Name: Tooba Khan
Entry Number: 2021JCS2245
Report for Assignment 3 of SIL765
Readme for Problem 1.

Task 1:

1. The cipher being used is: ('ECDHE-RSA-AES256-GCM-SHA384', 'TLSv1.2', 256)

2. The certificate is:

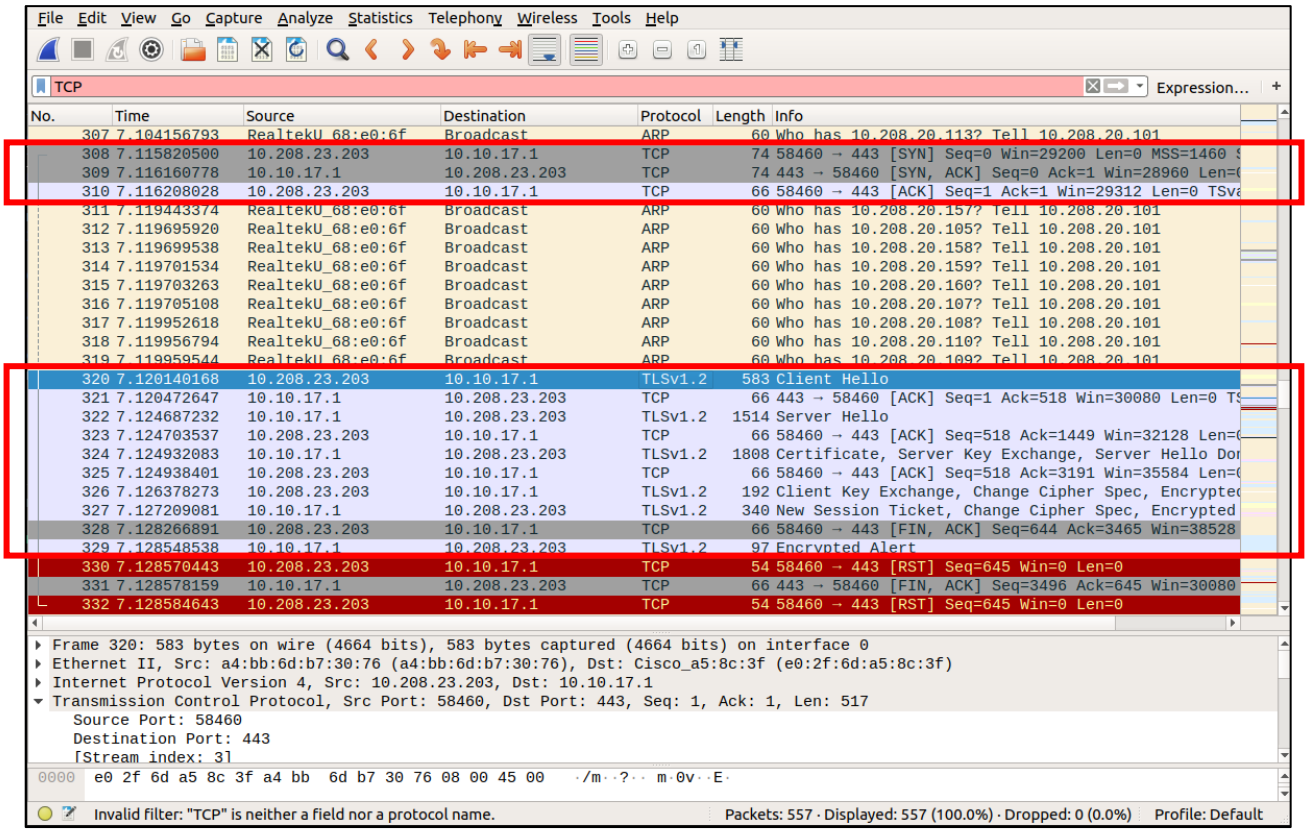
```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

(base) tooba@vr9-Precision-7820-Tower:~/Documents/Nss-3$ python cert.py moodle.iitd.ac.in 443 >> output1.txt
(base) tooba@vr9-Precision-7820-Tower:~/Documents/Nss-3$ python cert.py moodle.iitd.ac.in 443
TCP connection established.

Q.2 The server certifcate:
{'OCSP': ('http://ocsp.globalsign.com/gsrsaovsslca2018',),
 'caIssuers': ('http://secure.globalsign.com/cacert/gsrsaovsslca2018.crt',),
 'issuer': ((('countryName', 'BE'),),
            (('organizationName', 'GlobalSign nv-sa'),),
            (('commonName', 'GlobalSign RSA OV SSL CA 2018'),)),
 'notAfter': 'Sep 27 11:06:03 2022 GMT',
 'notBefore': 'Aug 26 11:06:03 2021 GMT',
 'serialNumber': '69E5A55860CB4DC21625A43C',
 'subject': ((('countryName', 'IN'),),
             (('stateOrProvinceName', 'Delhi'),),
             (('localityName', 'New Delhi'),),
             (('organizationName', 'Indian Institute of Technology Delhi'),),
             (('commonName', '*.iitd.ac.in'),)),
 'subjectAltName': (('DNS', '*.iitd.ac.in'),
                    ('DNS', 'www.iitd.ac.in'),
                    ('DNS', 'iitd.ac.in')),
 'version': 3}
(base) tooba@vr9-Precision-7820-Tower:~/Documents/Nss-3$
```

3. /etc/ssl/certs stores system certificates. This is the location for Linux/Ubuntu systems wehere certificates for different servers are stores.
These are SSL certificates for Certification authorities and servers. It can be updated using command: update-ca-certificates.
It stores certificates and ca-certificates.crt, a concatenated single-file list of certificates.

4. **Wireshark Output:**



It is clear from the wireshark output that first TCP handshake is completed(line 308-310) and the TLS handshake begins with a client hello message in line 320.
TCP handshake is followed by TLS handshake. This means that first TCP handshake is completed and then TLS handshake begins. The TLS handshake is treated as application data by TCP. TCP does not concern itself with TLS handshake and it's details. TCP connection established.

Task 2:

When folder is changed to /certs, I got the following error:

```
(base) tooba@vr9-Precision-7820-Tower:~/Documents/Nss-3$ mkdir certs
(base) tooba@vr9-Precision-7820-Tower:~/Documents/Nss-3$ python q2.py moodle.iitd.ac.in 443
TCP connection established.
Traceback (most recent call last):
  File "/home/tooba/Documents/Nss-3/q2.py", line 19, in <module>
    ssock.do_handshake() # Start the handshake
  File "/home/tooba/anaconda3/lib/python3.9/ssl.py", line 1309, in do_handshake
    self._sslobj.do_handshake()
ssl.SSLCertVerificationError: [SSL: CERTIFICATE_VERIFY_FAILED] certificate verify failed: unable to get local issuer certificate (_ssl.c:1129)
(base) tooba@vr9-Precision-7820-Tower:~/Documents/Nss-3$
```

After copying the CA certificate and generating hash and renaming, the program was able to talk to the server.
CA certificate was: GlobalSign_Root_CA_-_R3.pem
It's generated hash value was: 062cdee6

```
q2.py   output1.txt   cert.py
q2.py > ...
#!/usr/bin/python3
import socket, ssl, sys, pprint
hostname = sys.argv[1]
port = int(sys.argv[2])
cadir = './certs'
# Set up the TLS context
context = ssl.SSLContext(ssl.PROTOCOL_TLS_CLIENT)
context.load_verify_locations(capath=cadir)
context.verify_mode = ssl.CERT_REQUIRED
context.check_hostname = True

sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.connect((hostname, port))
print("TCP connection established.")
```

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

(base) tooba@vr9-Precision-7820-Tower:~/Documents/Nss-3$ openssl x509 -in certs/GlobalSign_Root_CA_-_R3.pem -noout -subject_hash
062cdee6
(base) tooba@vr9-Precision-7820-Tower:~/Documents/Nss-3$ cd certs
(base) tooba@vr9-Precision-7820-Tower:~/Documents/Nss-3/certs$ ls -l
total 4
lrwxrwxrwx 1 tooba tooba 62 Mar  8 18:02 GlobalSign_Root_CA_-_R3.pem -> /usr/share/ca-certificates/mozilla/GlobalSign_Root_CA_-_R3.crt
(base) tooba@vr9-Precision-7820-Tower:~/Documents/Nss-3/certs$ ln -s GlobalSign_Root_CA_-_R3.pem 062cdee6.0
(base) tooba@vr9-Precision-7820-Tower:~/Documents/Nss-3/certs$ cd ..
(base) tooba@vr9-Precision-7820-Tower:~/Documents/Nss-3$ python3 q2.py moodle.iitd.ac.in 443
TCP connection established.
{'OCSP': ('http://ocsp.globalsign.com/gsrsaovsslca2018',),
 'caIssuers': ('http://secure.globalsign.com/cacert/gsrsaovsslca2018.crt',),
 'issuer': ((('countryName', 'BE'),),
            (('organizationName', 'GlobalSign nv-sa'),),
            (('commonName', 'GlobalSign RSA OV SSL CA 2018'),)),
 'notAfter': 'Sep 27 11:06:03 2022 GMT',
 'notBefore': 'Aug 26 11:06:03 2021 GMT',
 'serialNumber': '69E5A55860CB4DC21625A43C',
 'subject': ((('countryName', 'IN'),),
             (('stateOrProvinceName', 'Delhi'),),
             (('localityName', 'New Delhi'),),
             (('organizationName', 'Indian Institute of Technology Delhi'),),
             (('commonName', '*.iitd.ac.in'),)),
 'subjectAltName': (('DNS', '*.iitd.ac.in'),
                    ('DNS', 'www.iitd.ac.in'),
                    ('DNS', 'iitd.ac.in')),
 'version': 3}
Handshake done.
(base) tooba@vr9-Precision-7820-Tower:~/Documents/Nss-3$
```

Part 3:

Dig command gives:

tls_client.py

```python
#!/usr/bin/python3
import socket, ssl, sys, pprint
from turtle import clear
hostname = sys.argv[1]
port = int(sys.argv[2])
cadir = '/etc/ssl/certs'
# Set up the TLS context
context = ssl.SSLContext(ssl.PROTOCOL_TLS_CLIENT)
context.load_verify_locations(capath=cadir)
context.verify_mode = ssl.CERT_REQUIRED
context.check_hostname = False
# print("List of all supported Ciphers:",context.get_ciphers())
# Create TCP connection
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.connect((hostname, port))
print("TCP connection established.")
# Add the TLS
ssock = context.wrap_socket(sock, server_hostname=hostname,do_handshake_on_connect=False)
ssock.do_handshake() # Start the handshake
# print("Certificates to be copied to the folder are:",context.get_ca_certs()) #To get the
print("Handshake done.")
# print("Answers for Part 1:")
# print("\n1 The cipher being used is: ",ssock.cipher())
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
(base) tooba@vr9-Precision-7820-Tower:~/Documents/Nss-3$ dig example.com

; <<>> DiG 9.11.3-1ubuntu1-Ubuntu <<>> example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 30262
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;example.com.                   IN      A

;; ANSWER SECTION:
example.com.            1384    IN      A       93.184.216.34

;; Query time: 1 msec
;; SERVER: 127.0.0.53#53(127.0.0.53)
;; WHEN: Tue Mar 08 19:13:37 IST 2022
;; MSG SIZE  rcvd: 56

(base) tooba@vr9-Precision-7820-Tower:~/Documents/Nss-3$
```

For both the values i.e True and False, following are the observations:

tls_client.py > ...

```python
#!/usr/bin/python3
import socket, ssl, sys, pprint
import requests
from turtle import clear
hostname = sys.argv[1]
port = int(sys.argv[2])
cadir = '/etc/ssl/certs'
# Set up the TLS context
context = ssl.SSLContext(ssl.PROTOCOL_TLS_CLIENT)
context.load_verify_locations(capath=cadir)
context.verify_mode = ssl.CERT_REQUIRED
context.check_hostname = False
print("Checking host name: ",context.check_hostname)
# print("List of all supported Ciphers:",context.get_ciphers())
# Create TCP connection
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.connect((hostname, port))
print("TCP connection established.")
# Add the TLS
ssock = context.wrap_socket(sock, server_hostname=hostname,do_handshake_on_connect=False)
ssock.do_handshake() # Start the handshake
# print("Certificates to be copied to the folder are:",context.get_ca_certs()) #To get the information about certificates.
print("Handshake done.")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL bash

```
(base) tooba@vr9-Precision-7820-Tower:~/Documents/Nss-3$ python3 tls_client.py www.example2020.com 443
Checking host name:  True
TCP connection established.
Traceback (most recent call last):
  File "/home/tooba/Documents/Nss-3/tls_client.py", line 21, in <module>
    ssock.do_handshake() # Start the handshake
  File "/home/tooba/anaconda3/lib/python3.9/ssl.py", line 1309, in do_handshake
    self._sslobj.do_handshake()
ssl.SSLCertVerificationError: [SSL: CERTIFICATE_VERIFY_FAILED] certificate verify failed: Hostname mismatch, certificate is not valid for 'www.example2020.com'. (_ssl.c:1129)
(base) tooba@vr9-Precision-7820-Tower:~/Documents/Nss-3$ python3 tls_client.py www.example2020.com 443
Checking host name:  False
TCP connection established.
Handshake done.
(base) tooba@vr9-Precision-7820-Tower:~/Documents/Nss-3$
```

1. When check_hostname is set to True:

tls_client.py > ...

```python
#!/usr/bin/python3
import socket, ssl, sys, pprint
import requests
from turtle import clear
hostname = sys.argv[1]
port = int(sys.argv[2])
cadir = '/etc/ssl/certs'
# Set up the TLS context
context = ssl.SSLContext(ssl.PROTOCOL_TLS_CLIENT)
context.load_verify_locations(capath=cadir)
context.verify_mode = ssl.CERT_REQUIRED
context.check_hostname = True
print("Checking host name: ",context.check_hostname)
# print("List of all supported Ciphers:",context.get_ciphers())
# Create TCP connection
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.connect((hostname, port))
print("TCP connection established.")
# Add the TLS
ssock = context.wrap_socket(sock, server_hostname=hostname,do_handshake_on_connect=False)
ssock.do_handshake() # Start the handshake
# print("Certificates to be copied to the folder are:",context.get_ca_certs()) #To get the information about certificates.
print("Handshake done.")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
(base) tooba@vr9-Precision-7820-Tower:~/Documents/Nss-3$ python3 tls_client.py www.example2020.com 443
Checking host name:  True
TCP connection established.
Traceback (most recent call last):
  File "/home/tooba/Documents/Nss-3/tls_client.py", line 21, in <module>
    ssock.do_handshake() # Start the handshake
  File "/home/tooba/anaconda3/lib/python3.9/ssl.py", line 1309, in do_handshake
    self._sslobj.do_handshake()
ssl.SSLCertVerificationError: [SSL: CERTIFICATE_VERIFY_FAILED] certificate verify failed: Hostname mismatch, certificate is not valid for 'www.example2020.com'. (_ssl.c:1129)
(base) tooba@vr9-Precision-7820-Tower:~/Documents/Nss-3$
```

2. When check_hostname is set to False:

```
tls_client.py > ...
#!/usr/bin/python3
import socket, ssl, sys, pprint
import requests
from turtle import clear
hostname = sys.argv[1]
port = int(sys.argv[2])
cadir = '/etc/ssl/certs'
# Set up the TLS context
context = ssl.SSLContext(ssl.PROTOCOL_TLS_CLIENT)
context.load_verify_locations(capath=cadir)
context.verify_mode = ssl.CERT_REQUIRED
context.check_hostname = False
print("Checking host name: ",context.check_hostname)
# print("List of all supported Ciphers:",context.get_ciphers())
# Create TCP connection
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.connect((hostname, port))
print("TCP connection established.")
# Add the TLS
ssock = context.wrap_socket(sock, server_hostname=hostname,do_handshake_on_connect=False)
ssock.do_handshake() # Start the handshake
# print("Certificates to be copied to the folder are:",context.get_ca_certs()) #To get the inf
print("Handshake done.")
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

(base) tooba@vr9-Precision-7820-Tower:~/Documents/Nss-3$ python3 tls_client.py www.example2020.com 443
Checking host name:  False
TCP connection established.
Handshake done.
(base) tooba@vr9-Precision-7820-Tower:~/Documents/Nss-3$
```

Etc/hosts file contains IP addresses of all hosts on the local network. It is used to resolve a hostname into IP address.

The above error is because a wrong hostname has been associated with the IP address of example.com in the hosts file. TCP connection got established with the IP address specified in the hosts file but failed to verify the associated hostname because it was a wrong hostname.

Hostname check is crucial because it checks whether the hostname matches any of the hostnames associated with the certificate. It verifies the server's identity.

Certificate validation depends on hostname verification that name of server matches one of the IDs in "Common name" or extension fields. If the hostname verification is turned off or is faulty, certificate validation and security guarantees of SSL/TLS will be compromised.

Part 4:

After adding code for sending and receiving HTTP requests and responses, the following output was generated:

tls_client.py

```python
#Send HTTP Request to Server
request = b"GET / HTTP/1.0\r\nHost: " + hostname.encode() + b"\r\n\r\n"
ssock.sendall(request)
# Read HTTP Response from Server
response = ssock.recv(2048)
while response:
    pprint.pprint(response.split(b"\r\n"))
    response = ssock.recv(2048)

ssock.shutdown(socket.SHUT_RDWR)
ssock.close()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
(base) tooba@vr9-Precision-7820-Tower:~/Documents/Nss-3$ python3 tls_client.py moodle.iitd.ac.in 443
TCP connection established.
Handshake done.
[b'HTTP/1.0 303 See Other',
 b'Date: Tue, 08 Mar 2022 13:31:44 GMT',
 b'Server: Apache/2.4.18 (Ubuntu)',
 b'Set-Cookie: MoodleSession=dkj6mfjf0ifij5vac5l0btjqc0; path=/',
 b'Expires: Thu, 19 Nov 1981 08:52:00 GMT',
 b'Cache-Control: no-store, no-cache, must-revalidate',
 b'Pragma: no-cache',
 b'Location: https://moodle.iitd.ac.in/login/index.php',
 b'Content-Language: en',
 b'Content-Length: 439',
 b'Connection: close',
 b'Content-Type: text/html; charset=utf-8',
 b'',
 b'']
[b'<!DOCTYPE html>\n<html  lang="en" xml:lang="en">\n<head>\n<meta http-equiv='
 b'"Content-Type" content="text/html; charset=utf-8" />\n\n<title>Redirect</t'
 b'itle>\n</head><body><div style="margin-top: 3em; margin-left:auto; margin'
 b'-right:auto; text-align:center;">This page should automatically redirect. If'
 b' nothing is happening please use the continue link below.<br /><a href="http'
 b's://moodle.iitd.ac.in/login/index.php">Continue</a></div></body></html>']
(base) tooba@vr9-Precision-7820-Tower:~/Documents/Nss-3$
```

<u>For fetching an image:</u>

1. I added the following code to the existing code:

```python
file = "/wikipedia/commons/3/39/Awful_wreck_of_the_steam_packet_Home.jpg"

request = b"GET "+file.encode(encoding='utf-8')+b" HTTP/1.1\r\nHost: " + \
hostname.encode(encoding='utf-8') + b"\r\n\r\n"
ssock.sendall(request)

print("The request sent is: ",request)

# Read HTTP Response from Server
response = ssock.recv(2048)
print("\n\nImage retrieved. Printing some headers in the response: ",response.split(b"\r\n")[:10])
while response:
    response = ssock.recv(2048)

ssock.shutdown(socket.SHUT_RDWR)
ssock.close()
print("\nConnection closed.\n")
```

2. Terminal screenshot:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL                                                    bash

(base) tooba@vr9-Precision-7820-Tower:~/Documents/Nss-3$ python3 tls_client.py upload.wikimedia.org 443
Checking host name:  True
TCP connection established.
Handshake done.
The request sent is:  b'GET /wikipedia/commons/3/39/Awful_wreck_of_the_steam_packet_Home.jpg HTTP/1.1\r\nHost: upload.wikimedia.org\r\n\r\n'


Image retrieved. Printing some headers in the response:  [b'HTTP/1.1 200 OK', b'date: Tue, 08 Mar 2022 19:57:10 GMT', b'etag: 0b0f52e6e69fc6d25511b1bbe33ad480', b'server: ATS/8.0.8', b'content-type
: image/jpeg', b'content-length: 155812', b'x-object-meta-sha1base36: c9qkmbg125v0ozpo7p2dle5vxmdjc8m', b'last-modified: Sat, 21 Jan 2017 20:12:09 GMT', b'age: 35483', b'x-cache: cp5003 hit, cp5006
 hit/3685']

Connection closed.

(base) tooba@vr9-Precision-7820-Tower:~/Documents/Nss-3$
```

How to run the code:

Execute: python3 tls_client.py upload.wikimedia.org 443

The program takes two arguments i.e hostname and port number.

Security aspects of https:

1. It provides encryption of data being exchanged.
2. https also provides server identity verification.
3. It is a combination of http and SSL/TLS, so it inherits all the security aspects of TLS.
4. It is secure from replay attacks by using nonces.
5. Certificates are validated and even the hostname checks are performed so that no one can forge an identity.
6. It also verifies the server with which client is communicating so that client can be assured of a secure communication.