PLAGIARISM STATEMENT

I certify that this assignment/report is my own work, based on my personal study and/or research and that I have acknowledged all material and sources used in its preparation, whether they be books, articles, reports, lecture notes, and any other kind of document, electronic or personal communication. I also certify that this assignment/report has not previously been submitted for assessment in any other course, except where specific permission has been granted from all course instructors involved, or at any other time in this course, and that I have not copied in part or whole or otherwise plagiarised the work of other students and/or persons. I pledge to uphold the principles of honesty and responsibility at CSE@IITH. In addition, I understand my responsibility to report honour violations by other students if I become aware of it.

Name: Akash Tadwai
Date: 13/02/2022
Signature: Akash Tadwai

Assignment 4: Decrypting TLS and HTTP(S) using Wireshark++

Individual Assignment

PART-A: Decrypt TLS handshake and HTTPS messages between your browser and the web server of Bank X

- Set SSLKEYLOGFILE environment variable in your host OS by following the instructions given in References 1-3 and then launch Chrome browser with a blank tab (for surfing the website of bank x) and wireshark (for capturing all the messages exchanged between your browser and bank website/DNS resolvers/CAs).
- Start packet capture in wireshark
- Type in the hostname of the bank X in the address bar of the browser. Let N be your (RollNo % 4 +1). If N==1, X=ICICI. If N==2, X=HDFC. If N==3, X=SBI. If N==4, X=Bank of America
    a. Click on the link that takes you to the online net banking page of the bank.
    b. Enter some arbitrary values against Username and Password so that the login process fails:-)
    c. Stop the packet capture in wireshark and save it as <RollNo-BankName.pcapng>. And also close your browser tab.
    d. Follow the steps in References 1-3 to specify the complete path of SSLkeyLog file in your computer for wireshark to decrypt TLS and HTTPS messages present in <RollNo-BankName.pcapng>.
        i. Note that <RollNo-BankName.pcapng> should only contain the messages exchanged between your browser and bank website/DNS resolvers/CAs

including sub-domains/redirections and 3rd party tracking/resource fetching sessions triggered by your visit to the bank's site. So, close all background Apps running on your computer to avoid capturing their messages in your wireshark capture or use appropriate display/capture filters to exclude other messages in your packet trace. **This is your Deliverable-1.**

ii. Before providing session keys which are present inside SSLkeyLog file to wireshark, you should find that all of the application traffic and most of the handshake (HTTPS) is encrypted and shown as TLS traffic with encrypted application data. Get a snapshot of it. **This is your Deliverable-2.**

iii. After providing session keys in SSLkeyLog file to wireshark, you should find that all of the application traffic along with handshake traffic is decrypted and shown as HTTP traffic along with TLS handshake messages in plain-text. Get a snapshot of it. **This is Deliverable-3.**

Answer the following queries by referring to the (decrypted) messages in your browsing session with the banking site using wireshark GUI. It is important to keep in mind that an Ethernet frame may contain either a partial, one or more TLS records. This is very different from HTTP(S), for which each Ethernet frame contains either one complete HTTP message or a portion of a HTTP message.

Whenever possible, when answering the questions given below, you should produce a screenshot of the packet(s) within the trace that you used to answer the question asked. Highlight portions of the snapshot to explain your answer. To print a packet in wireshark GUI, use *File->Print Option*, choose *Selected packet only*, choose *Packet summary line,* and select the minimum amount of packet detail that you need to answer the question.

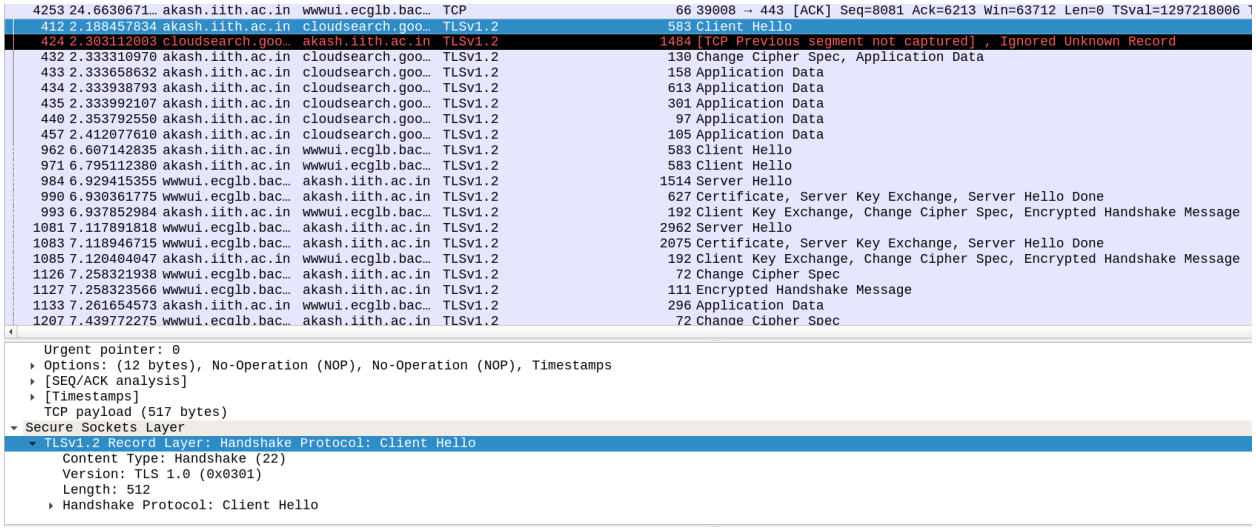
1. **What browser did you use, what's the version number?**
   Google Chrome 97.0.4692.99

2. **List out various protocols that you noticed in the column named "Protocol" in the wireshark GUI from the time you keyed in the hostname of the bank in the browser till you start viewing application data. For each such protocol, mention its purpose in brief.**
   ARP: It maps dynamic IP addresses to the MAC address of an interface
   DNS: It resolves hostnames to ip addresses on the internet
   MDNS: It is a protocol aimed at helping for name resolution in smaller networks
   STUN: It is a client side protocol used by VoIP utilities for communication between the machines hidden behind NAT gateways.
   TCP: It is a connection based reliable transport protocol to reliably share data between two applications.
   UDP - It is a connectionless datagram based transport protocol which focuses on lower response time rather than reliability.

HTTP2: - It provides a way for users to interact with web resources such as HTML files by transmitting hypertext messages between clients and servers

TLS: The Transport Layer Security is a security protocol that provides confidentiality and data integrity for Internet communications. Implementing TLS is standard practice for building secure web apps.

3. **Each of the TLS records begins with the same three fields (with possibly different values). One of these fields is "content type" and has a length of one byte. List all three fields and their lengths for the first 10 records in the trace.**



1. Content Type: Handshake (22) **(1byte)**

   Version: TLS 1.0 (0x0301) **(2bytes)**

   Length: 512 **(2bytes)**

2. Content Type: Change Cipher Spec (20) **(1byte)**

   Version: TLS 1.2 (0x0303) **(2bytes)**

   Length: 1 **(2bytes)**

3. Content Type: Application Data (23) **(1byte)**

   Version: TLS 1.2 (0x0303) **(2bytes)**

   Length: 4541 **(2bytes)**

4. Content Type: Application Data (23) **(1byte)**

   Version: TLS 1.2 (0x0303) **(2bytes)**

Length: 4541 **(2bytes)**

5. Content Type: Application Data (23) **(1byte)**

   Version: TLS 1.2 (0x0303) **(2bytes)**

   Length: 53 **(2bytes)**

6. Content Type: Application Data (23) **(1byte)**

   Version: TLS 1.2 (0x0303) **(2bytes)**

   Length: 87 **(2bytes)**

7. Content Type: Application Data (23) **(1byte)**

   Version: TLS 1.2 (0x0303) **(2bytes)**

   Length: 309 **(2bytes)**

8. Content Type: Application Data (23) **(1byte)**

   Version: TLS 1.2 (0x0303) **(2bytes)**

   Length: 541 **(2bytes)**

9. Content Type: Handshake (22) **(1byte)**

   Version: TLS 1.0 (0x0301) **(2bytes)**

   Length: 512 **(2bytes)**

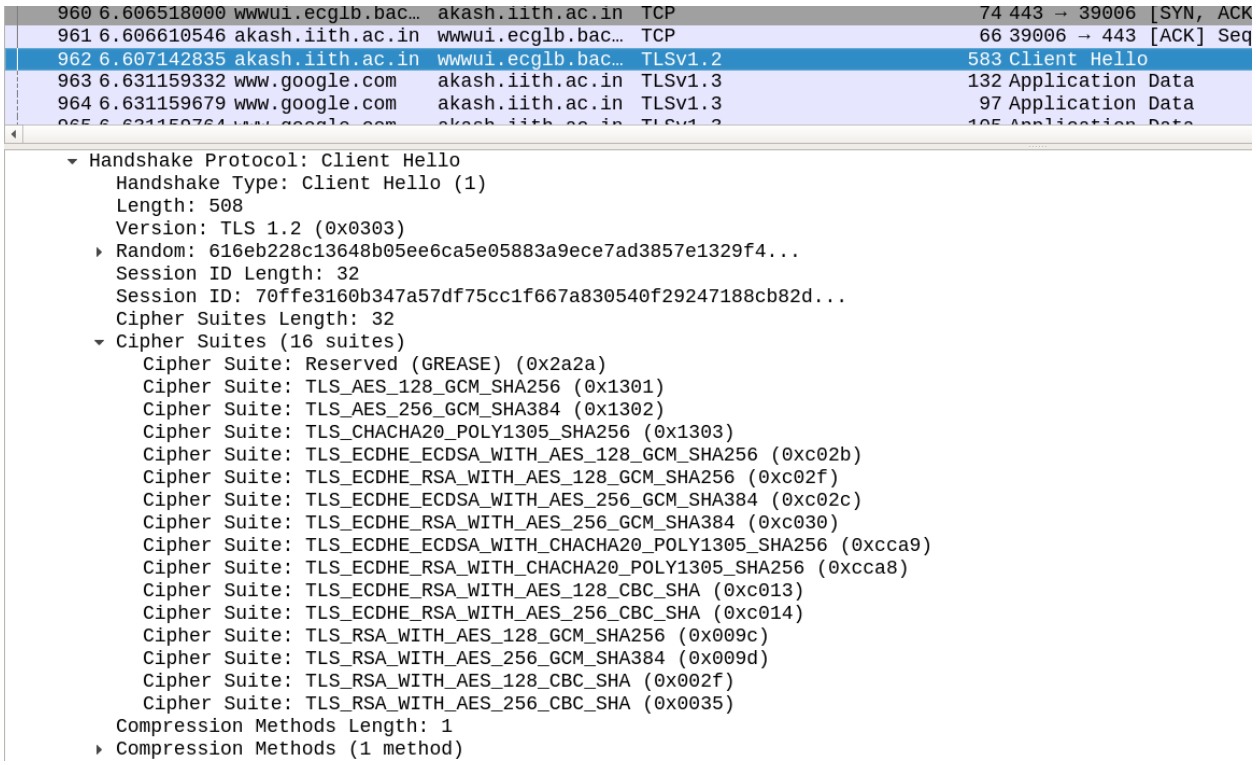
10. Content Type: Handshake (22) **(1byte)**

    Version: TLS 1.0 (0x0301) **(2bytes)**

    Length: 512 **(2bytes)**

4. **Cipher Suites in ClientHello Record: Look at the first two and the last cipher suites offered by the client and compare them. What cipher suite the server selected?**

Cipher Suites offered by client:



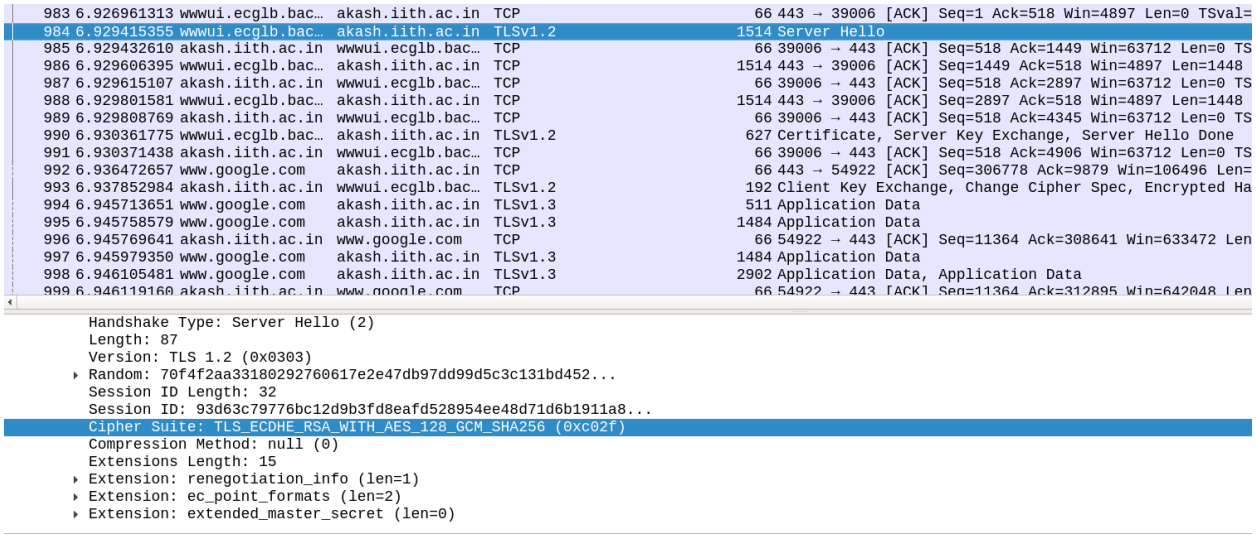
The first two cipher suites are :

- Cipher Suite: TLS_AES_128_GCM_SHA256 (0x1301)
- Cipher Suite: TLS_AES_256_GCM_SHA384 (0x1302)

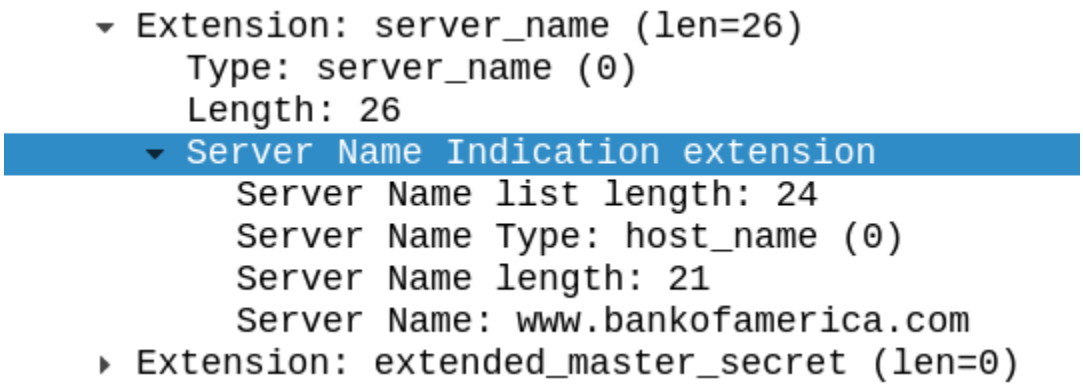
TLS_AES_256_GCM_SHA384 (0x1302) is the strongest cipher suite since it is using

AES_GCM_SHA384.

The server selected TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xco2f) cipher suite as shown in the below figure.

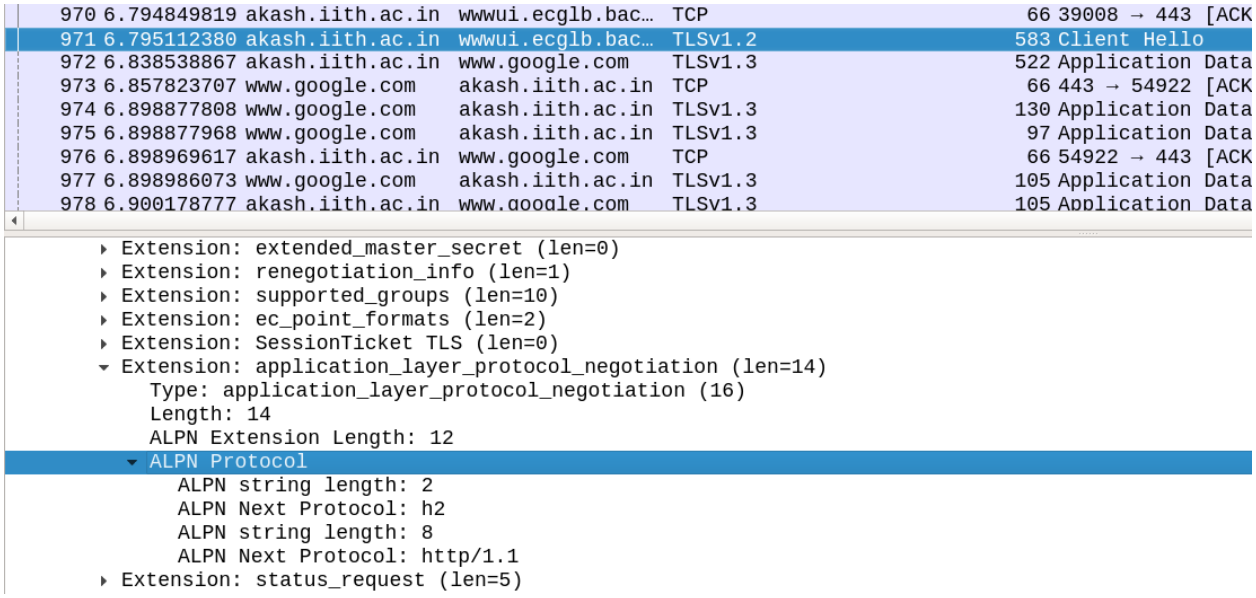


5. **What is the SNI value in ClientHello Record? What's its purpose? In other words, why is the client advertising it to the server?**

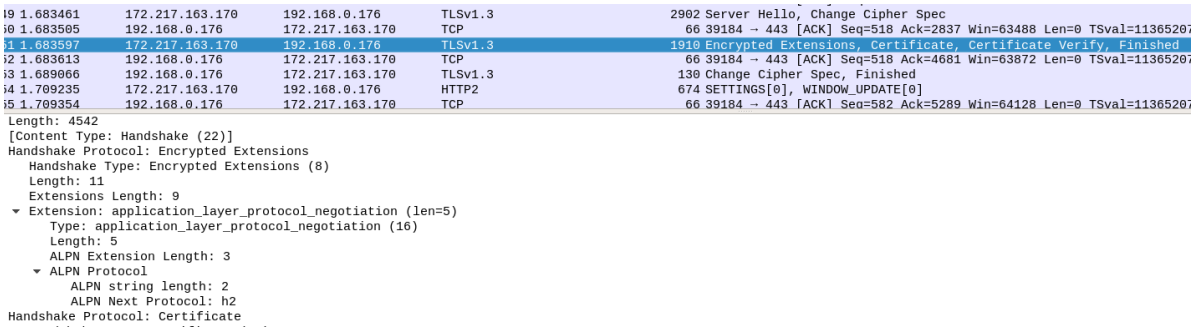


The fields in Server Name Indication extension (SNI) are as in the above snapshot. With the help of SNI extension, the server can host multiple TLS certificates for multiple websites under a single IP address. SNI extension contains the host name in it so as the websites can be uniquely identified.

6. **What is the ALPN value(s) in ClientHello Record? What's its purpose? Which one the server selected?**



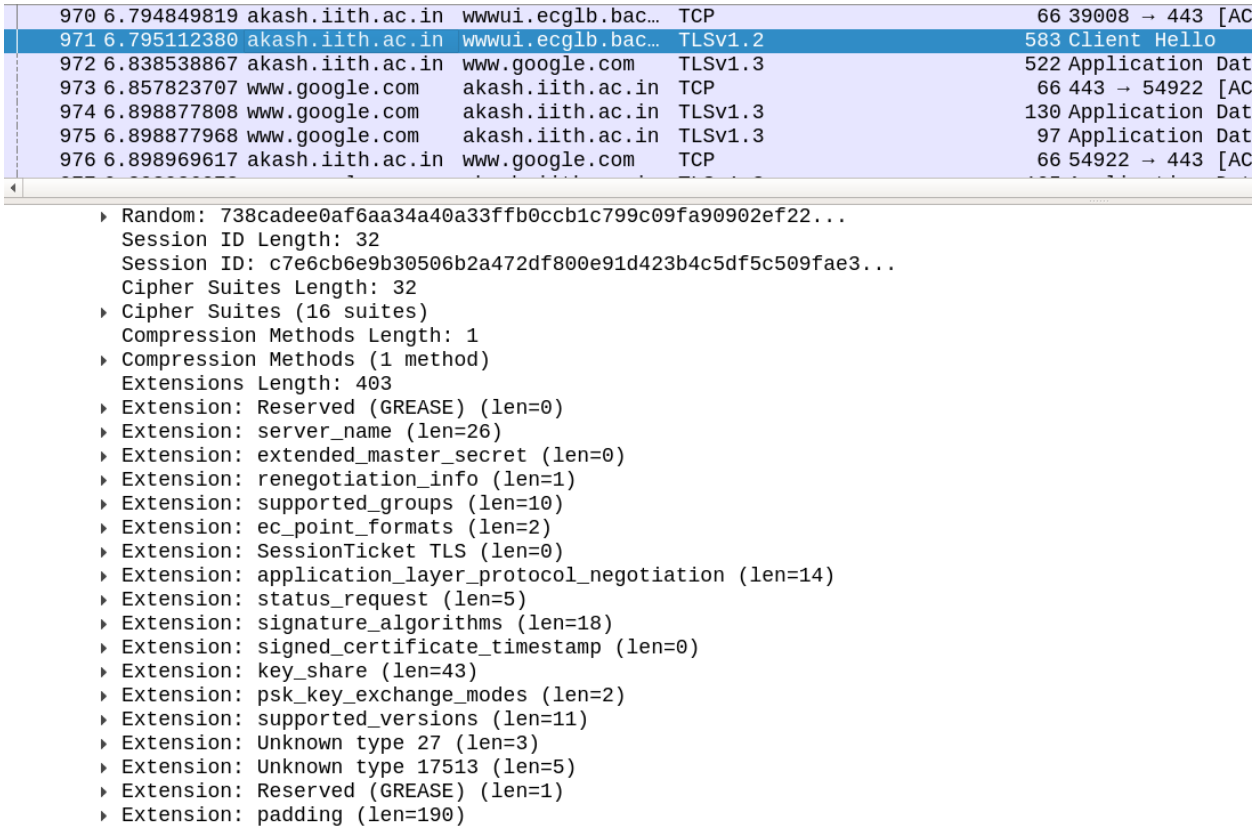
ALPN values in Client Hello are h2 and http/1.1.



Application-Layer Protocol Negotiation (ALPN) is a Transport Layer Security (TLS) extension that allows the application layer to negotiate which protocol should be performed over a secure connection in a manner that avoids additional round trips and which is independent of the application-layer protocols.

From the above figure we can see that the server had selected h2.

7. **Does the ClientHello contain status_request, supported_versions, psk_key_exchange_modes extensions? If so, what do they convey to the server?**



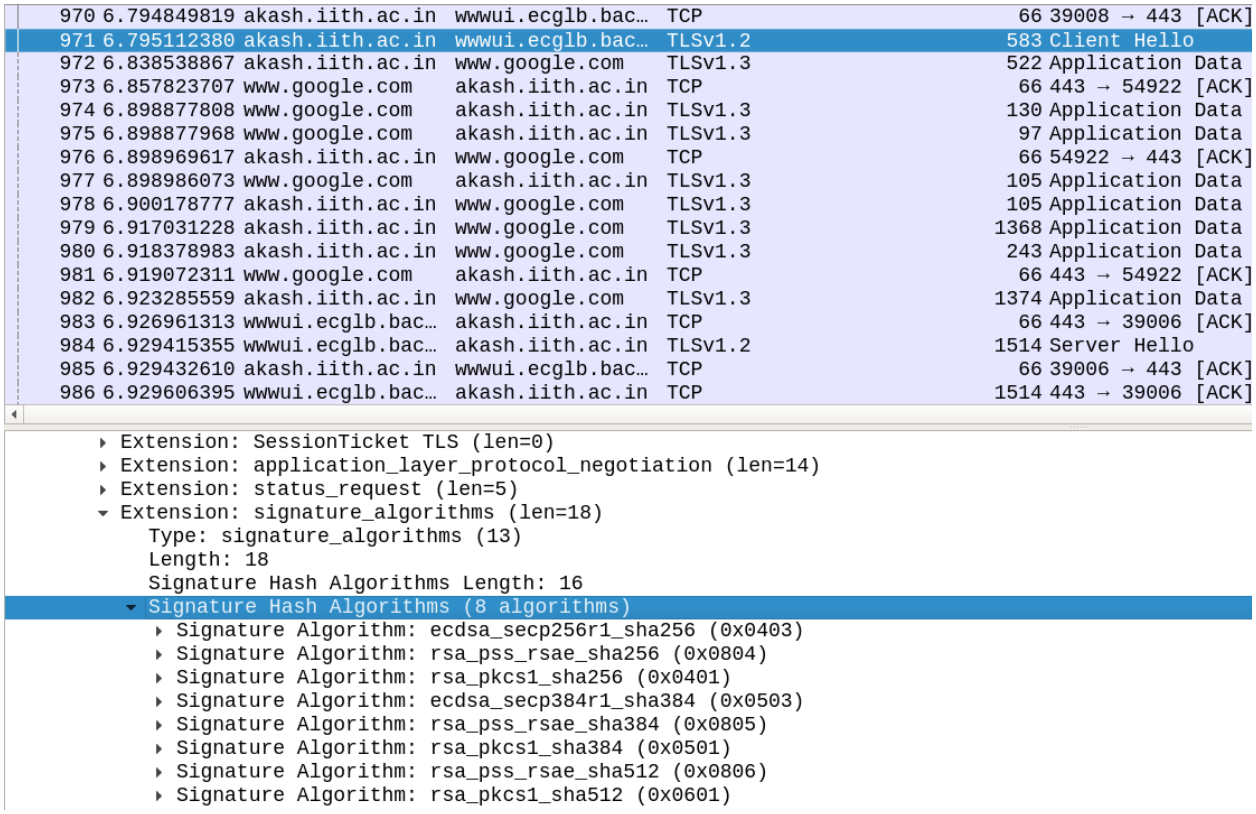
Yes, the ClientHello contains status_request, supported_versions, psk_key_exchange_modes extensions.

Status_request: Constrained clients may wish to use a certificate-status protocol such as OCSP [RFC2560] to check the validity of server certificates, in order to avoid transmission of CRLs and therefore save bandwidth on constrained networks. This extension allows for such information to be sent in the TLS handshake, saving round trips and resources.

Supported Versions: Supported Versions tell the server that the client supports a variety of TLS protocols offered in supported_versions extension.

psk_key_exchange_modes: The semantics of this extension are that the client only supports the use of PSKs with these modes, which restricts both the use of PSKs offered in this ClientHello and those which the server might supply via NewSessionTicket.

8. **Does ClientHello Record contain the Signature_algorithms extension? What's its purpose?**

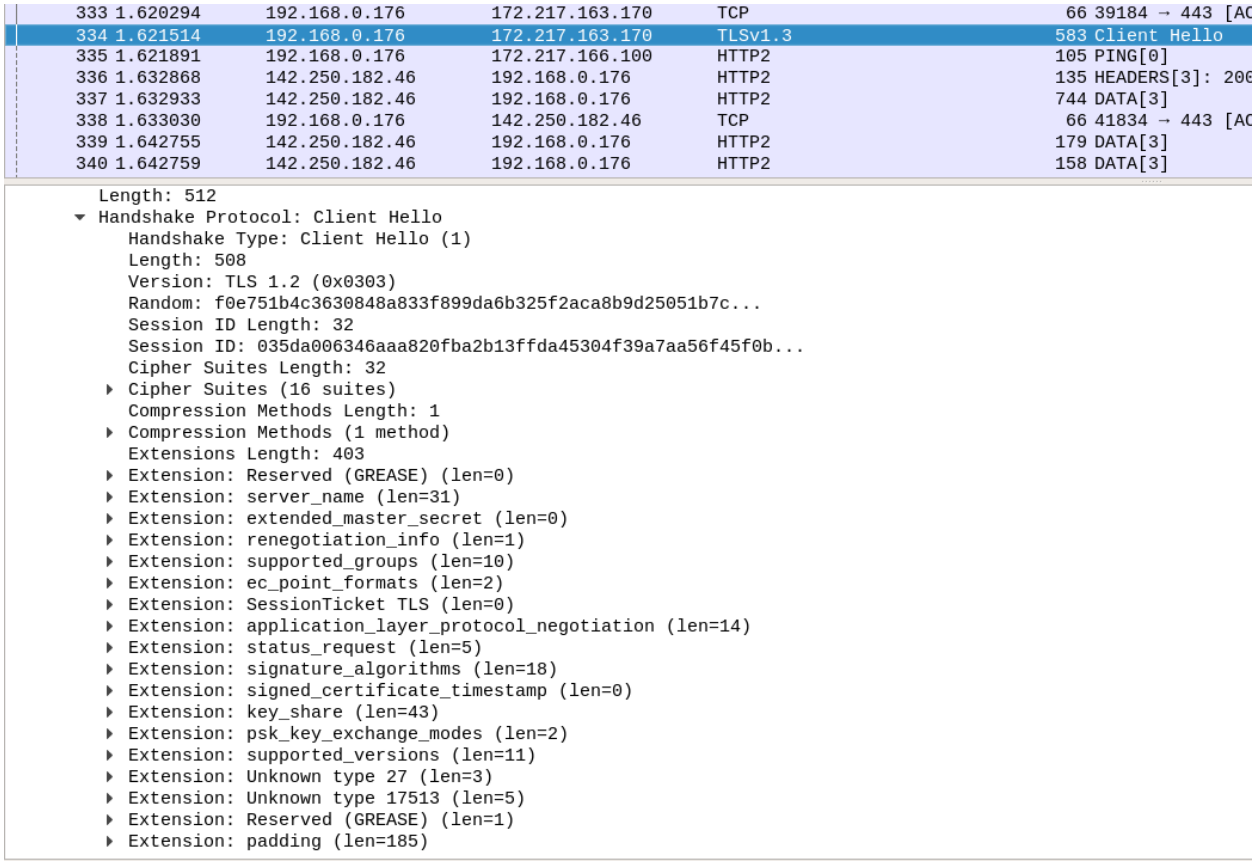


Yes, ClientHello Record contains the signature_algorithms extension. This extension provides the signature algorithms supported by clients out of which one algorithm is accepted and is used to sign all the TLS handshake messages.

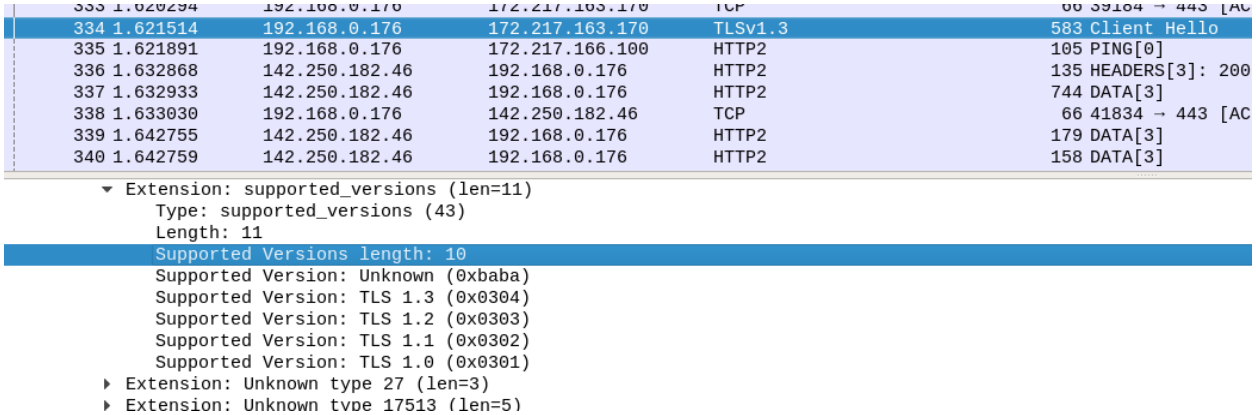
9. **Does the client offer any Random number, key share, Supported Groups and PSK in ClientHello Record? How will be these used by the Server?**

Yes, the below image shows that the client offers all of the Random number, key share, supported groups and PSK in ClientHello Record.

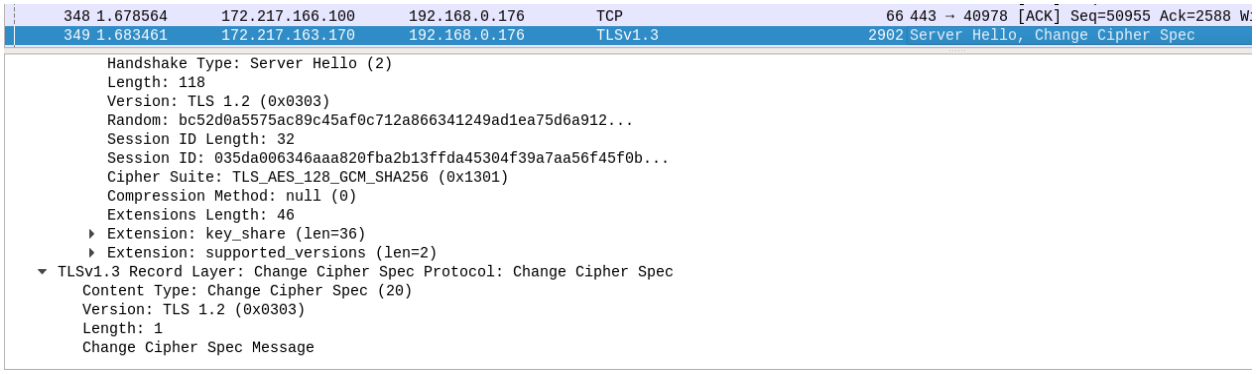
Random: This is a 32-byte random number. The client random and the server random are later used to generate the key for encryption.
key share: The "key_share" extension contains the endpoint's cryptographic parameters. Clients MAY send an empty client_shares vector in order to request group selection from the server, at the cost of an additional round trip.
Supported groups: When sent by the client, the "supported_groups" extension indicates the named groups which the client supports for key exchange, ordered from most preferred to least preferred.
PSK: These pre-shared keys are symmetric keys shared in advance among the client and server which can be established through out of band or session resumption.



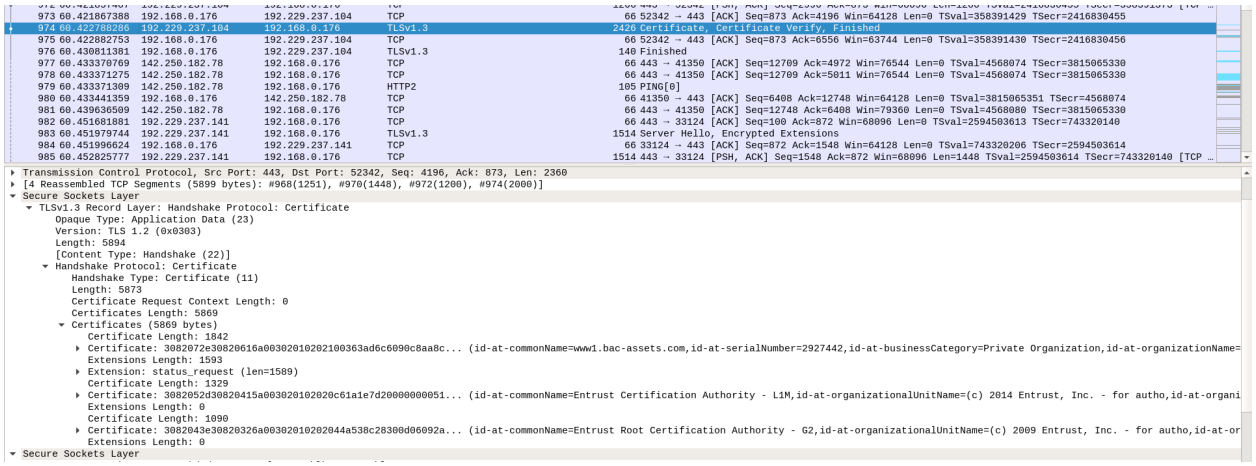
10. What TLS versions your browser/client is supporting? Which one the server selected?



As we can see above, browser/client supports TLS 1.0, 1.1, 1.2, 1.3 and the server selected, TLS 1.2 (figure is shown below)



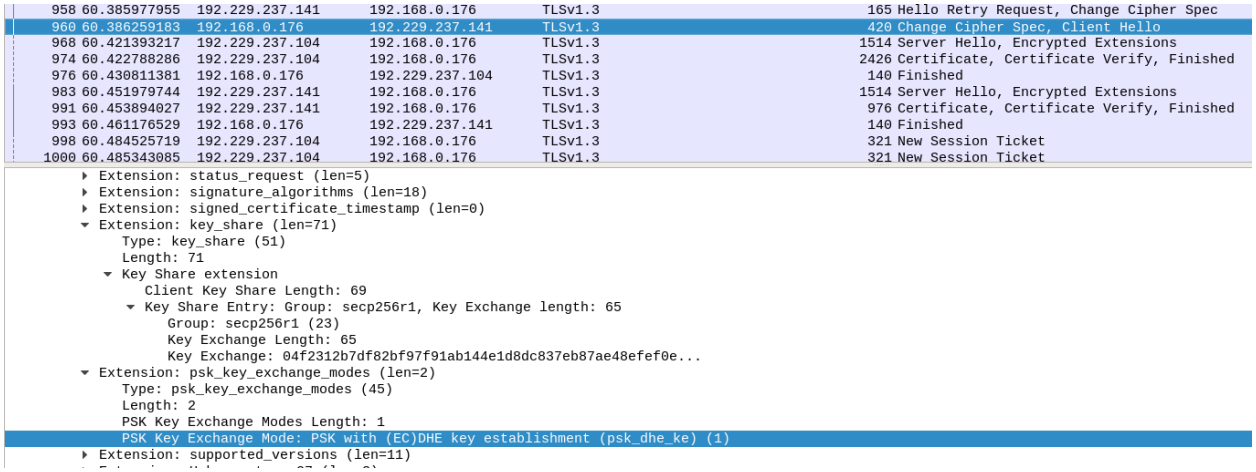
11. Look at Certificate Record from the server to the client: How many certificates did the server return and how are they related? Who is the issuer of the Bank's certificate? What type of public key the bank is using?

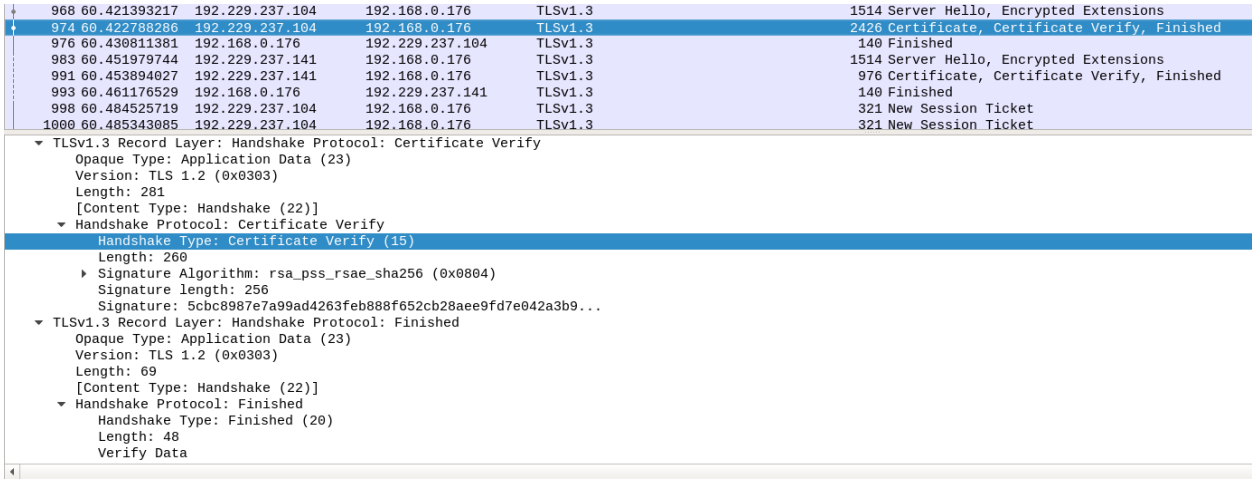


Server has returned 3 certificates in a chain of trust.
1. User- bankofamerica.com
2. Intermediate CA - Entrust Certification Authority
3. Root CA - Entrust Root Certification Authority

Issuer of the bank's certificate is Entrust Certification Authority. Bank is Using RSA public key.

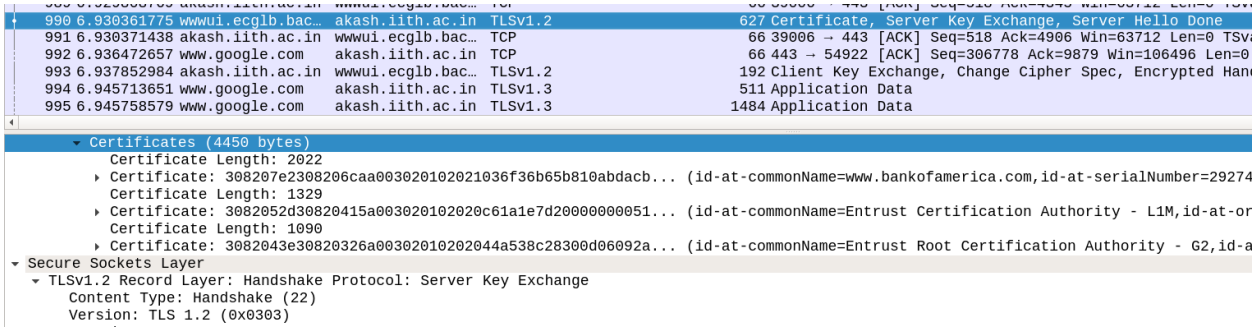
12. Comment on the key exchange algorithm agreed upon, what are the parameters that got exchanged between client and server to derive the session keys.





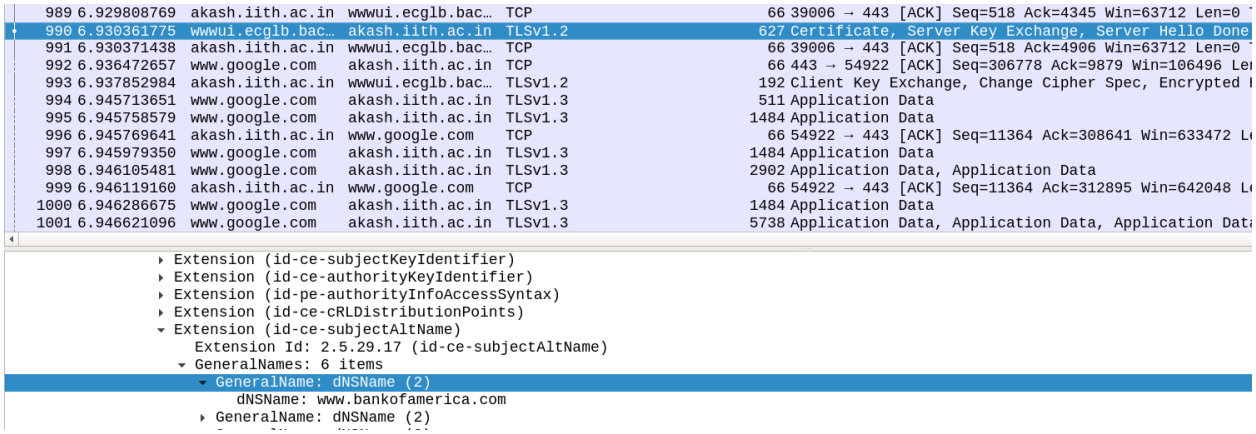
We observe that Client and Server exchange ECDHE key parameters i.e. Curve type, named curve, pubkey length and pubkey.

13. Which certificate type (DV/OV/EV) the bank is using?



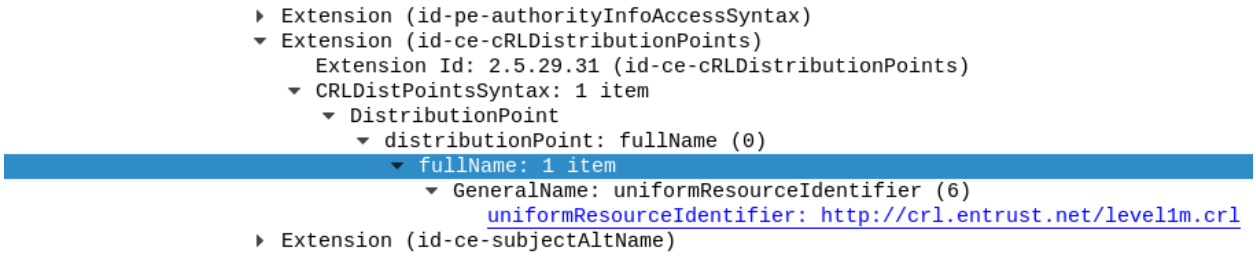
As the subject field contains jurisdictionCountryname, serial number and much more information, the certificate type bank is using is EV certificate.

14. Which certificate type (single or multi-domain or wild-card) the bank is using?



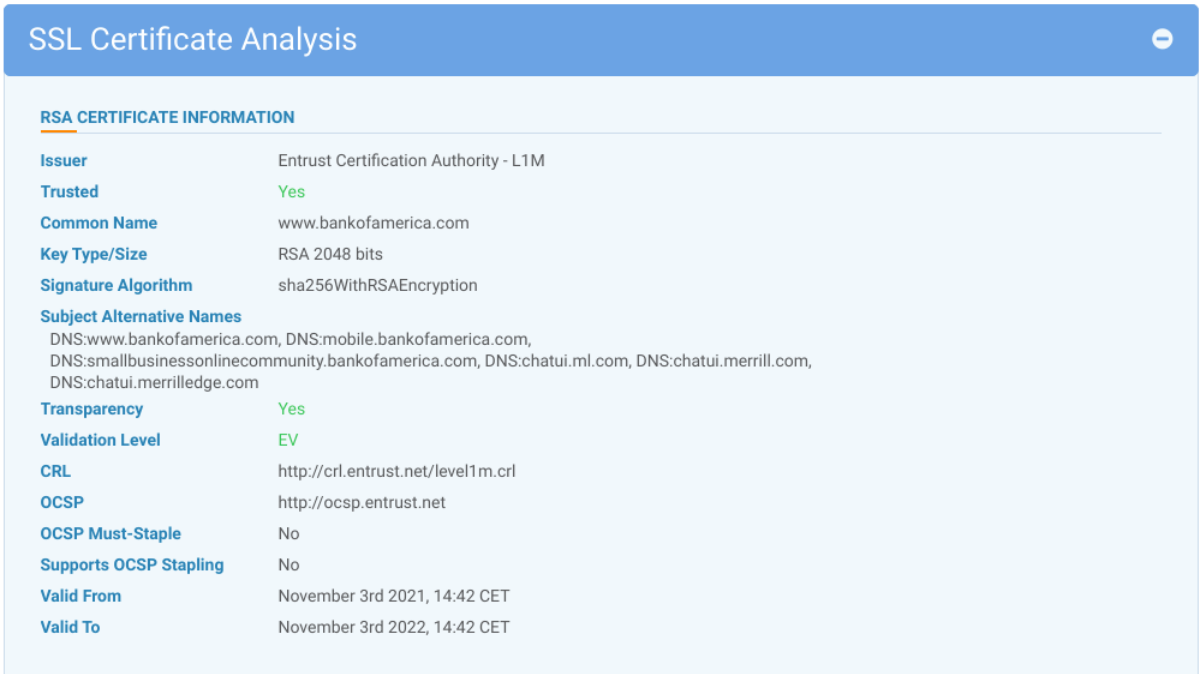
We observe from the id-ce-subjectAltName extension that the certificate is a single-domain certificate.

15. **How can the client check whether the certificate is revoked or not: OCSP/CRL? Does the server support OCSP stapling?**

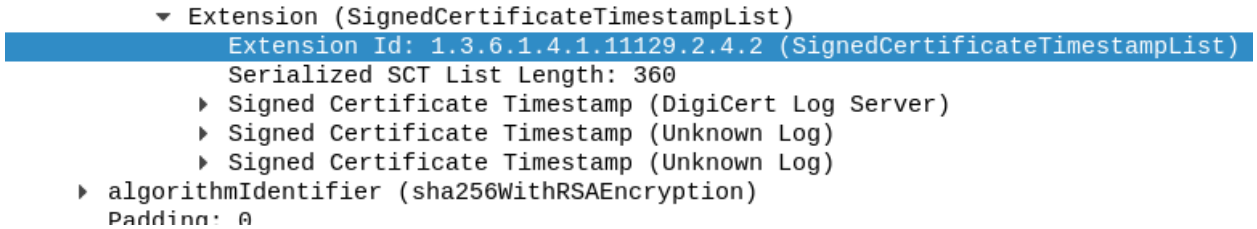


We can open the link provided in cRLDistributionPoints extension to check the OCSP status to see if the certificate is revoked or not.

No, the server doesn't support ocsp stapling as shown in the below screenshot.



16. **How many log servers logged the certificate of the bank? What role does the log server play in the Web PKI ecosystem? Refer: SCT extension.**



As we can see, there are 3 log servers which log the certificate of the bank. The role of log server in the Web PKI ecosystem is to maintain certificate transparency

17. **How is the application data being encrypted? Do the records containing application data include a separate MAC? Does Wireshark distinguish between the encrypted application data and the MAC?**
    - Application data is encrypted using “Symmetric encryption algorithm” (ECHDE in this case)
    - MAC is included in the records containing application data.
    - Wireshark cannot distinguish between encrypted applications data and MAC.
18. **Look at various keys logged in the file pointed to by the SSLKEYLOGFILE environment variable in your host OS and describe their usage. Also comment on how they are derived from nonces and other parameters using HKDF. Which entity in your system does this job on-the-fly?**

    The file contains 6 keys:

    1. **CLIENT_HANDSHAKE_TRAFFIC_SECRET:** This key is hex-encoded and used by the client for handshake.

    2. **SERVER_HANDSHAKE_TRAFFIC_SECRET**: This key is hex-encoded and used by the server for handshake.

    3. **CLIENT_TRAFFIC_SECRET_0**: It’s the client side’s first hex-encoded application traffic secret (for TLS 1.3).

    4. **SERVER_TRAFFIC_SECRET_0**: It’s the server side’s first hex-encoded application traffic secret (for TLS 1.3).

    5. **EXPORTER_SECRET**: The exporter secret encoded by the hex (For TLS 1.3).

    6. **CLIENT_RANDOM**: It is encoded as 96 hexadecimal characters, 48 bytes for the master secret.

19. **Do you see any support for session resumption in the trace? What do you find inside the session ticket, if it is used? Is it based on Session ID/Session ticket or**

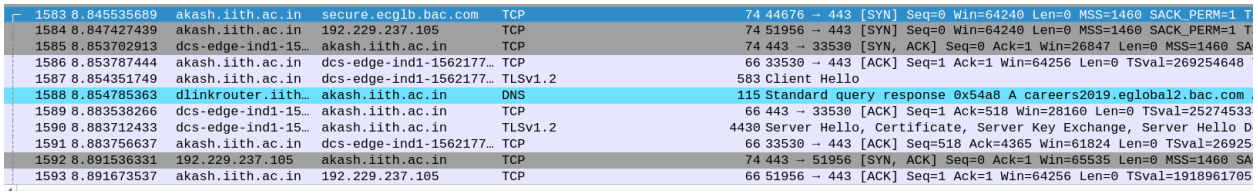
PSK based Session ticket? What role do the session IDs play in TLS 1.3?

Yes, there is support for the session resumption, as there is a New Session ticket issued by the server after the client sends the finished message. Session resumption here is based on Session Ticket. TLS Session ticket contains Lifetime hint, Nonce, Length and Session Ticket fields. TLS 1.3 replaces session IDs and session tickets with the concept of session resumption via pre-shared keys (PSK), hence session IDs doesn't play any role in TLS 1.3.

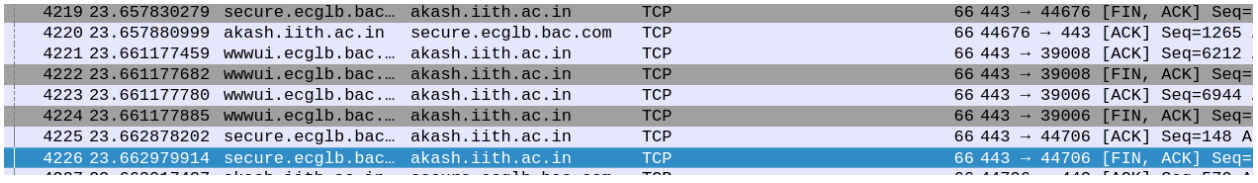
20. **How long does it take for TLS to establish a secure pipe? How much of it could be reduced when session resumption is used?**

    In TLS 1.2 it takes around 2 RTTs (approx 112ms) for establishing a secure connection, whereas TLS 1.3 takes 1 RTT. If resumption is used, TLS 1.2 takes 1 RTT whereas TLS 1.3 takes 0RTTs.

21. **What is the duration of the HTTPS session, how many IP packets are exchanged in the browsing session (starting from the first TCP SYN packet till TCP FIN packet)?**



1583	8.845535689	akash.iith.ac.in	secure.ecglb.bac.com	TCP	74	44676 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 T
1584	8.847427439	akash.iith.ac.in	192.229.237.105	TCP	74	51956 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 T
1585	8.853702913	dcs-edge-ind1-15…	akash.iith.ac.in	TCP	74	443 → 33530 [SYN, ACK] Seq=0 Ack=1 Win=26847 Len=0 MSS=1460 SA
1586	8.853787444	akash.iith.ac.in	dcs-edge-ind1-1562177…	TCP	66	33530 → 443 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=269254648
1587	8.854351749	akash.iith.ac.in	dcs-edge-ind1-1562177…	TLSv1.2	583	Client Hello
1588	8.854785363	dlinkrouter.iith…	akash.iith.ac.in	DNS	115	Standard query response 0x54a8 A careers2019.eglobal2.bac.com
1589	8.883538266	dcs-edge-ind1-15…	akash.iith.ac.in	TCP	66	443 → 33530 [ACK] Seq=1 Ack=518 Win=28160 Len=0 TSval=25274533
1590	8.883712433	dcs-edge-ind1-15…	akash.iith.ac.in	TLSv1.2	4430	Server Hello, Certificate, Server Key Exchange, Server Hello D
1591	8.883756637	akash.iith.ac.in	dcs-edge-ind1-1562177…	TCP	66	33530 → 443 [ACK] Seq=518 Ack=4365 Win=61824 Len=0 TSval=26925
1592	8.891536331	192.229.237.105	akash.iith.ac.in	TCP	74	443 → 51956 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 SA
1593	8.891673537	akash.iith.ac.in	192.229.237.105	TCP	66	51956 → 443 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1918961705



4219	23.657830279	secure.ecglb.bac…	akash.iith.ac.in	TCP	66	443 → 44676 [FIN, ACK] Seq=
4220	23.657880999	akash.iith.ac.in	secure.ecglb.bac.com	TCP	66	44676 → 443 [ACK] Seq=1265
4221	23.661177459	wwwui.ecglb.bac.…	akash.iith.ac.in	TCP	66	443 → 39008 [ACK] Seq=6212
4222	23.661177682	wwwui.ecglb.bac.…	akash.iith.ac.in	TCP	66	443 → 39008 [FIN, ACK] Seq=
4223	23.661177780	wwwui.ecglb.bac.…	akash.iith.ac.in	TCP	66	443 → 39006 [ACK] Seq=6944
4224	23.661177885	wwwui.ecglb.bac.…	akash.iith.ac.in	TCP	66	443 → 39006 [FIN, ACK] Seq=
4225	23.662878202	secure.ecglb.bac…	akash.iith.ac.in	TCP	66	443 → 44706 [ACK] Seq=148 A
4226	23.662979914	secure.ecglb.bac…	akash.iith.ac.in	TCP	66	443 → 44706 [FIN, ACK] Seq=

Total time of the browsing session = 23.662979914 - 8.845535689 = 14.817444225 sec.

Total number of IP packets exchanged = 4226-1592 = 2634

22. How many TLS connections are established?

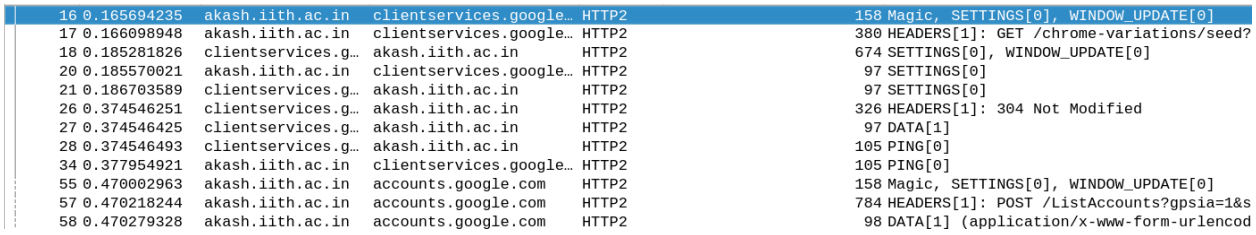
For calculating the number of connections established I have counted the number of “Finish” messages because it marks the end of session establishment in TLS.

Total count: 11

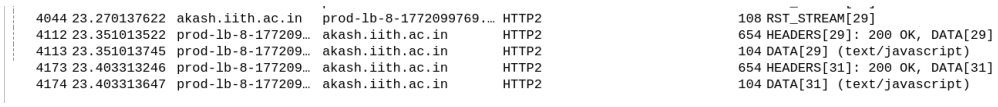
Count with bank of america: 5

23. How many HTTP request/response packets are exchanged in the browsing session? Identify the packet(s) that carried the response that included the Netbanking LOG-IN page of the bank. Do these response messages carry any security related directives like XSS, sameorigin, HSTS?

Total of 4158 packets are exchanged in the browsing session.

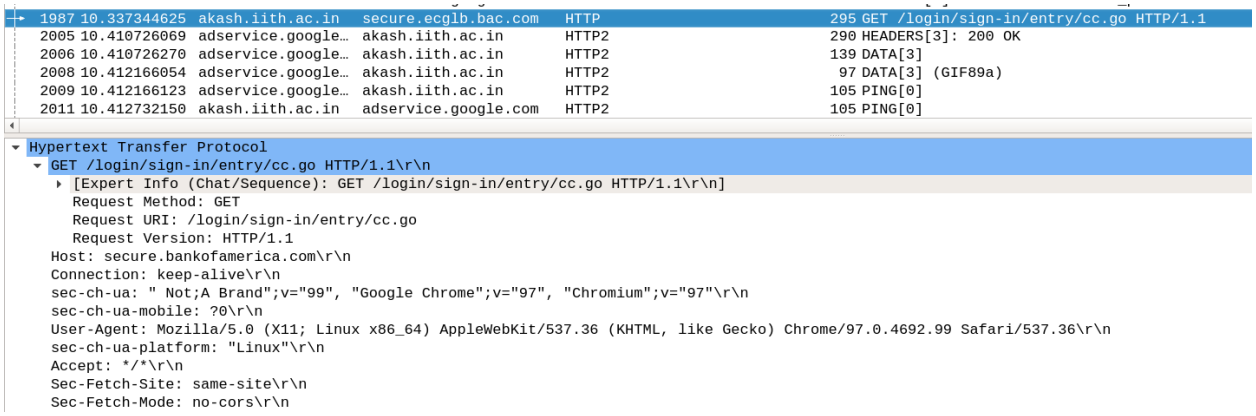


16	0.165694235	akash.iith.ac.in	clientservices.google…	HTTP2	158	Magic, SETTINGS[0], WINDOW_UPDATE[0]
17	0.166098948	akash.iith.ac.in	clientservices.google…	HTTP2	380	HEADERS[1]: GET /chrome-variations/seed?
18	0.185281826	clientservices.g…	akash.iith.ac.in	HTTP2	674	SETTINGS[0], WINDOW_UPDATE[0]
20	0.185570021	akash.iith.ac.in	clientservices.google…	HTTP2	97	SETTINGS[0]
21	0.186703589	clientservices.g…	akash.iith.ac.in	HTTP2	97	SETTINGS[0]
26	0.374546251	clientservices.g…	akash.iith.ac.in	HTTP2	326	HEADERS[1]: 304 Not Modified
27	0.374546425	clientservices.g…	akash.iith.ac.in	HTTP2	97	DATA[1]
28	0.374546493	clientservices.g…	akash.iith.ac.in	HTTP2	105	PING[0]
34	0.377954921	akash.iith.ac.in	clientservices.google…	HTTP2	105	PING[0]
55	0.470002963	akash.iith.ac.in	accounts.google.com	HTTP2	158	Magic, SETTINGS[0], WINDOW_UPDATE[0]
57	0.470218244	akash.iith.ac.in	accounts.google.com	HTTP2	784	HEADERS[1]: POST /ListAccounts?gpsia=1&s
58	0.470279328	akash.iith.ac.in	accounts.google.com	HTTP2	98	DATA[1] (application/x-www-form-urlencod



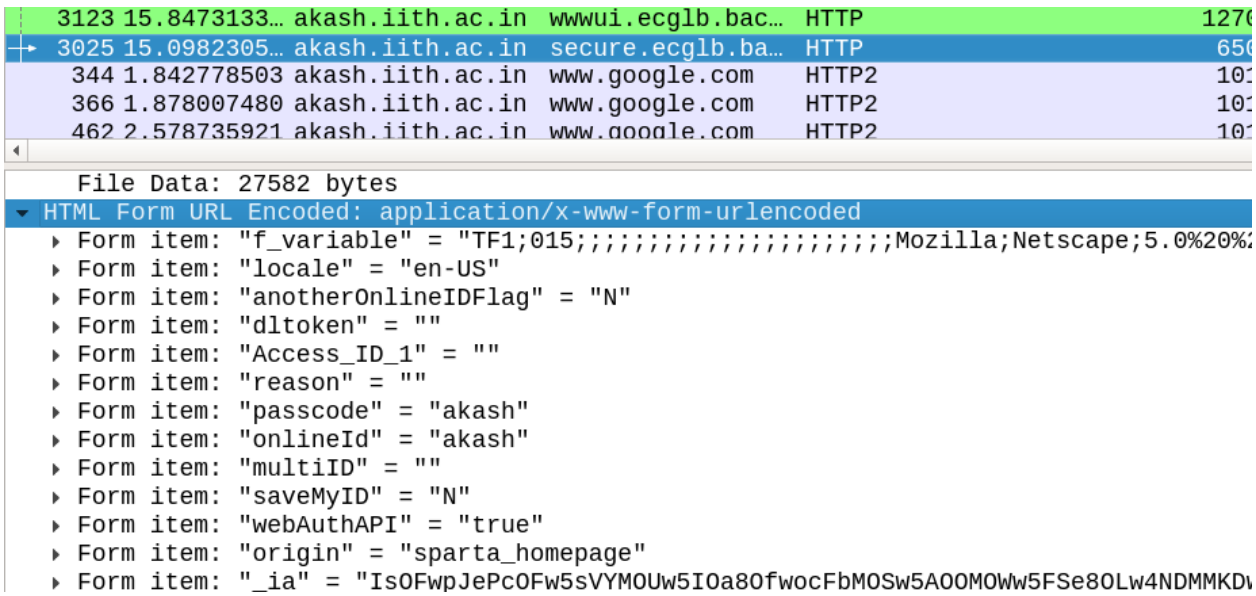
4044	23.270137622	akash.iith.ac.in	prod-lb-8-1772099769.…	HTTP2	108	RST_STREAM[29]
4112	23.351013522	prod-lb-8-177209…	akash.iith.ac.in	HTTP2	654	HEADERS[29]: 200 OK, DATA[29]
4113	23.351013745	prod-lb-8-177209…	akash.iith.ac.in	HTTP2	104	DATA[29] (text/javascript)
4173	23.403313246	prod-lb-8-177209…	akash.iith.ac.in	HTTP2	654	HEADERS[31]: 200 OK, DATA[31]
4174	23.403313647	prod-lb-8-177209…	akash.iith.ac.in	HTTP2	104	DATA[31] (text/javascript)

Below is the packet that carries the response that included the Net Banking LOG-IN page of the bank.



Fields under Hypertext Transfer Protocol carry security related directives.

24. **Identify the HTTP packet(s) that carried LOG-IN credentials supplied by you. Look at the raw bytes displayed in the wireshark GUI and identify strings that carried your LOG-IN credentials. Are you able to find both user id and password in the raw packet capture?1**
    a. **It's important that you only keyed in some arbitrary user id and password as part of this assignment for more safety!**

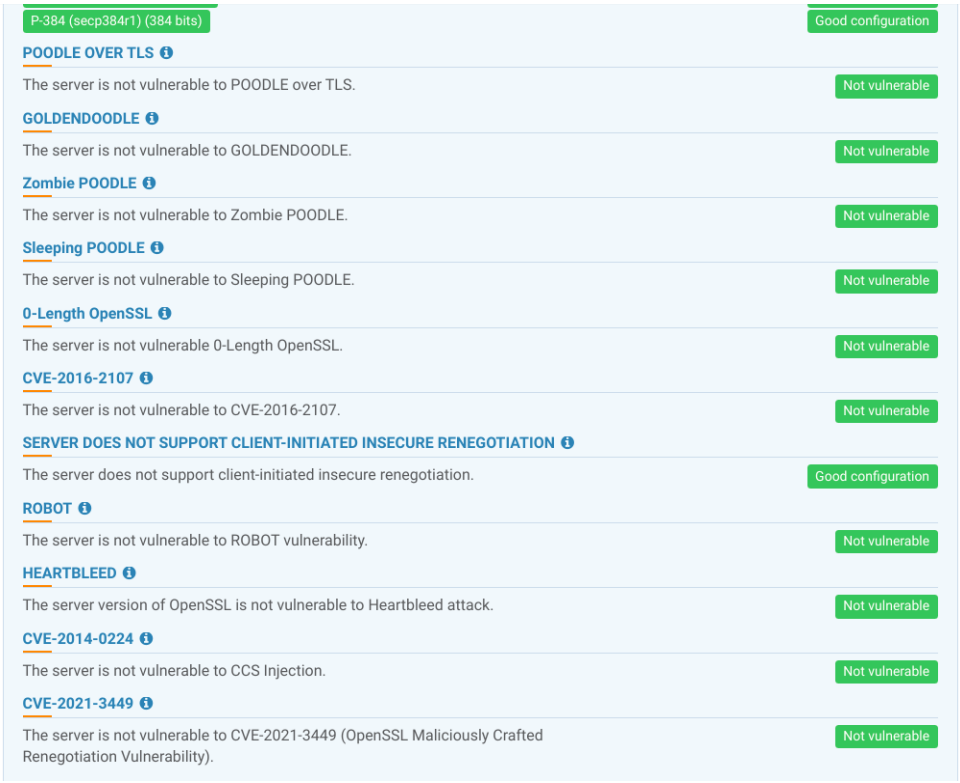


Yes, the userid and password are both visible in textual format after decrypting using sslkeyfile.

25. **Generate an SSL report of the bank using [SSL Server Test (Powered by Qualys]**

SSL Labs) and summarise what security features are implemented by the bank's web server for improved online banking by its customers. Does the report flag any issues with the security of the bank?

From the report shown below, some security features implemented by the bank are server supported secure Renegotiation, TLS_FALLBACK_SCSV usage to prevent downgrade attacks, heartbleed extension disabled and prevention from BEAST and POODLE attacks. It doesn't support OCSP stapling which is non-compliant with NIST guidelines. The server supports a client-initiated secure renegotiation that may be unsafe and allow Denial of Service attacks.



SSL Certificate Analysis

RSA CERTIFICATE INFORMATION

Issuer	Entrust Certification Authority - L1M
Trusted	Yes
Common Name	www.bankofamerica.com
Key Type/Size	RSA 2048 bits
Signature Algorithm	sha256WithRSAEncryption
Subject Alternative Names	DNS:www.bankofamerica.com, DNS:mobile.bankofamerica.com, DNS:smallbusinessonlinecommunity.bankofamerica.com, DNS:chatui.ml.com, DNS:chatui.merrill.com, DNS:chatui.merrilledge.com
Transparency	Yes
Validation Level	EV
CRL	http://crl.entrust.net/level1m.crl
OCSP	http://ocsp.entrust.net
OCSP Must-Staple	No
Supports OCSP Stapling	No
Valid From	November 3rd 2021, 14:42 CET
Valid To	November 3rd 2022, 14:42 CET

26. Comment on and explain anything else that you found interesting in the trace.

The server doesn't provide OCSP stapling but it provides Session resumption as it issued a new ticket.

Note 1: Bonus 30 marks if you complete TLS and HTTPS decrypting using openssl and shell/python scripting. Make sure your code is well documented.

Note 2: Add screenshots of relevant in your report in order to prove that the capture trace used for analysis is indeed of your own and makes the evaluation easy for TAs!!

PS: What's Wireshark++? Wireshark + Key log file!

Deliverables in GC as a tar ball:

- A readable PDF Report with name "TLSAsg-<RollNo>.PDF"
- Deliverables 1-3 (refer page-1 of the assignment)
- SSL Key Log File

- Shell scripts written for openssl based decrypting of TLS and HTTPS session (optional, to get bonus marks)

References:

1. Article: K50557518 - Decrypt SSL traffic with the SSLKEYLOGFILE environment variable on Firefox or Google Chrome using Wireshark (f5.com)
2. Wireshark Tutorial: Decrypting HTTPS Traffic (Includes SSL and TLS) (paloaltonetworks.com)
3. Decrypting TLS Streams With Wireshark: Part 1 | Didier Stevens
4. http://www.motobit.com/util/base64-decoder-encoder.asp
5. Dissecting TLS Using Wireshark (catchpoint.com)
6. https://tls13.ulfheim.net/
7. https://www.davidwong.fr/tls13/
8. SSL Server Test (Powered by Qualys SSL Labs)
9. Application-Layer Protocol Negotiation - Wikipedia
10. RFC 6066 - Transport Layer Security (TLS) Extensions
11. RFC 8446 - The Transport Layer Security (TLS) Protocol Version 1.3