1

CRYPTOGRAPHY (CTG)

Diploma in Cybersecurity and Digital Forensics (Dip in CSF)

Academic Year (AY) `21/`22 – Semester 2

WEEK 8.1

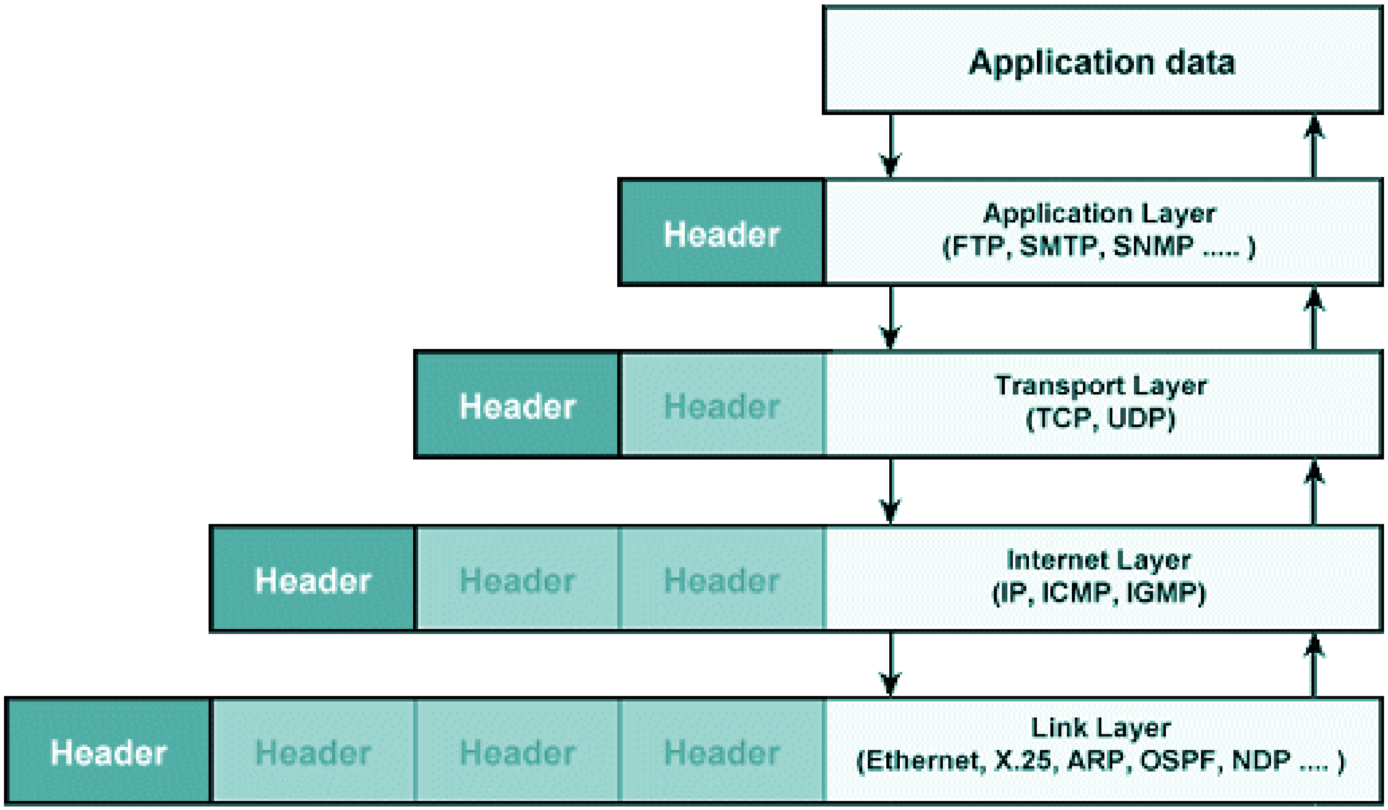
SECURE SOCKET LAYER (SSL) / TRANSPORT LAYER SECURITY (TLS)

Last Updated: 15/10/2021

2

What was the need for SSL/TLS?

No Data Encryption Mechanism in TCP/IP Layers!



Weakness in TCP/IP Protocol

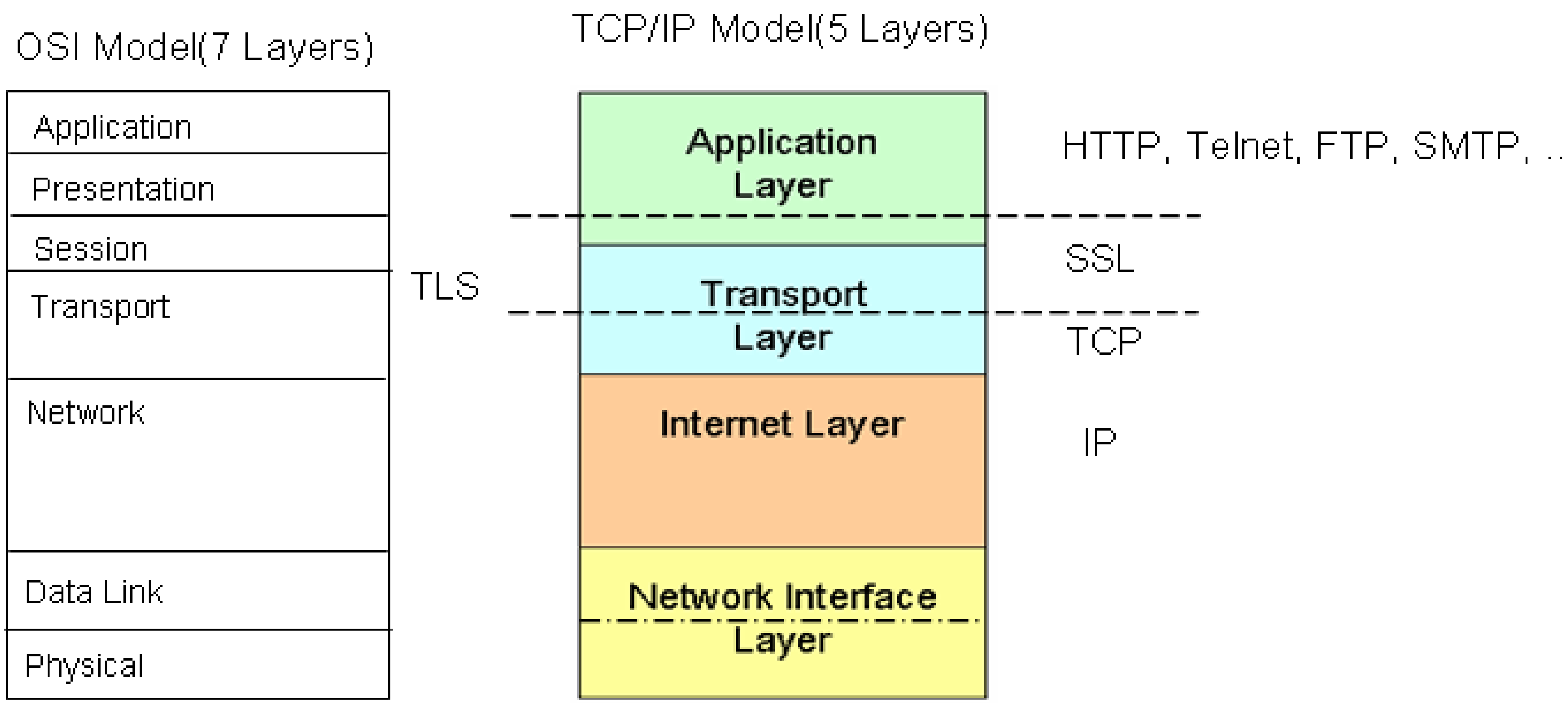
- None of the layers provide encryption to protect data transmitted from one end to another end.
- When the World-Wide-Web(WWW) was first introduced in the late 80s, the web browser/client and the web server communicate with one another in plaintext using the Hypertext Transfer Protocol (HTTP). These raw and unencrypted data traveling across the Internet can be easily sniffed and read by anyone.
- This problem together with the need to making sure that the web client is communicating with the right web server and vice versa, making the world-wide-web not an ideal platform for eCommerce.

The Solution

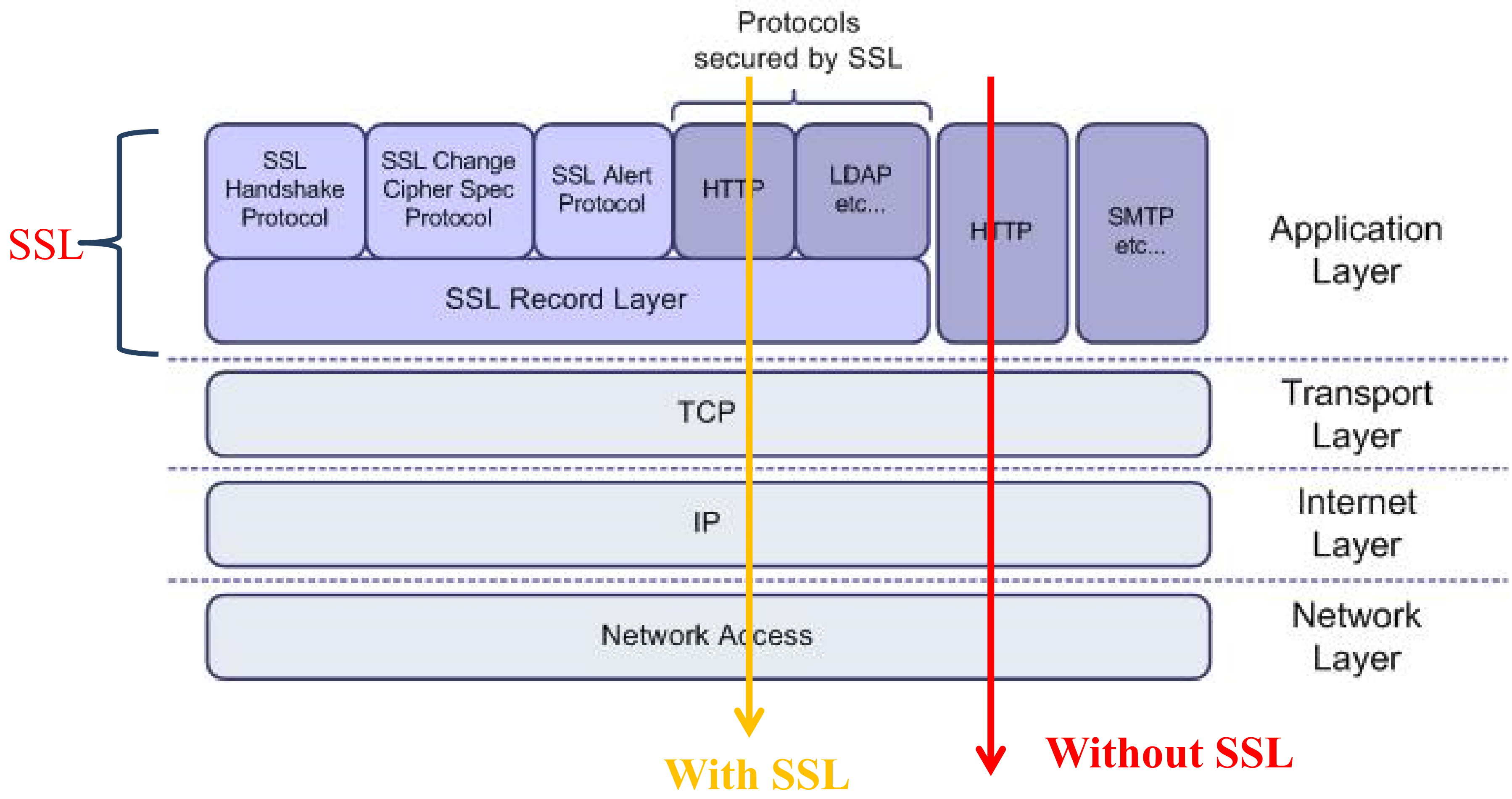
- SSL (Secure Socket Layer) version 2 was first introduced in 1994 by Netscape to solve the mentioned problems.
- In the subsequent year, Netscape introduced SSLv3 and its success led to IETF(Internet Engineering Task Force) to adopt it and to enhance it to become a standard.
- Subsequently, IETF released the standards: TLS v1.0 and TLS v1.2 and TLS v1.3 in 1999, 2006, and 2008 respectively.
  - TLS stands for Transport Layer Security
  - HTTPS = HTTP over SSL/TLS used in all the browsers.
- SSL v3.0 must not be used due to many recent attacks such as POODLE attack
- TLS v1.2 and TLS v1.3 are recommended for security

Knowledge Component

SSL/TLS in the TCP/IP Protocol Suite



SSL/TLS in the TCP/IP Protocol Suite



8

Transport Layer Security (TLS)

Purpose of TLS

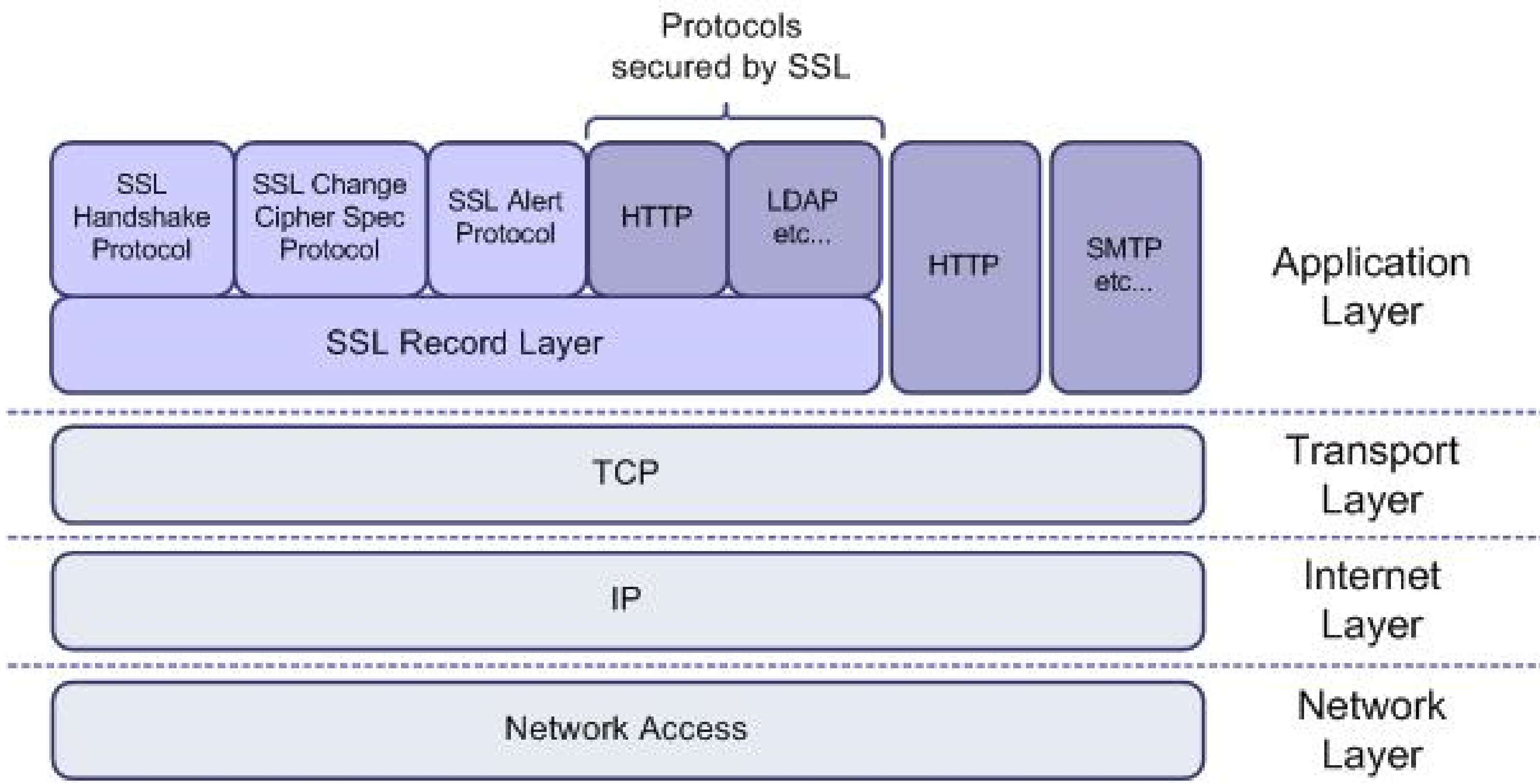
- TLS – Transport Layer Security
- It uses
  - Encryption
  - Cryptographic hash functions or message digests
  - Digital signatures
- To provide a secure transport connection between applications (e.g., a web server and a browser)
  - Client- server authentication
  - Data confidentiality
  - Data origin authentication
  - Data integrity

Two main parts of TLS

- Handshake Protocol
  - Establish shared secret key using public-key cryptography
  - Signed certificates for authentication
- Record Layer
  - Transmit data using negotiated key, encryption function
- We focus only on "Handshake Protocol"

Main Parts of SSL/TLS

SSL/ TLS Parts



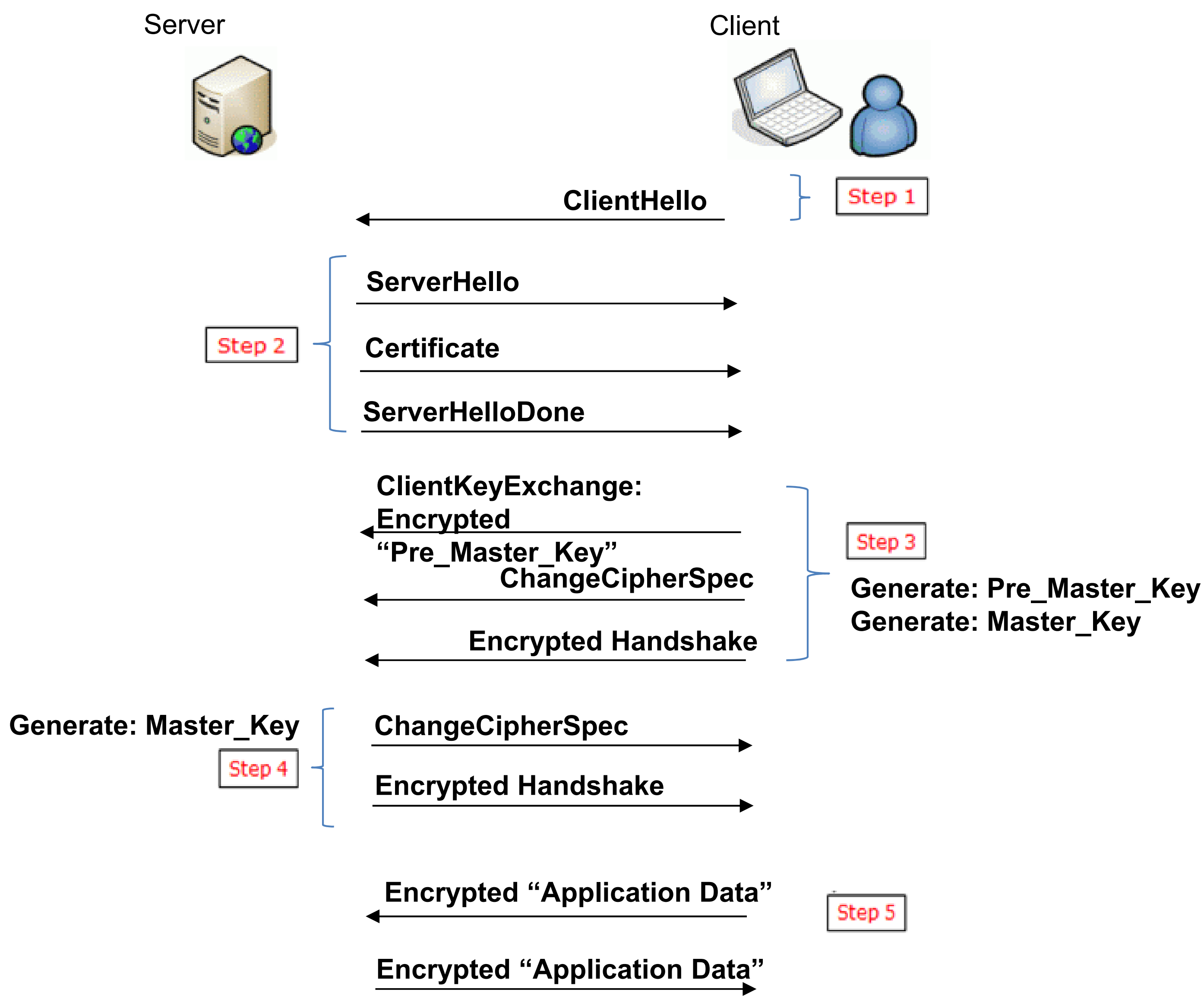
TLS - Handshake Protocol

Handshake Protocol

- To understand how the protocol works, you need to understand how the TLS secure channel is setup before encrypted data flow between two communicating devices.

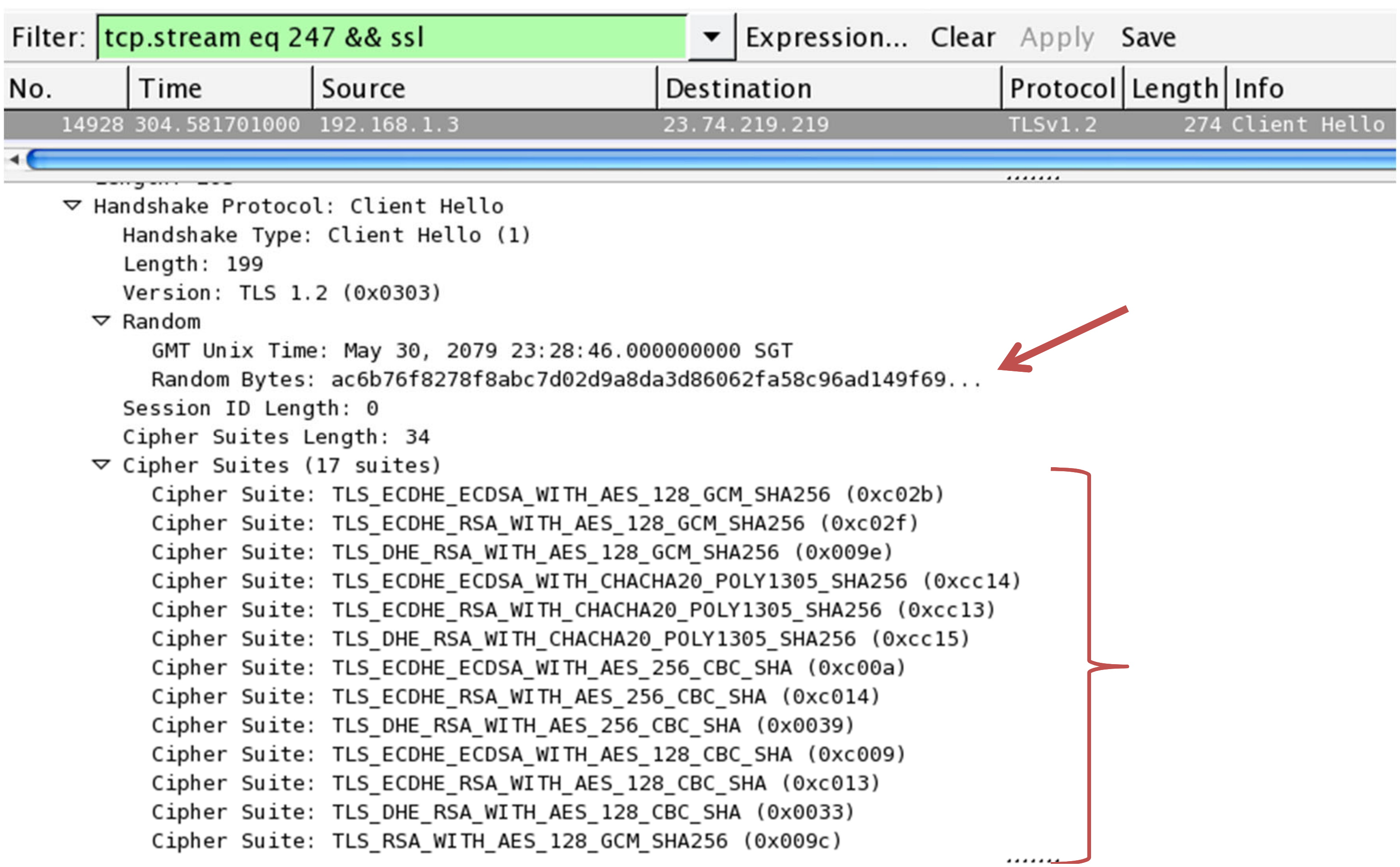
- The channel is setup by the TLS Handshake.
- The handshake protocol describes the rules used to establish common cryptographic parameters as well as authenticating the server and optionally, the client.

Handshake Protocol

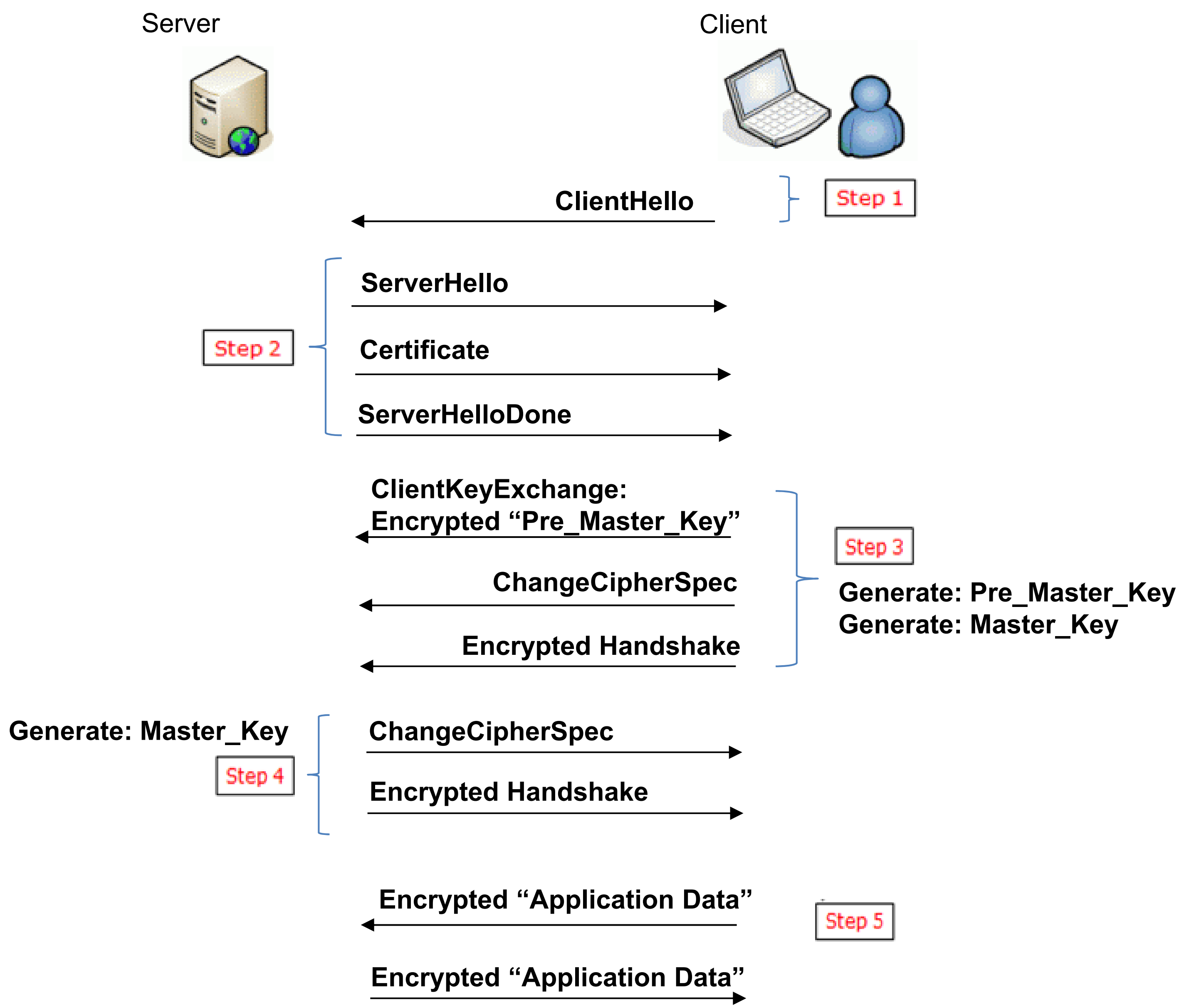


Step 1: Client Hello

- Client makes a connection request to the server by sending the client random number and a set of ciphers suites that it supports.

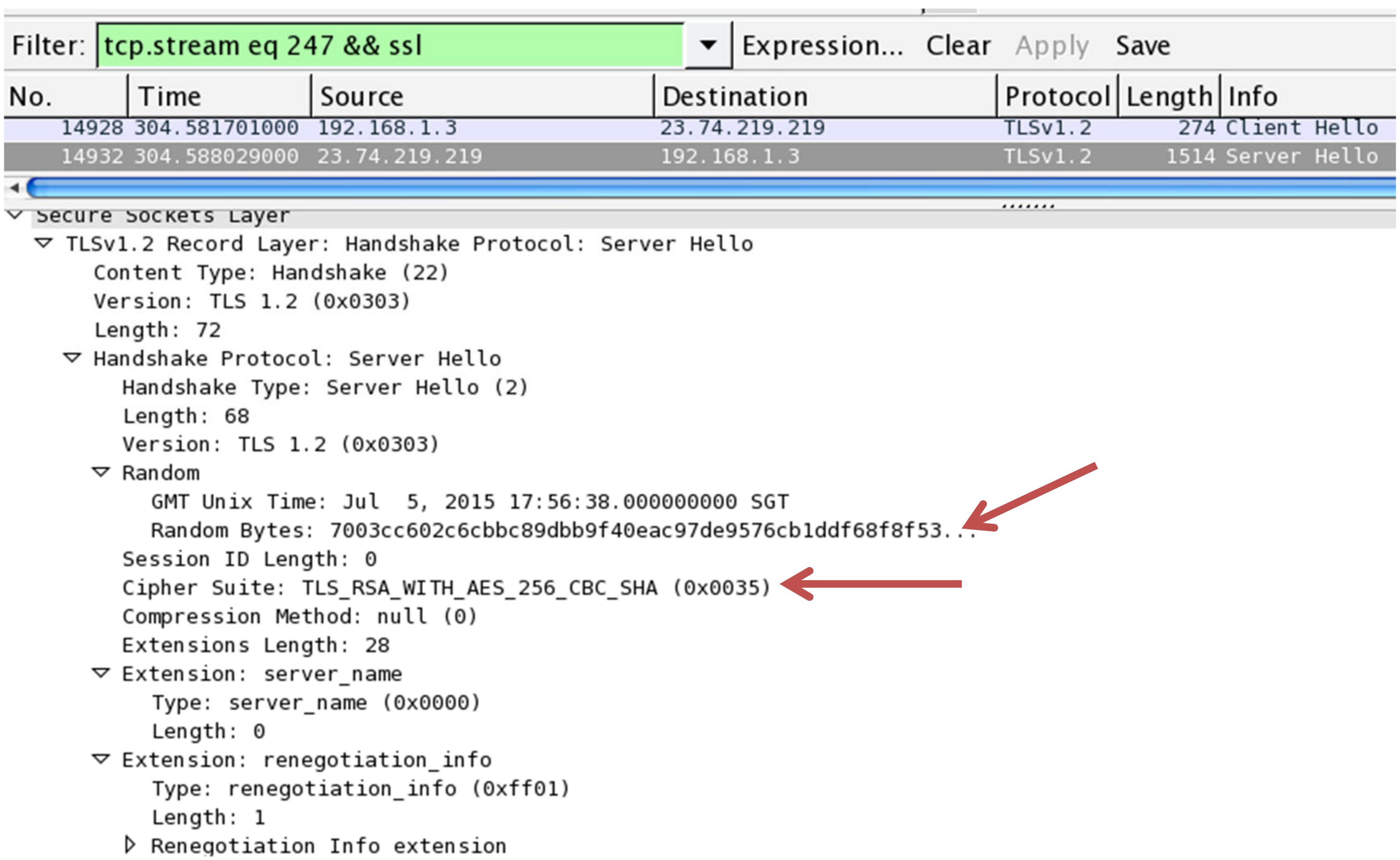


Handshake Protocol – Step 2

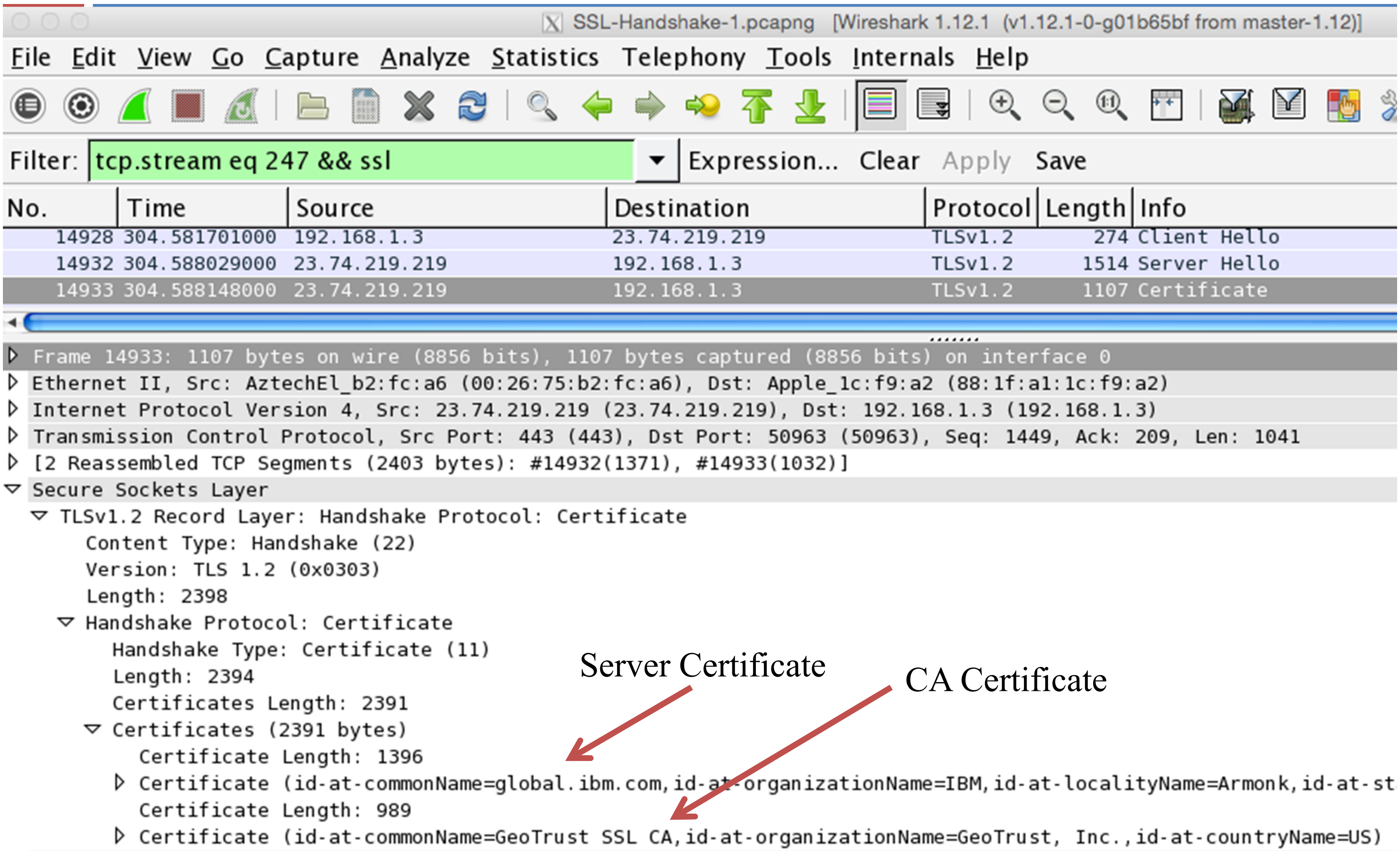


Step 2: Server Hello

- The server responds with a server random number, the chosen cipher (from the client cipher suites it received), and server certificate.

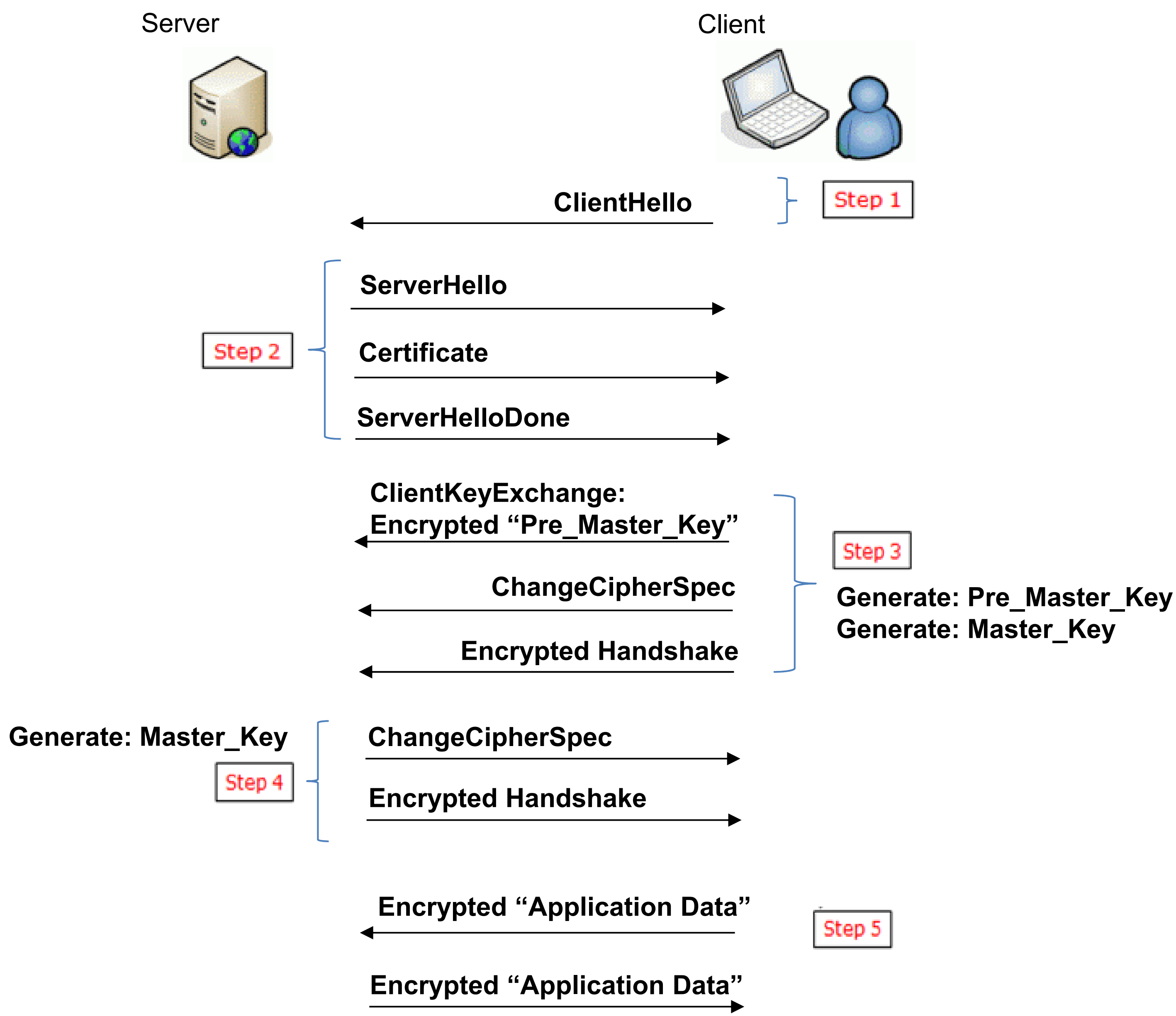


Step 2: Server Certificate



Knowledge Component

Handshake Protocol – Step 3



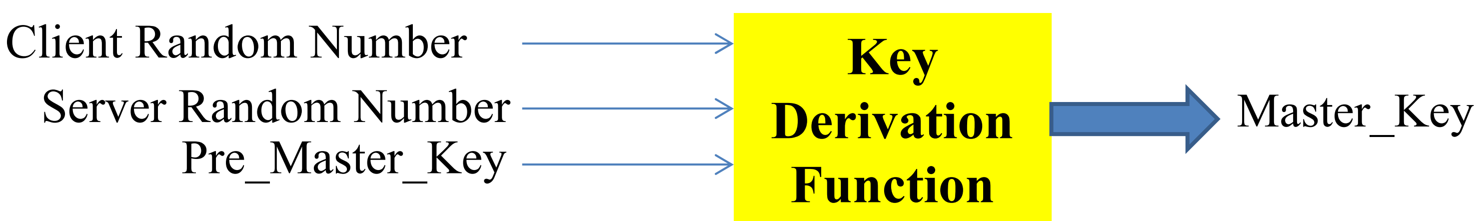
Step 3 - Client Key Exchange and Generation of Master_Key

- The client verifies the identity of the Server using the server's certificate and server's CA certificate.
  - This step is not necessary for repeated session within certain time frame.

- The client then generates another random number called the "Pre_Master_Key".

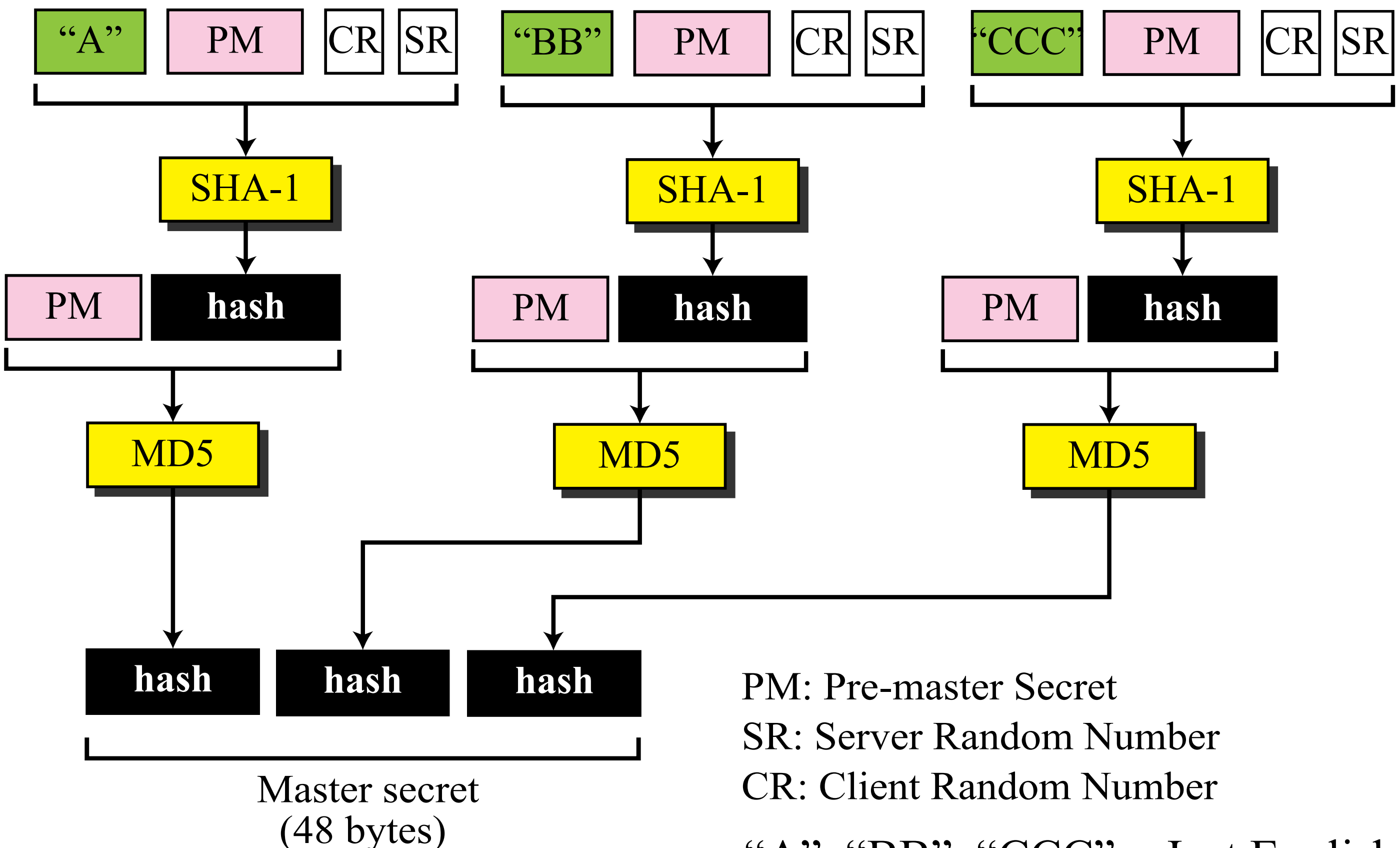
- The client encrypts Pre_Master_Key with the server's public key. Note that the client received the server's public key via the server certificate in Step 2. The client sends the encrypted Pre_Master_Key as the Client Key Exchange.

- The client goes on to compute the "Master_Key" by putting 3 numbers (as shown below) through a Key Derivation Function.




```
master_secret = PRF(pre_master_secret, "master secret", ClientHello.random + ServerHello.random)
```

Step 3 - Client Key Exchange and Generation of Master_Key



PM: Pre-master Secret
SR: Server Random Number
CR: Client Random Number

"A", "BB", "CCC" – Just English Text

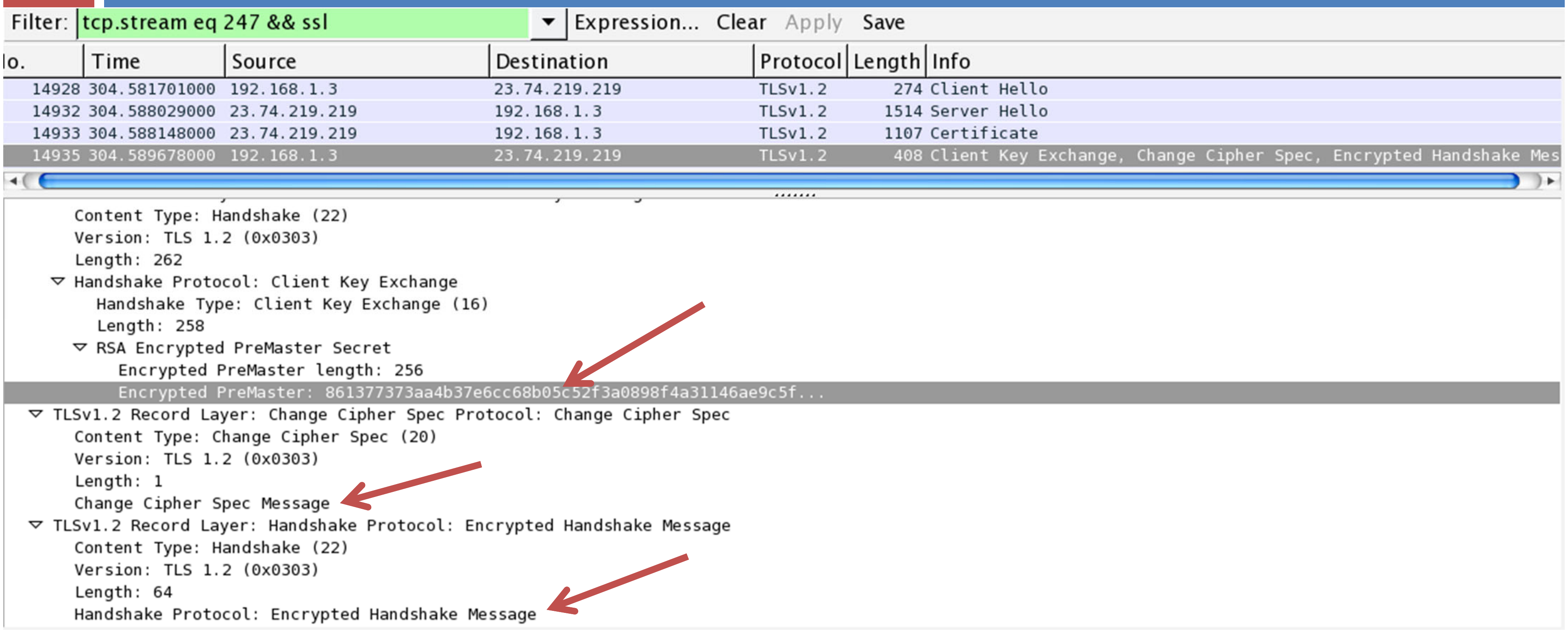
Step 3: Client Change Cipher Spec

- The client informs the server that all messages sent from now on will be encrypted with the generated Master_Key.

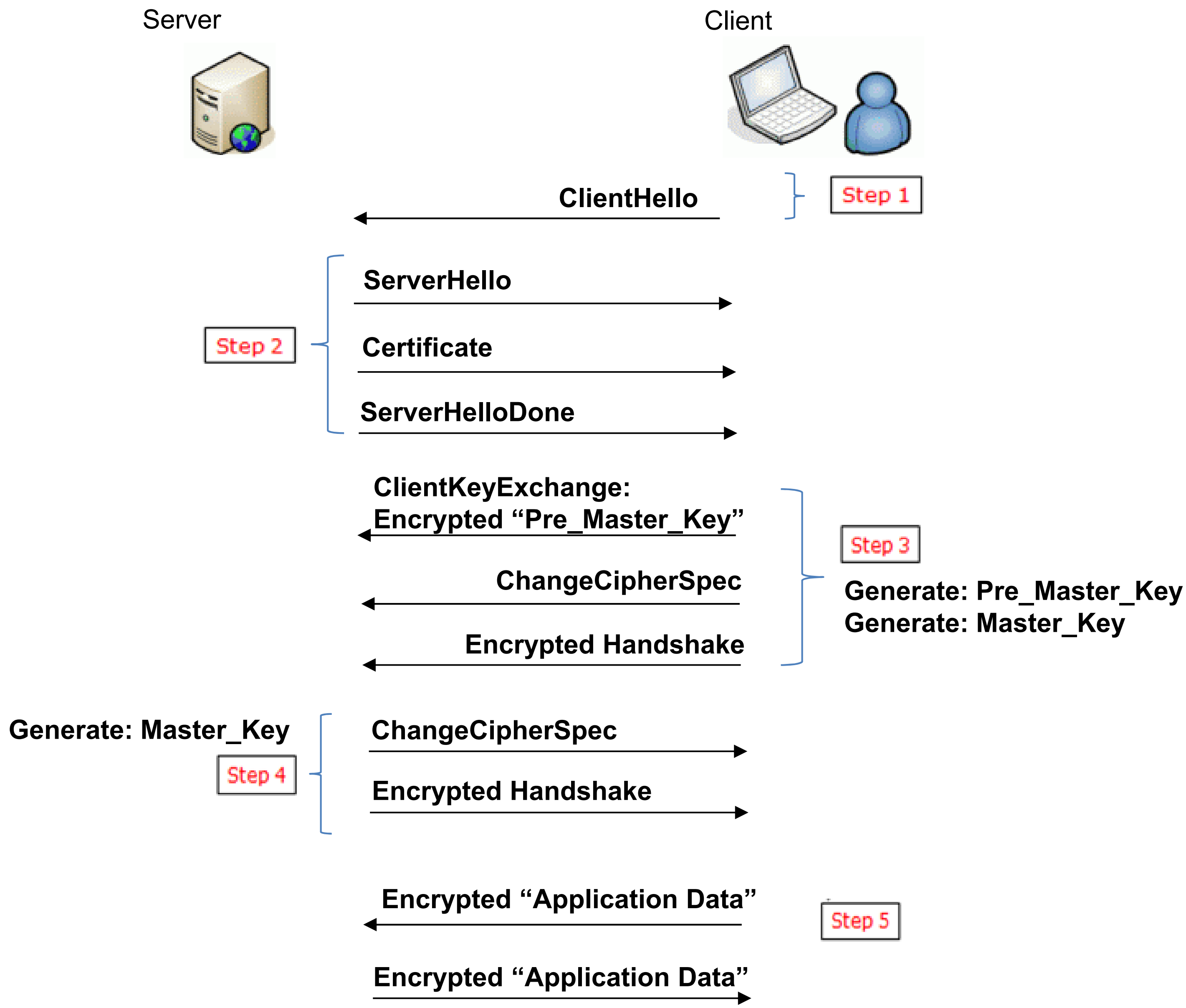
Step 3: Client Encrypted Handshake Message

- The client hashes all the previously exchanged messages and encrypts the message digest (MAC-Message Authentication Code) with the Master Key using the agreed symmetric algorithm.
- The client then sends the server the encrypted MAC code.

Step 3 - Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message



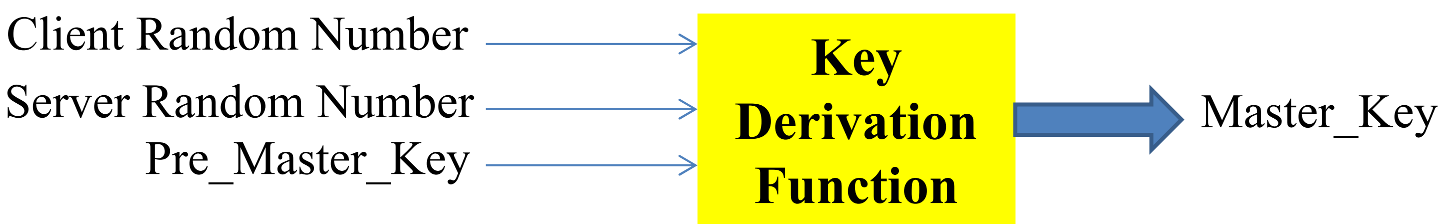
Handshake Protocol – Step 4



Step 4: Server Generates Master_Key

- The server decrypts the encrypted "Pre_Master_key" using its private key.

- It then goes on to compute the "Master_key" by putting the same three numbers through the same Key Derivation function.



Step 4: Verification of Client's Encrypted Handshake Message

- The server decrypts the client's encrypted handshake (MAC) with the Master_key
- It hashes all the previously exchanged messages and compares the hash with the decrypted MAC. If they match, it means all the previously exchanged messages were not tempered with.
- If the hashes do not match, the server terminates the handshake with an alert protocol.

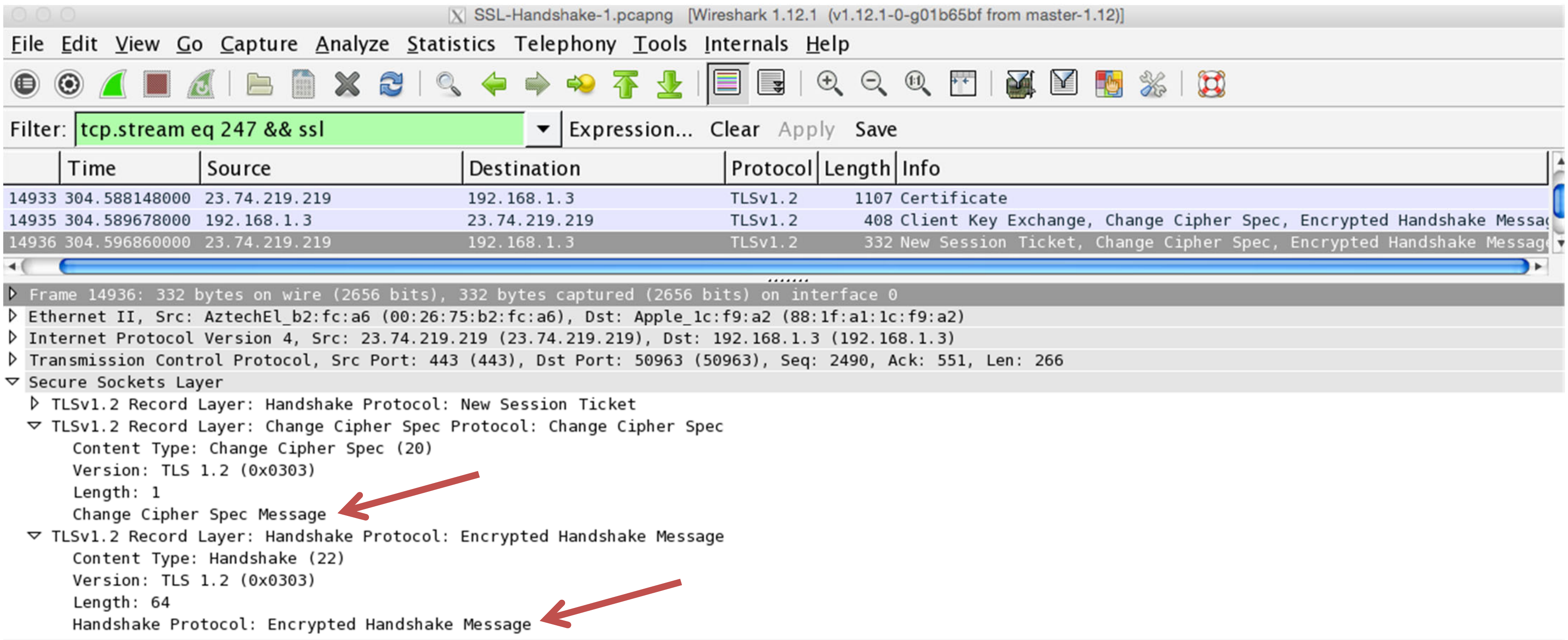
Step 4: Change Cipher Spec

- The server informs the client that all messages sent from now on will be encrypted with the generated Master_Key.

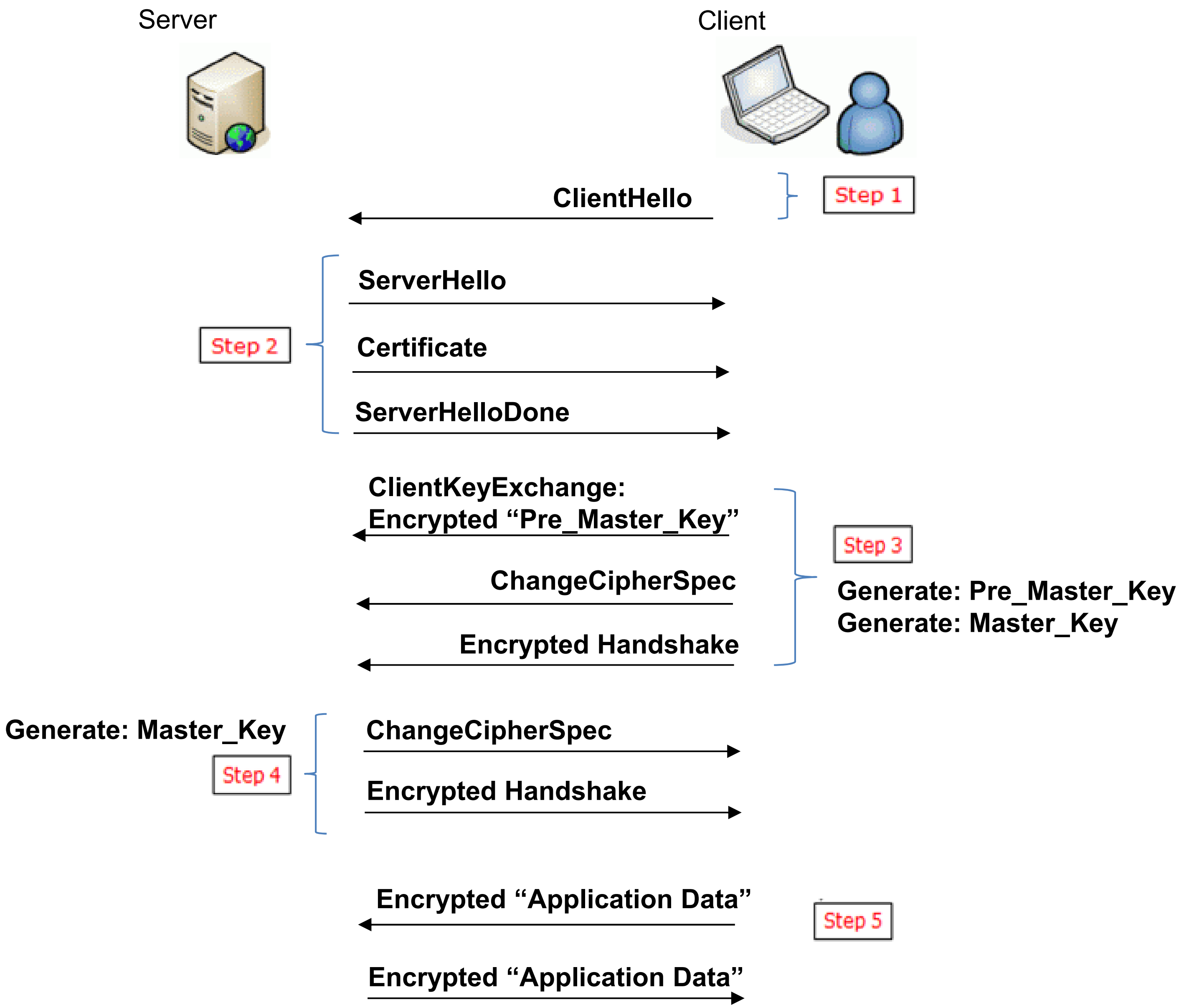
Step 4: Encrypted Handshake Message

- The server hashes all the previously exchanged messages and encrypts the message digest (MAC- Message Authentication Code) with the Master_Key using the agreed symmetric algorithm.
- The server then sends the client the encrypted handshake (MAC) code.

Step 4: Server Change Cipher Spec, Encrypted Handshake Message



Handshake Protocol – Step 5



Step 5: Verification of Server's Encrypted Handshake Message

- The client decrypts the server's encrypted handshake (MAC) with the Master_key

- It hashes all the previously exchanged messages and compares the hash with the decrypted MAC. If they match, it means all the previously exchanged messages were not tempered with.

- If the hashes do not match, the client terminates the handshake with an alert protocol.

Step 5: Encrypted Application Data

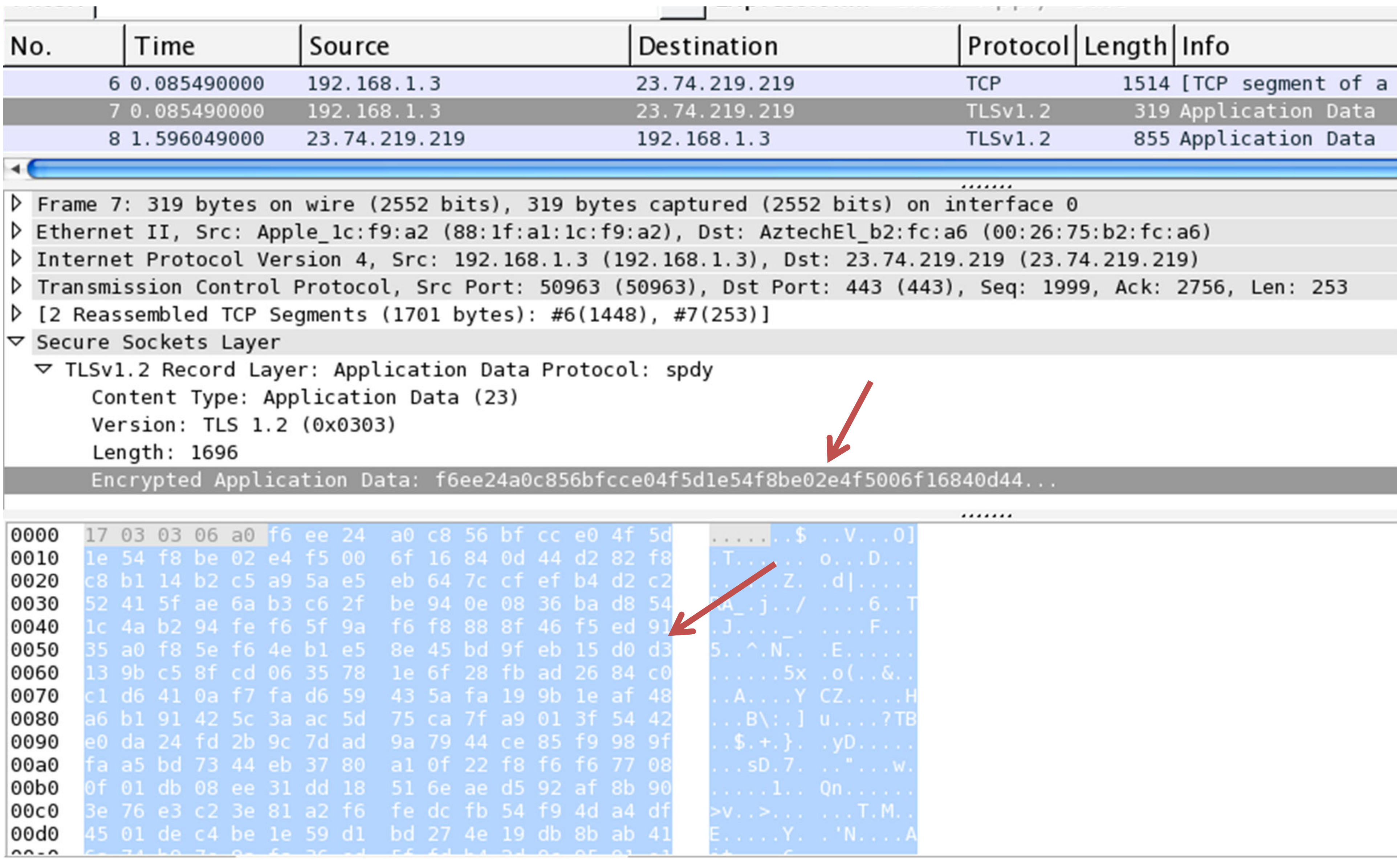
- If the messages were not tampered with, the client will start encrypting the application data using the Master_Key and send to the server. Upon receiving, the data will be decrypted using the same Master_Key.

- Likewise, the server will also encrypt the data using the Master_Key and send to the client. The client will decrypt the encrypted data using the Master_Key.

- The encrypted communication goes on until the client request termination.

Skill Component	Doing, Exploring, Collaborating: LCL Approach

Step 5: Encrypted Application Data



Exercises

Wireshark Captures

- Download and install Wireshark (if you have not already done so)
  - https://www.wireshark.org/download.html
- Choose the appropriate installer (64-bit or 32-bit)
- Follow the instructions provided.
- Please note that WinPcap needs to be installed as well (it should come with it when you are installing Wireshark)

Wireshark Captures

- Analyze the packets provided in the PCAP folder
- List the following
  - 8 bytes of Client Random
  - First 4 Client Cipher Suites
  - Server Name
  - 8 bytes of Server Random
  - Server Chosen Cipher Suite
  - Server certificate
    - Issued by
    - Validity period
    - 8 bytes of public key
  - 8 bytes of Encrypted Pre Master Key
  - 8 bytes of Client Encrypted Handshake Message
  - 8 bytes of Server Encrypted Handshake Message

Summary

Summary

ClientHello

- Client sends server the version of TLS that it would like to use, a list of supported ciphers and a random string that the server will need later.

ServerHello

- The server sends the TLS version, the cipher it has chosen and a random string that the client will need later.

Certificate

- The server sends its certificate as proof of its identity.

ServerHelloDone

- The server tells the client that it is done passing parameters.

ClientKeyExchange

- The client sends pre-master secret which will be used by both sides to generate session keys.

ChangeCipherSpec

- The client informs the server that all messages sent from now on will be encrypted with the generated session key.

Finished

- The client sends a hash of the handshake components, encrypted with the genenerated session key. The server will use this to verify the integrity of the handshake process.

ChangeCipherSpec

- The server informs the client that all messages sent from now on will be encrypted with the generated session key.

Finished

- The server sends a hash of the handshake components, encrypted with the genenerated session key. The client will use this to verify the integrity of the handshake process.

The TLS handshake using the key exchange method

Summary

- You learnt
  - Need for SSL/TLS
  - Purpose of SSL/TLS
  - Understanding TLS – Handshake Protocol
  - Analyzing SSL/TLS packets using Wireshark
- https://tools.ietf.org/html/rfc5280#page-23