

Lab3 (Report) – Problem-1

**VATSAL AGRAWAL**

**Master Of Technology  
COMPUTER SCIENCE and ENGINEERING  
2nd Sem**



**IIT Delhi**

Indian Institute of Technology Delhi

Entry Number - 2021MCS2157

**NETWORK &  
SYSTEM SECURITY  
SIL 765**

# Introduction

My programme is working correctly. The screenshot of working is below. I have read a python doc about SSL for help in solving documents. I have performed all tasks, and it's all part of all questions. I have used some info same from my previous submitted assignment to this course.

I have changed and added some of the code in the source code given to make all tasks work. The basic idea of solving code can also be seen in python docs. I have also provided analysis in the end as well as for each part.

The time for running the whole program is close to 1 second in real-time. We can see different tasks running by giving CLI arguments to our program- Thus, code is not hardcoded. I have tried three sites. w3schools and path of the image with check hostname as True is the default in our setup

1. [www.google.com](http://www.google.com);
2. [www.python.org](http://www.python.org);
3. [www.w3schools.com](http://www.w3schools.com) (for image getting) and
4. python.org

## INDEX

1. Task1
2. Task2
3. Task3
4. Task4
5. Analysis
6. Pre-Requisite
  - a. How To Make Executable and Run Programme
  - b. Structure Of Programme
7. References

# Problem 1

## Tasks 1

### 1. What is the cipher used between the client and the server?

Cipher Algo and TLS version and number of secret bit is

❖ For [www.google.com](http://www.google.com)

`TLS_AES_256_GCM_SHA384', 'TLSv1.3', 256`

❖ For [www.python.org](http://www.python.org)

`TLS_AES_256_GCM_SHA384', 'TLSv1.3', 256`

❖ For python.org / www.w3schools.com

`TLS_AES_256_GCM_SHA384', 'TLSv1.3', 256`

This shows us the current standard for protection for accessing HTTPS websites is with TLS version 1.3 and uses a size of 256 bit. The cipher used is AES GCM with 256-bit keys and 384 size bit hash in communication. SHA 384 provides a longer digest and also works faster than 256-bit hash. This task is printed in every configuration of passing CLI arguments.

I have used `.cipher()` to get cipher.

AES GCM pros are:-

- Lower encryption overheads
- Do both encryption and authentication
- Processed in parallel; faster than the combined effect of ctr and hashing
- Save from CPA and CCA (chosen ciphertext attack)

```
After getpercert. Press any key to continue .....  
( 'TLS_AES_256_GCM_SHA384', 'TLSv1.3', 256 )  
After cipher. Press any key to continue .....
```

Fig1. Cipher-Suite

### 2. Please print out the server certificate in the program.

The certificate of [www.google.com](http://www.google.com) is

The certificate displayed contains issuer info of certificate with country and other info. It contains the range of dates for which the certificate is valid (3 months approx.). It contains the server name and alternate name, along with DNS and version. It also contains other info like ca Issuers, CRL distribution points, etc. This all provides help to verify the server and see that hostnames

and time and other info needed for security are provided and valid. This task is printed in every configuration of passing CLI arguments.

I have used `getpeercert()` to get this

```
{'OCSP': ('http://ocsp.pki.goog/gts1c3',),
 'caIssuers': ('http://pki.goog/repo/certs/gts1c3.der',),
 'crlDistributionPoints': ('http://crls.pki.goog/gts1c3/Q0vJ0N1sT2A.crl',),
 'issuer': (((('countryName', 'US'),),
               (('organizationName', 'Google Trust Services LLC'),),
               (('commonName', 'GTS CA 1C3'),)),),
 'notAfter': 'May 12 11:32:41 2022 GMT',
 'notBefore': 'Feb 17 11:32:42 2022 GMT',
 'serialNumber': '097832DE3A58281C0A00000001378A25',
 'subject': (((('commonName', 'www.google.com'),),),),
 'subjectAltName': (('DNS', 'www.google.com'),),
 'version': 3}
```

```
After making TCP connection. Press any key to continue .....
{'OCSP': ('http://ocsp.pki.goog/gts1c3',),
 'caIssuers': ('http://pki.goog/repo/certs/gts1c3.der',),
 'crlDistributionPoints': ('http://crls.pki.goog/gts1c3/Q0vJ0N1sT2A.crl',),
 'issuer': (((('countryName', 'US'),),
               (('organizationName', 'Google Trust Services LLC'),),
               (('commonName', 'GTS CA 1C3'),)),),
 'notAfter': 'May 12 11:32:41 2022 GMT',
 'notBefore': 'Feb 17 11:32:42 2022 GMT',
 'serialNumber': '097832DE3A58281C0A00000001378A25',
 'subject': (((('commonName', 'www.google.com'),),),),
 'subjectAltName': (('DNS', 'www.google.com'),),
 'version': 3}
After getpercert. Press any key to continue .....|
```

Fig2. Server Certificate

### 3. Explain the purpose of `/etc/ssl/certs`.

`/etc/ssl/certs` are used to store CA certificates, and these certificates are used to validate the server-side certificate in our case. It also contains hash and .pem files, which is used to find certificate easily and to validate the connection. It can contain a public key for validating.

In our program, we have used `.load_verify_locations()` to load this directory and find the certificate in this using hash.

```
cadir = '/etc/ssl/certs'
context = ssl.SSLContext(ssl.PROTOCOL_TLS_CLIENT)
context.load_verify_locations(cafile="./certs/ca-certificates.crt", capath = cadir)
```

Fig3. Cadir path and loading of certificate code

**4. Use Wireshark to capture the network traffics during the execution of the program and explain your observation. In particular, explain which step triggers the TCP handshake and which step triggers the TLS handshake. Explain the relationship between the TLS handshake and the TCP handshake.**

In the First Step, we have made a TCP connection with the help of socket. connect for port number 443 for [www.python.org](http://www.python.org)

We can see three packet transactions in this, which includes SYN packet and ACK packet from both—this setups TCP handshake. We can also see the packet sequence number in the pic below. We can also see some retransmission and ack for duplicate packet also

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.0.5	199.232.20.223	TCP	66	57164 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
2	0.011680	199.232.20.223	192.168.0.5	TCP	66	443 → 57164 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1360 SACK_PERM=1 WS=512
3	0.011800	192.168.0.5	199.232.20.223	TCP	54	57164 → 443 [ACK] Seq=1 Ack=1 Win=131840 Len=0
4	1.024577	199.232.20.223	192.168.0.5	TCP	66	[TCP Retransmission] 443 → 57164 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1360 SACK_PERM=1 WS=512
5	1.024632	192.168.0.5	199.232.20.223	TCP	66	[TCP Dup ACK 3#1] 57164 → 443 [ACK] Seq=1 Ack=1 Win=131840 Len=0 SLE=0 SRE=1

Fig4. Wireshark output of TCP handshake

This is done with the help of a socket:-

```
sock = socket(AF_INET,SOCK_STREAM)
sock.connect((hostname,port))
```

In the second step, we have made a wrapped socket for authenticated and encrypted transactions between sender and receiver. This enables sharing of certificates and keys.

We can see nine packet transactions in this includes hello packet from the client and hello and ack packet from the server along with change cipher method. We also see several more ack and packet deliveries that may contain keys and certificates and change cipher protocol. This setups TLS handshake. We can also see the packet sequence number along with the TLS version in the pic below.

No.	Time	Source	Destination	Protocol	Length	Info
4	1.032947	192.168.0.5	199.232.20.223	TLSv1	571	Client Hello
5	1.043165	199.232.20.223	192.168.0.5	TCP	54	443 → 59630 [ACK] Seq=1 Ack=518 Win=147456 Len=0
6	1.044964	199.232.20.223	192.168.0.5	TLSv1.3	1414	Server Hello, Change Cipher Spec, Application Data
7	1.045081	192.168.0.5	199.232.20.223	TCP	54	59630 → 443 [ACK] Seq=518 Ack=1361 Win=131840 Len=0
8	1.047056	199.232.20.223	192.168.0.5	TCP	1414	443 → 59630 [PSH, ACK] Seq=1361 Ack=518 Win=147456 Len=1360 [TCP segment of a reassembled PDU]
9	1.047142	192.168.0.5	199.232.20.223	TCP	54	59630 → 443 [ACK] Seq=518 Ack=2721 Win=131840 Len=0
10	1.048980	199.232.20.223	192.168.0.5	TLSv1.3	1030	Application Data, Application Data, Application Data
11	1.050013	192.168.0.5	199.232.20.223	TLSv1.3	134	Change Cipher Spec, Application Data
12	1.060588	199.232.20.223	192.168.0.5	TCP	54	443 → 59630 [ACK] Seq=3697 Ack=598 Win=147456 Len=0

Fig5. Wireshark output of TLS handshake

This is done with the help of the SSL library:-

```
ssock = context.wrap_socket(sock,server_hostname = hostname,do_handshake_on_connect = False)
ssock.do_handshake()
```

In the Third Step, we can see a connection broke request. We can see some TCP fin packet and ack and some TLS packet transferred indicating finish of packet.

13	0.369507	192.168.0.5	199.232.20.223	TCP	54 60934 → 443 [FIN, ACK] Seq=598 Ack=3697 Win=130816 Len=0
14	0.380603	199.232.20.223	192.168.0.5	TCP	54 443 → 60934 [ACK] Seq=3697 Ack=599 Win=147456 Len=0
15	0.380681	199.232.20.223	192.168.0.5	TLSv1.3	78 Application Data
16	0.380747	199.232.20.223	192.168.0.5	TCP	54 443 → 60934 [FIN, ACK] Seq=3721 Ack=599 Win=147456 Len=0

Fig6. Wireshark output of Broke Connection

The difference between TCP and TLS handshake is that – TCP handshake is used to make reliable connection-oriented services. It helps in establishing initials like sequence number and window size, etc., which is needed for connection. In this, both sides send sequence number synchronization packet and acknowledgment. In comparison, TLS is used to establish a secure and authenticated connection. It helps in the exchange of keys and certificates, which can be used for authentication and encryption. TLS handshake happens after TCP handshake.

## Tasks 2

### 1. Create a folder called certs, and assign the cadir to ./certs.

This helps us in identifying the CA certificate used for verification. We have used this folder for every execution of tasks further.

```
cadir = './certs'
context = ssl.SSLContext(ssl.PROTOCOL_TLS_CLIENT)
context.load_verify_locations(cafile="./certs/ca-certificates.crt", capath = cadir)
#context.load_verify_locations(capath = cadir)
context.verify_mode = ssl.CERT_REQUIRED
context.check_hostname = True
```

Fig7. Cadir path

### 2. Run the client program. Since the folder is empty, the program will throw an error. Report your observation and the potential cause.

```
root@Pankhuri:~# time python3 main.py www.python.org
After making TCP connection. Press any key to continue .....
Traceback (most recent call last):
  File "main.py", line 19, in <module>
    ssock.do_handshake()
  File "/usr/lib/python3.8/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:1108)
```

Fig8. Error for empty Cadir folder

We can see the error that verification of server has failed due to reason of CA certificate missing as the folder is empty. SSLContext is set for TLS



protocol for a client with SSL cert required method, So it is necessary to obtain a certificate from the server and verify it. For verification, we need some CA authority, but as the folder is empty thus, no CA authority certificate can be found, and thus no verification can happen. Thus, this leads to error.

If we make a connection without authentication, this may lead to impersonation and other security breaches attacks. The code responsible for this is shown in the first part of this task.

**3. It may be possible to resolve the previous error by placing the corresponding CA's certificate into your certs folder. Please use the client program to find out what CA certificate is needed to verify the server's certificate, and then copy the certificate from the /etc/ssl/certs to the ./certs folder.**

We have copied the CA certificate named ca-certificates.crt to our new folder. Also, we have generated a hash of it and added a symbolic link to it.

*We have also provided the exact ca certificate file using attribute cafile. This helps us in eliminating errors.*

The command we have used is

```
context.load_verify_locations(cafile  
                             = "./certs/ca - certificates.crt",capath = cadir)
```

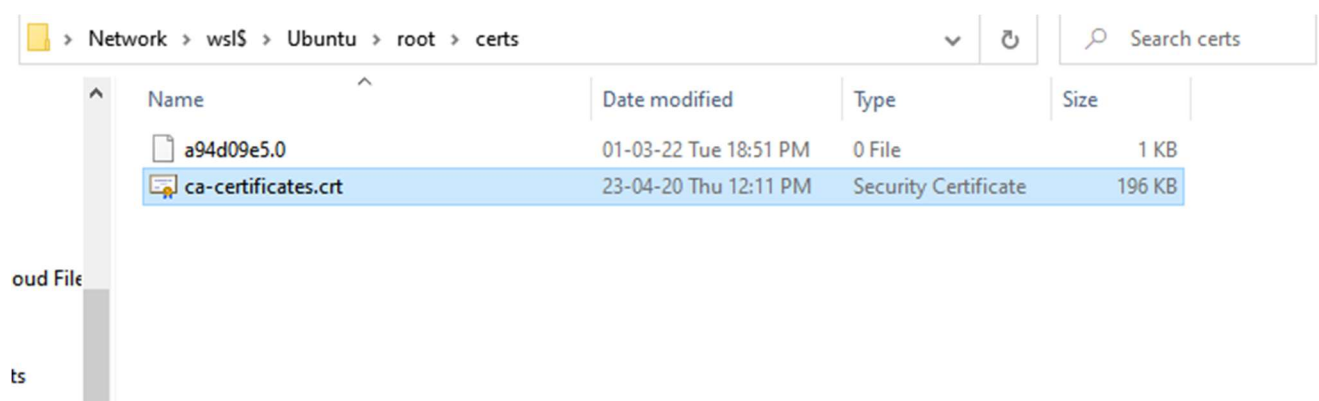


Fig9. Cadir folder

**4. Run the client program again and report the observation. If you have done everything correctly, your client program should be able to talk to the server.**

My client program is working efficiently due to the addition of the ca certificate and add cafile attribute, as shown below. This is because the certificate needed for verification is found by our program efficiently, and it uses that for validation. The running snapshot is for python.org.

I have also printed cadir for proof of changing the folder.

```
root@Pankhuri:~# time python3 main.py www.python.org
Cadir Folder path:
./certs
After making TCP connection. Press any key to continue .....
{'OCSP': ('http://ocsp.globalsign.com/ca/gsatlasr3dvtlscah22021',),
 'caIssuers': ('http://secure.globalsign.com/cacert/gsatlasr3dvtlscah22021.crt',),
 'crlDistributionPoints': ('http://crl.globalsign.com/ca/gsatlasr3dvtlscah22021.crl',),
 'issuer': (((('countryName', 'BE'),),
               (('organizationName', 'GlobalSign nv-sa'),),
               (('commonName', 'GlobalSign Atlas R3 DV TLS CA H2 2021'),)),
 'notAfter': 'Nov 23 18:41:10 2022 GMT',
 'notBefore': 'Oct 22 18:41:11 2021 GMT',
 'serialNumber': '01B7C6CD03E8E071BE48C2B1A7994075',
 'subject': (((('commonName', 'www.python.org'),),),),
 'subjectAltName': (('DNS', 'www.python.org'),
                    ('DNS', '*.python.org'),
                    ('DNS', 'docs.python.org'),
                    ('DNS', 'downloads.python.org'),
                    ('DNS', 'pypi.python.org')),
 'version': 3}
After getpercert. Press any key to continue .....
('TLS_AES_256_GCM_SHA384', 'TLSv1.3', 256)
After cipher. Press any key to continue .....

real    0m0.504s
user    0m0.047s
sys      0m0.063s
```

Fig10. Successful running with new Cadir path

## Tasks 3

**1. Get the IP address of the server using the dig command, such as the following:**

We can use the +shorts attribute with the dig command to get the IP address only. We can also see that sometimes the Ip address of the same site can differ due to aliasing.

```
root@Pankhuri:~# dig www.python.org +short
dualstack.python.map.fastly.net.
199.232.20.223
```

Fig11. Dig short command



IP address is

- ❖ For [www.google.com](http://www.google.com)  
**142.250.193.68**
- ❖ For [www.python.org](http://www.python.org)  
**199.232.20.223**
- ❖ For [www.w3schools.com](http://www.w3schools.com)  
**199.250.206.28**
- ❖ For python.org  
**138.197.63.241**

```
root@Pankhuri:~# dig www.python.org

; <<>> DiG 9.16.1-Ubuntu <<>> www.python.org
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 16547
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
;; QUESTION SECTION:
;www.python.org.                IN      A

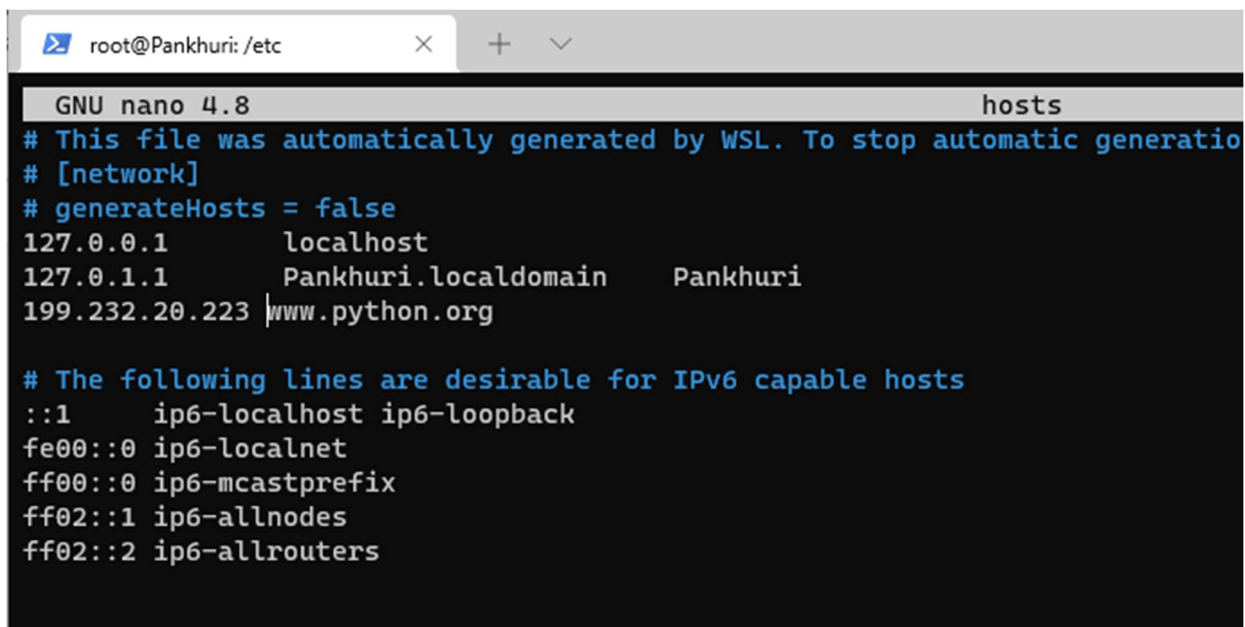
;; ANSWER SECTION:
www.python.org.                604800  IN      CNAME   dualstack.python.map.fastly.net.
dualstack.python.map.fastly.net. 30     IN      A       199.232.20.223

;; Query time: 69 msec
;; SERVER: 1.1.1.1#53(1.1.1.1)
;; WHEN: Wed Mar 02 02:13:45 IST 2022
;; MSG SIZE rcvd: 104
```

Fig12. Dig Full command

**2. Modify the /etc/hosts file add the above entry at the end of the file (the IP address is what you get from the dig command).**

I have modified the hosts' file with the IP address and hostname of [www.python.org](http://www.python.org), as shown below. It provides and helps in fast finding of the mapping of IP address and its hostname. It can be used by the client to match the certificate hostname and to receive an IP address.



```
root@Pankhuri: /etc
GNU nano 4.8 hosts
# This file was automatically generated by WSL. To stop automatic generation
# [network]
# generateHosts = false
127.0.0.1 localhost
127.0.1.1 Pankhuri.localdomain Pankhuri
199.232.20.223 www.python.org

# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

Fig13. Hosts file modified

**3. Switch the following line in the client program between True and False, and then connect your client program to [www.example2020.com](http://www.example2020.com). Describe and explain your observation.**

Turning off the checking of hostname can lead to some other person sending a different certificate, impersonating to be the original server. This can lead to man in middle attack also. It is used to show that hostname provided in the certificate is mapped to the IP we are connecting.

We can see the effect of this when we enter the wrong mapping in our hosts' file. We will see when true our handshake failed and when true it worked. We have not shown this in the screenshot because mapping needs to be done correctly in the hosts' file as stated.

We can see in the picture that when hostname checking is off, then our program works the same as before, but when it is on then, it gives an error when we provide only the Ip address instead of hostname. But when we provided hostname to our client, then it worked the same in both cases. The reason for our failure is hostname is missing to check the hostname. We can pass 2 CLI arguments with the first containing hostname and the second as boolean to check hostname.

The Code which helps us in this is

```
context = ssl.SSLContext(ssl.PROTOCOL_TLS_CLIENT)
context.check_hostname = True
```

```

root@Pankhuri:~# time python3 main.py 199.232.20.223
Cadir Folder path:
./certs
Check Hostname Attribute:
True
After making TCP connection. Press any key to continue .....

Traceback (most recent call last):
  File "main.py", line 23, in <module>
    ssock.do_handshake()
  File "/usr/lib/python3.8/ssl.py", line 1309, in do_handshake
    self._sslobj.do_handshake()
ssl.SSLCertVerificationError: [SSL: CERTIFICATE_VERIFY_FAILED] certificate verify failed: IP address mismatch, certificate is not valid for '199.232.20.223'. (_ssl.c:1108)

```

Fig14. Running with Check Hostname as True with Ip address

```

root@Pankhuri:~# time python3 main.py 199.232.20.223 False
Cadir Folder path:
./certs
Check Hostname Attribute:
False
After making TCP connection. Press any key to continue .....
{'OCSP': ('http://ocsp.globalsign.com/ca/gsatlasr3dvtlscah22021',),
 'caIssuers': ('http://secure.globalsign.com/cacert/gsatlasr3dvtlscah22021.crt',),
 'crlDistributionPoints': ('http://crl.globalsign.com/ca/gsatlasr3dvtlscah22021.crl',),
 'issuer': (((('countryName', 'BE'),),
               (('organizationName', 'GlobalSign nv-sa'),),
               (('commonName', 'GlobalSign Atlas R3 DV TLS CA H2 2021'),)),
 'notAfter': 'Nov 23 18:41:10 2022 GMT',
 'notBefore': 'Oct 22 18:41:11 2021 GMT',
 'serialNumber': '01B7C6CD03E8E071BE48C2B1A7994075',
 'subject': (((('commonName', 'www.python.org'),),),
 'subjectAltName': (('DNS', 'www.python.org'),
                    ('DNS', '*.python.org'),
                    ('DNS', 'docs.python.org'),
                    ('DNS', 'downloads.python.org'),
                    ('DNS', 'pypi.python.org')),
 'version': 3}
After getpercert. Press any key to continue .....
('TLS_AES_256_GCM_SHA384', 'TLSv1.3', 256)
After cipher. Press any key to continue .....
[b'GET / HTTP/1.0', b'Host: 199.232.20.223', b'', b'']

```

Fig15. Running with Check Hostname as True with Ip address

```

root@Pankhuri:~# time python3 main.py www.python.org True
Cadir Folder path:
./certs
Check Hostname Attribute:
True
After making TCP connection. Press any key to continue .....
{'OCSP': ('http://ocsp.globalsign.com/ca/gsatlasr3dvtlscah22021',),
 'caIssuers': ('http://secure.globalsign.com/cacert/gsatlasr3dvtlscah22021.crt',),
 'crlDistributionPoints': ('http://crl.globalsign.com/ca/gsatlasr3dvtlscah22021.crl',),
 'issuer': (((('countryName', 'BE'),),
               (('organizationName', 'GlobalSign nv-sa'),),
               (('commonName', 'GlobalSign Atlas R3 DV TLS CA H2 2021'),)),
 'notAfter': 'Nov 23 18:41:10 2022 GMT',
 'notBefore': 'Oct 22 18:41:11 2021 GMT',
 'serialNumber': '01B7C6CD03E8E071BE48C2B1A7994075',
 'subject': (((('commonName', 'www.python.org'),),),
 'subjectAltName': (('DNS', 'www.python.org'),
                    ('DNS', '*.python.org'),
                    ('DNS', 'docs.python.org'),
                    ('DNS', 'downloads.python.org'),
                    ('DNS', 'pypi.python.org')),
 'version': 3}
After getpercert. Press any key to continue .....
('TLS_AES_256_GCM_SHA384', 'TLSv1.3', 256)
After cipher. Press any key to continue .....

```

Fig16. Running with Check Hostname as True with hostname

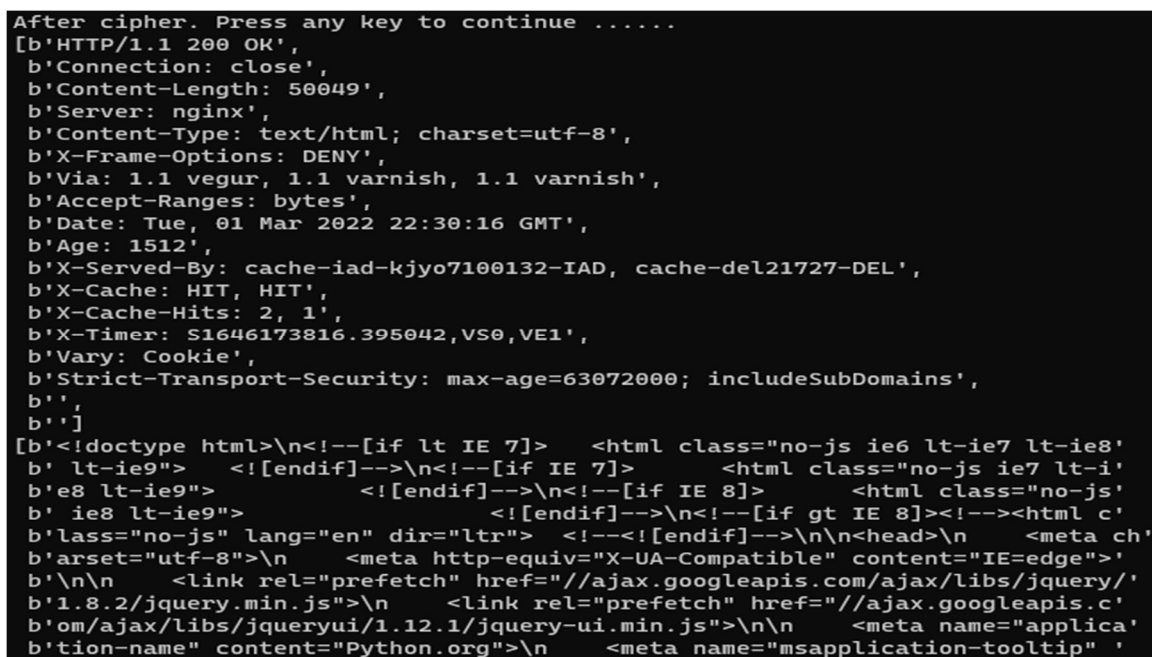
## Tasks 4

**1. Please add the data sending/receiving code to your client program, and report your observation.**

We have sent get request for HTTPS to the server and got the HTTPS response back. We split the response according to space (`\r\n`). We have received 2048 bytes or less in one iteration. We have used the same code was this

```
request = b"GET / HTTP/1.0\r\nHost: " + \
    hostname.encode('utf-8') + b"\r\n\r\n"
sock.sendall(request)
response = sock.recv(2048)
while response:
    pprint(response.split(b"\r\n"))
    response = sock.recv(2048)
```

We can see that data received contains HTTP ok message, content length, receiving response until all responses are over.



```
After cipher. Press any key to continue .....
[b'HTTP/1.1 200 OK',
 b'Connection: close',
 b'Content-Length: 50049',
 b'Server: nginx',
 b'Content-Type: text/html; charset=utf-8',
 b'X-Frame-Options: DENY',
 b'Via: 1.1 vegur, 1.1 varnish, 1.1 varnish',
 b'Accept-Ranges: bytes',
 b'Date: Tue, 01 Mar 2022 22:30:16 GMT',
 b'Age: 1512',
 b'X-Served-By: cache-iad-kjyo7100132-IAD, cache-del21727-DEL',
 b'X-Cache: HIT, HIT',
 b'X-Cache-Hits: 2, 1',
 b'X-Timer: S1646173816.395042,VS0,VE1',
 b'Vary: Cookie',
 b'Strict-Transport-Security: max-age=63072000; includeSubDomains',
 b'',
 b'']
[b'<!doctype html>\n<!--[if lt IE 7]> <html class="no-js ie6 lt-ie7 lt-ie8'
 b' lt-ie9"> <![endif]-->\n<!--[if IE 7]> <html class="no-js ie7 lt-i'
 b'e8 lt-ie9"> <![endif]-->\n<!--[if IE 8]> <html class="no-js'
 b' ie8 lt-ie9"> <![endif]-->\n<!--[if gt IE 8]><!--><html c'
 b'lass="no-js" lang="en" dir="ltr"> <!--<![endif]-->\n\n<head>\n <meta ch'
 b'arset="utf-8">\n <meta http-equiv="X-UA-Compatible" content="IE=edge">'
 b'\n\n <link rel="prefetch" href="//ajax.googleapis.com/ajax/libs/jquery/'
 b'1.8.2/jquery.min.js">\n <link rel="prefetch" href="//ajax.googleapis.c'
 b'om/ajax/libs/jqueryui/1.12.1/jquery-ui.min.js">\n\n <meta name="applica'
 b'tion-name" content="Python.org">\n <meta name="msapplication-tooltip" '
```

Fig17. HTTP response

**2. Please modify the HTTP request, so you can fetch an image file of your choice from an HTTPS server (there is no need to display the image).**

We have added the path to our file image in our HTTP request, which we want. We can see that the content type is changed from text/html to image/jpeg. This shows us that the HTTP response we get is the image file that we want.

Also, we can see the contrast in data we get; we have got similar values of many pixels. This shows us that data contains a pattern that is an indication of the image.

When we run our `tls_client`, this task will run by default if no CLI arguments are passed. If three arguments are passed, then the first argument should be a hostname, the second should be Boolean for hostname check, and 3<sup>rd</sup> should be the image path we want (without “/” in starting ).

```
[b'GET /images/picture.jpg HTTP/1.0', b'Host: www.w3schools.com', b'', b'']
[b'HTTP/1.0 200 OK',
 b'Accept-Ranges: bytes',
 b'Age: 9098',
 b'Cache-Control: public,max-age=14400,public',
 b'Content-Security-Policy: frame-ancestors 'self' https://mycourses.w3schools.'
 b'com;',
 b'Content-Type: image/jpeg',
 b'Date: Sun, 06 Mar 2022 23:37:14 GMT',
 b'Etag: "03365b19d2fd81:0"',
 b'Last-Modified: Fri, 04 Mar 2022 07:58:54 GMT',
 b'Server: ECS (ndl/D35C)',
 b'X-Cache: HIT',
 b'X-Content-Security-Policy: frame-ancestors 'self' https://mycourses.w3school'
 b's.com;',
 b'X-Powered-By: ASP.NET',
 b'Content-Length: 75672',
 b'Connection: close',
 b'',
 b'']
[b'\xff\xd8\xff\xe0\x00\x10JFIF\x00\x01\x01\x01\x00H\x00H\x00\x00\xff\xe1b\x1e'
 b'Exif\x00\x00II*\x00\x08\x00\x00\x00\x07\x00\x0f\x01\x02\x00\x18\x00\x00\x00'
 b'b\x00\x00\x00\x10\x01\x02\x00\x08\x00\x00\x00z\x00\x00\x001\x01\x02\x00'
 b'\x16\x00\x00\x00x82\x00\x00\x002\x01\x02\x00\x14\x00\x00\x00'
 b'\x98\x00\x00\x00x13\x02\x03\x00\x01\x00\x00\x00\x01\x00\x01\x00'
 b'i\x87\x04\x00\x01\x00\x00\x00xda\x00\x00\x00\xa5\xc4\x07\x00.\x00\x00\x00'
 b'\xac\x00\x00\x00\x00\x00\x00CASIO COMPUTER CO.,LTD \x00EX-Z750\x001.01'
 b'
 \x002008:07:06 11:45:14\x00PrintIM\x000300\x00\x00']
```

Fig17. HTTP response for image required

# Analysis

1. In our code, we first passed CLI arguments conditions. Then we mentioned initials like the CA folder. Then we set up the context of TLS and check the initials like check hostname as True, Then we use socket to set up the TCP connection and then wrap it using context for TLS handshake. We perform handshake and print cipher used and server certificate. Then we send HTTP to get requests and get a response and finally close the connection.
2. We have used TLS1.3, which is currently best for secure connection over the net. We do a handshake to agree on the cipher suite and get the master key and session key. We then use them for encryption and authentication.
3. Cipher suite obtained is AES 256 GCM because it is very fast due to parallel processing and 256-bit key encryption. Also, 384-bit hash provides a strong MAC.
4. Certificates provide lots of info about the server along with the public key. We should check valid range dates for authenticity along with hostname.
5. CA certificate is used to verify the server certificate. Missing this would not authenticate the server and may lead to an attack. The certificate is searched with the help of hash and symbolic links.
6. We can see in Wireshark different packet transfers for different protocols along with ack.
7. Hostname check can prevent middle man attack.
8. The HTTP response header contains lots of info of data we obtain like content type, etc



# Pre-Requisite

## How To Make Executable and Run Programme

- 1) **Run `tls_client` with no arguments:-** This will print the certificate, cipher used, image obtained from w3schools. It is used to fetch the image from w3school
- 2) **Run `tls_client` with one argument:-** This will print the certificate, cipher used, data obtained from the hostname provided in 1<sup>st</sup> argument.
- 3) **Run `tls_client` with two arguments:-** This will print the certificate, cipher used, data obtained from the hostname provided in 1<sup>st</sup> argument. 2<sup>nd</sup> argument must be bool for hostname check.
- 4) **Run `tls_client` with three arguments:-** This will print the certificate, cipher used, data obtained from the hostname provided in 1<sup>st</sup> argument. 2<sup>nd</sup> argument must be bool for hostname check. 3<sup>rd</sup> argument is for a specific path needed in the hostname, like a path for image (without "/" in starting)

## Structure Of Programme

Program is made up of directory structure which includes

- 1) Problem-1 – code for problem1
- 2) Problem 2 – code for problem2
- 3) Makefile – Makefile command
- 4) Problem-1/tls\_client –code we made using the library
- 5) Env.txt – Containing Environment info on which program was tested.

## References

I have completed my assignment by learning from the following sources

1. <https://www.openssl.org/>
2. <https://docs.python.org/3/library/ssl.html>