CS 342 : Assignment - 2

Name: VISHISHT PRIYADARSHI
Roll No: 180123053

Drive Link (for Traces) – https://drive.google.com/drive/folders/12un_H0VLoBYnq57Fr23SPzxlB35nPqxs?usp=sharing

Question 1:

The protocols used at different layers and their packet format are:

1. APPLICATION LAYER:

a) SSH (Secure Shell) Protocol –

The SSH is a protocol for secure remote login and other secure network services over an insecure network. It runs on the top of TCP/IP. The SSH protocol contains three major components:

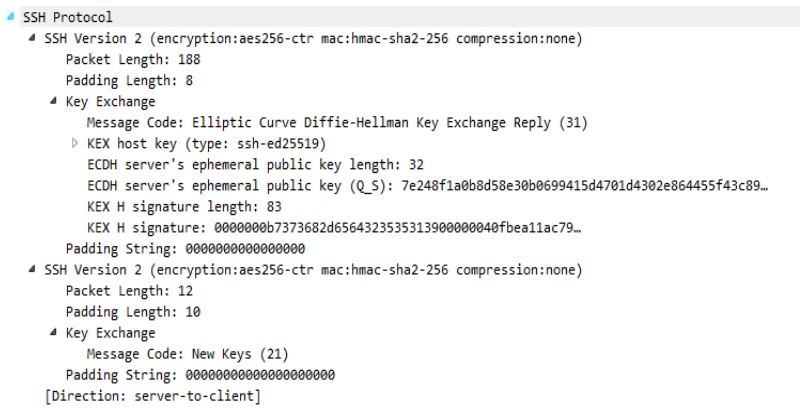
(i) SSH Transport Layer Protocol: It provides server authentication, confidentiality, and integrity.

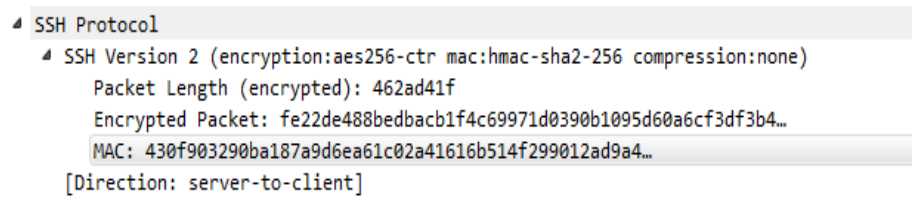
(ii) **SSH User Authentication Protocol:** It runs over the SSH Transport Layer Protocol, and has access to **session_id**. It supports three user authentication methods – "Public Key", "Password", "Host-based".

(iii) **SSH Connection Protocol:** It provides interactive login sessions, and remote execution of commands.

The SSH packet format is as follows:

Packet Length is the length of the packet in bytes. **Padding Length** is the length of the 'padding string' in bytes, which is used while "encrypting". **Payload** contains the useful contents of the packet, which gets encrypted after completion of SSH handshake. **Random Padding** is the 'random padding' which is required to encode the packet. Its size was specified by the "padding length" field. The transport protocol verifies the integrity of the data by adding a **Message Authentication Code (MAC)** to the packet. It is based on the shared secret (established by the key exchange), the packet sequence number, and the packet contents. It is calculated before encryption takes place.





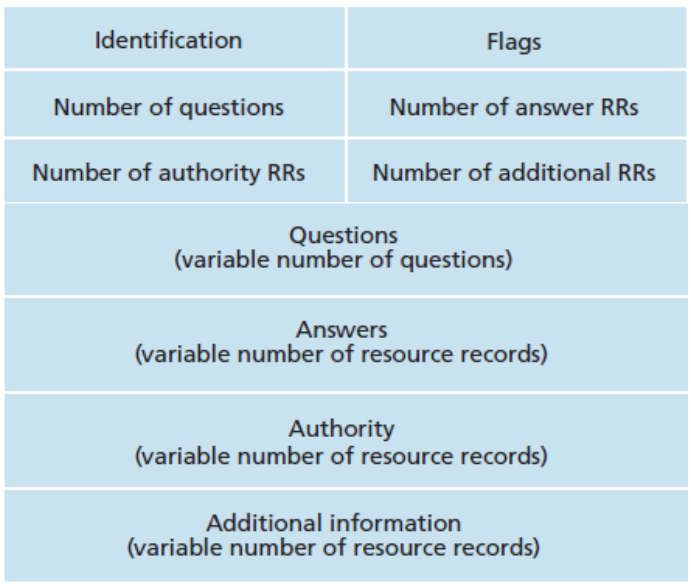
The left image is of the packet when the connection to the remote server is being set-up, while the second image is of when the interactive login session is in process.

Wireshark Packet Analysis

Packet Length	188	The length of the packet is 188 bytes.
Padding Length	8	8 bytes are allotted for random padding for security purposes.
Padding String	000000000..00	'Random padding' used to encode the packet.
Key Exchange	Elliptic Curve, Diffie-Hellman …	It specifies how one-time session keys are generated for encryption and for authentication, and how the server authentication is done. It generally specifies two versions of Diffie-Hellman Key Exchange. It sets up the keys that will be used to secure the connection,
MAC	430f903290b…	Message Authentication Code to verify the integrity of the packet.
Encrypted Packet	fe22de4…	Payload which has been encrypted after SSH handshake completion

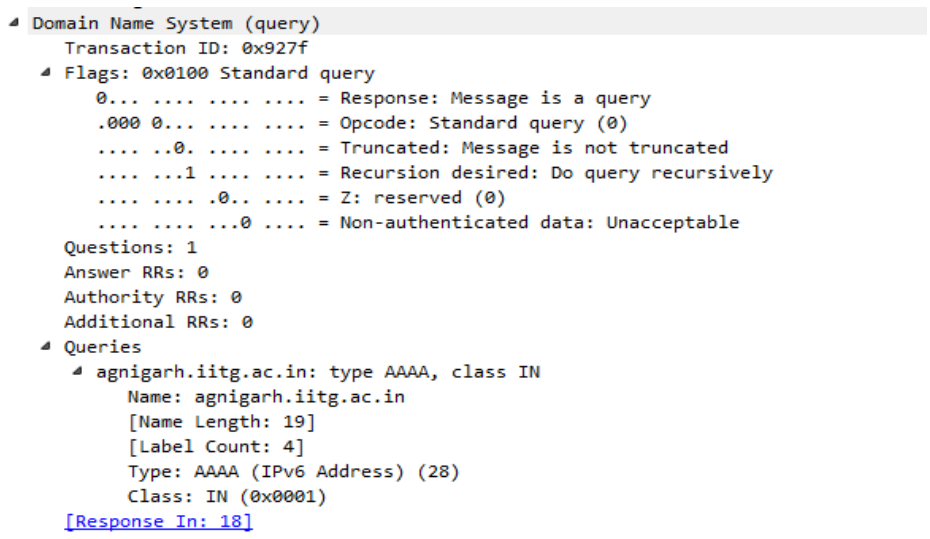
b) DNS (Domain Name system) Protocol –

DNS is a query/response protocol. The client queries an information in a single UDP request. This request is followed by a single UDP reply from the DNS server. A DNS query and a DNS reply share the same structure. The various fields include:



Transaction ID which is a 16-bit identification field generated by the device that creates the DNS query. **Flags** contain a number of sub fields like: **Query/Response** which contains 0/1 depending on whether it is a query or a response.

Opcode specifies the type of query the message is carrying. **Truncated** is et to 1 when message is truncated due to its length longer than the limit of transport medium used. (used in case of UDP). It contains further info regarding status and the query was recursive or not. **Questions** contain value provide the number of requests that are sent in the DNS query segment. **Answer RRs/ Authority RRs/ Additional RRs:** RR stands for Resources Records. They are used to store hostnames, IP addresses and other info in DNS name servers. **Queries** contains the domain name and type of record (A, AAAA, MX, TXT, etc.) being resolved.



Wireshark Packet Analysis

Transaction ID	0x927f	It is the 16 bit identification filed generated by my PC.
Flags	0x0100 Standard Query	Message is a query, and it is a standard query with no recursion
Questions	1	1 request sent in DNS query segment
Answer RRs/ Authority RRs/ Additional RRs	0	There are no RRs currently stored.
Name	agnigarh.iitg.ac.in	Name of the server
Type	AAAA	Query is for IPv6 address of the server

c) TLS (Transport Layer Security) Protocol –

It is a security protocol designed to facilitate privacy and data security for communication over the network. It is used for encrypting the communication between web applications and servers. The packet structure is as follows:

Byte	+0	+1	+2	+3
0	Content type			
1..4	Version		Length	
5..n	*Payload*			
n..m	MAC			
m..p	Padding (block ciphers only)			

Content Type can be of 4 types - Handshake, Change Cipher Spec, Alert and Application Data. **Version** is the TLS protocol version that the client wants to communicate with server. **Length** is the length of the application data being transferred. The transport protocol verifies the integrity of the data by adding a **Message Authentication Code (MAC)** to the packet. It is based on the shared secret (established by the key exchange), the packet sequence number, and the packet contents. It is calculated before encryption takes place.

Wireshark Packet Analysis

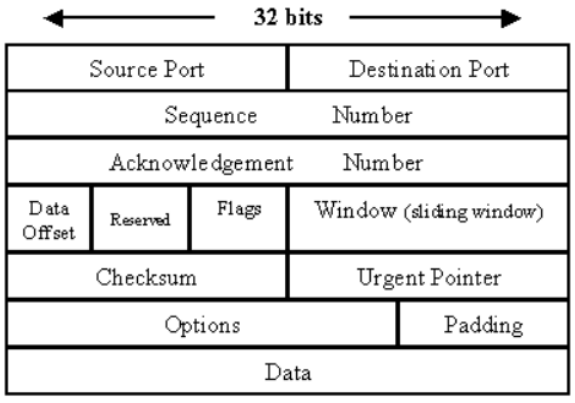
```
▲ Transport Layer Security
  ▲ TLSv1.2 Record Layer: Handshake Protocol: Server Hello
      Content Type: Handshake (22)
      Version: TLS 1.2 (0x0303)
      Length: 89
  ▲ Handshake Protocol: Server Hello
      Handshake Type: Server Hello (2)
      Length: 85
      Version: TLS 1.2 (0x0303)
    ▲ Random: d5b8c54f92268b6430ccddc299b85ec0b80802d1a15c87b1…
        GMT Unix Time: Aug 16, 2083 21:05:43.000000000 India Standard Time
        Random Bytes: 92268b6430ccddc299b85ec0b80802d1a15c87b1444f574e…
      Session ID Length: 32
      Session ID: b36e4923116d0ec30deb48cf9ebbd8578cd5687b9328d038…
      Cipher Suite: TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 (0x009f)
      Compression Method: null (0)
      Extensions Length: 13
    ▲ Extension: renegotiation_info (len=1)
        Type: renegotiation_info (65281)
        Length: 1
      ▲ Renegotiation Info extension
          Renegotiation info extension length: 0
    ▲ Extension: server_name (len=0)
        Type: server_name (0)
        Length: 0
    ▲ Extension: extended_master_secret (len=0)
        Type: extended_master_secret (23)
        Length: 0
```

Content Type	Handshake	It is a handshake packet.
Length	89	It is the length of the application data being transferred.
Random	d58c54f922…	32-byte pseudorandom number that is used in encryption key
Session ID	b36e492311…	Used by the client to identify the session
Cipher Suite	TLS_DHE_RSA…	List of cipher suites supported by the client

2. TRANSPORT LAYER :

a) TCP (Transmission Control Protocol) –

TCP is a connection-oriented protocol that defines how to establish and maintain a network connection. The various fields of a TCP packet are:



Source port is the port which is sending the packets. **Destination port** is the receiving port. **Sequence Number:** If SYN bit is set 1, then it is the initial sequence number, else this is the accumulated sequence number of the first data byte of this segment. **Acknowledgment Number:** If it had been 1, it meant that then the value of this field is the next sequence number that the sender of the ACK is expecting. **Header Length** is the size of the header specified im multiple of 4 bytes. **Window Size** is the size of receive window in bytes which is used to regulate the amount of data sent. **Checksum:** This field of 16 bit size is used for the error checking of the header and the data.

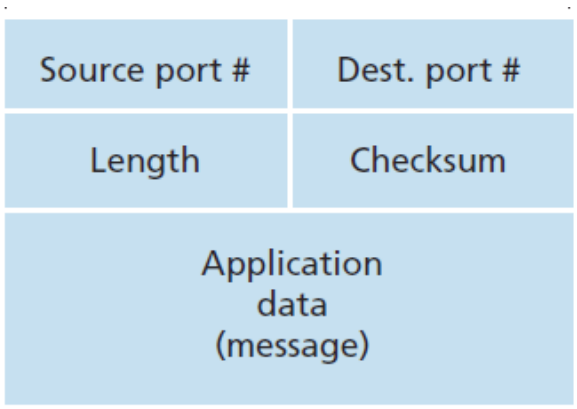
Wireshark Packet Analysis

```
▲ Transmission Control Protocol, Src Port: 49207, Dst Port: 10443, Seq: 0, Len: 0
    Source Port: 49207
    Destination Port: 10443
    [Stream index: 1]
    [TCP Segment Len: 0]
    Sequence number: 0    (relative sequence number)
    Sequence number (raw): 3145448350
    [Next sequence number: 1    (relative sequence number)]
    Acknowledgment number: 0
    Acknowledgment number (raw): 0
    1000 .... = Header Length: 32 bytes (8)
  ▷ Flags: 0x002 (SYN)
    Window size value: 8192
    [Calculated window size: 8192]
    Checksum: 0x3021 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  ▷ Options: (12 bytes), Maximum segment size, No-Operation (NOP), Window scale, No-Operation (NOP), No-Operation (NOP), SACK permitted
  ▷ [Timestamps]
```

Source Port	49207	This is the sending port of my pc.
Destination Port	10443	Destination port of the receiver.
Sequence Number	0	This is the first TCP packet to be exchanged.
Header Length	32 bytes (8)	Header length is 32 bytes but stored as 8 (multiple of 4 bytes)
Window size value	8192	Size of receive window in bytes
Checksum	0x3021	Used for error checking of header and data.

b) UDP (User Datagram Protocol) –

It is a connectionless protocol and faster than TCP. Various fields of a UDP packet (header) are: **Source Port** is the source port number of the packet. **Destination Port** is the destination port number of the packet. **Length** is the length of the UDP data and UDP header combined in bytes. **Checksum** is used to detect error in header and data if occurred during transmission

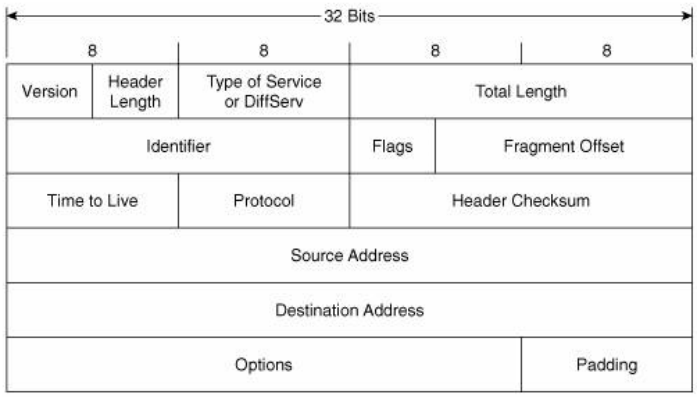



```
▲ User Datagram Protocol, Src Port: 50330, Dst Port: 53
    Source Port: 50330
    Destination Port: 53
    Length: 45
    Checksum: 0x7ec2 [unverified]
    [Checksum Status: Unverified]
    [Stream index: 0]
  ▲ [Timestamps]
      [Time since first frame: 0.000000000 seconds]
      [Time since previous frame: 0.000000000 seconds]
```

Source Port	50330	This is the sending port of my pc.
Destination Port	53	Destination port of the receiver.
Length	45	45 bytes is the length of UDP data and header combined.
Checksum	0x7ec2	Used for error detection

3. NETWORK LAYER : IPv4 (Internet Protocol version 4) –

IPv4 is the fourth version of IP. It is one of the core protocols of standards-based internetworking methods in the internet and other packet-switched networks. It is a connectionless protocol. Various field in IPv4 datagram header are:



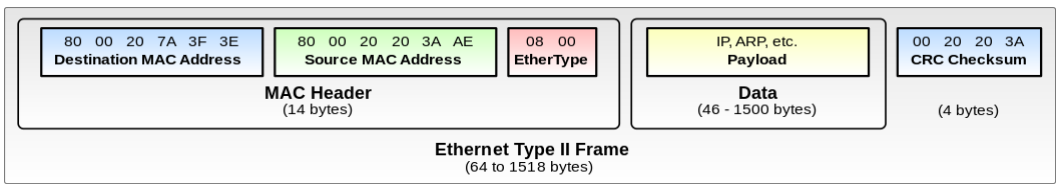
Version tells which version we are using. Only IPv4 uses this header. **Header length** tells the length of the IP header in 32 bits increment. The minimum length of an IP header is 20 bytes, so with 32 bit increments, we see a value of 5 there. **Differentiated Services Field** is used for particular quality of service (QoS) that is needed. **Total Length** shows the entire size of IP packet in bytes. **Identification:** If the IP packet is fragmented then each fragmented packet will use the same 16 bit identification number to identify to which IP packet they belong to. **Flags:** These 3 bits are used to identify or control the fragmentation. **Time to Live** is used to prevent packets from looping around forever.

```
▲ Internet Protocol Version 4, Src: 192.168.43.147, Dst: 14.139.196.11
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▷ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 52
    Identification: 0x0816 (2070)
  ▷ Flags: 0x4000, Don't fragment
    Fragment offset: 0
    Time to live: 128
    Protocol: TCP (6)
    Header checksum: 0x33dc [validation disabled]
    [Header checksum status: Unverified]
    Source: 192.168.43.147
    Destination: 14.139.196.11
```

Wireshark Packet Analysis

Version	4	Version 4 of IP, i.e., IPv4 is used.
Header Length	20 bytes (5)	Header length is 20 bytes but stored as 5 (multiple of 4 bytes)
Time To Live	128	After 128 hops, this packet will be dropped.
Source	192.168.43.147	IP address of the source, i.e., my PC
Destination	14.139.196.11	IP address of the destination, i.e., agnigarh.iitg.ac.in

4. LINK LAYER : Ethernet II -



Ethernet is used to connect devices in a network and is a popular form of network connection. There is no **preamble** in the fields shown in Wireshark. The preamble is a **physical layer mechanism** to help the NIC identify the start of a frame. It carries no useful data and is not received like other fields

```
▲ Ethernet II, Src: D-LinkIn_a5:64:b9 (ec:22:80:a5:64:b9), Dst: da:32:e3:d5:40:52 (da:32:e3:d5:40:52)
  ▲ Destination: da:32:e3:d5:40:52 (da:32:e3:d5:40:52)
      Address: da:32:e3:d5:40:52 (da:32:e3:d5:40:52)
      .... ..1. .... .... .... .... = LG bit: Locally administered address (this is NOT the factory default)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
  ▲ Source: D-LinkIn_a5:64:b9 (ec:22:80:a5:64:b9)
      Address: D-LinkIn_a5:64:b9 (ec:22:80:a5:64:b9)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    Type: IPv4 (0x0800)
```

Wireshark Packet Analysis

Destination	da:32:e3:d5:40:52	MAC address of the destination
Source	ec:22:80:a5:64:b9	MAC address of the router with which my PC is connected.
Type	IPv4	It is used to indicate which protocol is encapsulated in the payload of the frame. Here, it is IPv4.

Question 2:

The two important functionalities of the FortiClient VPN are:

i) Establish VPN connection:

- This feature uses **DNS and TLS protocol** at the **Application layer**, **TCP and UDP** at **Transport layer**, **IPv4** at **Network Layer** and **Ethernet II** at the **Link Layer**.
- TLS protocol generates a VPN connection where the web browser is the client and user access is prohibited to specific applications instead of the entire network. The **TLS encrypts all data packets** travelling between an internet connected device and the VPN server. The encryption ensures that all the data passed from the VPN's subscriber device is private and secure. The third parties are unable to intercept the encrypted data. TLS works by executing a TLS handshake at the beginning of the connection which creates **a cryptographically secure channel** over which all the exchange of packets take place. TLS makes use of **public key cryptography** (asymmetric key cryptography) which allows the peers to negotiate a **shared secret key** without having to establish any prior knowledge of each other, and to do so over an unencrypted channel.
- At the transport layer, TCP is used. This is because TLS is designed to offer a **secure channel** on the top of a **reliable transport channe**l and TCP offers this facility. TLS itself does not handle transport errors, lost packets or other disturbances that may occur with IP. So, due to this reason a reliable transport channel is required by TLS.

ii) ssh remote machine:

- This feature uses **DNS and SSH protocol** at the **Application layer**, **TCP and UDP** at **Transport layer**, **IPv4** at **Network Layer** and **Ethernet II** at the **Link Layer**.
- SSH protocol is a method for **secure login** from one computer to another. As evident it sounds, SSH is used in the Application layer to perform this task. It is a secure alternative to the non-protected login protocols (like Telnet) and insecure file transfer methods (like FTP). SSH provides strong encryption and integrity protection. Once a connection has been established between the SSH client and server, the data is transmitted in an encrypted fashion which is implemented using **encryption algorithms like AES (Advanced Encryption Standard**). SSH protocol ensures the integrity of the transmitted data using standard hash algorithms like SHA-2 (Standard Hashing Algorithm). SSH creates a tunnel for transfer of encrypted data between the two connected devices.
- TLS and SSH both provide cryptographic method to establish the connection. But the tunnel that is created in both cases differ. SSH has its own set of protocols for managing the tunnel more efficiently (multiplexing several transfers, password based authentication, terminal management, etc). But there is no such thing in SSL/TLS. So, **SSH is used in Application layer for remote connection**.
- SSH uses TCP as Transport layer since TCP is a **connection oriented protocol** and SSH requires a **secure channel for communication**.

In both the features, we observed **DNS protocol being used over UDP**. DNS protocol helps Internet users and network devices discover websites using human-readable hostnames, instead of IP addresses. DNS generally uses UDP to serve DNS queries. A DNS query is a single UDP request from the DNS Client followed by a single UDP reply from the server. DNS can also use TCP if DNS response is larger than 512 bytes, but in my analysis, I have seen it using UDP only. UDP is a connectionless communication protocol that is primarily used for establishing low-latency and loss-tolerating connections between applications.

Similarly, in both of them IPv4 was also used. IPv4 is **a connectionless protocol** used in packet switched layer networks, such as Ethernet. The network layer is responsible **for carrying data from one host to another**. Network layer defines the data path, the packets should follow to reach the destination. Routers work on this layer and provides mechanism to route data to its destination. Internet Protocol being a layer-3 protocol (OSI) takes data Segments from layer-4 (Transport) and divides it into packets. IP packet encapsulates data unit received from above layer and add to its own header information. So, that is why it is used here in both the features

Ethernet II is a **standard protocol** used across all the parts of networking equipment. It is responsible for the **error-free transmission and separation of the bit stream into blocks**. It stores **the MAC address of the source** and **the destination hosts**.

Since FortiClient is a paid software, I tried to find more functionalities, but was unable to do so. In the free version, only these 2 functionalities were available.

Question 3:

The two functionalities of FortiClient along with messages sequences observed are:

i) Establish VPN connection –

a) DNS Packet:

(ip.dst == 14.139.196.11) || (ip.src == 14.139.196.11) || (udp.stream eq 11)

No.	Time	Source	Destination	Protocol	Length	Info
34	10.156889	192.168.43.147	192.168.43.1	DNS	79	Standard query 0x6864 A agnigarh.iitg.ac.in
35	10.164231	192.168.43.1	192.168.43.147	DNS	95	Standard query response 0x6864 A agnigarh.iitg.ac.in A 14.139.196.11

The first packets that are exchanged are DNS packets which are used to identify the **IP address of the destination**, i.e., agnigarh.iitg.ac.in

b) TCP Packet:

11	8.024917	192.168.43.147	14.139.196.11	TCP	66 49207 → 10443 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
12	8.140535	14.139.196.11	192.168.43.147	TCP	66 10443 → 49207 [SYN, ACK] Seq=0 Ack=1 Win=18352 Len=0 MSS=1370 SACK_PERM=1 WS=1024
13	8.140633	192.168.43.147	14.139.196.11	TCP	54 49207 → 10443 [ACK] Seq=1 Ack=1 Win=16384 Len=0

After extracting the IP address using DNS, my pc sets up TCP connection with the destination. First of all, **TCP Handshaking** is done using **3-way handshake (SYN, SYN-ACK, ACK**). To begin any further communication, both the source and destination must synchronize the **Sequence Numbers** they both have. So, my PC sends SYN message with its own sequence number x. Destination replies with a synchronize-acknowledgement (SYN-ACK) message with its own sequence number y, and acknowledgement number x + 1. Then my PC replies with an acknowledgement message (ACK) with acknowledgement number y + 1. So, the handshaking gets succeeded and further transfer of messages start.

c) TLS Packet:

No.	Time	Source	Destination	Protocol	Length	Info
52	13.384006	192.168.43.147	14.139.196.11	TLSv1.2	274	Client Hello
54	13.530200	14.139.196.11	192.168.43.147	TLSv1.2	1424	Server Hello
56	13.530314	14.139.196.11	192.168.43.147	TLSv1.2	1308	Certificate, Server Key Exchange, Server Hello Done
58	13.572139	192.168.43.147	14.139.196.11	TLSv1.2	372	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
59	13.716690	14.139.196.11	192.168.43.147	TLSv1.2	105	Change Cipher Spec, Encrypted Handshake Message

After TCP handshake, **TLS handshake** is done. After TLS handshake only, the session will start to use **TLS encryption**. During handshake, first both parties agree on the TLS version they will use and cipher suites they will be using. The authentication of the identity of the server is being done using server's public key and the SSL certificate authority's digital signature. Then the session keys are generated in order to use symmetric encryption. We can see various messages exchanged like **Client Hello**, **Server Hello**, **Certificate**, **Client Key Exchange**, **Change Cipher Spec**, etc.

d) Data Packets:

114	9.274243	192.168.43.147	14.139.196.11	TLSv1.2	136 Application Data
116	9.323737	14.139.196.11	192.168.43.147	TCP	54 10443 → 49689 [ACK] Seq=4058 Ack=4419 Win=42 Len=0
117	9.328640	14.139.196.11	192.168.43.147	TLSv1.2	188 Application Data
118	9.329439	192.168.43.147	14.139.196.11	TLSv1.2	300 Application Data
119	9.423349	14.139.196.11	192.168.43.147	TCP	54 10443 → 49689 [ACK] Seq=4192 Ack=4673 Win=42 Len=0
120	9.466001	14.139.196.11	192.168.43.147	TLSv1.2	284 Application Data
121	9.466111	14.139.196.11	192.168.43.147	TLSv1.2	556 Application Data
122	9.466157	192.168.43.147	14.139.196.11	TCP	54 49689 → 10443 [ACK] Seq=4919 Ack=4924 Win=60 Len=0
123	9.466802	192.168.43.147	14.139.196.11	TLSv1.2	124 Application Data
125	9.543619	14.139.196.11	192.168.43.147	TLSv1.2	236 Application Data
126	9.663714	14.139.196.11	192.168.43.147	TCP	54 10443 → 49689 [ACK] Seq=5106 Ack=4989 Win=44 Len=0
127	9.747884	192.168.43.147	14.139.196.11	TCP	54 49689 → 10443 [ACK] Seq=4989 Ack=5106 Win=60 Len=0
128	9.748057	192.168.43.147	14.139.196.11	TLSv1.2	124 Application Data
129	9.893864	14.139.196.11	192.168.43.147	TCP	54 10443 → 49689 [ACK] Seq=5106 Ack=5059 Win=44 Len=0

After TLS handshaking is done, the data is transferred using TLS protocol, and is termed as **Application Data.** The client acknowledges with an ACK message through TCP for each packet it successfully receives. Further these messages are actually covered up in TLS packets so as to avoid any third party to intercept them and analyse them. I have explained in further sections how to deeply analyse these **Application Data** packets.

e) Terminating VPN session:

14	0.271249	14.139.196.11	192.168.43.147	TCP	54 10443 → 49217 [ACK] Seq=1 Ack=359 Win=265 Len=0
32	1.607371	192.168.43.147	14.139.196.11	TCP	54 49217 → 10443 [FIN, ACK] Seq=359 Ack=1 Win=61 Len=0
33	1.785935	14.139.196.11	192.168.43.147	TCP	54 10443 → 49217 [FIN, ACK] Seq=1 Ack=360 Win=265 Len=0
34	1.786033	192.168.43.147	14.139.196.11	TCP	54 49217 → 10443 [ACK] Seq=360 Ack=2 Win=61 Len=0

When the session is closed, **TCP termination handshake** is performed to closed the connection. First Client sends a FIN packet, along with an ACK packet (which was an acknowledgement to an earlier packet). The server then sends an ACK packet as an acknowledgement to the earlier FIN packet. Following this, the server sends its own FIN packet to close the connection. The client receives this packet, and closes the connection after sending an ACK packet. So, the handshake is different from 3-way handshake. It is an actually **a pair of 2-way handshake**. So, the TCP packets are used to terminate the connection.

ii) ssh remote machine: Open Connection –

On my local IP, all the packets were of similar nature (as in **Establish TCP connection**) when the SSH session is initiated. Same process was followed – first DNS packets were exchanged, followed by TCP and TLS packets along with handshaking. After the session initiated, the packets exchanged were termed as application data and it occurred using TLS packets.

But VPN creates an IP address for me and all the activities are routed through that particular IP address. So to properly analyse what all packets are exchanged during remote ssh connection, Wireshark need to observe the packets over that particular assigned IP address. When I did that, the actual packets that were exchanged were visible to me now, which were earlier marked as "Application Data" under TLS packet.

a) Establish SSH connection:

17	5.213191	172.18.16.4	172.16.70.64	SSHv2	82 Client: Protocol (SSH-2.0-PuTTY_Release_0.74)
19	5.360156	172.16.70.64	172.18.16.4	SSHv2	95 Server: Protocol (SSH-2.0-OpenSSH_7.2p2 Ubuntu-4ubuntu2.8)
20	5.365443	172.18.16.4	172.16.70.64	SSHv2	1222 Client: Key Exchange Init
21	5.505883	172.16.70.64	172.18.16.4	SSHv2	1030 Server: Key Exchange Init
22	5.512007	172.18.16.4	172.16.70.64	SSHv2	102 Client: Elliptic Curve Diffie-Hellman Key Exchange Init
23	5.653755	172.16.70.64	172.18.16.4	SSHv2	262 Server: Elliptic Curve Diffie-Hellman Key Exchange Reply, New Keys
24	5.687338	172.18.16.4	172.16.70.64	SSHv2	134 Client: New Keys, Encrypted packet (len=64)

We can see that SSH Packets are used for the remote connection. To first establish the connection using SSH, first of all **SSH handshake** is being performed. It is a **cryptographic handshake** that is made to ensure that the client can verify that it is communicating with the correct server. Then the transport layer connecting between the client and the server (here, TCP) is encrypted using a **symmetric cipher**. Following this, the client authenticates itself to the server. Ultimately after all these steps, the client interacts with the remote host over the **encrypted connection**.

b) Data Packets:

7	2.351768	172.18.16.4	172.16.70.64	SSH	118 Client: Encrypted packet (len=64)
9	2.516793	172.16.70.64	172.18.16.4	SSH	118 Server: Encrypted packet (len=64)
14	2.671636	172.18.16.4	172.16.70.64	SSH	118 Client: Encrypted packet (len=64)
15	2.796909	172.16.70.64	172.18.16.4	SSH	118 Server: Encrypted packet (len=64)
16	2.927584	172.18.16.4	172.16.70.64	SSH	118 Client: Encrypted packet (len=64)
17	3.066386	172.16.70.64	172.18.16.4	SSH	118 Server: Encrypted packet (len=64)
19	3.367635	172.18.16.4	172.16.70.64	SSH	118 Client: Encrypted packet (len=64)
20	3.506323	172.16.70.64	172.18.16.4	SSH	118 Server: Encrypted packet (len=64)
22	4.527612	172.18.16.4	172.16.70.64	SSH	118 Client: Encrypted packet (len=64)

These data packets were observed on the virtual IP allotted by VPN. These are SSH packets which **got encrypted after SSH handshake**. Both client and server sends packets in turn so as to acknowledge each other that it received an earlier packet.

61	13.717122	192.168.43.147	14.139.196.11	TLSv1.2	222 Application Data
62	13.861326	14.139.196.11	192.168.43.147	TLSv1.2	667 Application Data
64	13.862333	192.168.43.147	14.139.196.11	TLSv1.2	223 Application Data
71	13.991634	14.139.196.11	192.168.43.147	TLSv1.2	976 Application Data
73	13.998915	192.168.43.147	14.139.196.11	TLSv1.2	529 Application Data
75	14.144118	14.139.196.11	192.168.43.147	TLSv1.2	918 Application Data
77	14.144922	192.168.43.147	14.139.196.11	TLSv1.2	510 Application Data
80	14.280087	14.139.196.11	192.168.43.147	TLSv1.2	326 Application Data
82	14.281167	192.168.43.147	14.139.196.11	TLSv1.2	514 Application Data

These data packets were observed on my local IP when the SSH session was going on. The packets are covered up as TLS packets hiding its underlying details by FortiClient. It is necessary because its VPN duty to establish a secure connection so that traffic can be protected from the third party and can't be easily snooped. But when we moved onto that particular IP, then all the packets can be easily determined. Yet we cannot determine the contents of those packets then also as SSH is a secure connection, and all the packets get encrypted before being transferred.

c) Terminating SSH connection:

25 4.687074	172.18.16.4	172.16.70.64	TCP	54 49213 → 22 [ACK] Seq=321 Ack=385 Win=510 Len=0
27 4.689681	172.18.16.4	172.16.70.64	TCP	54 49213 → 22 [FIN, ACK] Seq=321 Ack=561 Win=509 Len=0
28 4.826800	172.16.70.64	172.18.16.4	TCP	54 22 → 49213 [FIN, ACK] Seq=561 Ack=322 Win=288 Len=0
29 4.826903	172.18.16.4	172.16.70.64	TCP	54 49213 → 22 [ACK] Seq=322 Ack=562 Win=509 Len=0

On terminating SSH connection, similar procedure was followed as in **Terminating VPN connection.** A pair of **2-way handshake** between client and server is performed and after this, the connection is closed.

Question 4:

i) Establish VPN connection:

	Network – 1 (Home WiFi)			Network – 2 (Mobile Hotspot)		
Details	**9 am**	**2 pm**	**8 pm**	**9 am**	**2 pm**	**8 pm**
Throughput *(in bytes/sec)*	2980	2749	1764	2413	2950	3147
RTT *(in ms)*	62.60	33.5	33.78	31.41	23.59	24.89
Packet Size *(in bytes)*	242	252	209	217	254	229
No of Packets Lost	0	0	0	0	0	0
No of UDP Packets	2	2	6	4	2	4
No of TCP Packets	141	129	181	177	126	173
No. of responses per one request sent	0.72	0.70	0.83	0.81	0.75	0.75

ii) ssh remote machine – Open Connection:

	Network – 1 (Home WiFi)			Network – 2 (Mobile Hotspot)		
Details	**9 am**	**2 pm**	**8 pm**	**9 am**	**2 pm**	**8 pm**
Throughput *(in bytes/sec)*	780	910	653	734	929	866
RTT *(in ms)*	34.12	206.92	300.86	102.42	106.34	102.22
Packet Size *(in bytes)*	236	211	218	211	211	236
No of Packets Lost	0	0	0	0	0	0
No of UDP Packets	0	0	0	0	0	0
No of TCP Packets	28	39	32	39	39	28
No. of responses per one request sent	0.87	0.86	1	0.86	0.86	0.87

iii) ssh remote machine – Close Connection:

	Network – 1 (Home WiFi)			Network – 2 (Mobile Hotspot)		
Details	**9 am**	**2 pm**	**8 pm**	**9 am**	**2 pm**	**8 pm**
Throughput *(in bytes/sec)*	748	1470	390	701	1297	1487
RTT *(in ms)*	125.6	21.25	89.12	149.26	36.02	25.54
Packet Size *(in bytes)*	103	121	112	115	124	121
No of Packets Lost	0	0	0	0	0	0
No of UDP Packets	0	0	0	0	0	0
No of TCP Packets	18	28	46	33	27	28
No. of responses per one request sent	0.8	0.87	0.70	0.83	0.93	0.87

Question 5:

In each case, the **destination IP address** was **14.139.196.11** (agnigarh.iitg.ac.in). But the IPs assigned by VPN to me (**source**) varied with each fresh login. Some of them were:

i. 172.18.16.3
ii. 172.18.16.18
iii. 172.18.16.16
iv. 172.18.16.3

These IPs were **private IP addresses**. It may be that my PC gets connected to the different routers at the institute and so, at different times of the day, the connection varied depending upon the traffic conditions.

So, the IP address of the institute server remained same to me at different times of the day, but those assigned by FortiClient varied. It is clear why those varied since VPN creates Virtual Network, and it will assign different IPs to each client with each connection.

But sometimes servers have a number of IP addresses and each time when we try to connect, we might connect to a different IP. This is done to balance the huge amount of traffic, and improve network speed. It is also reliable since failure of some servers will not affect other servers, which may cater to the needs of incoming traffic.