

APPLIED CRYPTOGRAPHY

NAME: VISHWAS M

SRN: PES2UG20CS390

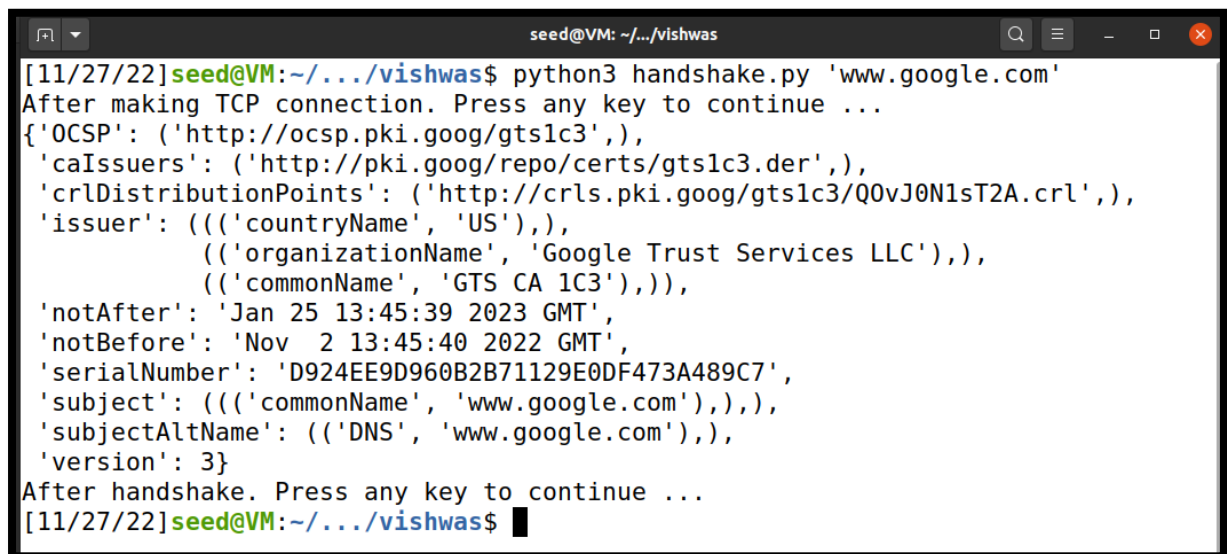
SEC: F

LAB: TLS Lab

Task 1:

Task1.a:

- Server Certificate from the program:

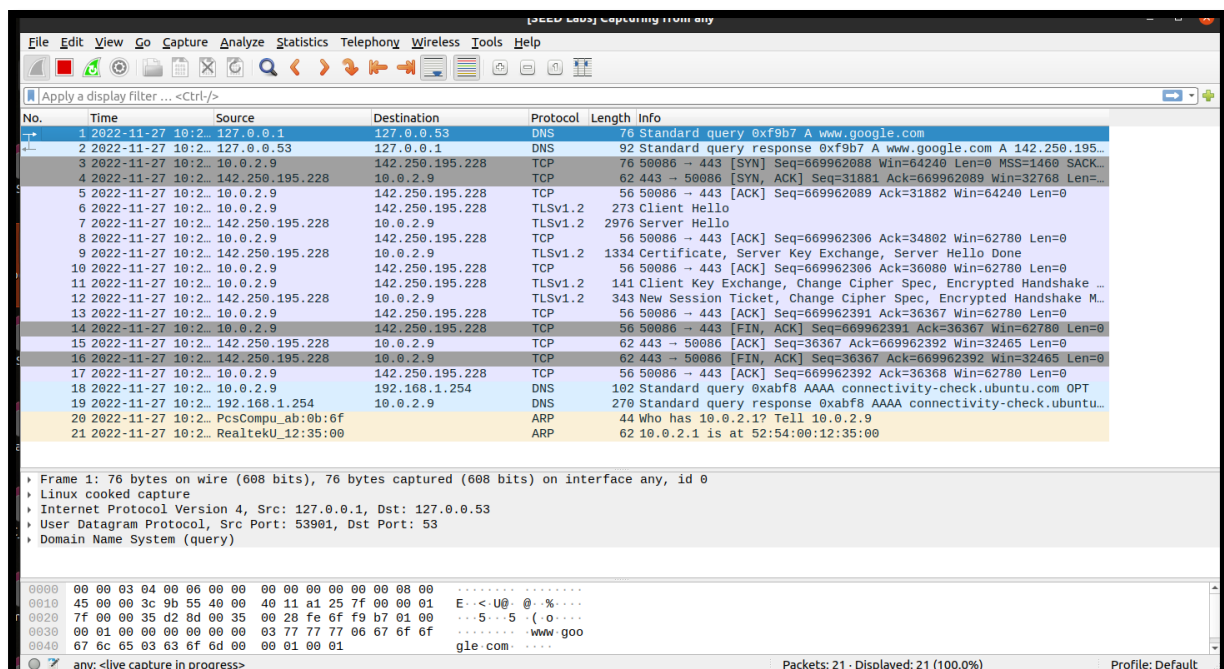


```
seed@VM: ~/.../vishwas
[11/27/22]seed@VM:~/.../vishwas$ python3 handshake.py 'www.google.com'
After making TCP connection. Press any key to continue ...
{'OCSP': ('http://ocsp.pki.goog/gts1c3',),
 'caIssuers': ('http://pki.goog/repo/certs/gts1c3.der',),
 'crlDistributionPoints': ('http://crls.pki.goog/gts1c3/Q0vJ0N1sT2A.crl',),
 'issuer': (((('countryName', 'US'),),
               (('organizationName', 'Google Trust Services LLC'),),
               (('commonName', 'GTS CA 1C3'),)),),
 'notAfter': 'Jan 25 13:45:39 2023 GMT',
 'notBefore': 'Nov  2 13:45:40 2022 GMT',
 'serialNumber': 'D924EE9D960B2B71129E0DF473A489C7',
 'subject': (((('commonName', 'www.google.com'),),),),
 'subjectAltName': (('DNS', 'www.google.com'),),),
 'version': 3}
After handshake. Press any key to continue ...
[11/27/22]seed@VM:~/.../vishwas$
```

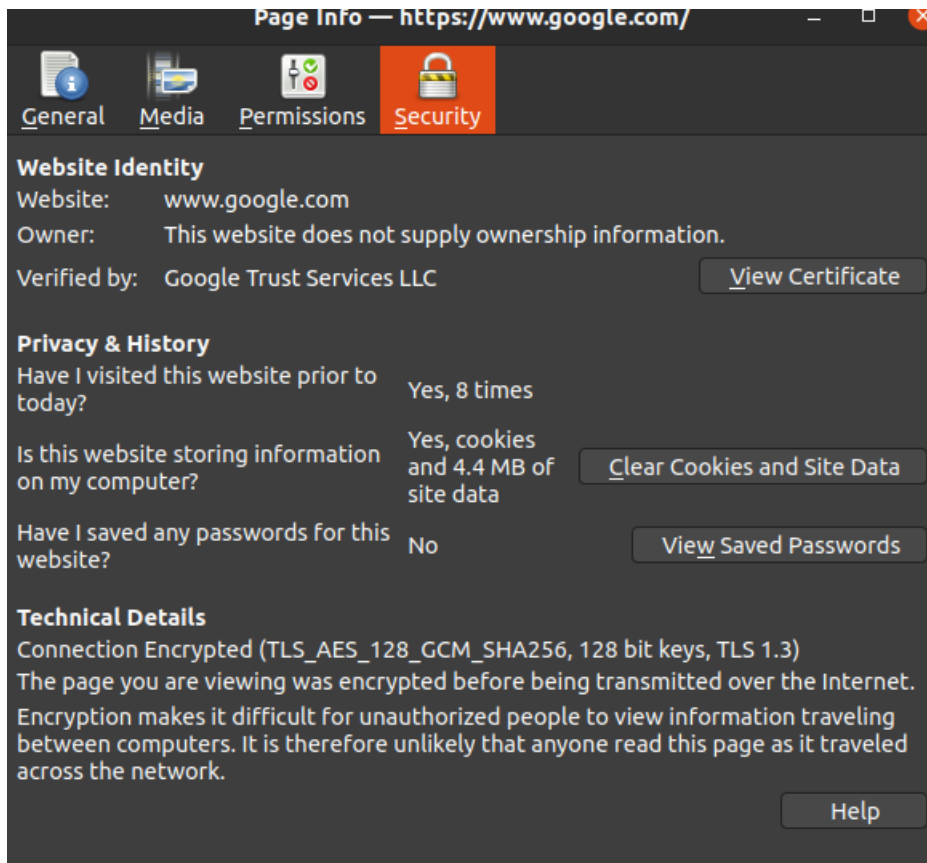
- Using Wireshark, we can see how TCP and TLS handshake are done. After TCP handshake, TLS handshake takes place and the certificates are shared between the client and the server.

Then a new session key is shared between the client and server. The client will encrypt the messages and send those messages to server. Then server will decrypt the messages using the same shared key. After communication between client and server is finished TLS connection is closed and then TCP connection is closed.

- The Client initiates the TLS handshake by sending “Client Hello” message as we can see in the below screenshot.



- We can see that AES encryption is used from the below screenshot.



- “/etc/ssl/certs” is the directory of a folder which contains the list of all the certificates

Task 1.b: CA's Certificate

After changing the cadir directory to

‘/home/seed/Desktop/certs/vishwas’, we get this error:

```
seed@VM: ~/.../vishwas
[11/27/22]seed@VM:~/.../vishwas$ python3 handshake.py 'www.example.com'
After making TCP connection. Press any key to continue ...
Traceback (most recent call last):
  File "handshake.py", line 19, in <module>
    ssock.do_handshake() # Start the 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
y failed: unable to get local issuer certificate (_ssl.c:1123)
[11/27/22]seed@VM:~/.../vishwas$
```

We have to change the name of the file that we copied:

```
seed@VM: ~/.../vishwas
[11/27/22]seed@VM:~/.../vishwas$ openssl x509 -in DigiCert_Global_Root_CA.crt -noout -subject_hash
3513523f
[11/27/22]seed@VM:~/.../vishwas$ ln -s someCA.crt 3513523f.0
[11/27/22]seed@VM:~/.../vishwas$ ls -l
total 4
lrwxrwxrwx 1 seed seed 10 Nov 27 12:49 3513523f.0 -> someCA.crt
lrwxrwxrwx 1 seed seed 62 Nov 27 12:34 DigiCert_Global_Root_CA.crt -> /usr/share/ca-certificates/mozilla/DigiCert_Global_Root_CA.crt
[11/27/22]seed@VM:~/.../vishwas$
```

Task 1.c: Experiment with the hostname check

Step1: dig cmd

```
[11/27/22]seed@VM:~/.../vishwas$ dig www.example.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 38217
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

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

;; ANSWER SECTION:
www.example.com.                1142    IN      A      93.184.216.34

;; Query time: 0 msec
;; SERVER: 127.0.0.53#53(127.0.0.53)
;; WHEN: Sun Nov 27 12:56:20 IST 2022
;; MSG SIZE rcvd: 60

[11/27/22]seed@VM:~/.../vishwas$
```

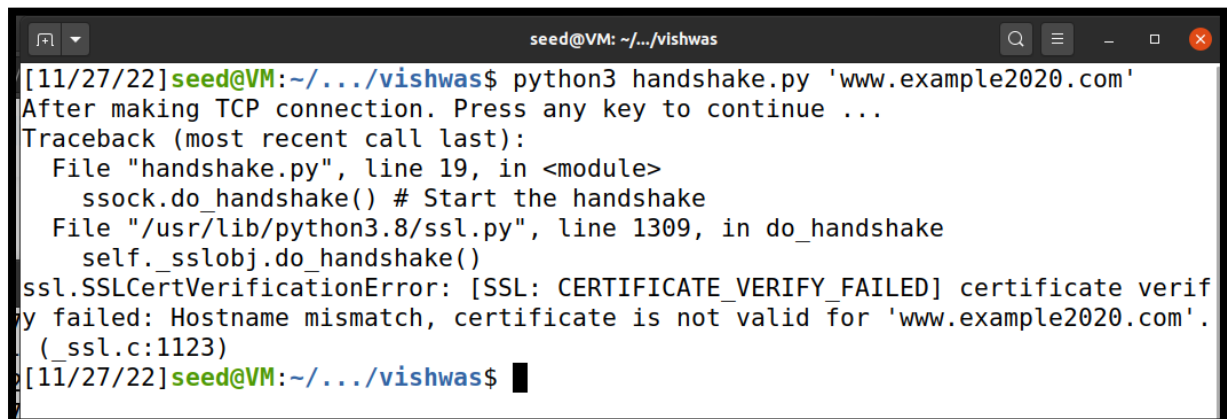
Step2:

Modifying /etc/hosts file

```
34 # For PKI lab
35 10.9.0.80      example.com
36 93.184.216.34 www.example2020.com
37
```

Step3:

When we change 'Context.check_hostname=True', then:

A terminal window titled 'seed@VM: ~/.../vishwas' showing the execution of a Python script. The user runs 'python3 handshake.py 'www.example2020.com''. The script prompts 'After making TCP connection. Press any key to continue ...'. It then shows a traceback for 'ssl.SSLError: [SSL: CERTIFICATE_VERIFY_FAILED] certificate verification failed: Hostname mismatch, certificate is not valid for 'www.example2020.com''. The error occurs in 'handshake.py' at line 19 and in the standard library 'ssl.py' at line 1309. The terminal ends with the prompt '[11/27/22]seed@VM:~/.../vishwas\$' and a cursor.

```
[11/27/22]seed@VM:~/.../vishwas$ python3 handshake.py 'www.example2020.com'
After making TCP connection. Press any key to continue ...
Traceback (most recent call last):
  File "handshake.py", line 19, in <module>
    sock.do_handshake() # Start the handshake
  File "/usr/lib/python3.8/ssl.py", line 1309, in do_handshake
    self._sslobj.do_handshake()
ssl.SSLError: [SSL: CERTIFICATE_VERIFY_FAILED] certificate verification failed: Hostname mismatch, certificate is not valid for 'www.example2020.com'.
(_ssl.c:1123)
[11/27/22]seed@VM:~/.../vishwas$
```

When we change 'Context.check_hostname=False', then:

```
seed@VM: ~/.../vishwas
[11/27/22]seed@VM:~/.../vishwas$ python3 handshake.py 'www.example2020.com'
After making TCP connection. Press any key to continue ...
{'OCSP': ('http://ocsp.digicert.com',),
 'caIssuers': ('http://cacerts.digicert.com/DigiCertTLRSASHA2562020CA1-1.crt',),
 'crlDistributionPoints': ('http://crl3.digicert.com/DigiCertTLRSASHA2562020CA1-4.crl',
                           'http://crl4.digicert.com/DigiCertTLRSASHA2562020CA1-4.crl'),
 'issuer': (((('countryName', 'US'),),
               (('organizationName', 'DigiCert Inc'),),
               (('commonName', 'DigiCert TLS RSA SHA256 2020 CA1'),)),),
 'notAfter': 'Mar 14 23:59:59 2023 GMT',
 'notBefore': 'Mar 14 00:00:00 2022 GMT',
 'serialNumber': '0FAA63109307BC3D414892640CCD4D9A',
 'subject': (((('countryName', 'US'),),
                (('stateOrProvinceName', 'California'),),
                (('localityName', 'Los Angeles'),),
                (('organizationName',
                  'Internet\\xa0Corporation\\xa0for\\xa0Assigned\\xa0Names\\xa0and\\xa0'
                  'Numbers'),),
                (('commonName', 'www.example.org'),)),),
 'subjectAltName': (('DNS', 'www.example.org'),
                    ('DNS', 'example.net'),
                    ('DNS', 'example.edu'),
                    ('DNS', 'example.com'),
                    ('DNS', 'example.org'),
                    ('DNS', 'www.example.com'),
                    ('DNS', 'www.example.edu'),
                    ('DNS', 'www.example.net')),
 'version': 3}
After handshake. Press any key to continue ...
[11/27/22]seed@VM:~/.../vishwas$
```

Based on the above observation we can conclude that it is necessary to check the hostname.

Task 1.d: Sending and Getting Data

1) After adding the data sending/receiving code to client program.

```
seed@VM: ~/.../vishwas$ python3 handshake.py 'www.example.com'
After making TCP connection. Press any key to continue ...
{'OCSP': ('http://ocsp.digicert.com',),
 'caIssuers': ('http://cacerts.digicert.com/DigiCertTLRSASHA2562020CA1-1.crt',),
 'crlDistributionPoints': ('http://crl3.digicert.com/DigiCertTLRSASHA2562020CA1-4.crl',
 'http://crl4.digicert.com/DigiCertTLRSASHA2562020CA1-4.crl'),
 'issuer': (((('countryName', 'US'),),
               (('organizationName', 'DigiCert Inc'),),
               (('commonName', 'DigiCert TLS RSA SHA256 2020 CA1'),)),
 'notAfter': 'Mar 14 23:59:59 2023 GMT',
 'notBefore': 'Mar 14 00:00:00 2022 GMT',
 'serialNumber': '0FAA63109307BC3D414892640CCD4D9A',
 'subject': (((('countryName', 'US'),),
                 (('stateOrProvinceName', 'California'),),
                 (('localityName', 'Los Angeles'),),
                 (('organizationName',
                  'Internet\\xa0Corporation\\xa0for\\xa0Assigned\\xa0Names\\xa0and\\xa0'
                  'Numbers'),),
                 (('commonName', 'www.example.org'),)),
 'subjectAltName': (('DNS', 'www.example.org'),
                    ('DNS', 'example.net'),
                    ('DNS', 'example.edu'),
                    ('DNS', 'example.com'),
                    ('DNS', 'example.org'),
                    ('DNS', 'www.example.com'),
                    ('DNS', 'www.example.edu'),
                    ('DNS', 'www.example.net')),
 'version': 3}
After handshake. Press any key to continue ...
[b'HTTP/1.0 200 OK',
 b'Age: 488758',
 b'Cache-Control: max-age=604800',
 b'Content-Type: text/html; charset=UTF-8',
```

```
seed@VM: ~/.../vishwas$
b'Age: 488758',
b'Cache-Control: max-age=604800',
b'Content-Type: text/html; charset=UTF-8',
b'Date: Sun, 27 Nov 2022 07:50:20 GMT',
b'Etag: "3147526947+ident"',
b'Expires: Sun, 04 Dec 2022 07:50:20 GMT',
b'Last-Modified: Thu, 17 Oct 2019 07:18:26 GMT',
b'Server: ECS (dcb/7F5E)',
b'Vary: Accept-Encoding',
b'X-Cache: HIT',
b'Content-Length: 1256',
b'Connection: close',
b'',
b'']
[b'<!doctype html>\n<html>\n<head>\n    <title>Example Domain</title>\n\n    '
 b' <meta charset="utf-8" />\n    <meta http-equiv="Content-type" content="t'
 b'ext/html; charset=utf-8" />\n    <meta name="viewport" content="width=device-width, initial-scale=1" />\n    <style type="text/css">\n        body {\n            background-color: #f0f0f2;\n            margin: 0;\n            padding: 0;\n            font-family: -apple-system, system-ui, BlinkMacSystemFont, "Segoe UI", "Open Sans", "Helvetica Neue", Helvetica, Arial, sans-serif;\n        }\n        div {\n            width: 600px;\n            margin: 5em auto;\n            padding: 2em;\n            background-color: #fdfdff;\n            border-radius: 0.5em;\n            box-shadow: 2px 3px 7px 2px rgba(0,0,0,0.02);\n        }\n        a:link, a:visited {\n            color: #38488f;\n            text-decoration: none;\n        }\n        @media (max-width: 700px) {\n            div {\n                margin: 0 auto;\n                width: auto;\n            }\n        }\n    </style>\n</head>\n\n<body>\n<div>\n    <h1>Example Domain</h1>\n    <p>This domain is for use in illustrative examples in documents. You may use this domain in literature without prior coordination or asking for permission.</p>\n    <p><a href="https://www.iana.org/domains/example">More information</a></p>\n</div>\n</body>\n</html>\n']
[11/27/22]seed@VM:~/.../vishwas$
```


2) Modifying the HTTP GET request to get an image from www.google.com

```
seed@VM: ~/.../vishwas
[11/27/22]seed@VM:~/.../vishwas$ python3 handshake.py 'www.google.com'
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': 'Jan 25 13:45:39 2023 GMT',
 'notBefore': 'Nov 2 13:45:40 2022 GMT',
 'serialNumber': 'D924EE9D960B2B71129E0DF473A489C7',
 'subject': (((('commonName', 'www.google.com'),),),),
 'subjectAltName': (('DNS', 'www.google.com'),),
 'version': 3}
After handshake. Press any key to continue ...
[b'HTTP/1.0 200 OK',
 b'Accept-Ranges: bytes',
 b'Content-Type: image/png',
 b'Cross-Origin-Resource-Policy: cross-origin',
 b'Cross-Origin-Opener-Policy-Report-Only: same-origin; report-to="static-on-bi',
 b'gtable"',
 b'Report-To: {"group":"static-on-bigtable","max_age":2592000,"endpoints":[{"ur',
 b'l":"https://csp.withgoogle.com/csp/report-to/static-on-bigtable"}]}',
 b'Content-Length: 7108',
 b'Date: Sun, 27 Nov 2022 08:14:36 GMT',
 b'Expires: Sun, 27 Nov 2022 08:14:36 GMT',
 b'Cache-Control: private, max-age=31536000',
 b'Last-Modified: Tue, 22 Oct 2019 18:30:00 GMT',
 b'X-Content-Type-Options: nosniff',
 b'Server: sffe',
 b'X-XSS-Protection: 0',
 b'Alt-Svc: h3=":443"; ma=2592000,h3-29=":443"; ma=2592000,h3-Q050=":443"; ma=2',
 b'592000,h3-Q046=":443"; ma=2592000,h3-Q043=":443"; ma=2592000,quic=":443"; ma=']
```

```

b'Server: sfpe',
b'X-XSS-Protection: 0',
b'Alt-Svc: h3=":443"; ma=2592000,h3-29=":443"; ma=2592000,h3-Q050=":443"; ma=2'
b'592000,h3-Q046=":443"; ma=2592000,h3-Q043=":443"; ma=2592000,quic=":443"; ma'
b'=2592000; v="46,43"',
b'',
b'\x89PNG',
b'\x1a\n\x00\x00\x00\rIHDR\x00\x00\x02 \x00\x00\x00\xb8\x08\x03'
b'\x00\x00\x00\xed\xfd\xa7)\x00\x00\x03\x00PLTE\x00\x00\x00\xff\xff'
b'\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff'
b'\xfe\xfe\xfe\xff\xff\xff\xff\xfe\xfe\xfe\xff\xff\xff\xff\xff\xff'
b'\xff\xff\xfe\xfe\xfe\xfe\xfe\xfe\xfe\xff\xff\xff\xff\xff\xff\xff'
b'\xff\xfe\xfe\xfe\xfe\xfe\xfe\xff\xff\xff\xff\xff\xff\xff\xff'
b'\xfe\xfe\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff'
b'\xfe\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff'
b'\xfe\xfe\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff'
b'\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff'
b'\xfe\xfe\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff'
b'\xfe\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff'
b'\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff'
b'\xfe\xff\xff\xff\xff\xff\xff\xff\xff\xff'
b'\xff\xff\xff\xff\xff\xff\xff\xff\xff'
b'\xfe\xff\xff\xff\xff\xff\xff\xff'
b'\xff\xff\xff\xff\xff\xff'
b'\xfe\xff\xff'
b'\xff\xff'
b'\x00'

```


Task 2: TLS Server

Handshake.py:

```
1#!/usr/bin/python3
2import socket, ssl, sys, pprint
3hostname = sys.argv[1]
4port = 443
5cadir = '/home/seed/Desktop/AC_lab9/vishwas'
6# Set up the TLS context
7context = ssl.SSLContext(ssl.PROTOCOL_TLS_CLIENT) # For Ubuntu 20.04 VM
8#context = ssl.SSLContext(ssl.PROTOCOL_TLSv1_2) # For Ubuntu 16.04 VM
9context.load_verify_locations(capath=cadir)
10context.verify_mode = ssl.CERT_REQUIRED
11context.check_hostname = False
12# Create TCP connection
13sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
14sock.connect((hostname, port))
15# Create a TLS connection. Does any key to establish ...
```

Server.py:

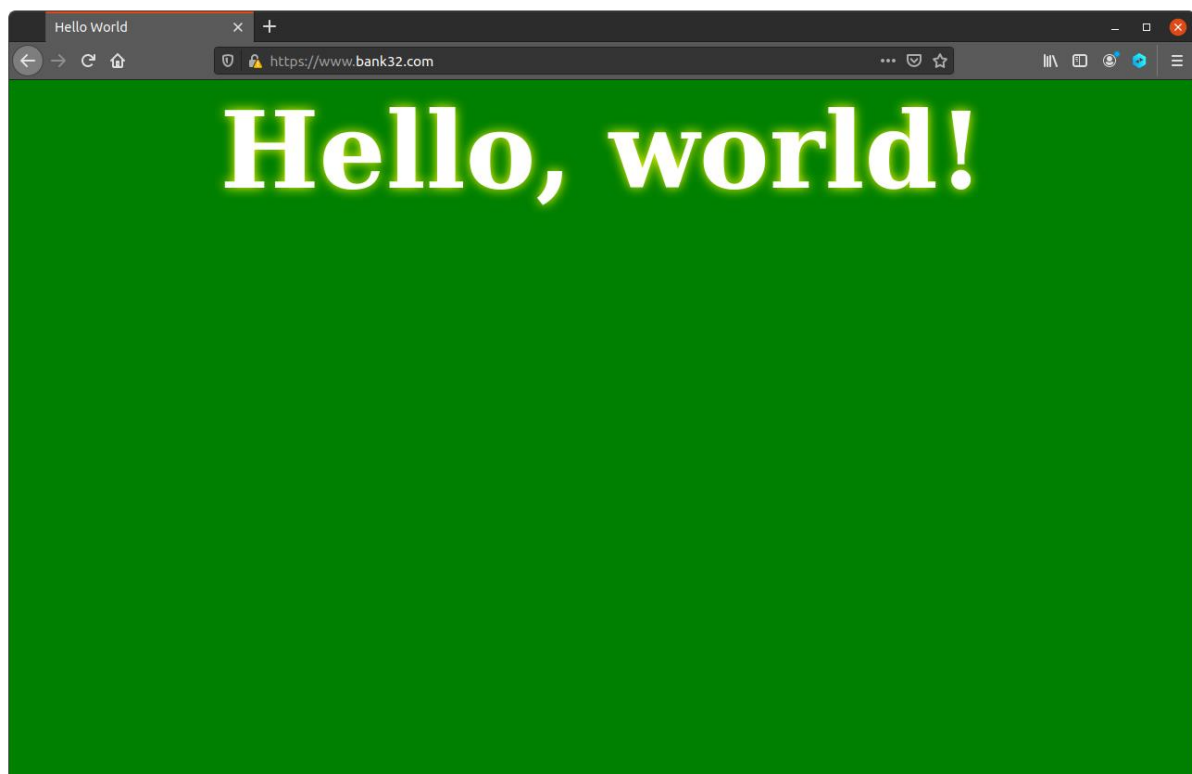
```
1#!/usr/bin/python3
2import socket
3import ssl
4html = """
5HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n
6<!DOCTYPE html><html><body><h1>Hello, world!</h1></body></html>
7"""
8SERVER_CERT = '/home/seed/Desktop/AC_lab9/vishwas/server.crt'
9SERVER_PRIVATE = '/home/seed/Desktop/AC_lab9/vishwas/server.key'
10# context = ssl.SSLContext(ssl.PROTOCOL_TLS_SERVER) # For Ubuntu 20.04 VM
11context = ssl.SSLContext(ssl.PROTOCOL_TLSv1_2) # For Ubuntu 16.04 VM
12context.load_cert_chain(SERVER_CERT, SERVER_PRIVATE)
13sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM, 0)
14sock.bind(('0.0.0.0', 443))
15sock.listen(5)
16while True:
17    newsock, fromaddr = sock.accept()
18    ssock = context.wrap_socket(newsock, server_side=True)
19    data = ssock.recv(1024) # Read data over TLS
20    ssock.sendall(html.encode('utf-8')) # Send data over TLS
21    ssock.shutdown(socket.SHUT_RDWR) # Close the TLS connection
22    ssock.close()
```

In the process of debugging the server.py program, we may encounter the following situations:

```
[11/27/22]seed@VM:~/.../vishwas$ python3 server.py
Enter PEM pass phrase:
Traceback (most recent call last):
  File "server.py", line 12, in <module>
    context.load_cert_chain(SERVER_CERT, SERVER_PRIVATE)
OSError: [Errno 22] Invalid argument
[11/27/22]seed@VM:~/.../vishwas$
```

Task 2.b: Testing the server program using browsers

Since in the previous lab experiment, the root CA certificate created by yourself has been installed in the browser, you can directly test it. After running the server.py program, you will get the following results in the browser:



Task 2.c: Certificate with multiple names

Modifying the /etc/hosts file

```
34 # For PKI lab
35 10.9.0.80      example.com
36 93.184.216.34 www.example2020.com
37
```

Checking in browser:

Since we have imported the private ca.crt certificate in the browser, and the server certificate we generated is issued by this ca.crt, it will pass the verification:



Of course, we can also test on other hosts, just need to configure a static IP in the etc/hosts file to make an IP modification.

Task 3: A Simple HTTPS Proxy

```
1 import threading
2
3 while True:
4     sock_for_browser, fromaddr = sock_listen.accept()
5     ssock_for_browser = context_srv.wrap_socket(sock_for_browser,
6     server_side=True)
7     x = threading.Thread(target=process_request, args=(ssock_for_browser,))
8     x.start()
9
10 def process_request(ssock_for_browser):
11     hostname = 'www.example.com'
12     real_ip_address = "93.184.216.34"
13     # Make a connection to the real server
14     sock_for_server = socket.create_connection((real_ip_address , 443))
15     ssock_for_server = ... # [Code omitted]: Wrap the socket using TLS
16     request = ssock_for_browser.recv(2048)
17     if request:
18         # Forward request to server
19         ssock_for_server.sendall(request)
20         # Get response from server, and forward it to browser
21         response = ssock_for_server.recv(2048)
22         while response:
23             ssock_for_browser.sendall(response) # Forward to browser
24             response = ssock_for_server.recv(2048)
25 ssock_for_browser.shutdown(socket.SHUT_RDWR)
26 ssock_for_browser.close()
27
28
```

Since the hostname `www.example.com` is already mapped to the localhost, we cannot use this name in the proxy code to connect to the real web server. We will get the IP address of this domain, and directly use the IP address in our proxy code, instead of using the hostname.