while(True):
    recv_encrypted_msg = client.recv(1024)
    recv_msg = f.decrypt(recv_encrypted_msg)

    print("Client: " , recv_msg.decode())

    msg = input("Enter your message:")
    encrypted_msg = f.encrypt(msg.encode())
    client.send(encrypted_msg)
```