CS 315: Computer Networks Lab
Spring 2024-25, IIT Dharwad
Assignment-12
Wireshark Lab: 802.11 WiFi & TLS
April 7, 2025

Chidurala Tejaswini
(220010012 / CS22BT012)

Part 0: Paste a screenshot of your system IP address, using ipconfig (on Windows) or ifconfig (on Mac and Linux), and fill out this Google form to submit the details of your system. The same system must be used to attempt all exercises of this lab.

```
user@sysad-HP-Elite-Tower-600-G9-Desktop-PC:~$ ifconfig
eno1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.240.118.97  netmask 255.255.248.0  broadcast 10.240.119.255
        inet6 fe80::1d6b:1bfb:2bd6:ef0d  prefixlen 64  scopeid 0x20<link>
        ether e0:73:e7:0a:99:9a  txqueuelen 1000  (Ethernet)
        RX packets 186173  bytes 143688016 (143.6 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 55113  bytes 7668580 (7.6 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 19  memory 0x80900000-80920000

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 4723  bytes 521713 (521.7 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 4723  bytes 521713 (521.7 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Introduction

In this lab, we'll investigate the 802.11 wireless network protocol. In all of the Wireshark labs thus far, we've captured frames on a wired Ethernet connection. Here, since 802.11 is a wireless link-layer protocol, we'll be capturing frames "in the air." Unfortunately, some device drivers for wireless 802.11 NICs still don't provide the hooks to capture/copy received 802.11 frames for use in Wireshark. Thus, in this lab, we'll provide a trace of captured 802.11 frames for you to analyze and assume in the questions below that you are using this trace.

Getting Started

Download the file `WiFi_Trace.pcap` from the link. This trace was collected using AirPcap and Wireshark running on a computer in the home network of one of the authors, consisting of a Linksys 802.11g combined access point/router, with two wired PCs and one wireless host PC attached to the access point/router. The author is fortunate to have other access points in neighbouring houses available as well. In this trace file, we'll see frames captured on channel 6. Since the host and AP that we are interested in are not the only devices using channel 6, we'll see a lot of frames that we're not interested in

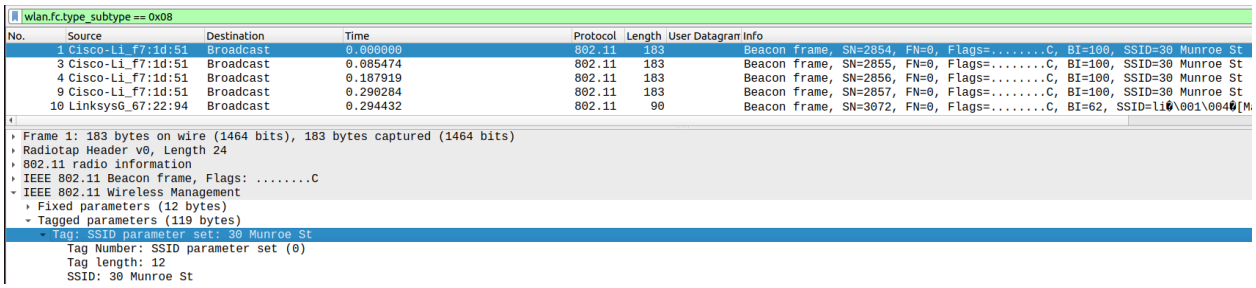
for this lab, such as beacon frames advertised by a neighbour's AP also operating on channel 6. The recorded wireless host activities in the trace file are as follows:

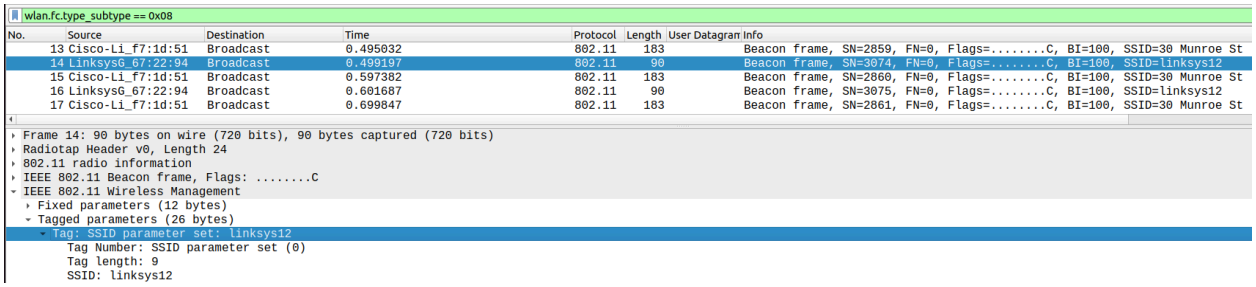
- At the start of the trace, the host is already connected to the *30 Munroe St access point (AP)*.
- At t = 24.82, the host sends an HTTP request to `http://gaia.cs.umass.edu/wireshark-labs/alice.txt`, with the destination IP address 128.119.245.12.
- At t = 32.82, the host makes another HTTP request to `http://www.cs.umass.edu`, which resolves to 128.119.240.19.
- At t = 49.58, the host disconnects from the *30 Munroe St AP* and attempts to connect to *linksys_ses_24086*, a secured access point. However, the connection attempt is unsuccessful.
- At t = 63.0, after failing to associate with *linksys_ses_24086*, the host reconnects to the *30 Munroe St AP*.

Part-1: Beacon Frames

Recall that beacon frames are used by an 802.11 AP to advertise its existence. To answer some of the questions below, you'll want to look at the details of the "IEEE 802.11" frame and subfields in the middle Wireshark window.

1. **What are the SSIDs of the two access points that are issuing most of the beacon frames in this trace?**

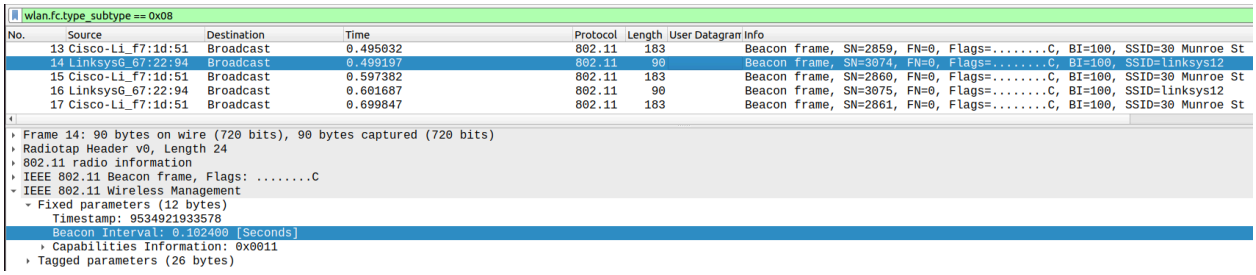


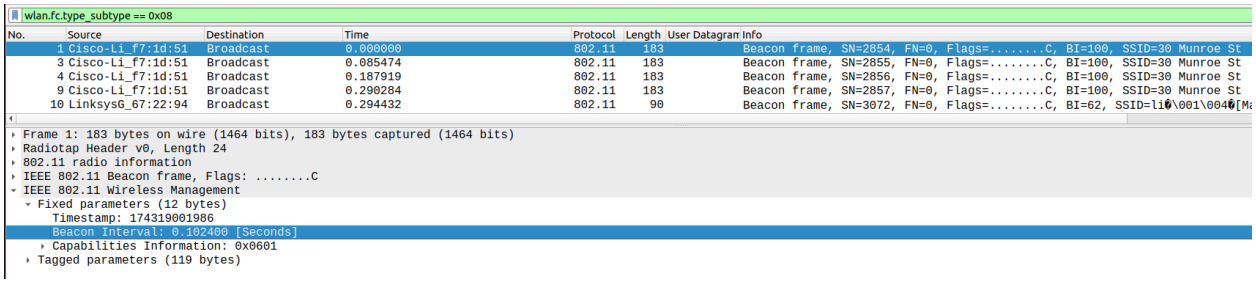


The SSIDs of the two access points that are issuing most of the beacon frames in this trace are:

Access Point	SSID
Munroe ST	30 Munroe St
link_ses_24086	linksys12

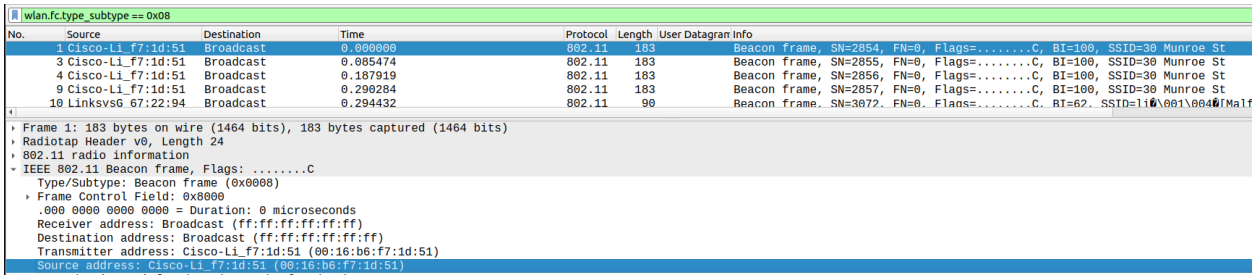
2. **What are the 'beacon interval' field values in the *linksys_ses_24086* access point and the *30 Munroe St.* access point?**





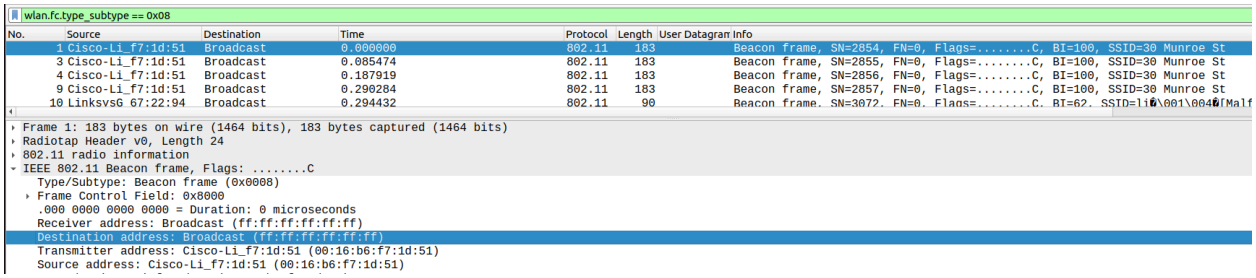
The 'beacon interval' field values in the linksys_ses_24086 access point and the 30 Munroe St. access point are 0.102400 seconds.

3. **What (in hexadecimal notation) is the source MAC address on the beacon frame from *30 Munroe St*?**



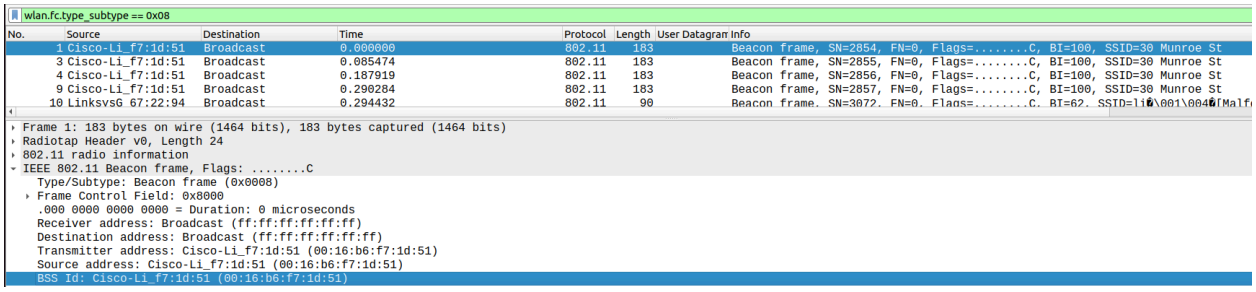
Source MAC address: Cisco-Li_f7:1d:51 (00:16:b6:f7:1d:51)

4. **What (in hexadecimal notation) is the destination MAC address on the beacon frame from *30 Munroe St*?**



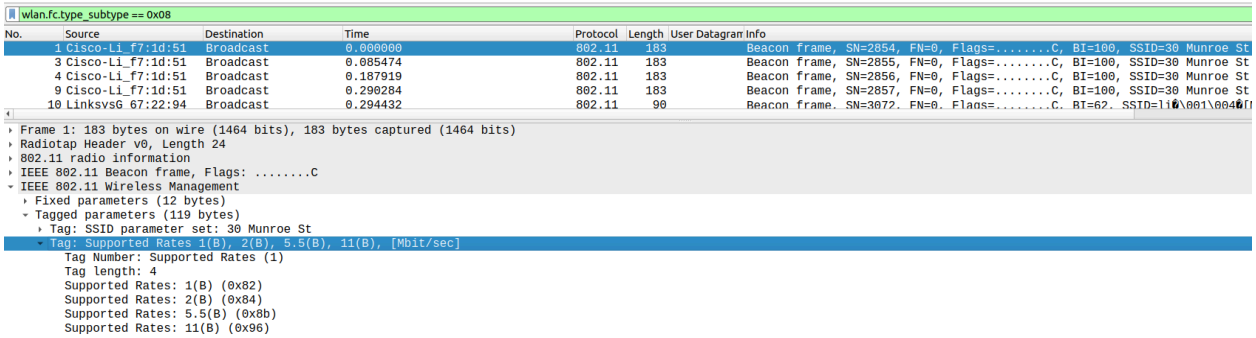
Destination MAC address: Broadcast (ff:ff:ff:ff:ff:ff)
→indicating a **broadcast** beacon frame

5. **What (in hexadecimal notation) is the MAC BSS id on the beacon frame from *30 Munroe St*?**



- BSS Id: Cisco-Li_f7:1d:51 (00:16:b6:f7:1d:51)
- **BSS ID** will often be the same as the Source MAC for APs.

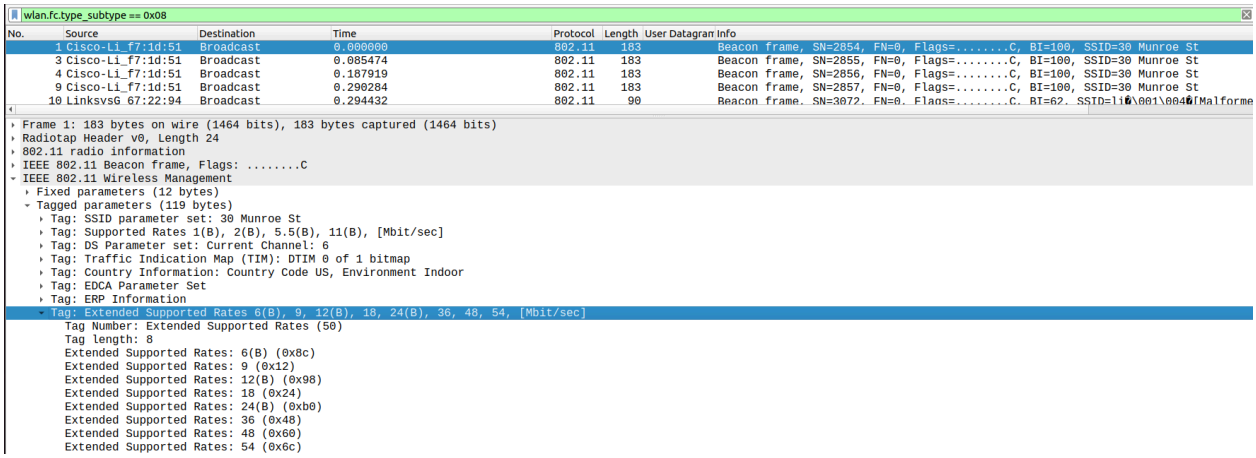
6. **The beacon frames from the *30 Munroe St* access point advertise that the access point can support four data rates and eight additional "extended supported rates." What are these rates?**



Supported Data rates are

- 1 (B), 2 (B), 5.5 (B), 11(B) [Mbits/sec]

The (B) indicates **basic rates** that must be supported by all devices in the network.



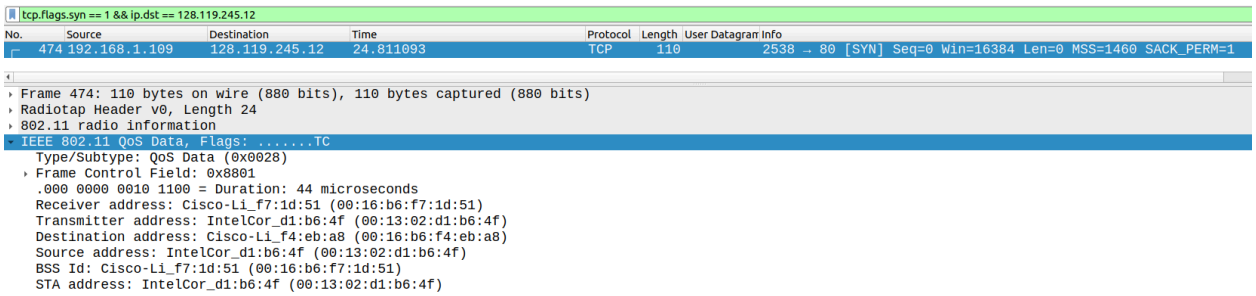
Extended Supported Rates are

- 6 (B) , 9, 12(B), 18, 24(B), 36, 48, 54 [Mbit/sec]

Part-2: Data Transfer

Since the trace starts with the host already associated with the AP, let's first look at data transfer over an 802.11 association before looking at AP association/disassociation. Recall that in this trace, at *t = 24.82*, the host makes an HTTP request to `http://gaia.cs.umass.edu/wireshark-labs/alice.txt`. The IP address of `gaia.cs.umass.edu` is 128.119.245.12. Then, at *t=32.82,* the host makes an HTTP request to `http://www.cs.umass.edu`.

1. **Find the 802.11 frame containing the SYN TCP segment for this first TCP session (that downloads `alice.txt`). What are three MAC address fields in the 802.11 frame? Which MAC address in this frame corresponds to the wireless host (give the hexadecimal representation of the MAC address for the host)? To the access point? To the first-hop router? What is the IP address of the wireless host sending this TCP segment? What is the destination IP address?**



Frame containing the SYN TCP segment for this first TCP session (that downloads alice.txt):

- Frame Number: 474

Three MAC Address Fields in the 802.11 Frame:

1. Receiver Address (Access Point): `00:16:b6:f7:1d:51`

2. Transmitter Address (Wireless Host): `00:13:02:d1:b6:4f`
3. Destination Address (First-Hop Router): `00:16:b6:f4:eb:a8`
4. (Also shown: Source Address = same as Transmitter)

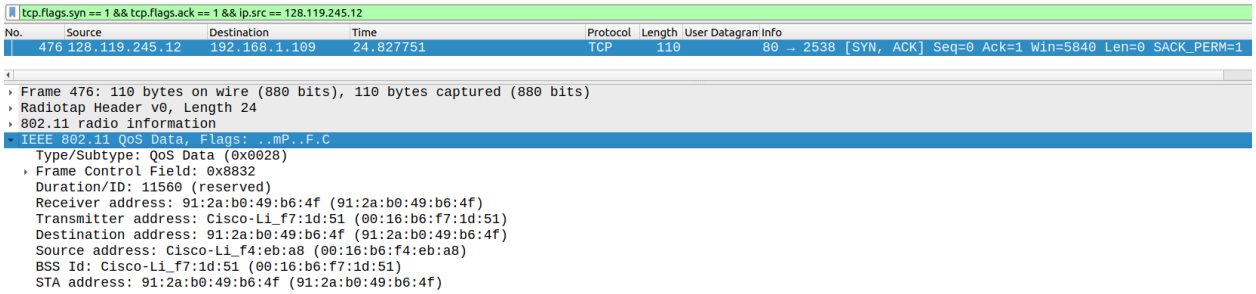
The MAC address corresponding to:

- **Wireless Host MAC**: `00:13:02:d1:b6:4f` (Transmitter)
- **Access Point MAC**: `00:16:b6:f7:1d:51` (Receiver)
- **First-Hop Router MAC**: `00:16:b6:f4:eb:a8` (Destination)

IP Address Information:

- **Source IP (Wireless Host)**: `192.168.1.109`
- **Destination IP (Server)**: `128.119.245.12`

2. **Find the 802.11 frame containing the SYNACK segment for this TCP session. What are three MAC address fields in the 802.11 frame? Which of these are the MAC addresses corresponding to the host sending SYNACK, destination and BSS? What is the IP address of the server sending the TCP SYNACK?**



Frame containing the SYNACK TCP segment:
Frame Number containing the SYNACK segment for this TCP session: 476

The three MAC address fields in the 802.11 frame are:

- **Receiver Address**: `91:2a:b0:49:b6:4f`
- **Transmitter Address**: `00:16:b6:f7:1d:51`
- **Source Address**: `00:16:b6:f4:eb:a8`

The MAC address corresponding to:

- **Host sending SYNACK (Server-side MAC)**: `00:16:b6:f4:eb:a8`(Source Address)
- **Destination (Wireless host/client MAC)**: `91:2a:b0:49:b6:4f`(Receiver Address)
- **BSS (Basic Service Set) /(Access Point MAC)**: `00:16:6b:f7:1d:51`(Transmitter Address)

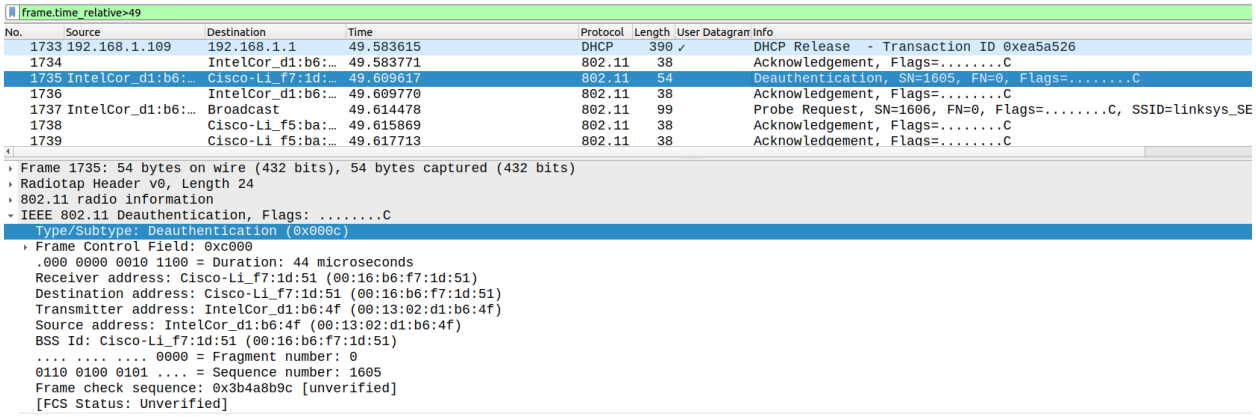
IP address of the server sending the TCP SYNACK:

- **Source IP Address**: 128.119.245.12

Part-3: Association/Disassociation

In the text, a host must first *associate* with an access point before sending data. Association in 802.11 is performed using the ASSOCIATE REQUEST frame (sent from the host to AP, with a frame type 0 and subtype 0) and the ASSOCIATE RESPONSE frame (sent by the AP to a host with a frame type 0 and subtype of 1, in response to a received ASSOCIATE REQUEST).

1. **What two actions are taken (i.e., frames are sent) by the host in the trace just after *t=49*, to end the association with the *30 Munroe St* AP that was initially in place when trace collection began? (Hint: one is an IP-layer action, and one is an 802.11-layer action). Looking at the 802.11 specification, is there another frame that you might have expected to see, but don't see here?**

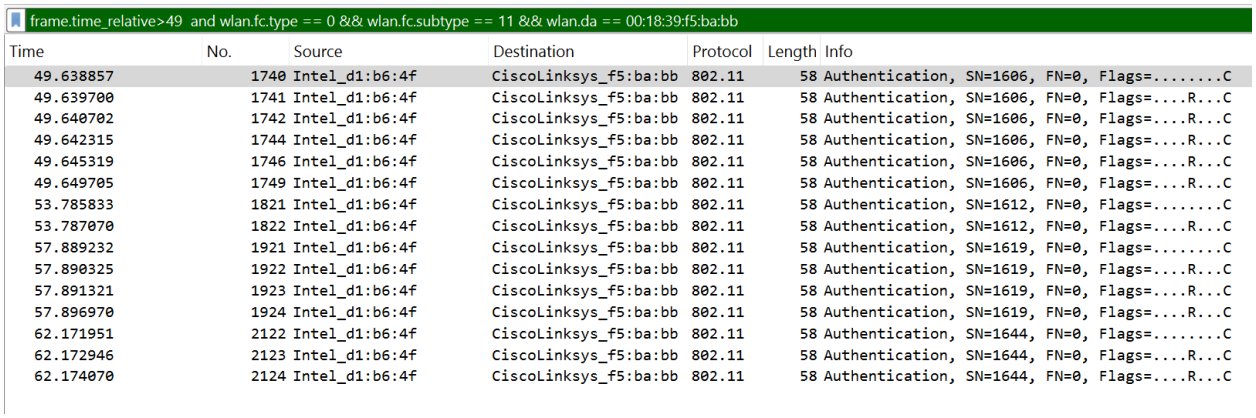


After t = 49, the following actions are taken to end the association with the 30 Munroe St:

- At t = 49.583615, DHCP Release is done. The host is releasing its IP address back to the DHCP server, and is exiting the network.
- At t = 49.609617, Deauthentication is done, to terminate a Wi-Fi connection.

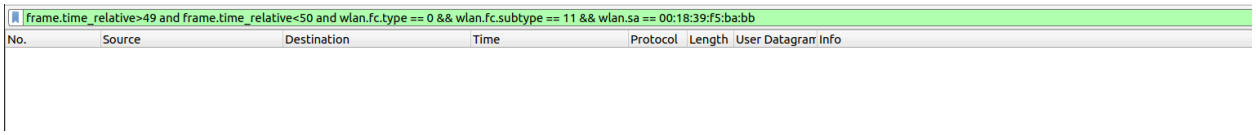
One would expect a DISASSOCIATION request, but that is not observed here.

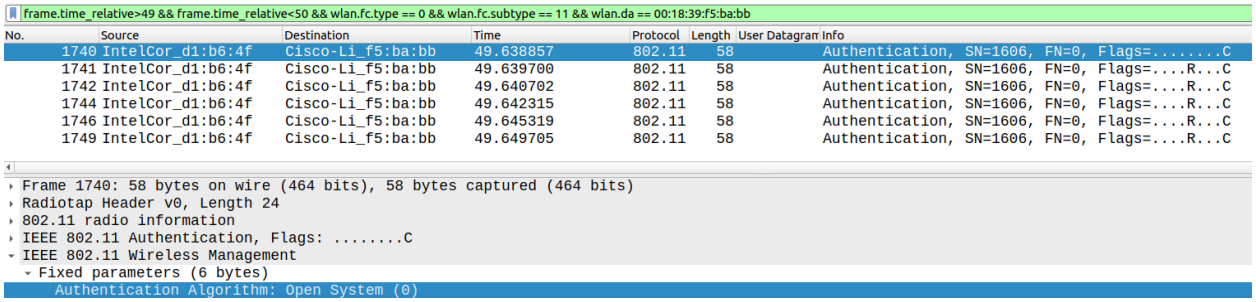
2. **Examine the trace file and look for AUTHENTICATION frames sent from the host to an AP and vice versa. How many AUTHENTICATION messages are sent from the wireless host to the *linksys_ses_24086* AP (which has a MAC address of Cisco_Li_f5:ba:bb) starting at around *t=49?***



A total of 15 AUTHENTICATION messages are sent from the wireless host to the linksys_ses_24086 AP (which has a MAC address of Cisco_Li_f5:ba:bb) starting at around t=49.

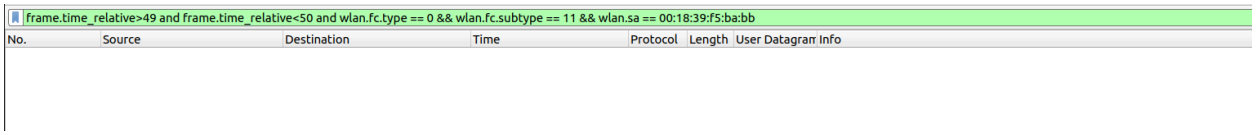
3. **Does the host want the authentication to require a key or be open?**





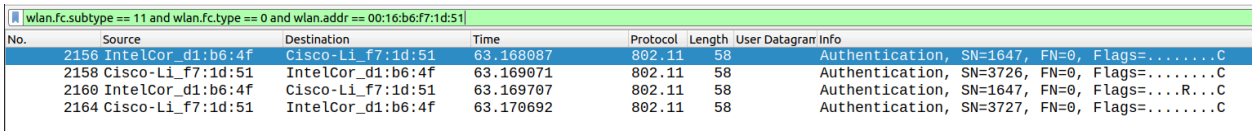
- Based on the **Authentication Algorithm field** in the AUTHENTICATION frames,
- The value is **0**, which corresponds to **Open System authentication** (i.e., **no shared key** required).
- So, the **host wants the authentication to be Open**.

4. **Do you see a reply AUTHENTICATION from the *linksys_ses_24086* AP in the trace?**



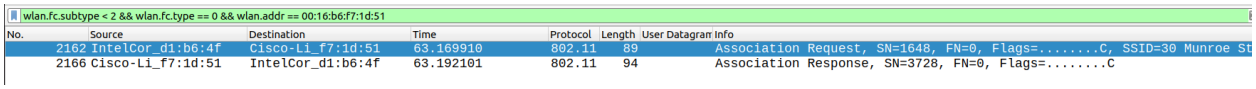
A reply AUTHENTICATION means the access point is acknowledging the host's request to authenticate. In this trace, no authentication reply is observed from the AP **(00:18:39:f5:ba:bb),** indicating the authentication attempt failed and the host did not successfully connect to this AP.

5. **Now let's consider what happens as the host gives up trying to associate with the *linksys_ses_24086* AP and now tries to associate with the *30 Munroe St* AP. Look for AUTHENTICATION frames sent from the host to an AP and vice versa. At what times is there an AUTHENTICATION frame from the host to *30 Munroe St.* AP, and when is there a reply AUTHENTICATION sent from that AP to the host reply? (Note that you can use the filter expression "wlan.fc.subtype == 11 && wlan.fc.type == 0 && wlan.addr == 00:13:02:d1:b6:4f" to display only the AUTHENTICATION frames in this trace for this wireless host.)**



- At times 63.168087 seconds and 63.169707 seconds , there is an AUTHENTICATION frame from the host to the AP.
- At times 63.169071 seconds and 63.170692 seconds , there is an AUTHENTICATION reply

6. **An ASSOCIATE REQUEST from the host to AP and a corresponding ASSOCIATE RESPONSE frame from AP to the host is used for the host to be associated with an AP. At what time is there an ASSOCIATE REQUEST from the host to *30 Munroe St* AP? When is the corresponding ASSOCIATE REPLY sent? (Note that you can use the filter expression "`wlan.fc.subtype < 2 && wlan.fc.type == 0 && wlan.addr == 00:13:02:d1:b6:4f`" to display only the ASSOCIATE REQUEST and ASSOCIATE RESPONSE frames for this trace.)**

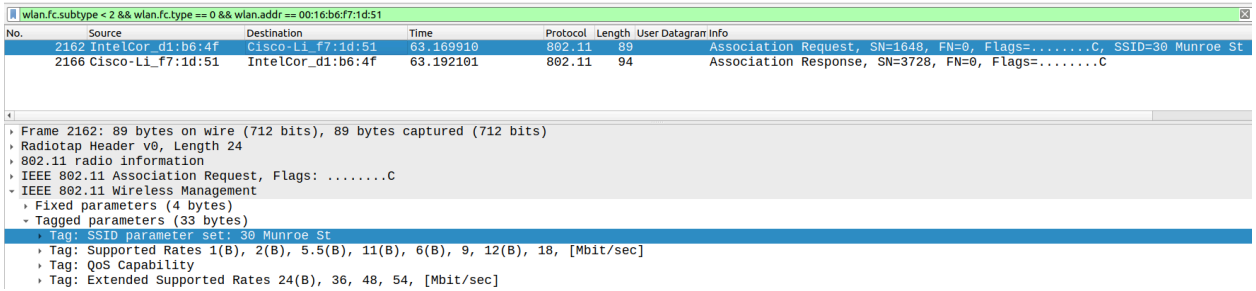


- An ASSOCIATION REQUEST (i.e., the ASSOCIATE REQUEST) is sent from host to the 30 Munroe St AP at time t = 63.169910 seconds.
- An ASSOCIATION RESPONSE (i.e., the ASSOCIATE REPLY) is sent from that AP to the host at time t = 63.192101 seconds.

7. **What transmission rates is the host willing to use? The AP? To answer this question, you will need to look into the parameter fields of the 802.11 wireless LAN management frame.**

To determine what transmission rates the host and the access point (AP) are willing to use, we examine the **Supported Rates** and **Extended Supported Rates** fields in **802.11 management frames** (typically in **Probe Request/Response** or **Association Request/Response** frames).

Host's Supported Rates

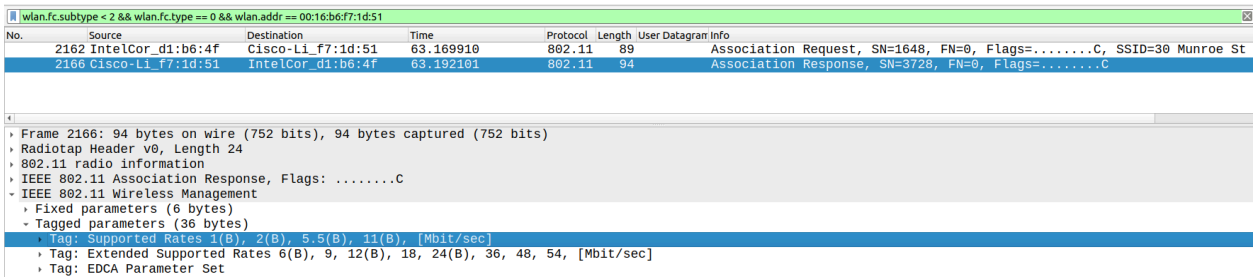


Rates advertised in request

From the 802.11 frame (e.g., Probe Request or Association Request):

- **Supported Rates:**
  `1(B), 2(B), 5.5(B), 11(B), 6(B), 9, 12(B), 18,` Mbps
- **Extended Supported Rates:**
  `24(B), 36, 48, 54` Mbps

The `(B)` indicates **basic rates** that must be supported by all devices in the network.



Rates advertised in response

Access Point's (AP's) Supported Rates

From the AP's Probe Response or Association Response:

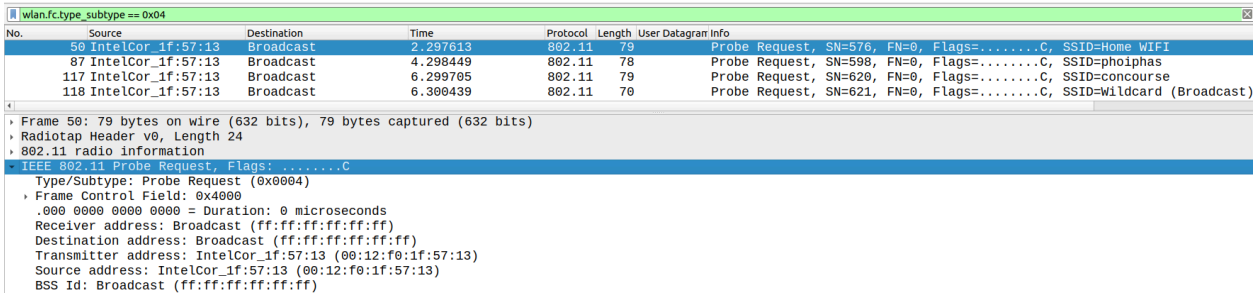
- **Supported Rates:**
  `1(B), 2(B), 5.5(B), 11(B),` Mbps
- **Extended Supported Rates:**
  `6(B), 9, 12(B), 18, 24(B), 36, 48, 54` Mbps

Note: The AP also supports the same total set of rates but distributes them slightly differently between **Supported** and **Extended Supported Rates** fields.

Part-4: Other Frame types

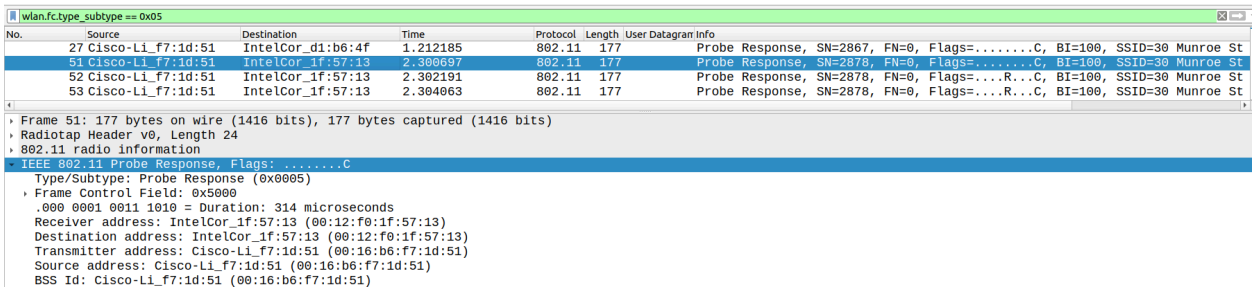
Our trace contains a number of PROBE REQUEST and PROBE RESPONSE frames.

1. **What are the sender, receiver and BSS ID MAC addresses in these frames? What is the purpose of these two types of frames?**



First PROBE REQUEST Frame (at t = 2.297613):

- **Source / Sender MAC address:** `00:12:f0:1f:57:13` (wireless host)
- **Receiver MAC address:** `ff:ff:ff:ff:ff:ff` (broadcast address)
- **BSS ID MAC address:** `ff:ff:ff:ff:ff:ff` (broadcast address)



Corresponding PROBE RESPONSE Frame (at t = 2.300697):

- **Source / Sender MAC address:** `00:16:b6:f7:1d:51` (Access Point)
- **Receiver MAC address:** `00:12:f0:1f:57:13` (wireless host)
- **BSS ID MAC address:** `00:16:b6:f7:1d:51` (same as AP MAC address)

Purpose of Probe Request & Response Frames

🔹 Probe Request (Type/Subtype: 0x04)

- **Purpose**: Sent by a client (station) to discover nearby Wi-Fi networks.
- **Use Case**: Especially useful when the SSID is hidden or when actively scanning.

🔹 Probe Response (Type/Subtype: 0x05)

- **Purpose**: Sent by an Access Point (AP) in reply to a Probe Request.
- **Includes**: SSID, supported data rates, capabilities, etc.

TLS

Introduction

In this lab, we'll investigate Transport Layer Security (known as TLS) and aspects of the authentication, data integrity, and confidentiality services provided by TLS. TLS is the successor to the now-deprecated Secure Sockets Layer (SSL).

We'll investigate TLS by analyzing a Wireshark packet trace captured during the retrieval of a web page via HTTPS - a secure version of HTTP, which implements TLS on top of HTTP. We'll look at TLS's client-server handshaking protocol in some detail since that's where most of the interesting action happens. You may use online resources for learning more about TLS here, here, and here; and, of course, in RFC 5246.

Capturing packets in a TLS session

The first step in this lab is to capture the packets in a TLS session. To do this, you should start Wireshark and begin packet capture, retrieve the homepage from https://www.cics.umass.edu using the browser of your choice, and then stop Wireshark packet capture. The 's' after 'http' will cause the **Hypertext Transfer Protocol Secure** (**HTTPS**) – an extension of HTTP – to be used to securely retrieve the homepage from `www.cics.umass.edu`. Here, "securely" means that the `www.cics.umass.edu` server will be authenticated by your web browser, that the transmission of your client HTTP GET request and the server's reply will be encrypted, and the integrity of all message content will be cryptographically verified. Of course, the authentication, integrity and encryption of a computer science department's web page may not be as critical as that for Internet commerce and banking sites, but the same TLS protocol and TLS messages are used in all cases.

Part-5: A first look at the captured trace

Let's first set Wireshark's display so that only the packets to and from `www.cics.umass.edu`, are displayed.

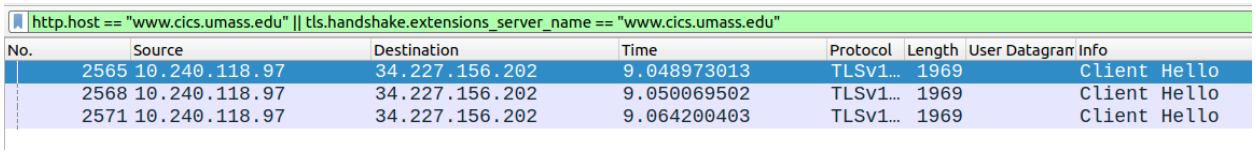
It's important to keep in mind that an Ethernet frame (containing an IP datagram containing a TCP segment) may contain one or more TLS records. (This is very different from HTTP, for which each frame contains either one complete HTTP message or a portion of an HTTP message.) Also, a TLS record may not completely fit into an Ethernet frame, in which case multiple frames will be needed to carry the record.

We've said earlier that HTTPS implements TLS running "over" TCP. That means that a TCP connection must first be established between your browser and the web server for `www.cics.umass.edu` before TLS and HTTP messages can be exchanged, just as we saw with the vanilla (non-TLS) HTTP protocol**.**

Answer the following questions:

1. **What is the IP address of the domain `www.cics.umass.edu`?**

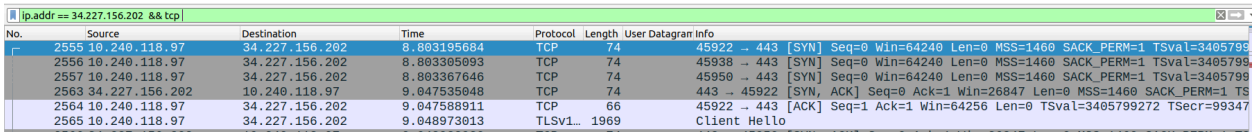
   IP Address of the domain www.cics.umass.edu **is**



   Destination IP: `34.227.156.202`

   This is the IP address your system resolves for `www.cics.umass.edu` in the trace.

2. **Does your system set up a TCP connection with the server of `www.cics.umass.edu`? Provide the packet numbers of the three-way handshake, with the corresponding screenshot.**



Three-Way Handshake Packet Numbers (from the screenshot):

TCP Stream (Client Port: 45922)

1. **Packet 2555:** `45922 → 443 [SYN]`
   Client initiates the connection to the server.
2. **Packet 2563:** `443 → 45922 [SYN, ACK]`
   Server responds with SYN-ACK.
3. **Packet 2564:** `45922 → 443 [ACK]`
   Client sends ACK to complete the handshake.

The screenshot clearly shows:

- All three packets (2555, 2563, 2564) in sequence.
- TLS Client Hello follows right after, in **Packet 2565**.

Summary:

- **Yes**, the TCP connection is successfully established.
- **Three-way handshake packets: 2555 (SYN)**, **2563 (SYN-ACK)**, **2564 (ACK)**.

3. **Is the TCP connection set up before or after the first TLS message is sent from the client to the server?**

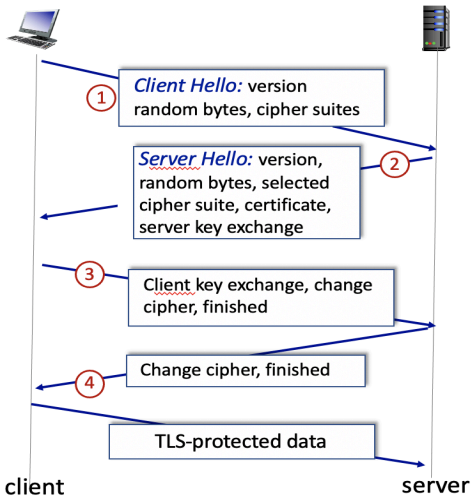
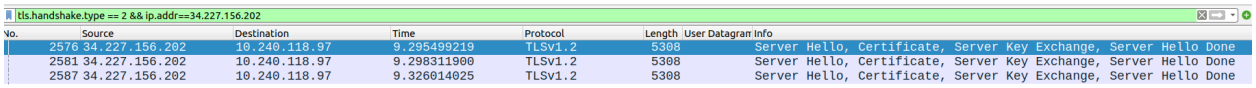
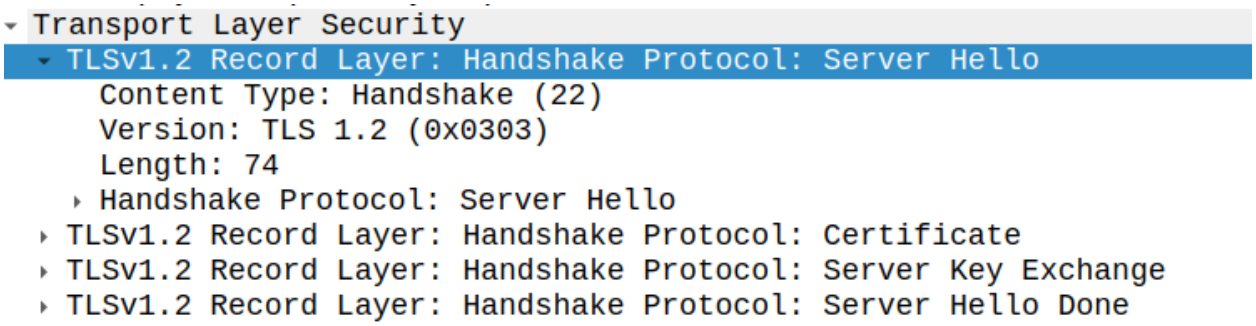


Figure 2: TLS handshake

- Yes — TLS runs over TCP.
- The **TCP connection is set up before** the first TLS message (Client Hello) is sent from the client to the server.
- As shown in **Figure 2: TLS handshake**, the TLS handshake begins **after** the three-way TCP handshake is completed. This ensures a reliable transport layer connection is in place before exchanging any TLS-specific messages

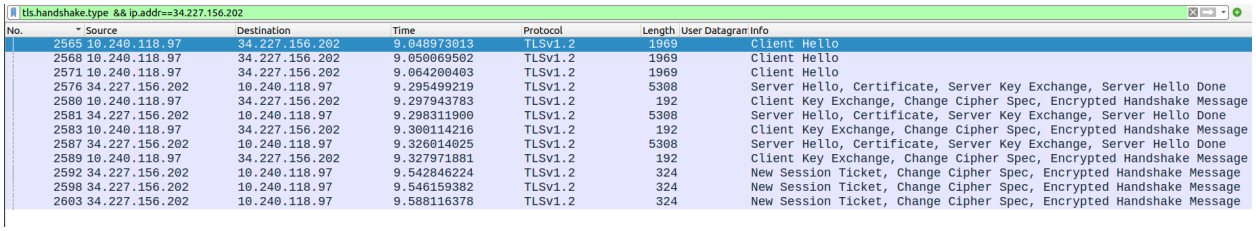
4. **What TLS version is used by the `www.cics.umass.edu` server?**





Version: TLS 1.2
Thus, the TLS version used by www.cics.umass.edu is TLS 1.2.

5. **List the various TLS messages between the client and the server?**



tls.handshake.type && ip.addr==34.227.156.202

No.	Source	Destination	Time	Protocol	Length	User Datagram	Info
2565	10.240.118.97	34.227.156.202	9.048973013	TLSv1.2	1969		Client Hello
2568	10.240.118.97	34.227.156.202	9.050069502	TLSv1.2	1969		Client Hello
2571	10.240.118.97	34.227.156.202	9.064200403	TLSv1.2	1969		Client Hello
2576	34.227.156.202	10.240.118.97	9.295499219	TLSv1.2	5308		Server Hello, Certificate, Server Key Exchange, Server Hello Done
2580	10.240.118.97	34.227.156.202	9.297943783	TLSv1.2	192		Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
2581	34.227.156.202	10.240.118.97	9.298311900	TLSv1.2	5308		Server Hello, Certificate, Server Key Exchange, Server Hello Done
2583	10.240.118.97	34.227.156.202	9.300114216	TLSv1.2	192		Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
2587	34.227.156.202	10.240.118.97	9.326014025	TLSv1.2	5308		Server Hello, Certificate, Server Key Exchange, Server Hello Done
2589	10.240.118.97	34.227.156.202	9.327971881	TLSv1.2	192		Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
2592	34.227.156.202	10.240.118.97	9.542846224	TLSv1.2	324		New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
2598	34.227.156.202	10.240.118.97	9.546159382	TLSv1.2	324		New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
2603	34.227.156.202	10.240.118.97	9.588116378	TLSv1.2	324		New Session Ticket, Change Cipher Spec, Encrypted Handshake Message

Based on the screenshot, the TLS handshake between the client (`10.240.118.97`) and the server (`34.227.156.202`) includes the following TLS messages:These packets represent the full TLS handshake exchange for setting up a secure connection using **TLS 1.2**.

The various TLS messages between the client and the server are

1. Client Hello:

- Sent by the client to initiate the TLS handshake.
- Contains supported cipher suites, TLS version, random data, and other information.

2. Server Hello:

- Sent by the server in response to the Client Hello.
- Includes the selected cipher suite, session ID, and server random data.

3. Certificate:

- Sent by the server.
- Contains the server's certificate for authentication.

4. Server Key Exchange:

- Sent by the server if required by the chosen cipher suite.
- Includes key exchange parameters.

5. Server Hello Done:

- Indicates that the server has finished its part of the handshake.

6. Client Key Exchange:

- Sent by the client.
- Contains key exchange information (e.g., pre-master secret).

7. Change Cipher Spec:

- Sent by both client and server to indicate that subsequent messages will be encrypted using the negotiated cipher suite.

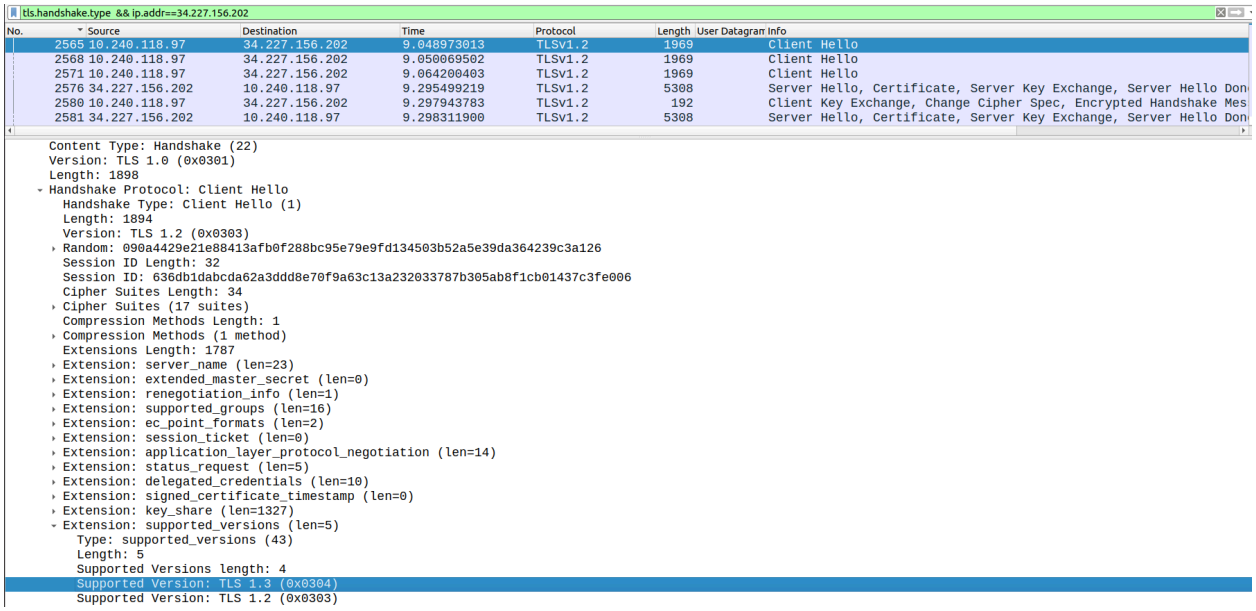
8. Encrypted Handshake Message:

- Sent by both client and server.
- Confirms that encryption and keys are correctly set up.

9. New Session Ticket (Optional):

- Sent by the server to provide a session ticket for resuming sessions

6. **In the `Client Hello` message, what are the two versions of the TLS, and are they different and why?**

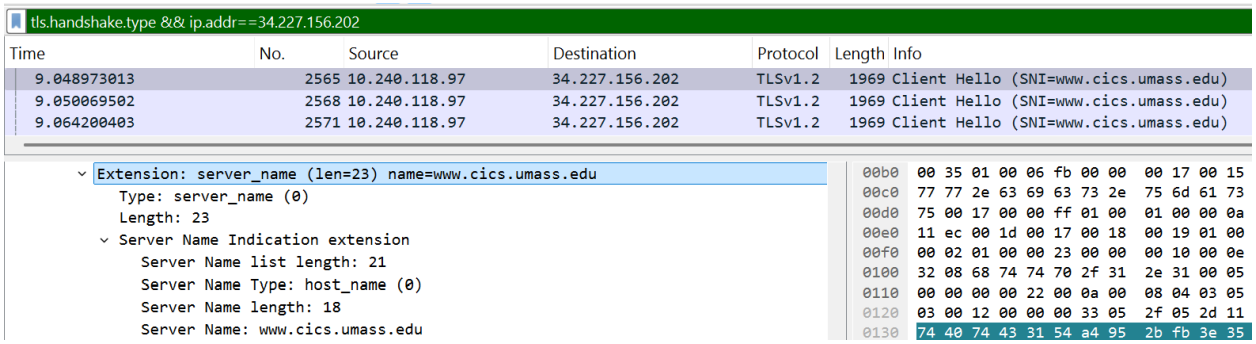


In the **Client Hello**:

- **Version field:** TLS 1.2 (0x0303)
- **Supported Versions extension:** TLS 1.3(0x0304)

Yes, they are different.The reason the client advertises both versions is to ensure that it can communicate with a wider variety of servers. If the server supports TLS 1.3, it will typically select that version; otherwise, the connection will fall back to TLS 1.2. This negotiation helps maintain both security and interoperability during the TLS handshake.

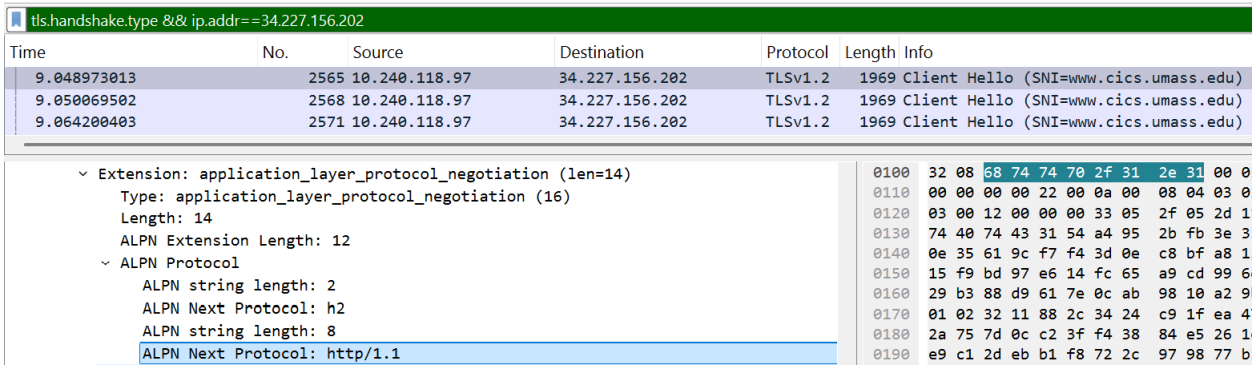
7. **Which field in the `Client Hello` message has the value of the domain? Is it encrypted or in plaintext?**



Field: `Server Name Indication (SNI)` under **Extension: server_name** Shows `www.cics.umass.edu`

It is **plaintext**, not encrypted (used to let the server know which certificate to send) even though it's part of the TLS handshake. This means anyone capturing the traffic (e.g., via Wireshark) can see which domain the client is trying to reach.

8. **Traverse through the '`Extension:`' parameters of the Client Hello message and provide the field name that contains the version of the HTTP, and what is the HTTP version?**



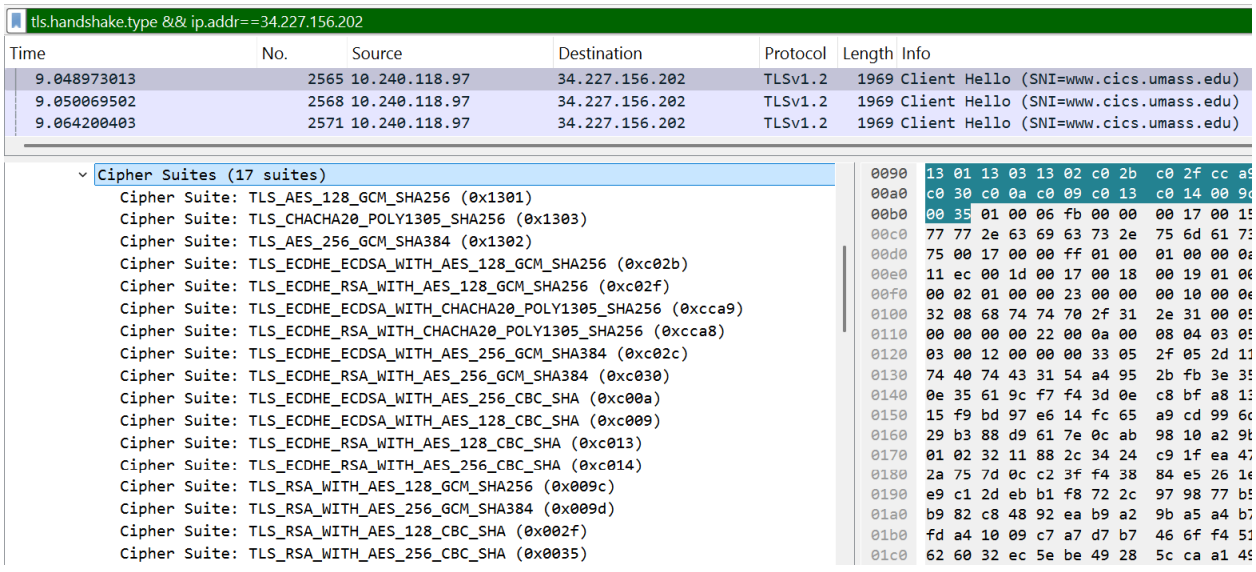
In the Client Hello message,

a. Field: `ALPN Next Protocol`
under **Extension: application_layer_protocol_negotiation(ALPN)**

b. So the HTTP version is: **"HTTP/2 (h2) and HTTP/1.1 (http/1.1)"**

9. **List all the cipher suites the client offers the server to choose from for further communication.**

The cipher suites offered by the client to the server in the Client Hello message are as follows:



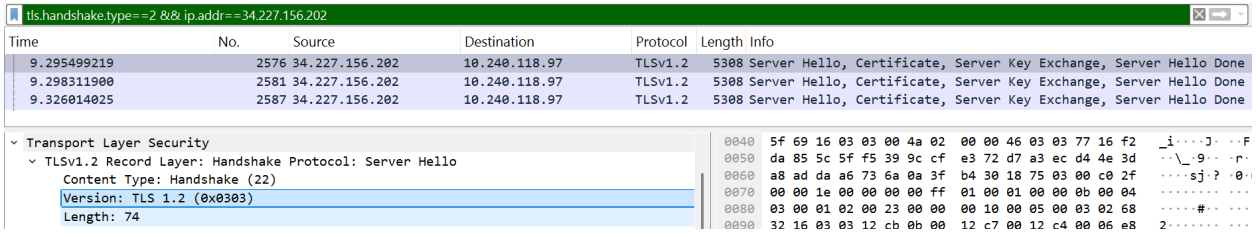
Cipher Suites (17 suites)

1. Cipher Suite: TLS_AES_128_GCM_SHA256 (0x1301)
2. Cipher Suite: TLS_CHACHA20_POLY1305_SHA256 (0x1303)
3. Cipher Suite: TLS_AES_256_GCM_SHA384 (0x1302)
4. Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (0xc02b)
5. Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)
6. Cipher Suite: TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256 (0xcca9)
7. Cipher Suite: TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 (0xcca8)
8. Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (0xc02c)
9. Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030)
10. Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA (0xc00a)
11. Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA (0xc009)
12. Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc013)
13. Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014)
14. Cipher Suite: TLS_RSA_WITH_AES_128_GCM_SHA256 (0x009c)
15. Cipher Suite: TLS_RSA_WITH_AES_256_GCM_SHA384 (0x009d)
16. Cipher Suite: TLS_RSA_WITH_AES_128_CBC_SHA (0x002f)
17. Cipher Suite: TLS_RSA_WITH_AES_256_CBC_SHA (0x0035)

These cipher suites represent the encryption algorithms and key exchange methods supported by the client, allowing the server to select one for secure communication based on mutual compatibility.

Now, analyse the corresponding `Server Hello` message and answer the following questions.

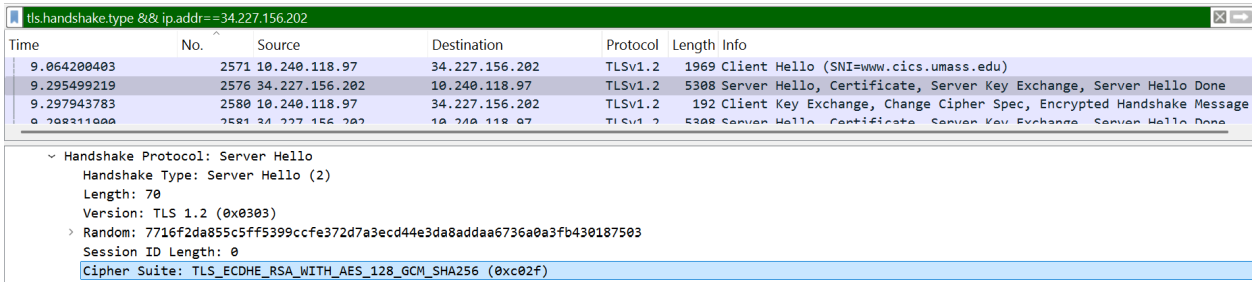
10. Which TLS version has the server agreed on to set the TLS handshake?



From the provided Wireshark capture, the server has agreed to use TLS 1.2 for the TLS handshake. This is evident in the Server Hello message, where the "Version" field specifies TLS 1.2 (0x0303).

Version: TLS 1.2 (0x0303)

11. Which cipher suite has the server agreed on for further communication?

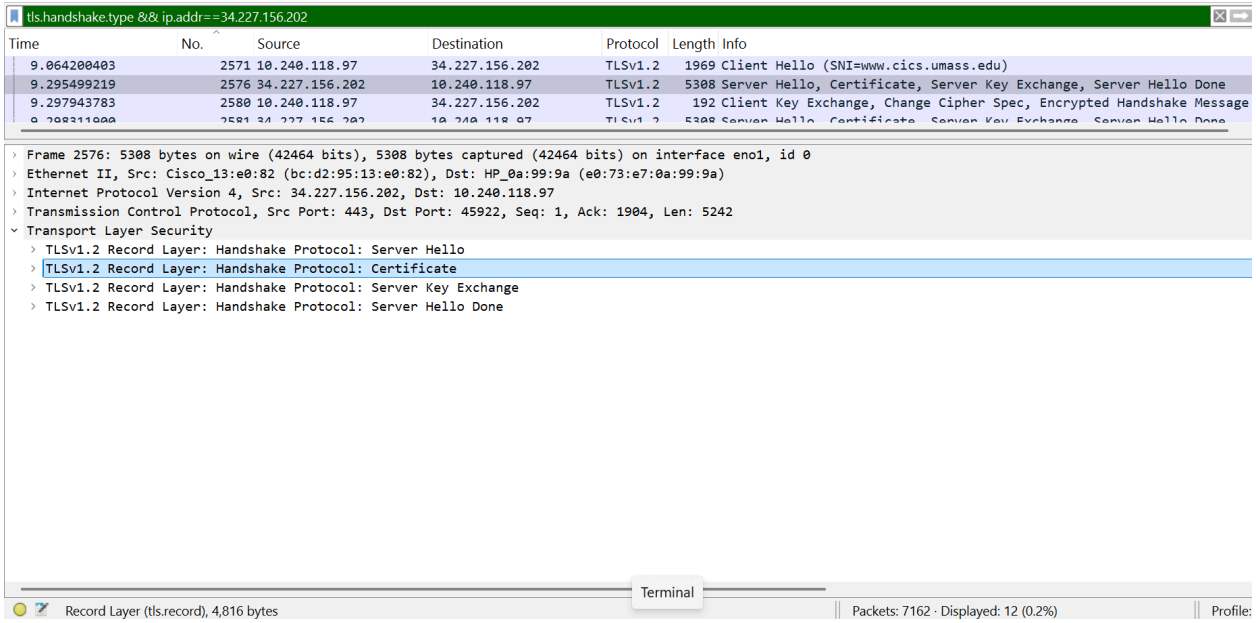


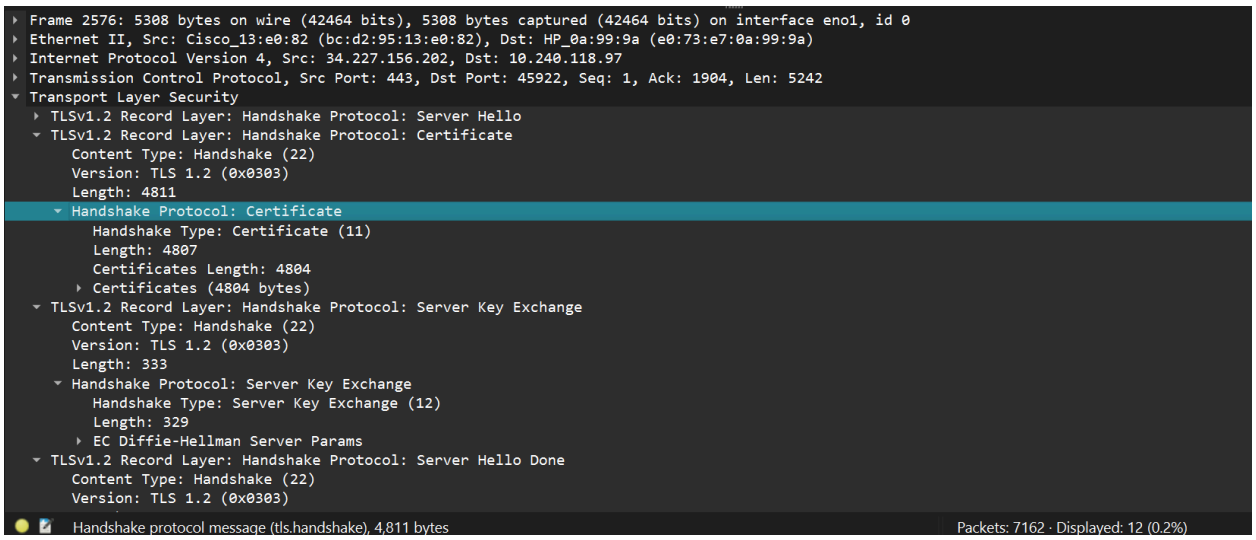
Cipher Suite agreed by the server for further communication is : TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)

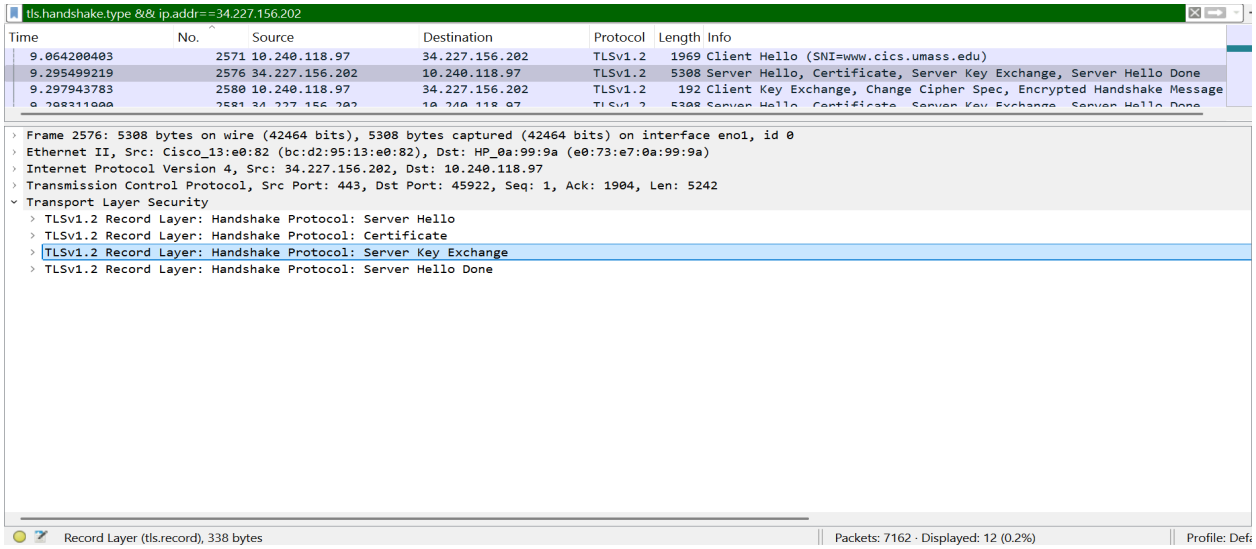
Analyse the `Certificate`, `Server Key Exchange`, `Server Hello Done` packet to answer the following questions.

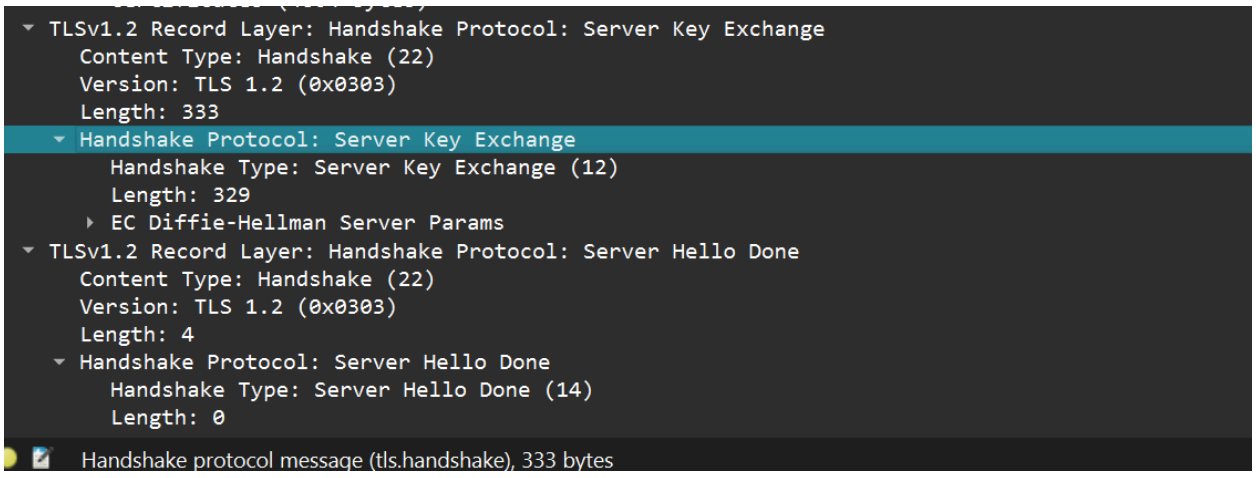
12. **What is the size (in bytes) of the `Certificate`, `Server Key Exchange`, `Server Hello Done` messages?**

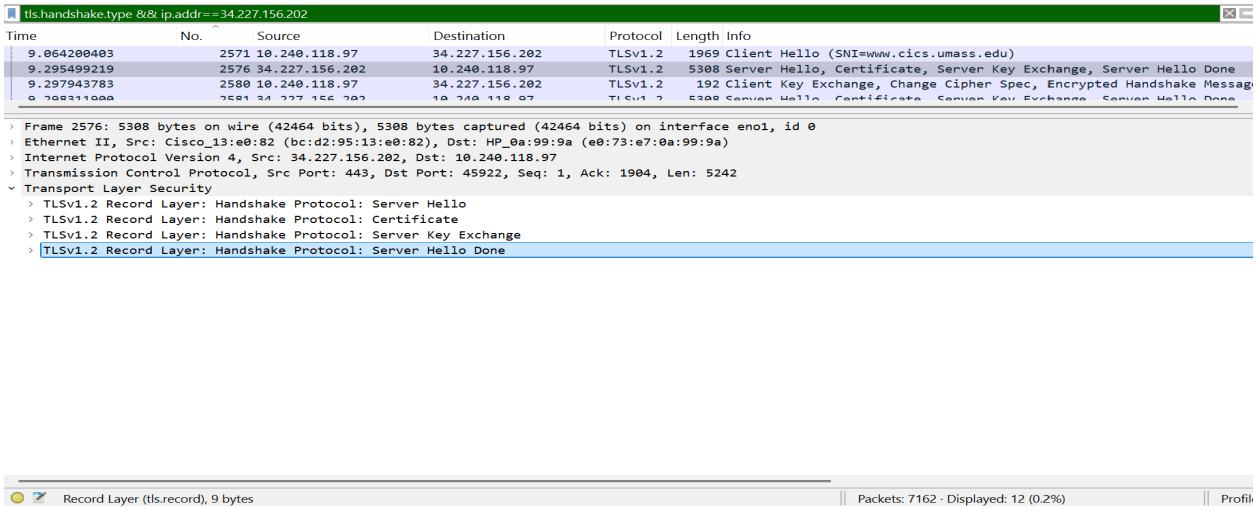
Message	**Size of Record Layer (in Bytes)**	**Size of Handshake Protocol Message(in Bytes)**
Certificate	4816 bytes	4811 bytes
Server Key Exchange	338 bytes	333 bytes
Server Hello Done	9 bytes	4 bytes

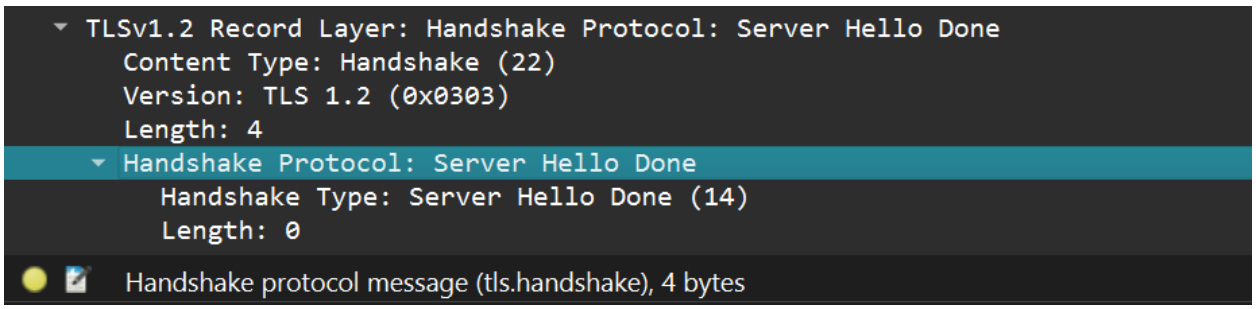



Frame 2576: 5308 bytes on wire (42464 bits), 5308 bytes captured (42464 bits) on interface eno1, id 0
Ethernet II, Src: Cisco_13:e0:82 (bc:d2:95:13:e0:82), Dst: HP_0a:99:9a (e0:73:e7:0a:99:9a)
Internet Protocol Version 4, Src: 34.227.156.202, Dst: 10.240.118.97
Transmission Control Protocol, Src Port: 443, Dst Port: 45922, Seq: 1, Ack: 1904, Len: 5242
Transport Layer Security
TLSv1.2 Record Layer: Handshake Protocol: Server Hello
TLSv1.2 Record Layer: Handshake Protocol: Certificate
Content Type: Handshake (22)
Version: TLS 1.2 (0x0303)
Length: 4811
Handshake Protocol: Certificate
Handshake Type: Certificate (11)
Length: 4807
Certificates Length: 4804
Certificates (4804 bytes)
TLSv1.2 Record Layer: Handshake Protocol: Server Key Exchange
Content Type: Handshake (22)
Version: TLS 1.2 (0x0303)
Length: 333
Handshake Protocol: Server Key Exchange
Handshake Type: Server Key Exchange (12)
Length: 329
EC Diffie-Hellman Server Params
TLSv1.2 Record Layer: Handshake Protocol: Server Hello Done
Content Type: Handshake (22)
Version: TLS 1.2 (0x0303)
Handshake protocol message (tls.handshake), 4,811 bytes
Packets: 7162 · Displayed: 12 (0.2%)

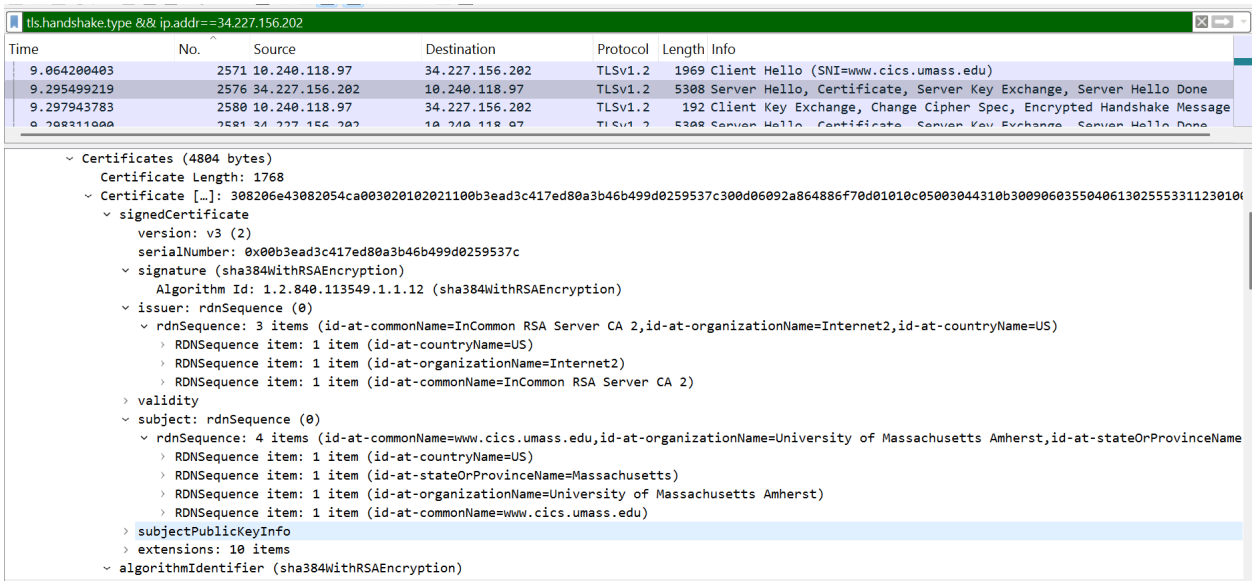

tls.handshake.type && ip.addr==34.227.156.202
Time No. Source Destination Protocol Length Info
9.064200403 2571 10.240.118.97 34.227.156.202 TLSv1.2 1969 Client Hello (SNI=www.cics.umass.edu)
9.295499219 2576 34.227.156.202 10.240.118.97 TLSv1.2 5308 Server Hello, Certificate, Server Key Exchange, Server Hello Done
9.297943783 2580 10.240.118.97 34.227.156.202 TLSv1.2 192 Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
Frame 2576: 5308 bytes on wire (42464 bits), 5308 bytes captured (42464 bits) on interface eno1, id 0
Ethernet II, Src: Cisco_13:e0:82 (bc:d2:95:13:e0:82), Dst: HP_0a:99:9a (e0:73:e7:0a:99:9a)
Internet Protocol Version 4, Src: 34.227.156.202, Dst: 10.240.118.97
Transmission Control Protocol, Src Port: 443, Dst Port: 45922, Seq: 1, Ack: 1904, Len: 5242
Transport Layer Security
TLSv1.2 Record Layer: Handshake Protocol: Server Hello
TLSv1.2 Record Layer: Handshake Protocol: Certificate
TLSv1.2 Record Layer: Handshake Protocol: Server Key Exchange
TLSv1.2 Record Layer: Handshake Protocol: Server Hello Done
Record Layer (tls.record), 338 bytes
Packets: 7162 · Displayed: 12 (0.2%)


TLSv1.2 Record Layer: Handshake Protocol: Server Key Exchange
Content Type: Handshake (22)
Version: TLS 1.2 (0x0303)
Length: 333
Handshake Protocol: Server Key Exchange
Handshake Type: Server Key Exchange (12)
Length: 329
EC Diffie-Hellman Server Params
TLSv1.2 Record Layer: Handshake Protocol: Server Hello Done
Content Type: Handshake (22)
Version: TLS 1.2 (0x0303)
Length: 4
Handshake Protocol: Server Hello Done
Handshake Type: Server Hello Done (14)
Length: 0
Handshake protocol message (tls.handshake), 333 bytes





13. Which certificate is agreed upon by the server?



The certificate agreed upon by the server is the end-entity certificate presented in the Handshake Protocol: Certificate message (Frame 2793)1. Its key details are:

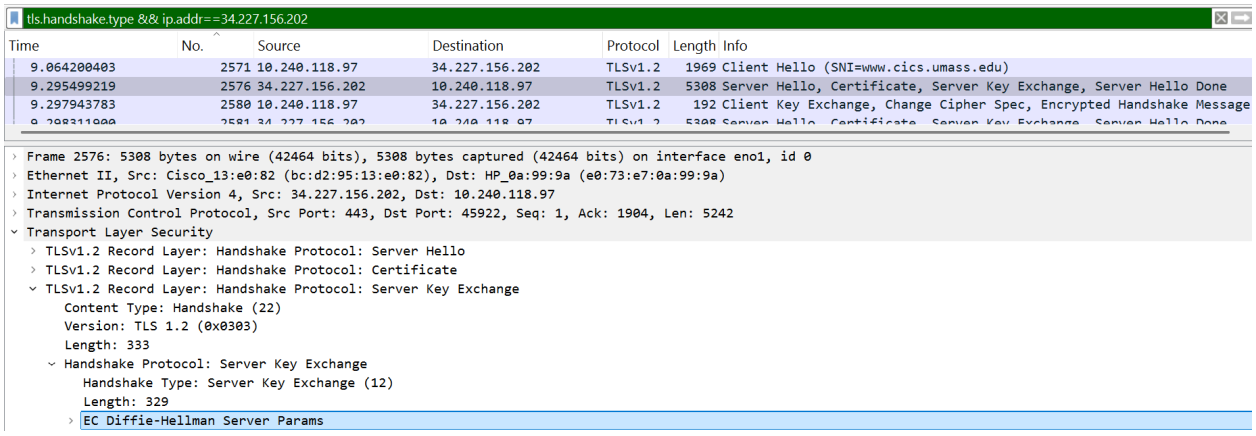
- Certificate Length: 1768 bytes.
- Version: v3 (2)1.
- Serial Number: 0x00b3ead3c417ed80a3b46b499d0259537c.

● Signature Algorithm: sha384WithRSAEncryption.
● Issuer: Common Name: InCommon RSA Server CA 2, Organization: Internet, Country: US.
● Validity: Dates specified as notBefore(2024-07-23) and notAfter values(2025-08-23)
● Subject: Common Name: www.cics.umass.edu, Organization: University of Massachusetts Amherst, State: Massachusetts, Country: US.
● Subject Public Key Info:
    ● Algorithm: rsaEncryption.
    ● Public Key: (hexadecimal representation provided in the capture).

This certificate, along with the intermediate certificates (InCommon RSA Server CA 2 issued by USERTrust RSA Certification Authority, and USERTrust RSA Certification Authority issued by AAA Certificate Services), forms the chain presented by the server for authentication.

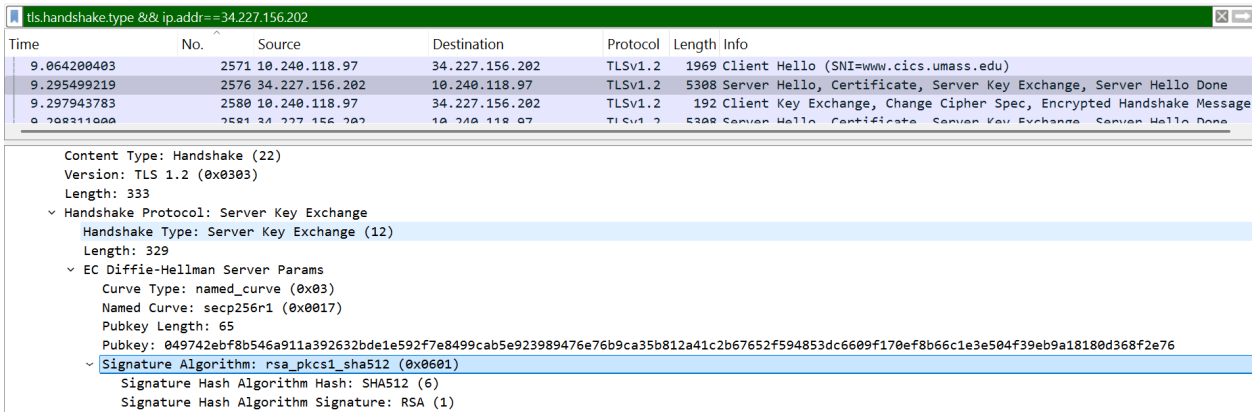
The certificate agreed upon by the server is issued to **`www.cics.umass.edu`**, and it is signed by **`InCommon RSA Server CA 2`**, using the **`sha384WithRSAEncryption`** algorithm.

14. Which handshake protocol is used in the `Server Key Exchange` message?



The handshake protocol used in the Server Key Exchange message is EC Diffie-Hellman Server Params i.e **ECDHE** (Elliptic Curve Diffie-Hellman Ephemeral).

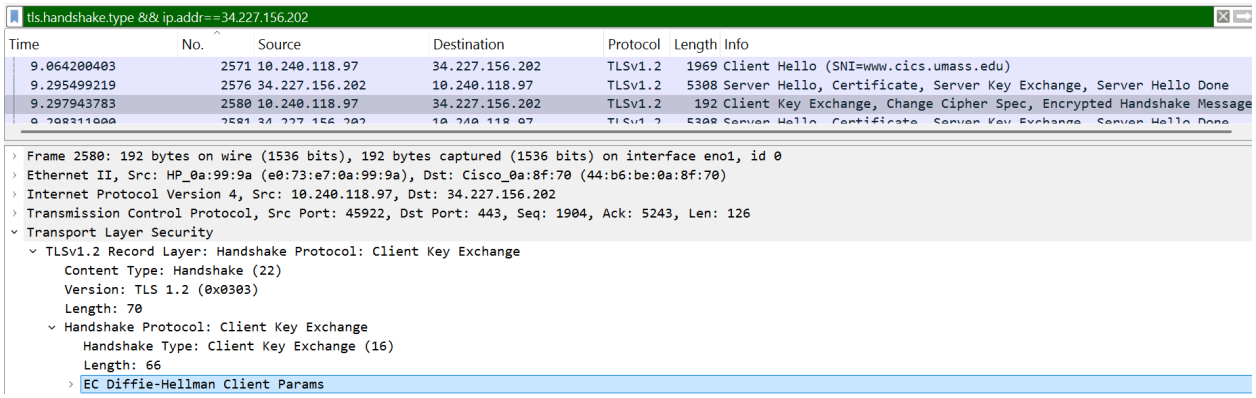
15. Which signature algorithm does the server send to the client?



The server sends **`rsa_pkcs1_sha512 (0x0601)`** as the signature algorithm to the client.

Analyse the `Client Key Exchange`, `Change Cipher Spec`, `Finished` message and answer the following questions.

16. Does the client agree on the same handshake protocol?



Yes, the client agrees on the **same handshake protocol (ECDHE)** as indicated by the **Client Key Exchange** message and continues with **Change Cipher Spec** and **Finished**, completing the handshake using the agreed **ECDHE protocol.**

Submission Details

- Write your answers in a single doc/tex file, and submit its PDF named after your IIT Dharwad roll number, which contains all answers (with screenshots, if necessary).