

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 certificate:
{'OCSP': ('http://ocsp.globalsign.com/gsrsoavsslca2018',),
 'caIssuers': ('http://secure.globalsign.com/cacert/gsrsoavsslca2018.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 where certificates for different servers are stored. 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:

No.	Time	Source	Destination	Protocol	Length	Info
307	7.104156793	RealtekU_68:e0:6f	Broadcast	ARP	60	Who has 10.208.20.113? Tell 10.208.20.101
308	7.115820500	10.208.23.203	10.10.17.1	TCP	74	58460 → 443 [SYN] Seq=0 Win=29200 Len=0 MSS=1460
309	7.116160778	10.10.17.1	10.208.23.203	TCP	74	443 → 58460 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0
310	7.116208028	10.208.23.203	10.10.17.1	TCP	66	58460 → 443 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSv
311	7.119443374	RealtekU_68:e0:6f	Broadcast	ARP	60	Who has 10.208.20.157? Tell 10.208.20.101
312	7.119695920	RealtekU_68:e0:6f	Broadcast	ARP	60	Who has 10.208.20.105? Tell 10.208.20.101
313	7.119699538	RealtekU_68:e0:6f	Broadcast	ARP	60	Who has 10.208.20.158? Tell 10.208.20.101
314	7.119701534	RealtekU_68:e0:6f	Broadcast	ARP	60	Who has 10.208.20.159? Tell 10.208.20.101
315	7.119703263	RealtekU_68:e0:6f	Broadcast	ARP	60	Who has 10.208.20.160? Tell 10.208.20.101
316	7.119705108	RealtekU_68:e0:6f	Broadcast	ARP	60	Who has 10.208.20.107? Tell 10.208.20.101
317	7.119952618	RealtekU_68:e0:6f	Broadcast	ARP	60	Who has 10.208.20.108? Tell 10.208.20.101
318	7.119956794	RealtekU_68:e0:6f	Broadcast	ARP	60	Who has 10.208.20.110? Tell 10.208.20.101
319	7.119959544	RealtekU_68:e0:6f	Broadcast	ARP	60	Who has 10.208.20.109? Tell 10.208.20.101
320	7.120140168	10.208.23.203	10.10.17.1	TLSv1.2	583	Client Hello
321	7.120472647	10.10.17.1	10.208.23.203	TCP	66	443 → 58460 [ACK] Seq=1 Ack=518 Win=30080 Len=0 TSv
322	7.124687232	10.10.17.1	10.208.23.203	TLSv1.2	1514	Server Hello
323	7.124703537	10.208.23.203	10.10.17.1	TCP	66	58460 → 443 [ACK] Seq=518 Ack=1449 Win=32128 Len=0
324	7.124932083	10.10.17.1	10.208.23.203	TLSv1.2	1808	Certificate, Server Key Exchange, Server Hello Don
325	7.124938401	10.208.23.203	10.10.17.1	TCP	66	58460 → 443 [ACK] Seq=518 Ack=3191 Win=35584 Len=0
326	7.126378273	10.208.23.203	10.10.17.1	TLSv1.2	192	Client Key Exchange, Change Cipher Spec, Encrypted
327	7.127209081	10.10.17.1	10.208.23.203	TLSv1.2	340	New Session Ticket, Change Cipher Spec, Encrypted
328	7.128266891	10.208.23.203	10.10.17.1	TCP	66	58460 → 443 [FIN, ACK] Seq=644 Ack=3465 Win=38528
329	7.128548538	10.10.17.1	10.208.23.203	TLSv1.2	97	Encrypted Alert
330	7.128570443	10.208.23.203	10.10.17.1	TCP	54	58460 → 443 [RST] Seq=645 Win=0 Len=0
331	7.128578159	10.10.17.1	10.208.23.203	TCP	66	443 → 58460 [FIN, ACK] Seq=3496 Ack=645 Win=30080
332	7.128584643	10.208.23.203	10.10.17.1	TCP	54	58460 → 443 [RST] Seq=645 Win=0 Len=0

Frame 320: 583 bytes on wire (4664 bits), 583 bytes captured (4664 bits) on interface 0
 Ethernet II, Src: a4:bb:6d:b7:30:76 (a4:bb:6d:b7:30:76), Dst: Cisco_a5:8c:3f (e0:2f:6d:a5:8c:3f)
 Internet Protocol Version 4, Src: 10.208.23.203, Dst: 10.10.17.1
 Transmission Control Protocol, Src Port: 58460, Dst Port: 443, Seq: 1, Ack: 1, Len: 517
 Source Port: 58460
 Destination Port: 443
 [Stream index: 3]

Invalid filter: "TCP" is neither a field nor a protocol name.

Packets: 557 · Displayed: 557 (100.0%) · Dropped: 0 (0.0%) · Profile: Default

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  X  output1.txt  cert.py
q2.py > ...
1  #!/usr/bin/python3
2  import socket, ssl, sys, pprint
3  hostname = sys.argv[1]
4  port = int(sys.argv[2])
5  cadir = './certs'
6  # Set up the TLS context
7  context = ssl.SSLContext(ssl.PROTOCOL_TLS_CLIENT)
8  context.load_verify_locations(capath=cadir)
9  context.verify_mode = ssl.CERT_REQUIRED
10 context.check_hostname = True
11
12 sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
13 sock.connect((hostname, port))
14 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/gsrsoavsslca2018',),
 'caIssuers': ('http://secure.globalsign.com/cacert/gsrsoavsslca2018.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 x
tls_client.py > ...
1  #!/usr/bin/python3
2  import socket, ssl, sys, pprint
3  from turtle import clear
4  hostname = sys.argv[1]
5  port = int(sys.argv[2])
6  cadir = '/etc/ssl/certs'
7  # Set up the TLS context
8  context = ssl.SSLContext(ssl.PROTOCOL_TLS_CLIENT)
9  context.load_verify_locations(capath=cadir)
10 context.verify_mode = ssl.CERT_REQUIRED
11 context.check_hostname = False
12 # print("List of all supported Ciphers:",context.get_ciphers())
13 # Create TCP connection
14 sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
15 sock.connect((hostname, port))
16 print("TCP connection established.")
17 # Add the TLS
18 ssock = context.wrap_socket(sock, server_hostname=hostname,do_handshake_on_connect=False)
19 ssock.do_handshake() # Start the handshake
20 # print("Certificates to be copied to the folder are:",context.get_ca_certs()) #To get the
21 print("Handshake done.")
22 # print("Answers for Part 1:")
23 # 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-lubuntu1-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 > ...
1  #!/usr/bin/python3
2  import socket, ssl, sys, pprint
3  import requests
4  from turtle import clear
5  hostname = sys.argv[1]
6  port = int(sys.argv[2])
7  cadir = '/etc/ssl/certs'
8  # Set up the TLS context
9  context = ssl.SSLContext(ssl.PROTOCOL_TLS_CLIENT)
10 context.load_verify_locations(capath=cadir)
11 context.verify_mode = ssl.CERT_REQUIRED
12 context.check_hostname = False
13 print("Checking host name: ",context.check_hostname)
14 # print("List of all supported Ciphers:",context.get_ciphers())
15 # Create TCP connection
16 sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
17 sock.connect((hostname, port))
18 print("TCP connection established.")
19 # Add the TLS
20 ssock = context.wrap_socket(sock, server_hostname=hostname,do_handshake_on_connect=False)
21 ssock.do_handshake() # Start the handshake
22 # print("Certificates to be copied to the folder are:",context.get_ca_certs()) #To get the information about certificates.
23 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$ 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 > ...
1  #!/usr/bin/python3
2  import socket, ssl, sys, pprint
3  import requests
4  from turtle import clear
5  hostname = sys.argv[1]
6  port = int(sys.argv[2])
7  cadir = '/etc/ssl/certs'
8  # Set up the TLS context
9  context = ssl.SSLContext(ssl.PROTOCOL_TLS_CLIENT)
10 context.load_verify_locations(capath=cadir)
11 context.verify_mode = ssl.CERT_REQUIRED
12 context.check_hostname = True
13 print("Checking host name: ",context.check_hostname)
14 # print("List of all supported Ciphers:",context.get_ciphers())
15 # Create TCP connection
16 sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
17 sock.connect((hostname, port))
18 print("TCP connection established.")
19 # Add the TLS
20 ssock = context.wrap_socket(sock, server_hostname=hostname,do_handshake_on_connect=False)
21 ssock.do_handshake() # Start the handshake
22 # print("Certificates to be copied to the folder are:",context.get_ca_certs()) #To get the information about certificates.
23 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 > ...
1  #!/usr/bin/python3
2  import socket, ssl, sys, pprint
3  import requests
4  from turtle import clear
5  hostname = sys.argv[1]
6  port = int(sys.argv[2])
7  cadir = '/etc/ssl/certs'
8  # Set up the TLS context
9  context = ssl.SSLContext(ssl.PROTOCOL_TLS_CLIENT)
10 context.load_verify_locations(capath=cadir)
11 context.verify_mode = ssl.CERT_REQUIRED
12 context.check_hostname = False
13 print("Checking host name: ",context.check_hostname)
14 # print("List of all supported Ciphers:",context.get_ciphers())
15 # Create TCP connection
16 sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
17 sock.connect((hostname, port))
18 print("TCP connection established.")
19 # Add the TLS
20 ssock = context.wrap_socket(sock, server_hostname=hostname,do_handshake_on_connect=False)
21 ssock.do_handshake() # Start the handshake
22 # print("Certificates to be copied to the folder are:",context.get_ca_certs()) #To get the info
23 print("Handshake done.")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
(base) tooha@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) tooha@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 x
tls_client.py > ...

35 #Send HTTP Request to Server
36 request = b"GET / HTTP/1.0\r\nHost: " + hostname.encode() + b"\r\n\r\n"
37 ssock.sendall(request)
38 # Read HTTP Response from Server
39 response = ssock.recv(2048)
40 while response:
41     pprint.pprint(response.split(b"\r\n"))
42     response = ssock.recv(2048)
43
44 ssock.shutdown(socket.SHUT_RDWR)
45 ssock.close()

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

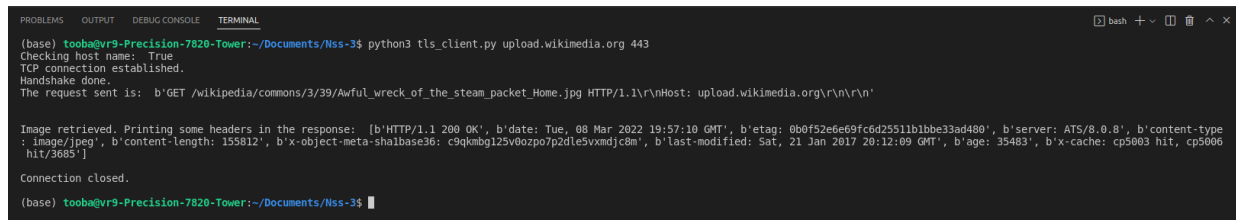
(base) toooba@vr9-Precision-7820-Tower:~/Documents/Nss-3$ python3 tls_client.py moodle.iitd.ac.in 443
TCP connection established.
Handshake done.
[b'HTTP/1.0 303 See Other',
 b'Date: Tue, 08 Mar 2022 13:31:44 GMT',
 b'Server: Apache/2.4.18 (Ubuntu)',
 b'Set-Cookie: MoodleSession=dkj6mfjf0ifij5vac5l0btjqc0; 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) toooba@vr9-Precision-7820-Tower:~/Documents/Nss-3$
```

For fetching an image:

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

```
31 file = "/wikipedia/commons/3/39/Awful_wreck_of_the_steam_packet_Home.jpg"
32
33 request = b"GET "+file.encode(encoding='utf-8')+b" HTTP/1.1\r\nHost: " + \
34 hostname.encode(encoding='utf-8') + b"\r\n\r\n"
35 ssock.sendall(request)
36
37 print("The request sent is: ",request)
38
39 # Read HTTP Response from Server
40 response = ssock.recv(2048)
41 print("\n\nImage retrieved. Printing some headers in the response: ",response.split(b"\r\n")[:10])
42 while response:
43     response = ssock.recv(2048)
44
45 ssock.shutdown(socket.SHUT_RDWR)
46 ssock.close()
47 print("\n\nConnection closed.\n")
```

2. Terminal screenshot:



```
(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: c9qkmbgl25v0ozp07p2d0e5vxdjcm', 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.