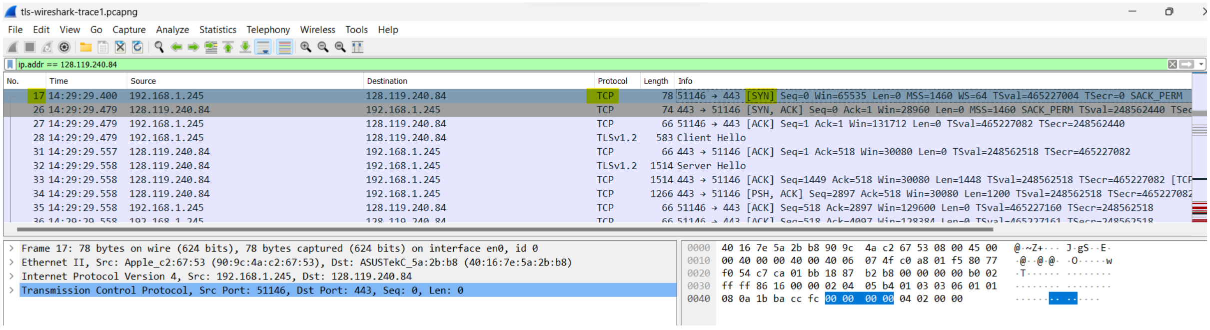
Lab 07_TLS

I. Capturing and analyzing Ethernet frames

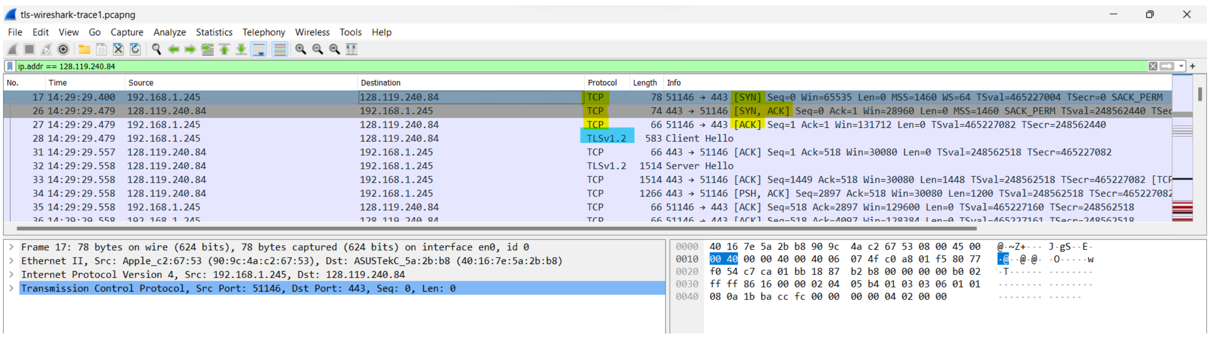
1. What is the packet number in your trace that contains the initial TCP SYN message? (By "packet number," we meant the number in the "No." column at the left of the Wireshark display, not the sequence number in the TCP segment itself).

The packet number in your trace that contains the initial TCP SYN message: 17



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

TLS is built on top of TCP/IP, TCP connection has to be established (and thus the TCP handshake successfully finished) before the TLS handshake can start, the client must first complete the 3-way TCP handshake with the server.



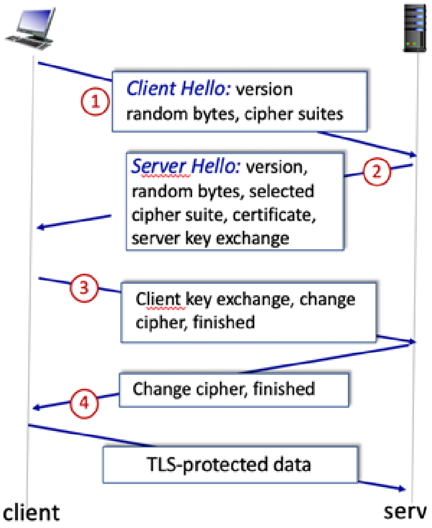
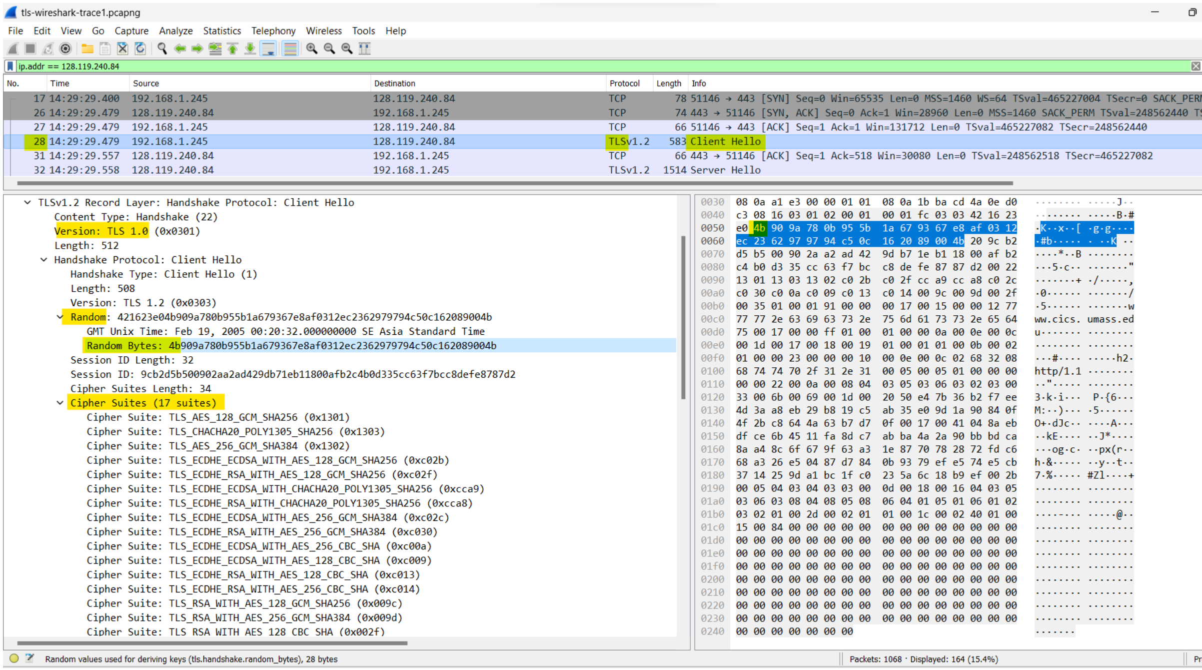


Figure 2: TLS handshaking

- **The TLS Handshake: Client Hello message**



3. **What is the packet number in your trace that contains the TLS *Client Hello* message?**

   the packet number in your trace that contains the TLS *Client Hello* message is: 28

4. **What version of TLS is your client running, as declared in the *Client Hello* message?**

   version of TLS is your client running: 1.0

5. **How many cipher suites are supported by your client, as declared in the *Client Hello* message? A cipher suite is a set of related cryptographic algorithms that determine how session keys will be derived, and how data will be encrypted and be digitally signed via a HMAC algorithm.**

   They are 17 Cipher suites supported by your client.

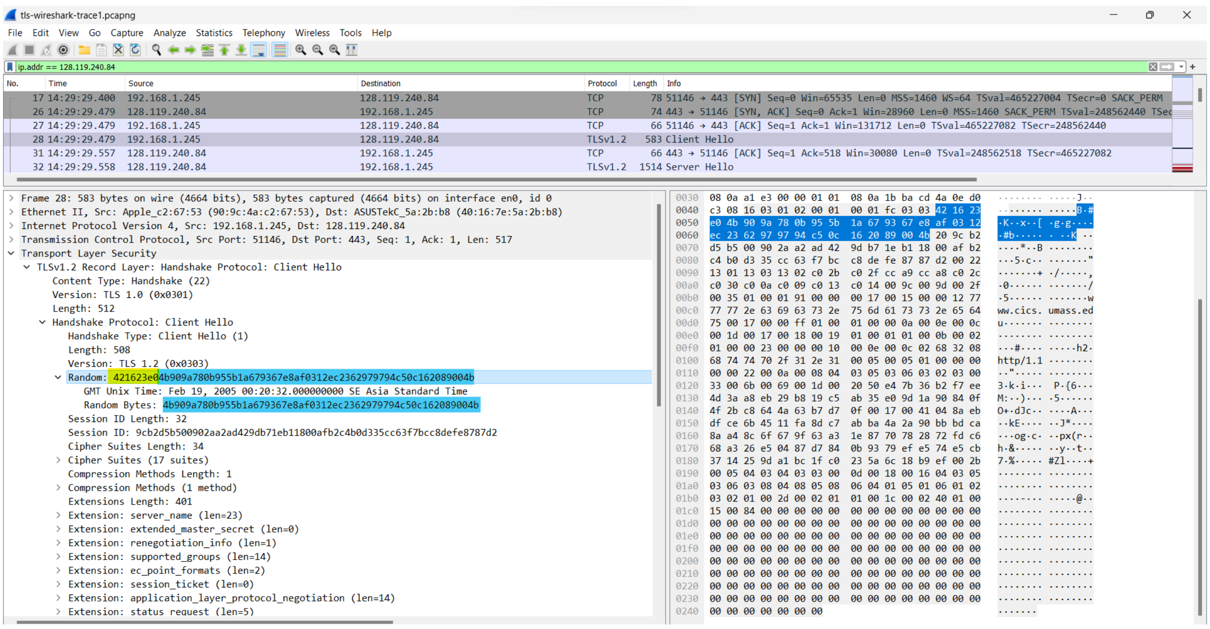
6. **Your client generates and sends a string of "random bytes" to the server in the *Client Hello* message. What are the first two hexadecimal digits in the random bytes field of the *Client Hello* message? Enter the two hexadecimal digits (without spaces between the hex digits and without any leading '0x' , using lowercase letters where needed). *Hint:* be careful to fully dig into the Random field to find the Random Bytes subfield (do not consider the GMT UNIX Time subfield of Random)**.

   The first two hexadecimal digits in the random bytes field of the *Client Hello* message are: 4b

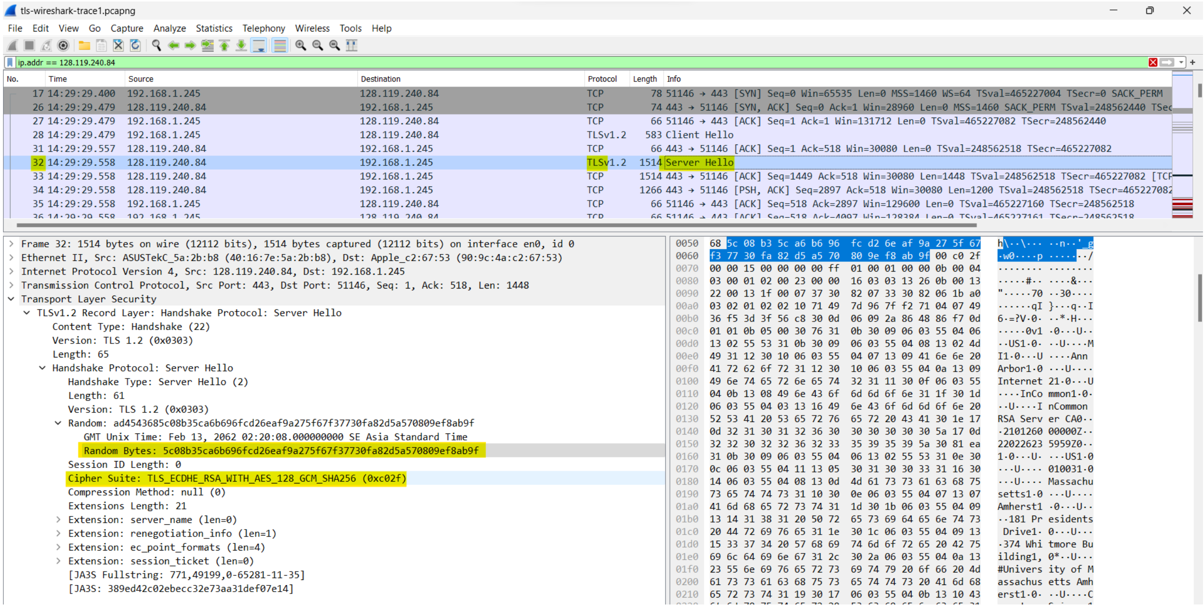
7. **What is the purpose(s) of the "random bytes" field in the *Client Hello* message? Note: you'll have do some searching and reading to get the answer to this question; see section 8.6 and in [RFC 5246](RFC 5246) (section 8.1 in RFC 5246 in particular).**

   Random bytes (28 bytes) should be generated by the client using a secure random number generator. The source of entropy for random number generation (to generate the key for encryption) will depend on the operating system and implementation of the client software.

   Its main purpose is to stop replay attacks. In essence, the addition of this unique number prevents an attacker from being able to send the client the very same packets that it had received in a previous connection.



- **The TLS Handshake: Server Hello message**



8. **What is the packet number in your trace that contains the TLS *Server Hello* message?**

   The packet number in your trace that contains the TLS *Server Hello* message is: 32

9. **Which cipher suite has been chosen by the server from among those offered in the earlier *Client Hello* message?**

   Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)

10. **Does the *Server Hello* message contain random bytes, similar to how the *Client Hello* message contained random bytes? And if so, what is/are their purpose(s)?**

Server: Random Bytes: 5c08b35ca6b696fcd26eaf9a275f67f37730fa82d5a570809ef8ab9f

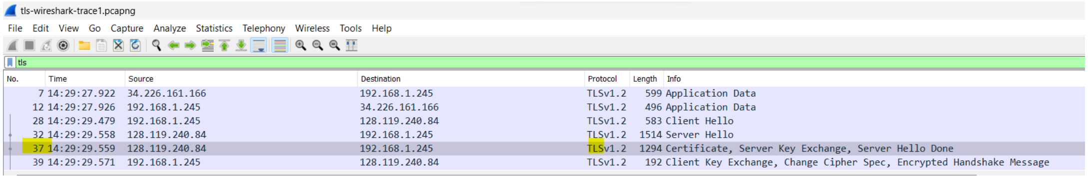
Client: Random Bytes: 4b909a780b955b1a679367e8af0312ec2362979794c50c162089004b

Different.

- **The TLS Handshake: Server Hello message -** public key certificate

11. **What is the packet number in your trace for the TLS message part that contains the public key certificate for the www.cics.umass.edu server (actually the www.cs.umass.edu server)?**

    The packet number in your trace for the TLS message part that contains the public key certificate for the www.cics.umass.edu server is 37

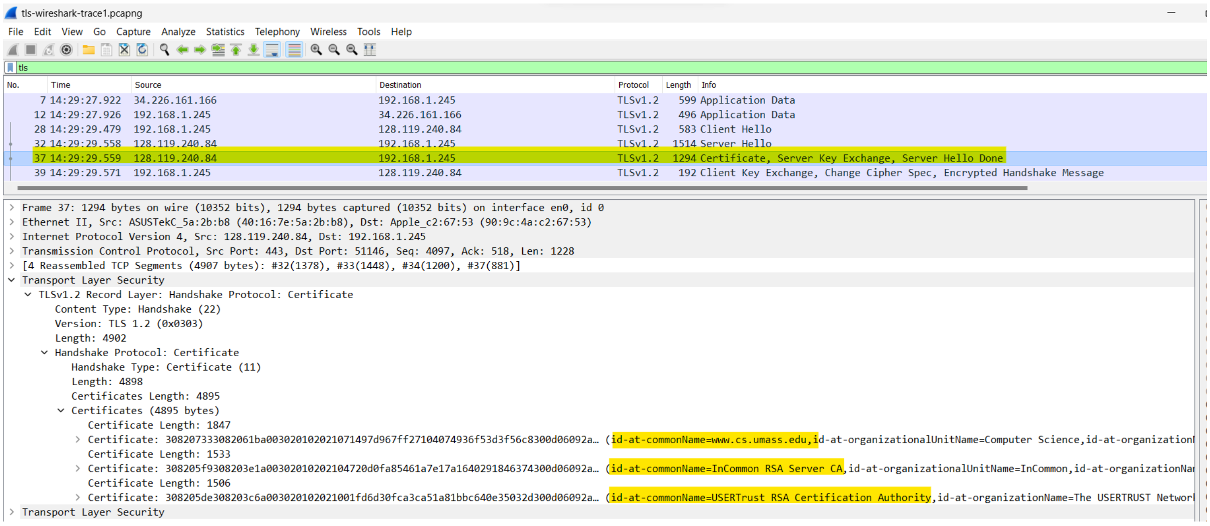


12. **A server may return more than one certificate. If more than one certificate is returned, are all of these certificates for www.cs.umass.edu? If not all are for www.cs.umass.edu, then who *are* these other certificates for? You can determine who the certificate is for by checking the id-at-commonName field in the retuned certificate.**

    If more than one certificate is returned, are all of these certificates not for www.cs.umass.edu.

    Certificate: id-at-commonName=InCommon RSA Server CA

    Certificate: id-at-commonName=USERTrust RSA Certification Authority

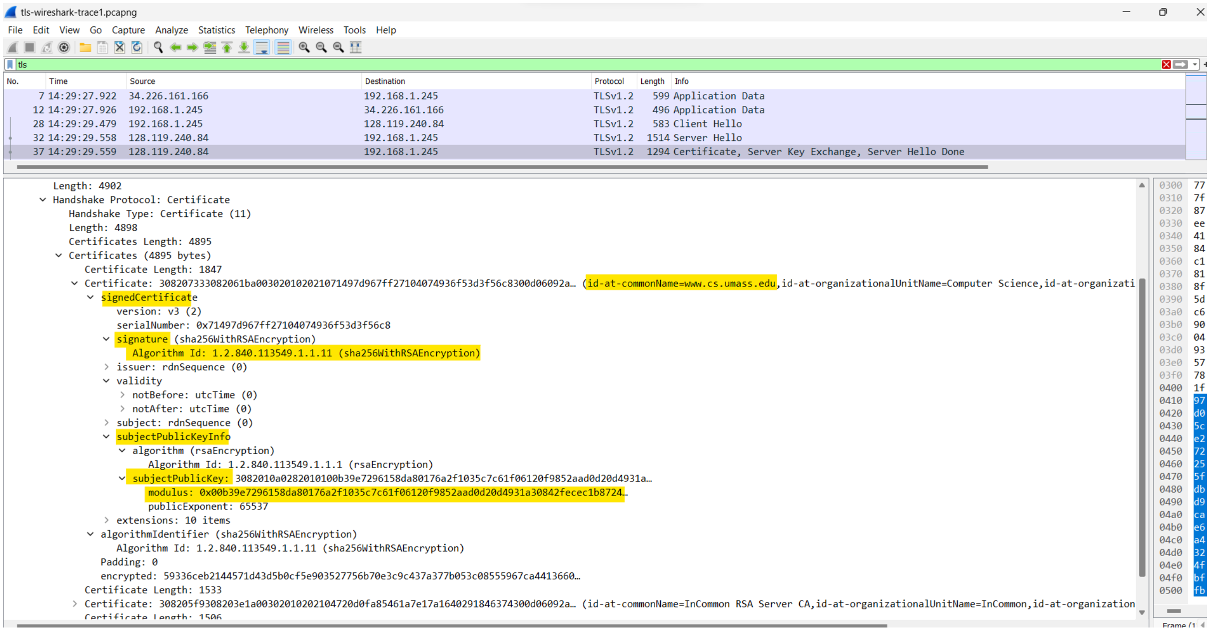


13. **What is the name of the certification authority that issued the certificate for id-at-commonName=www.cs.umass.edu?**

    Certificate: 308207333082061ba003020102021071497d967ff27104074936f53d3f56c8300d06092a…(id-at-commonName=www.cs.umass.edu,id-at-organizationalUnitName=Computer Science,id-at-organizationName=University of Massachusetts Amherst,id-at-stre

14. **What digital signature algorithm is used by the CA to sign this certificate? Hint: this information can be found in signature subfield of the SignedCertificate field of the certificate for www.cs.umass.edu.**

    Algorithm Id: 1.2.840.113549.1.1.11 (sha256WithRSAEncryption)



15. **Let's take a look at what a real public key looks like! What are the first four hexadecimal digits of the modulus of the public key being used by www.cics.umass.edu? Enter the four hexadecimal digits (without spaces between the hex digits and without any leading '0x' , using lowercase letters where needed, and including any leading 0s after '0x'). Hint: this information can be found in subjectPublicKeyInfo subfield of the SignedCertificate field of the certificate for www.cs.umass.edu.**

    The first four hexadecimal digits of the modulus of the public key being used by www.cics.umass.edu are: 00b3

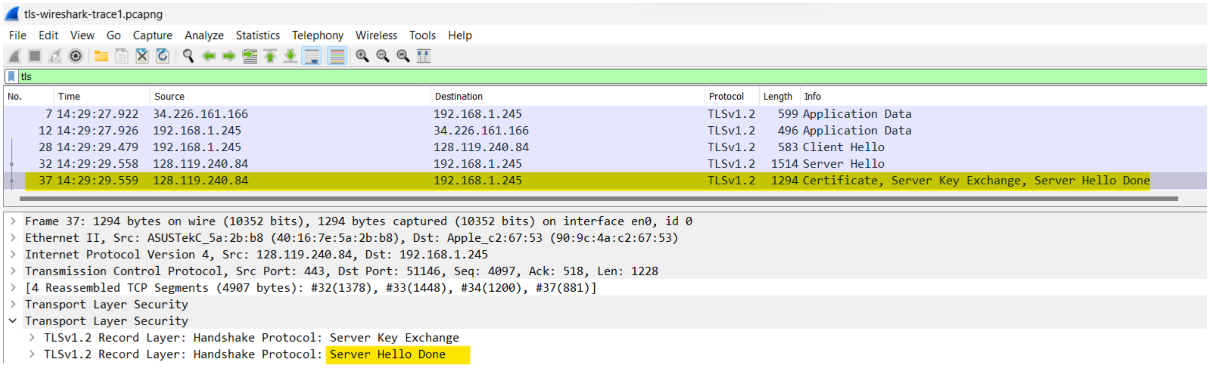
16. **Look in your trace to find messages between the client and a CA to get the CA's public key information, so that the client can verify that the CA-signed certificate sent by the server is indeed valid and has not been forged or altered. Do you see such message in your trace? If so, what is the number in the trace of the first packet sent from your client to the CA? If not, explain why the client did not contact the CA.**

    Not see. The client did not contact the CA because the certificate is signed by a CA about which the client has no knowledge at all. In this case the client treats the certificate as invalid.

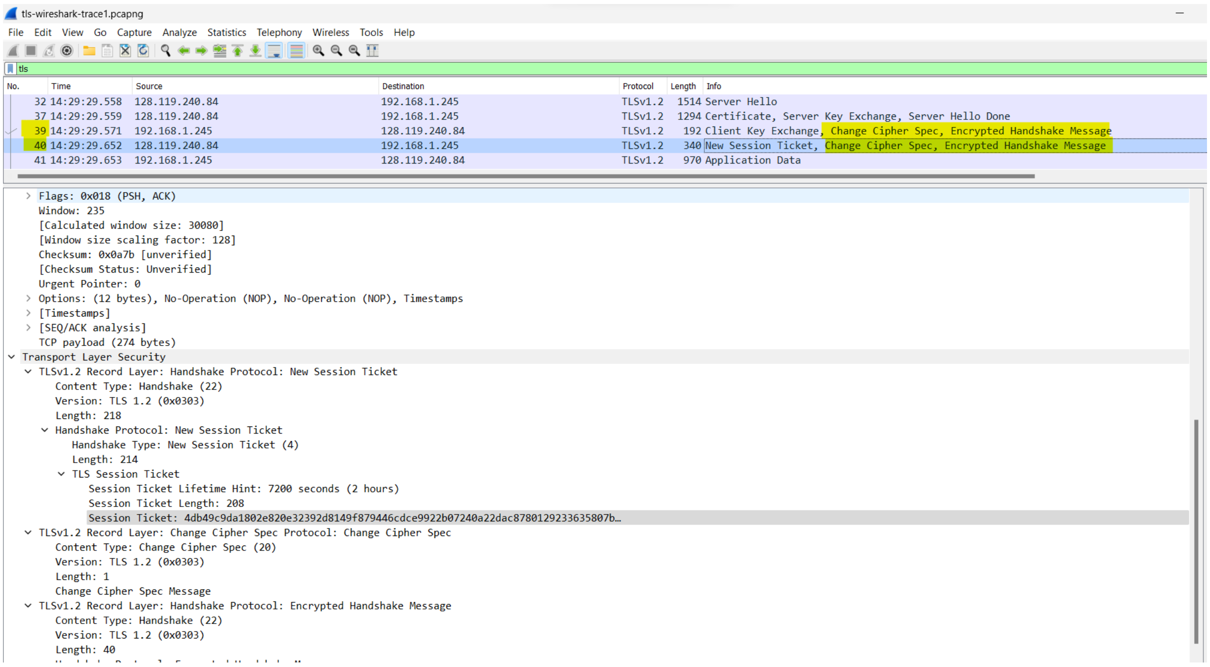
The *Server Hello* message is always terminated by an explicit *Server Hello Done* record.

17. **What is the packet number in your trace for the TLS message part that contains the *Server Hello Done* TLS record?**

    The packet number in your trace for the TLS message part that contains the *Server Hello Done* TLS record is: 37



- **The TLS Handshake: wrapping up the handshake**



18. What is the packet number in your trace for the TLS message that contains the public key information, *Change Cipher Spec,* and *Encrypted Handshake* message, being sent from client to server?

The packet number in your trace for the TLS message that contains the public key information, *Change Cipher Spec,* and *Encrypted Handshake* message, being sent from client to server is: 39

19. Does the client provide its own CA-signed public key certificate back to the server? If so, what is the packet number in your trace containing your client's certificate?

The packet number in your trace containing your client's certificate is: 40

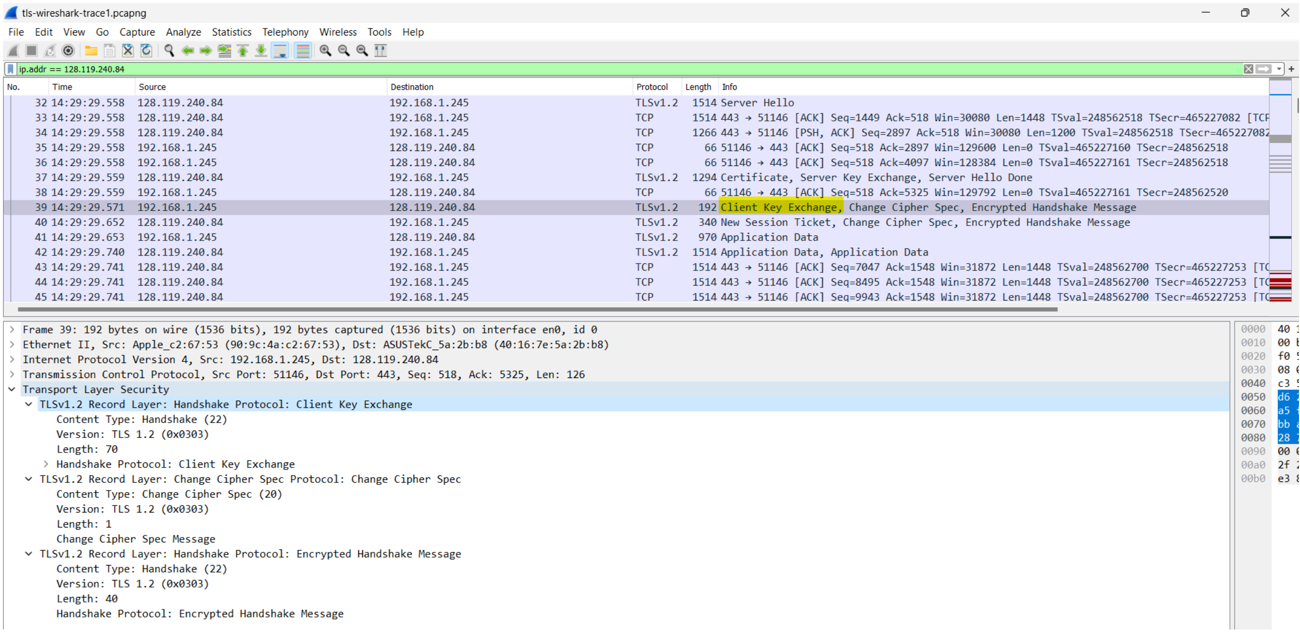
- **Application data**

20. What symmetric key cryptography algorithm is being used by the client and server to encrypt application data (in this case, HTTP messages)?

TLS protocol

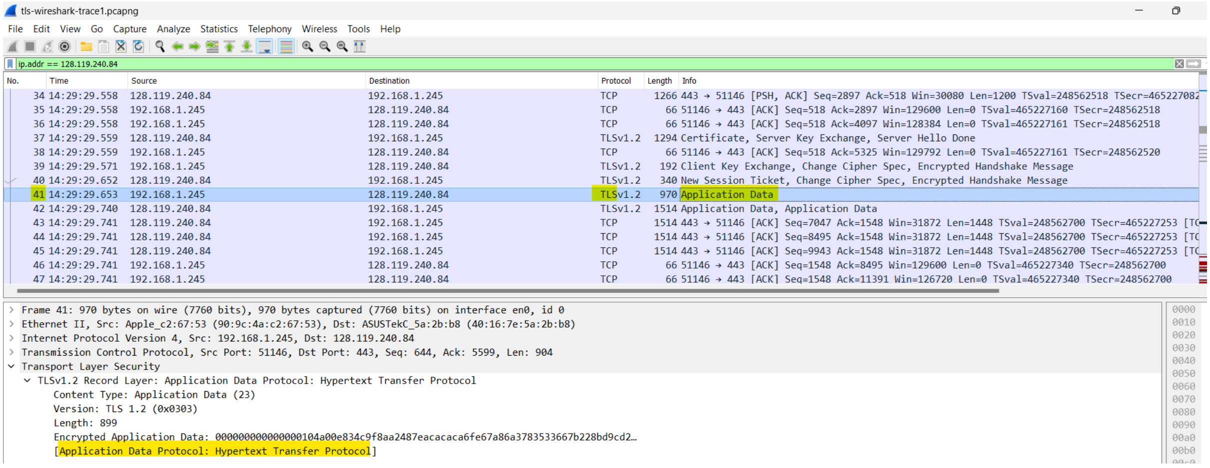
21. In which of the TLS messages is this symmetric key cryptography algorithm finally decided and declared?

The symmetric key cryptography algorithm is decided and declared in the "ClientKeyExchange" message. This message contains the client's pre-master secret, which is used to generate the symmetric key for encryption and decryption of data in the TLS session.

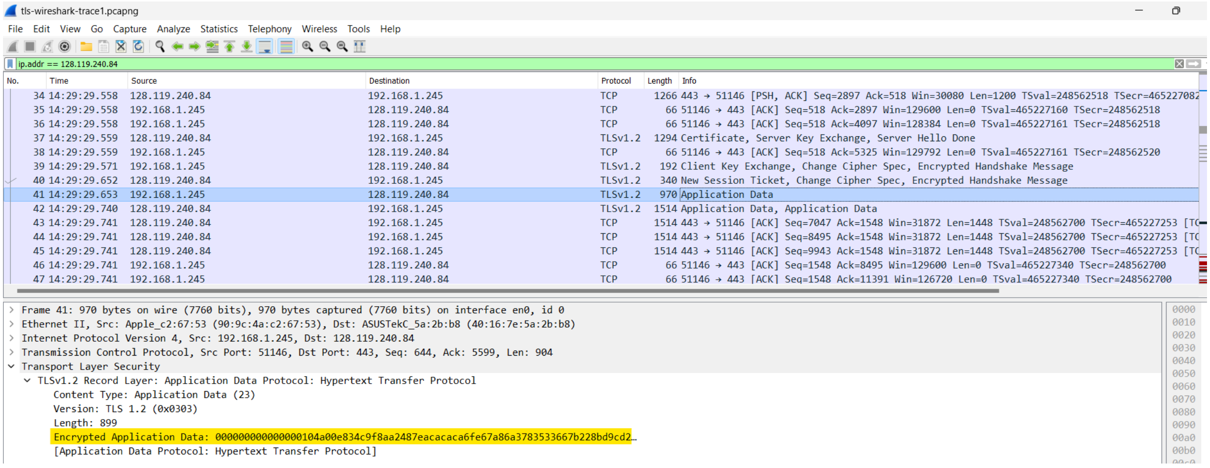


22. What is the packet number in your trace for the first encrypted message carrying application data from client to server?

The packet number in your trace for the first encrypted message carrying application data from client to server is 41



23. What do you think the content of this encrypted application-data is, given that this trace was generated by fetching the homepage of www.cics.umass.edu?



Encrypted Application Data: 0000000000000001104a00e834c9f8aa2487eacacaca6fe67a86a3783533667b228bd9cd2…

24. What packet number contains the client-to-server TLS message that shuts down the TLS connection? Because TLS messages are encrypted in our Wireshark traces, we can't actually look *inside* a TLS message and so we'll have to make an educated guess here.

Packet number contains the client-to-server TLS message that shuts down the TLS connection: 41