Rishabh Verma 20103027

Applied Cryptography

Assignment 1

Wireshark getting started:

1. List 3 different protocols that appear in the protocol column in the unfiltered packet-listing window in step 7 above.

Ans: UDP, TCP and DNS.

2. How long did it take from when the HTTP GET message was sent until the HTTP OK reply was received? (By default, the value of the Time column in the packet listing window is the amount of time, in seconds, since Wireshark tracing began. To display the Time field in time-of-day format, select the Wireshark View pull-down menu, then select Time Display Format, then select Time-of-day.)

Ans: Time taken = 9.753566 - 9.735362 = 0.018204 seconds

3. What is the Internet address of gaia.cs.umass.edu (also known as wwwnet.cs.umass.edu)? What is the Internet address of your computer?

Ans: IP of source(my computer) : 192.168.1.4
IP of gaia.cs.umass.edu : 128.119.245.12

4. Print the two HTTP messages (GET and OK) referred to in question 2 above. To do so, select Print from the Wireshark File command menu, select the “Selected Packet Only” and “Print as displayed” radial buttons, and then click OK.

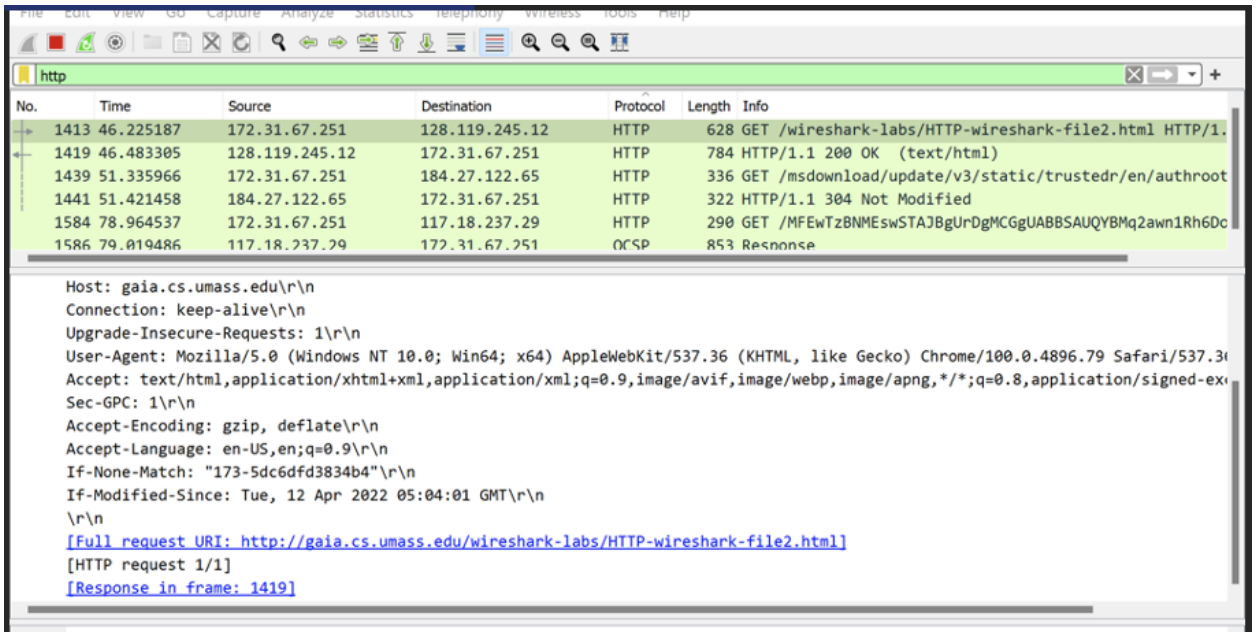
```
No.     Time           Source                Destination           Protocol Length Info
     96 5.303407       128.119.245.12        192.168.1.4           HTTP     504    HTTP/1.1 200
OK  (text/html)
Frame 96: 504 bytes on wire (4032 bits), 504 bytes captured (4032 bits) on interface en0, id 0
Ethernet II, Src: zte_41:64:40 (f4:f6:47:41:64:40), Dst: Apple_46:a8:76 (f8:ff:c2:46:a8:76)
Internet Protocol Version 4, Src: 128.119.245.12, Dst: 192.168.1.4
Transmission Control Protocol, Src Port: http (80), Dst Port: 58386 (58386), Seq: 1, Ack: 519,
Len: 438
Hypertext Transfer Protocol
Line-based text data: text/html (3 lines)
```

/var/folders/70/6mdh7ckj7nz56t9705xll_bh0000gn/T/wireshark_Wi-FiTM08Q1.pcapng 141 total packets, 2 shown

```
No.     Time           Source                Destination           Protocol Length Info
     59 5.020353       192.168.1.4           128.119.245.12        HTTP     584    GET /
wireshark-labs/INTRO-wireshark-file1.html HTTP/1.1
Frame 59: 584 bytes on wire (4672 bits), 584 bytes captured (4672 bits) on interface en0, id 0
Ethernet II, Src: Apple_46:a8:76 (f8:ff:c2:46:a8:76), Dst: zte_41:64:40 (f4:f6:47:41:64:40)
Internet Protocol Version 4, Src: 192.168.1.4, Dst: 128.119.245.12
Transmission Control Protocol, Src Port: 58386 (58386), Dst Port: http (80), Seq: 1, Ack: 1,
Len: 518
Hypertext Transfer Protocol
```

HTTP:

1. The Basic HTTP GET



1. Is your browser running HTTP version 1.0 or 1.1?What version of HTTP is the server running?
Ans: 1.1

2. What languages (if any) does your browser indicate that it can accept to the server?

Ans: English-US

3. What is the IP address of your computer? Of the gaia.cs.umass.edu server?

Ans: Computer IP – 172.31.67.251
Server IP – 128.119.245.12

4. What is the status code returned from the server to your browser?

Ans: 200

5. When was the HTML file that you are retrieving last modified at the server?

Ans:12 April,2022

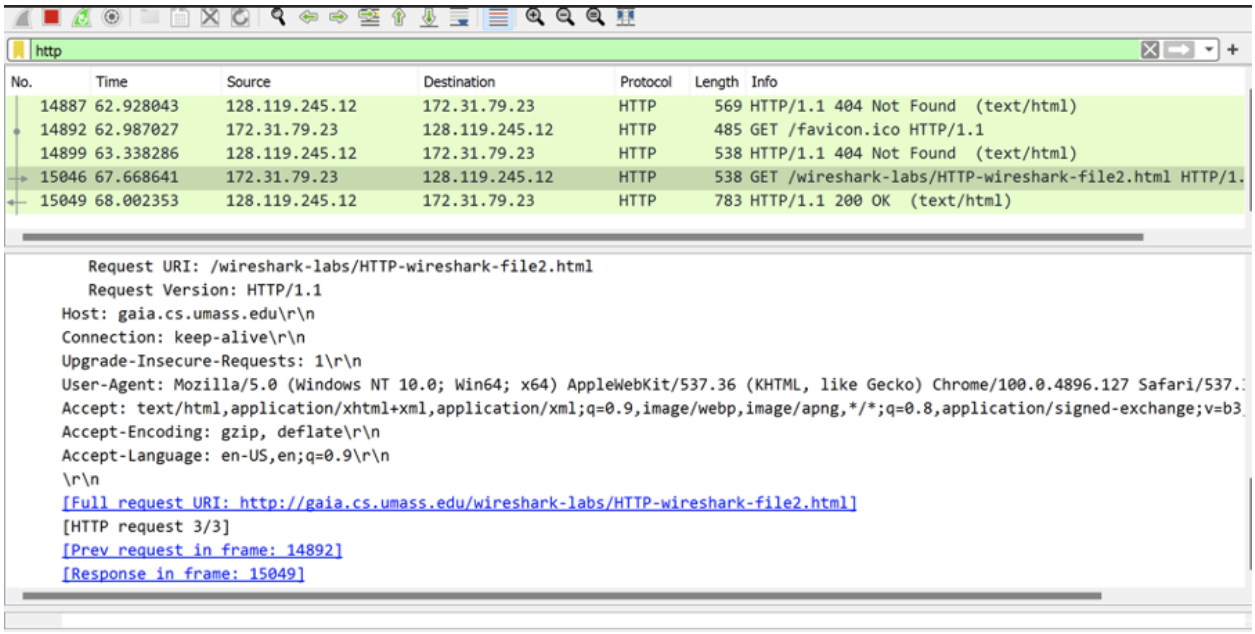
6. How many bytes of content are being returned to your browser?

Ans:569 Bytes

7. By inspecting the raw data in the packet content window, do you see any headers within the data that are not displayed in the packet-listing window?If so name one.

Ans: No, I do not see any headers that are not displayed in the packet window.

2. The HTTP Conditional GET



8. Inspect the contents of the first HTTP GET request from your browser to the server. Do you see an “IF-MODIFIED-SINCE” line in the HTTP GET?

Ans: No, there is no “IF-MODIFIED-SINCE” line in the HTTP GET.

9.Inspect the contents of the server response. Did the server explicitly return the contents of the file? How can you tell?

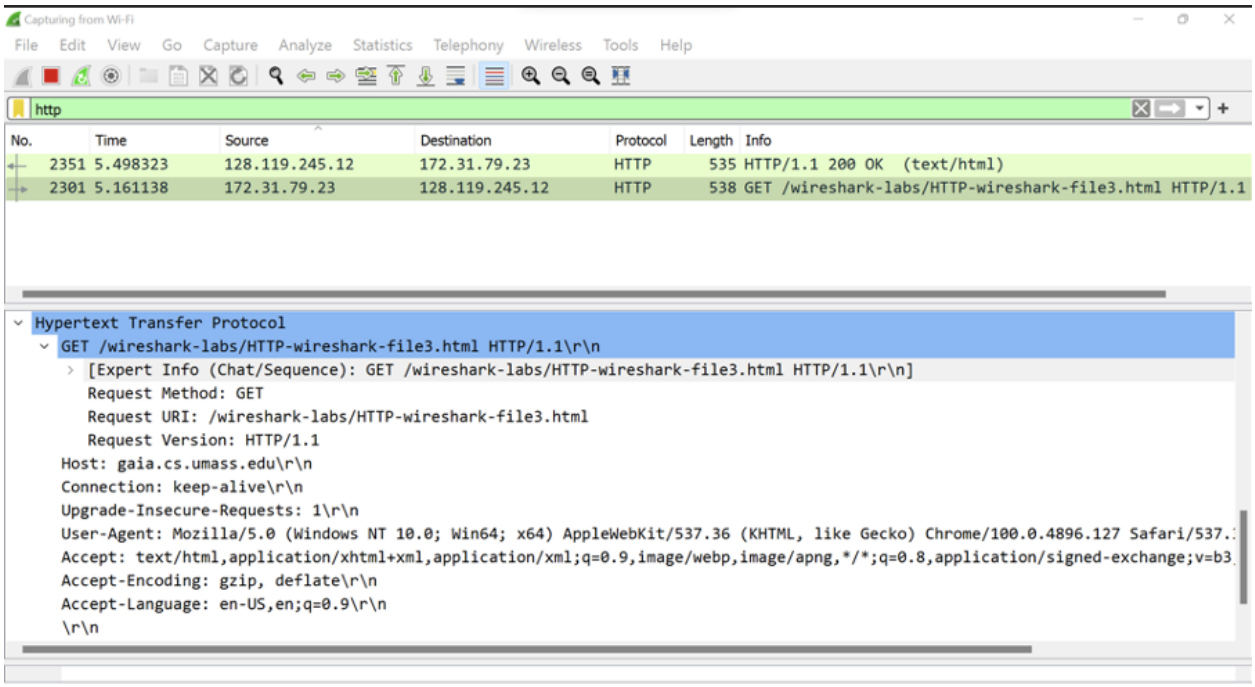
Ans: Yes, the server explicitly returned the contents of the file. I am able to tell this because of the Line-based text data in the OK response to the GET.

10.Now inspect the contents of the second HTTP GET request from your browser to the server. Do you see an “IF-MODIFIED-SINCE:” line in the HTTP GET? If so, what information follows the “IF-MODIFIED-SINCE:” header?
Ans: An “IF-MODIFIED-SINCE:” line in the HTTP GET was present. The information that followed the “IF-MODIFIED-SINCE:” header that was not present in the other HTTP GET was the date it was modified.

11.What is the HTTP status code and phrase returned from the server response to this second HTTP GET? Did the server explicitly return the contents of the file? Explain.

Ans: The HTTP status code that the server responded with was a 200 OK, surprisingly. This means that the server explicitly returned the contents of the file.

3. Retrieving Long Documents



12. How many HTTP GET request messages were sent by your browser?

Ans: My browser sent only one HTTP GET request message. Packet number 18 contained the GET message for the Bill of Rights

13. How many data-containing TCP segments were needed to carry the single HTTP response?

Ans: Packet 23 contained the response to the HTTP GET request

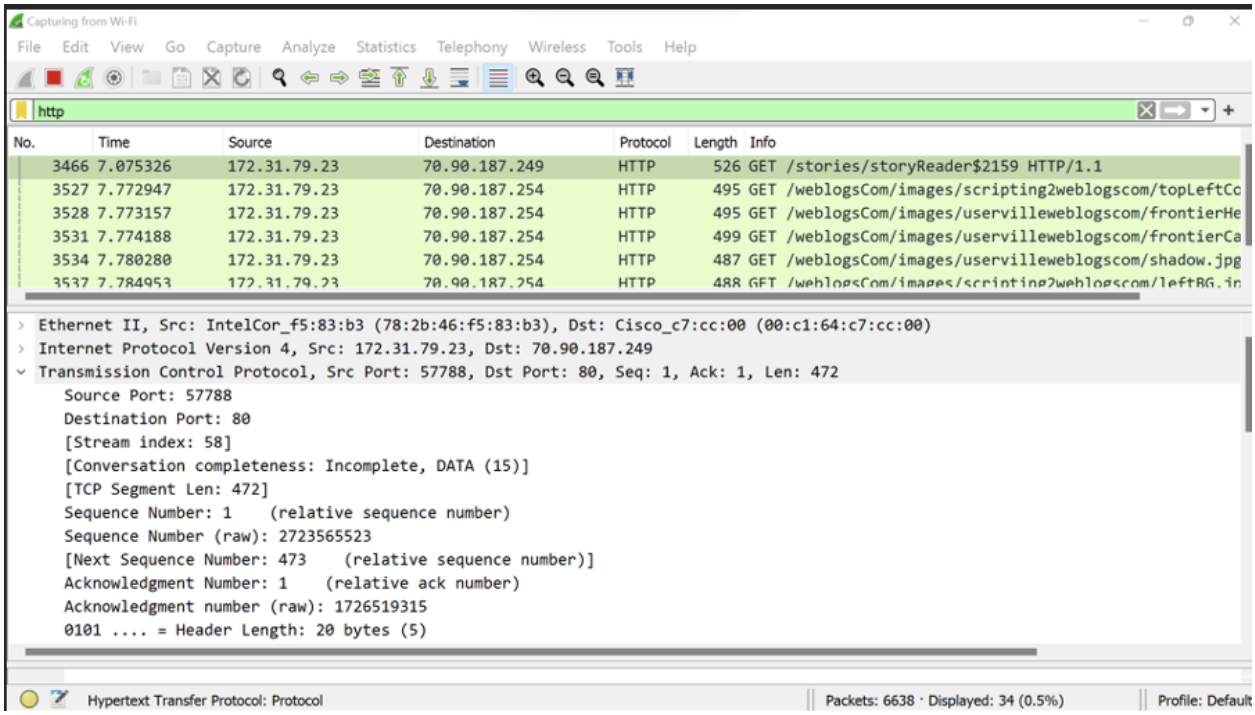
14. What is the status code and phrase associated with the response to the HTTP GET request?

Ans: The status code from this packet was a 200, and the phrase was an OK.

15. Are there any HTTP status lines in the transmitted data associated with a TCP-induced “Continuation”?

Ans: 3 data-containing TCP segments were needed to carry the single HTTP response and the text of the Bill of Rights.

4. HTML Documents with Embedded Objects



16. How many HTTP GET request messages were sent by your browser? To which Internet addresses were these GET requests sent?

Ans: 3 HTTP GET request messages were sent by my browser. It sent them to 128.119.245.12, 165.193.140.14, and 128.119.240.90. The last two gets are for the separate locations of the images on the initial web page.

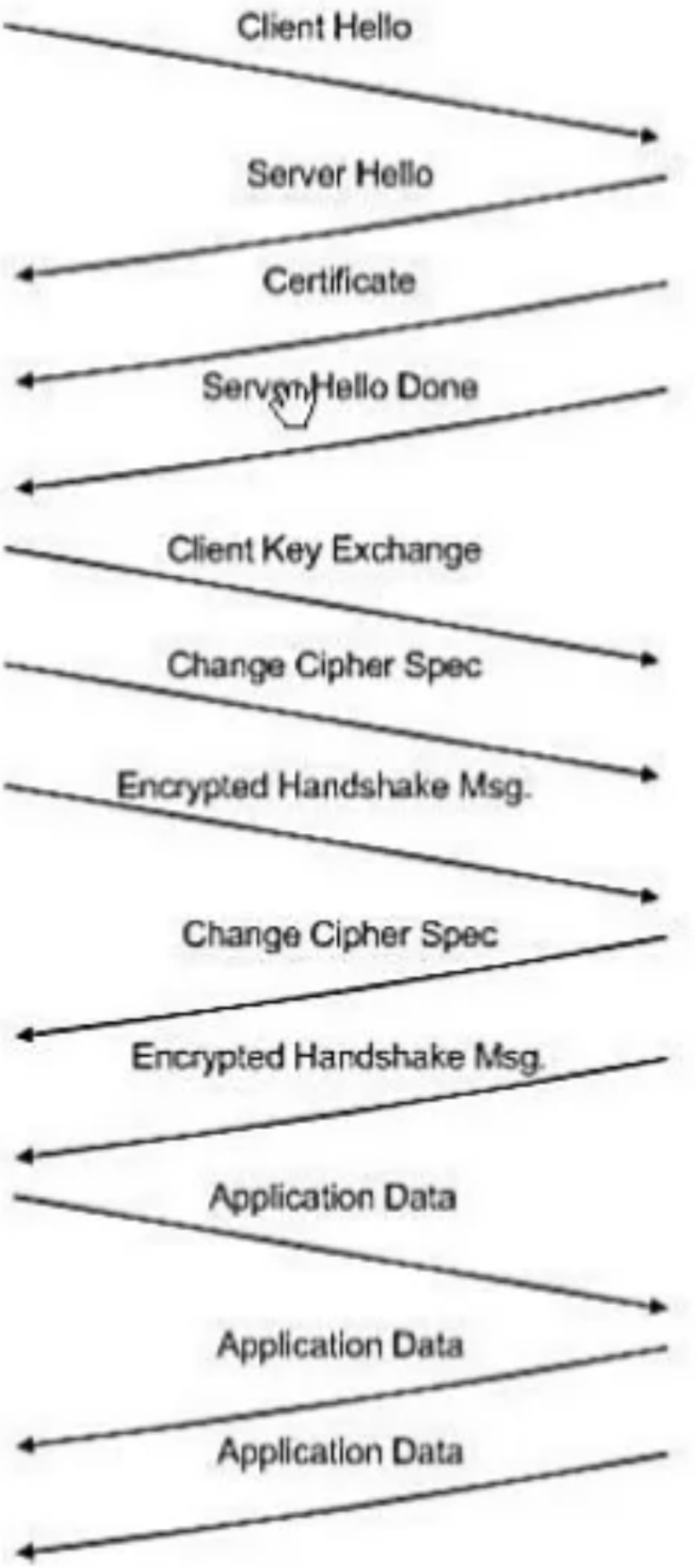
17. Can you tell whether your browser downloaded the two images serially, or whether they were downloaded from the two web sites in parallel? Explain.

Ans: From the looks of it, it would appear they are downloaded serially. This is because a GET is sent out after an OK response is seen. From a quick Google search, it appears this is not common. The Google search suggests that these pictures were downloaded in parallel.

SSL:

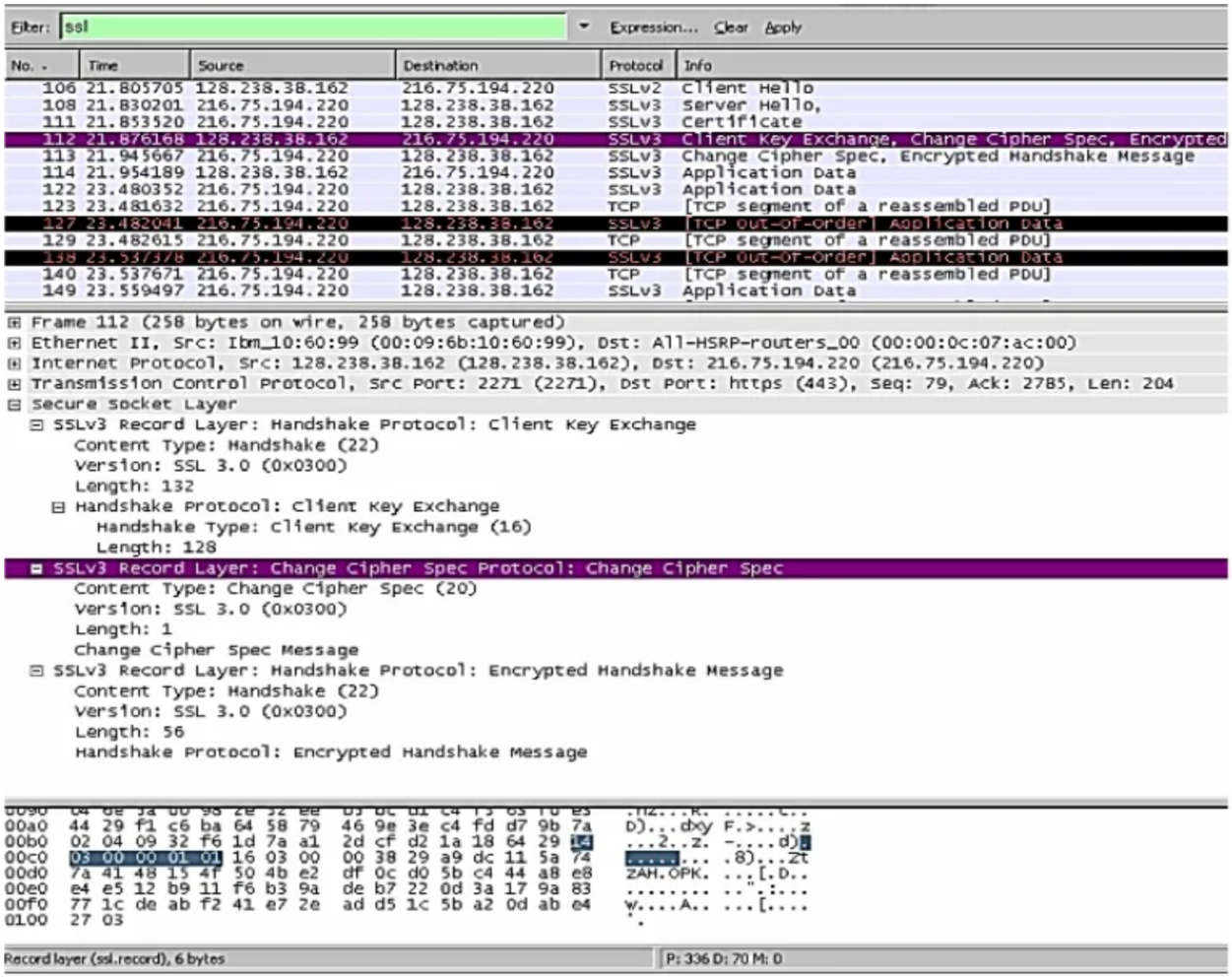
1. For each of the first 8 Ethernet frames, specify the source of the frame (client or server), determine the number of SSL records that are included in the frame, and list the SSL record types that are included in the frame. Draw a timing diagram between client and server, with one arrow for each SSL record.

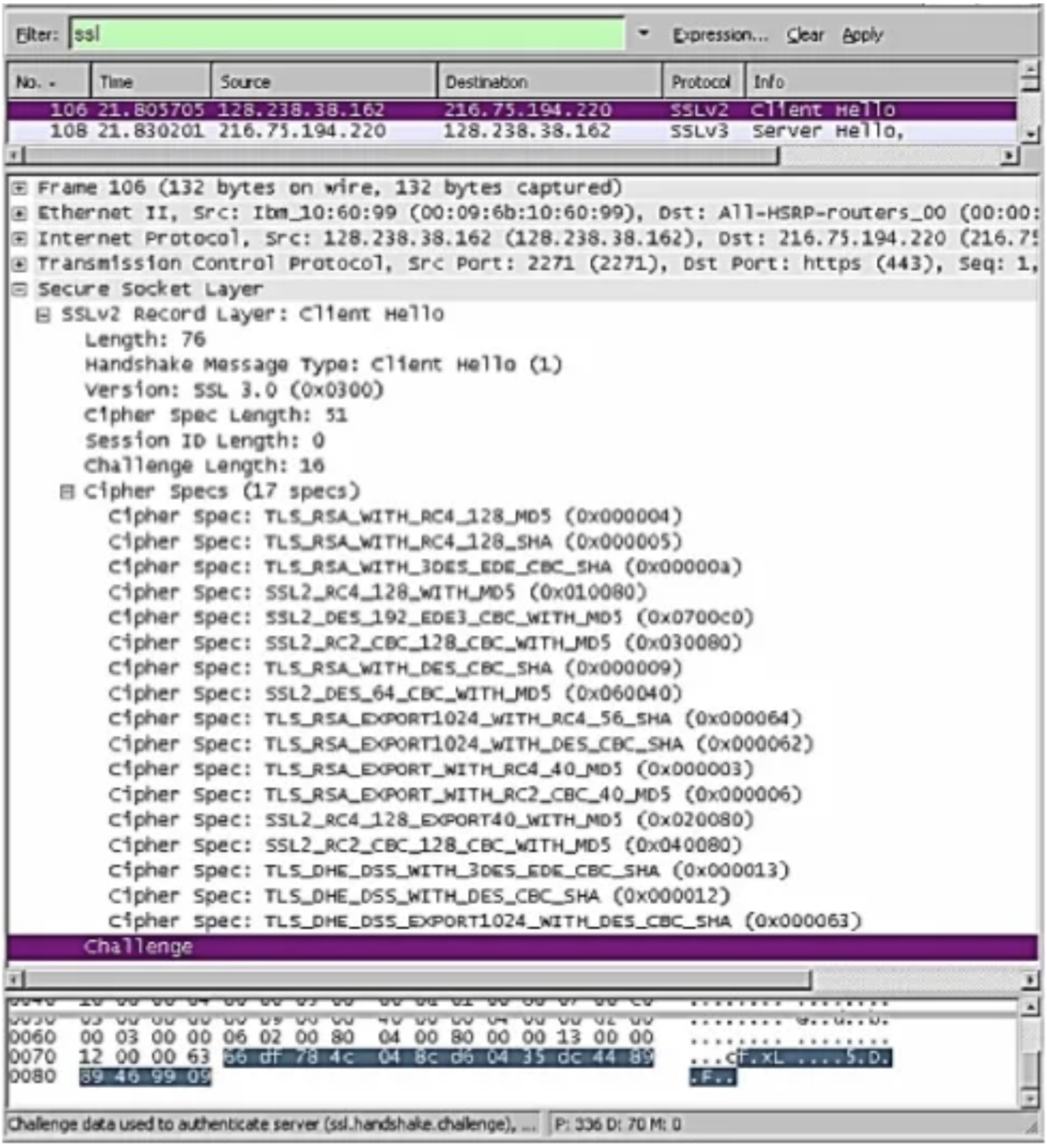
Frame	Source	SSL Count	SSL Type
106	Client	1	Client Hello
108	Server	1	Server Hello
111	Server	2	Certificate Server Hello Done
112	Client	3	Client Key Exchange Change Cipher Spec Encrypted Handshake Message
113	Server	2	Change Cipher Spec Encrypted Handshake Message
114	Client	1	Application Data
122	Server	1	Application Data
127	Server	1	Application Data


Client Hello
Server Hello
Certificate
Server Hello Done
Client Key Exchange
Change Cipher Spec
Encrypted Handshake Msg.
Change Cipher Spec
Encrypted Handshake Msg.
Application Data
Application Data
Application Data

2. Each of the SSL records begins with the same three fields (with possibly different values). One of these fields is “content type” and has length of one byte. List all three fields and their lengths.

Ans: Content type: 1byte
Version: 2bytes
Length: 2bytes





ClientHello Record:

3. Expand the ClientHello record. (If your trace contains multiple ClientHello records, expand the frame that contains the first one.) What is the value of the content type?

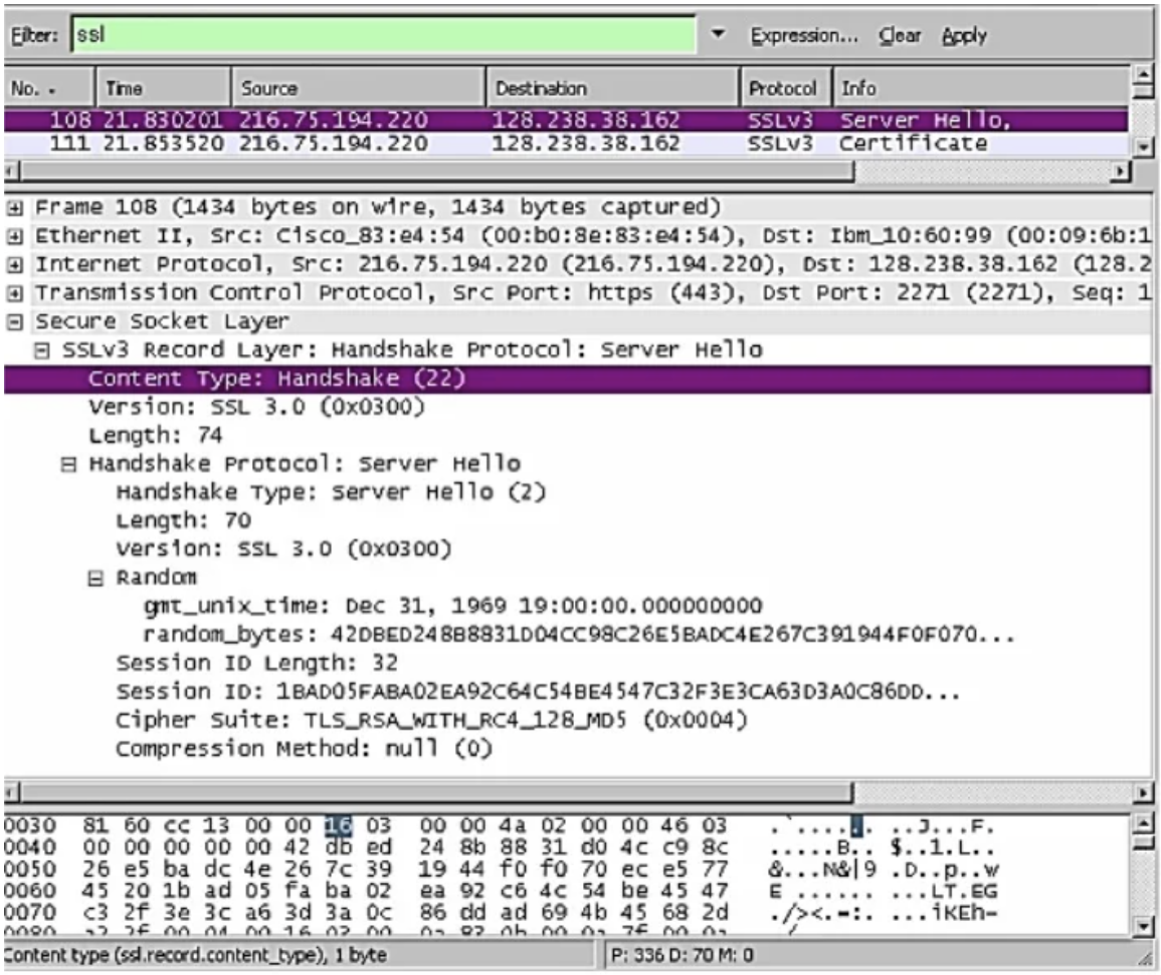
Ans: The content type is 22, for handshake message, with a handshake type of 01, client hello.

4. Does the ClientHello record contain a nonce (also known as a “challenge”)? If so, what is the value of the challenge in hexadecimal notation?

Ans: The client hello challenge is 66df 784c 048c 35dc 4489 8946 9909

5. Does the ClientHello record advertise the cyber suites it supports? If so, in the first listed suite, what are the public-key algorithm, the symmetric-key algorithm, and the hash algorithm?

Ans: The first suite uses RSA for public key cryptography, RC4 for the symmetric key cipher, and uses MD5 hash algorithm.



ServerHello Record:

6. Locate the ServerHello SSL record. Does this record specify a chosen cipher suite? What are the algorithms in the chosen cipher suite?

Ans: The cipher suite uses RSA for public key cryptography, RC4 for the symmetric key cipher, and uses MD5 hash algorithm.

7. Does this record include a nonce? If so, how long is it? What is the purpose of the client and server nonces in SSL?

Ans: Yes this record does include a nonce listed under Random. The nonce is 32-bit long, 28 for data and 4 for time. The purpose is to prevent a reply attack.

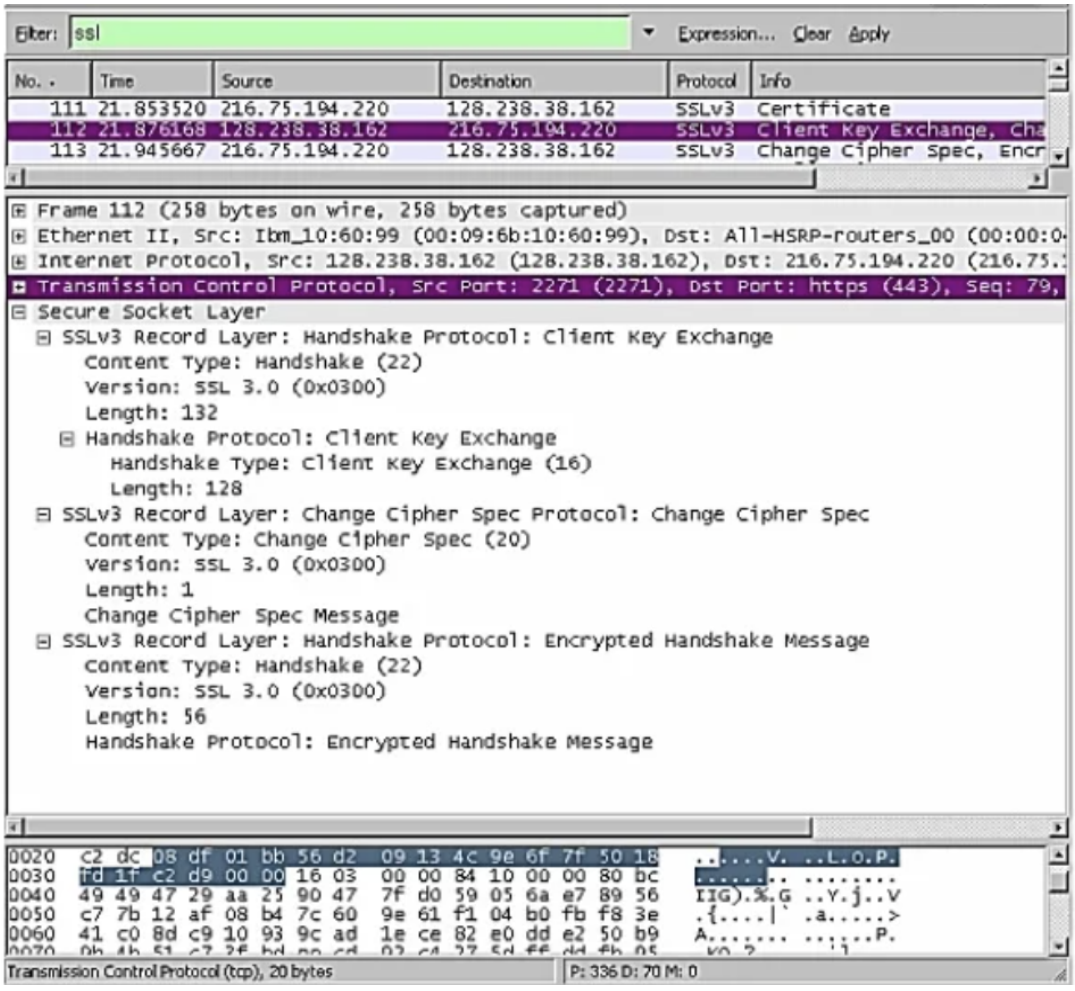
8. Does this record include a session ID? What is the purpose of the session ID?

Ans: Yes it does. It provides a unique persistent identifier for the SSL session which is sent in the clear. The client may resume the same session later by using the server-provided session ID when it sends the ClientHello.

9. Does this record contain a certificate, or is the certificate included in a separate record? Does the certificate fit into a single Ethernet frame?

Ans: There is no certificate; it is in another record. It does fit into a single Ethernet frame.

Client Key Exchange Record:



10. Locate the client key exchange record. Does this record contain a pre-master secret? What is this secret used for? Is the secret encrypted? If so, how? How long is the encrypted secret?
Ans:

Yes, it does contain a premaster secret. It is used by both the server and client to make a master secret, which is used to generate session keys for MAC and encryption. The secret gets encrypted using the server's public key, which the client extracted from the certificate sent by the server. The secret is 128 bytes long.

Change Cipher Spec Record (sent by the client) and Encrypted Handshake Record:

11. What is the purpose of the Change Cipher Spec record? How many bytes is the record in your trace?
Ans:

The purpose of the Change Cipher Spec record is to indicate that the contents of the following SSL records sent by the client (data, not header) will be encrypted. This record is 6 bytes long: 5 for the header and 1 for the message segment.

12. In the encrypted handshake record, what is being encrypted? How?
Ans:

In the encrypted handshake record, a MAC of the concatenation of all the previous handshake messages sent from this client is generated and sent to the server.

13. Does the server send a changed cipher record and an encrypted handshake record to the client? How are those records different from those sent by the client?
Ans:

Yes the server will also send a Change Cipher Spec record and encrypted handshake to the client. The server's encrypted handshake record is different from that sent by the client because it contains the concatenation of all the handshake messages sent from the server rather than from the client. Otherwise the records would end up being the same.

Application Data

14. How is the application data being encrypted? Do the records containing application data include a MAC? Does Wireshark distinguish between the encrypted application data and the MAC?
Ans:

Application data is encrypted using symmetric key encryption algorithm chosen in the handshake phase (RC4) using the keys generated using the pre-master key and nonces from both client and server. The client encryption key is used to encrypt the data being sent from client to server and the server encryption key is used to encrypt the data being sent from the server to the client.

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

Ans:

The version of SSL used changes from SSLv2 in the initial ClientHello message to SSLv3 in all following message exchanges.

Also, during resumes the handshake process is slightly different from the initial one. The client does not need another cert so the server never sends it. It just has to send a new nonce followed by Change Cipher Spec and Encrypted Handshake records from the server to client. After a response from the client then application data can be sent.