

# ELEC-C7420 - Basic principles in networking

## Laboratory exercise SSL/TLS

---

### Introduction

The main target of this experience is for you to get familiar with the implementation of the encryption in the network in the transport layer, to this end we will observe the SSL/TLS (Secure Sockets Layer / Transport Layer Security) in action. In this sense we will use SSL/TLS to encrypt and make secure the TCP connections in one widely used protocol as is HTTP by implementing HTTPS: HTTP over SSL.

The main motivation to implement HTTPS over the webpages we access daily is to provide a secure means to access the information by employing authentication techniques of the accessed website, protecting the integrity and the privacy of the data we share with the web browsers, among other benefits. The implementation of HTTPS or SSL/TLS to secure the TCP connections during the HTTP connection, helps us to prevent:

- Man-in-the-middle attacks.
- Eavesdropping.
- Tampering the communication

The bi-directional encryption of communications between the client and the servers enables a secure way for us to ensure that we are on a secure communication channel with the website we are accessing without any means of interference from attackers or for being attacked by an impostor of the original website. This is why, historically the HTTPS implementation was primarily intended to be used for payment transactions over the World Wide Web, securing e-mail messages or to exchange sensitive information between corporations. However, in the recent years the HTTPS protocol has been extended to practically all the websites in the network that were not originally intended to use encryption like entertainment webpages, protecting the page authenticity on all type of websites, securing user accounts and keeping communications safe and private.

## Step 1: open a trace

- 1) Download the trace from MyCourses and open it in your wireshark, in there you will be able to see the SSL messages as shown in the following figure:

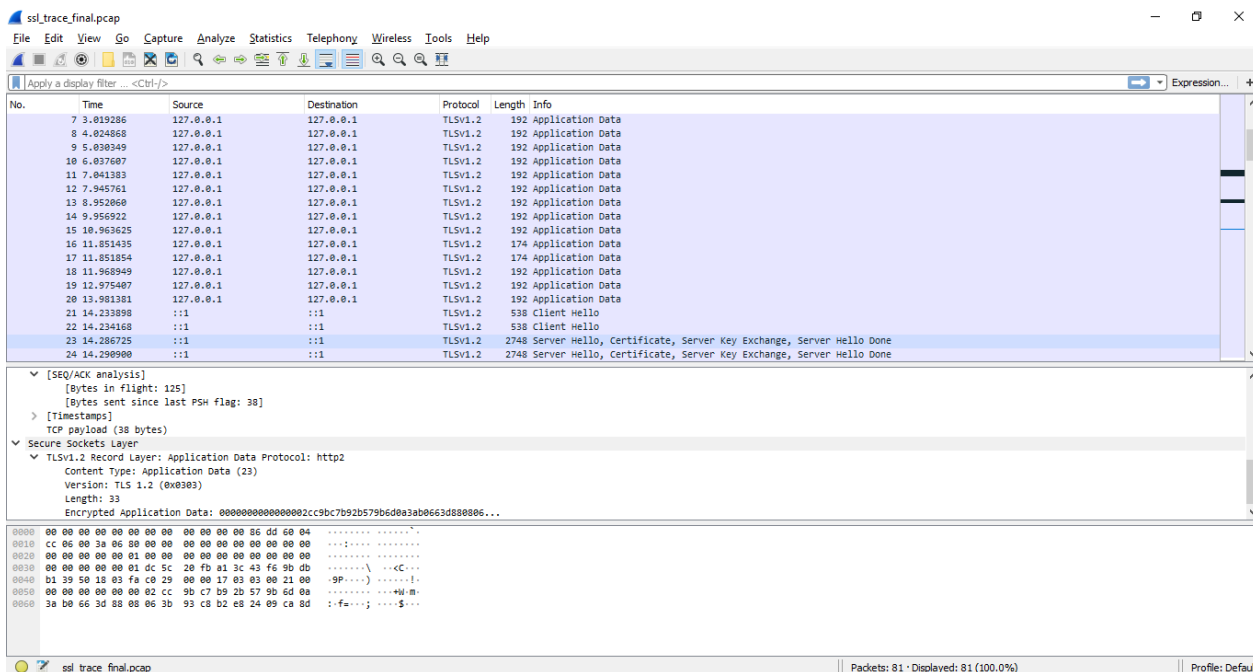


Figure. Trace of HTTPS traffic

## Step 2: inspect the trace

- 2) In this case the trace is already filtered to show only the SSL packets, however, if you capture your own trace you may apply the “ssl” filter protocol in wireshark, this filter will help you to simplify the display by showing only SSL and TLS messages. It will exclude other TCP segments that are part of the trace, such as Acks and connection open/close.
- 3) Select a TLS message somewhere in the middle of your trace for which the Info reads “Application Data” & expand its Secure Sockets Layer block (by using the “+” expander or icon). For instance, we select the packet #17 and we expand this packet as shown in the figure below:

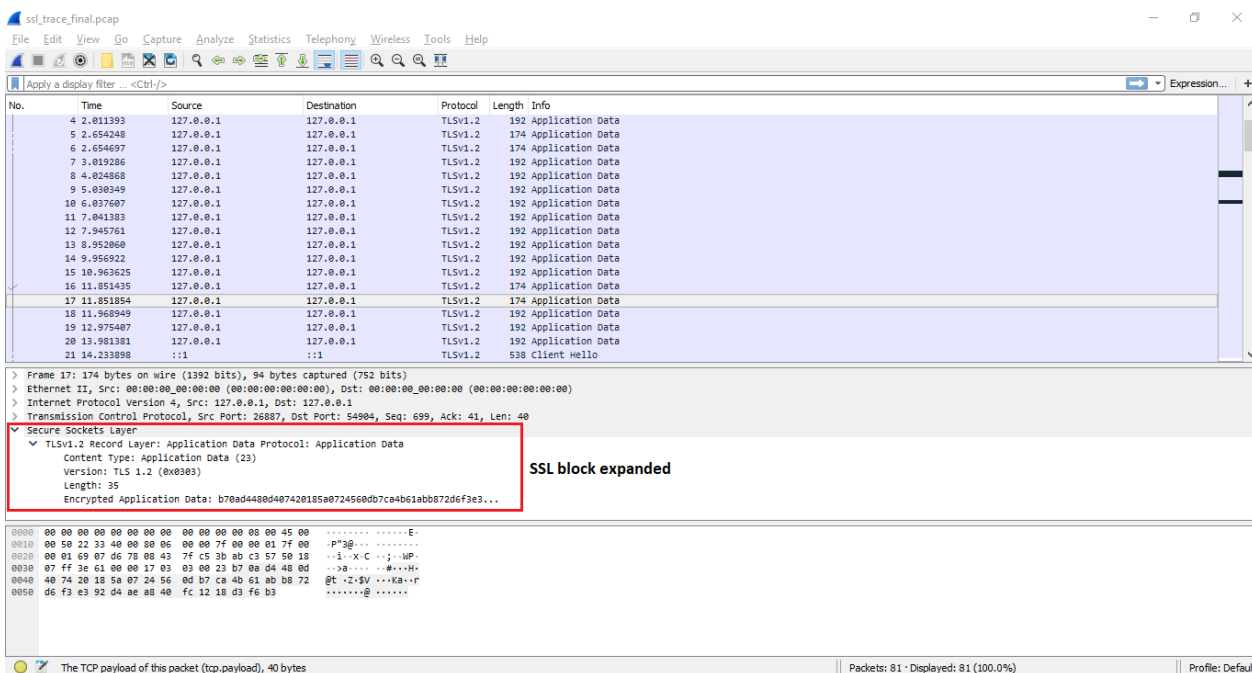


Figure. Example of SSL/TLS packet

If we observe the “Application Data” packet we can see that it is a generic TLS message carrying the encrypted information from the application it is encrypting for, in our case the contents of the webpage in HTTP.

As we know from the lectures, the lower layer protocol stack are TCP and IP because SSL/TLS runs on top of TCP/IP. The SSL layer contains a “TLS Record Layer”. This is the foundational sublayer for TLS. All messages contain records. Expand this block to see its content details:

- Content Type field: this indicates what is in the content of the corresponding record.
- Version identifier: It will be a constant value for the SSL connection, in our case we are working in the TLSv1.2.
- Length field: giving the length of the record.
- Application Data records are sent after SSL has secured the connection, so the contents will show up as encrypted data.

### Step 3: SSL handshake

An important part of SSL is the initial handshake that establishes a secure connection. The handshake proceeds in several phases. There are slight differences for different versions of TLS and depending on the encryption scheme that is in use. The usual outline for a brand-new connection is:

- a. Client (the browser) and Server (the web server) both send their Hellos.
- b. Server sends its certificate to Client to authenticate (and optionally asks for Client Certificate).
- c. Client sends keying information and signals a switch to encrypted data.
- d. Server signals a switch to encrypted data.
- e. Both Client and Server send encrypted data.
- f. An Alert is used to tell the other party that the connection is closing.

Note that there is also a mechanism to resume sessions for repeat connections between the same client and server to skip most of steps b and c.

- 4) The client sends a “Hello” packet to the server with a group of information that the server will use to establish the session. We can see several important fields here worth mentioning. First, the time (GMT seconds since midnight Jan 1, 1970) and random bytes (size 28) are included. This will be used later in the protocol to generate our symmetric encryption key. The client can send an optional session ID to quickly resume a previous TLS connection and skip portions of the TLS handshake. Arguably the most important part of the Client Hello message is the list of cipher suites, which dictate the key exchange algorithm, bulk encryption algorithm (with key length), MAC, and a pseudo-random function. The list should be ordered by client preference. The collection of these choices is a “cipher suite”, and the server is responsible for choosing a secure one it supports or return an error if it doesn’t support any. The final field specified in the specification is for compression methods. However, secure clients will advertise that they do not support compression (by passing “null” as the only algorithm) to avoid the CRIME

attack. Finally, the Client Hello can have a number of different extensions. A common one is server name, which specifies the host-name the connection is meant for, so web servers hosting multiple sites can present the correct certificate[1].

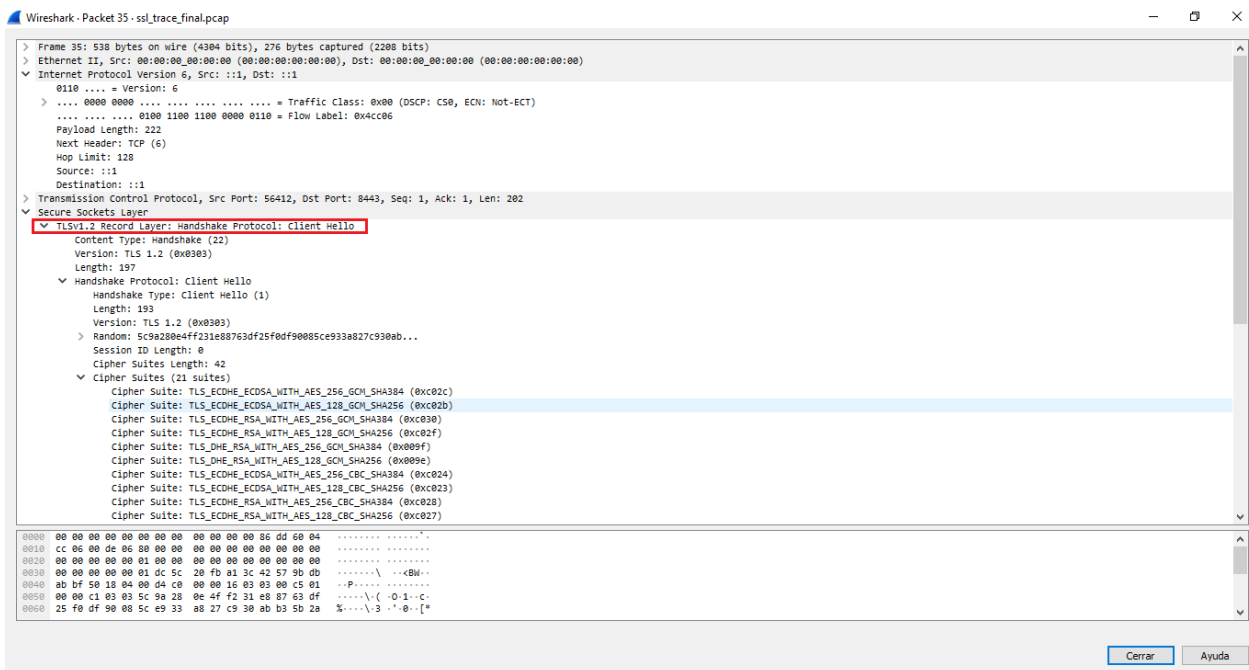


Figure. Client Hello packet

- 5) The server will send this message in response to a ClientHello message when it was able to find an acceptable set of algorithms. If it cannot find such a match, it will respond with a handshake failure alert. An important field in the ServerHello packet is the "session\_id" field, this is the identity of the session corresponding to this connection. If the ClientHello.session\_id was non-empty, the server will look in its session cache for a match. If a match is found and the server is willing to establish the new connection using the specified session state, the server will respond with the same value as was supplied by the client. This indicates a resumed session and dictates that the parties must proceed directly to the Finished messages. Otherwise, this field will contain a different value identifying the new session. The server may return an empty session\_id to indicate that the session will not be cached and therefore cannot be resumed. If a session is

resumed, it must be resumed using the same cipher suite it was originally negotiated with. Note that there is no requirement that the server resume any session even if it had formerly provided a session\_id. Clients MUST be prepared to do a full negotiation (including negotiating new cipher suites) during any handshake[2].

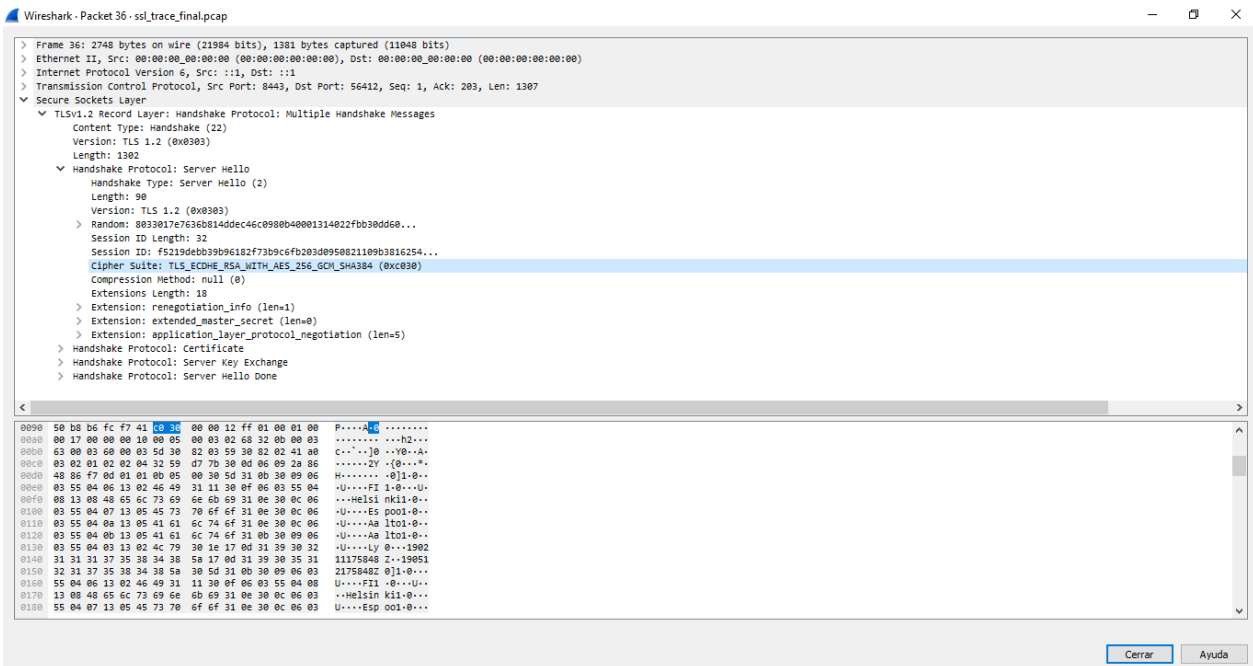


Figure. ServerHello message

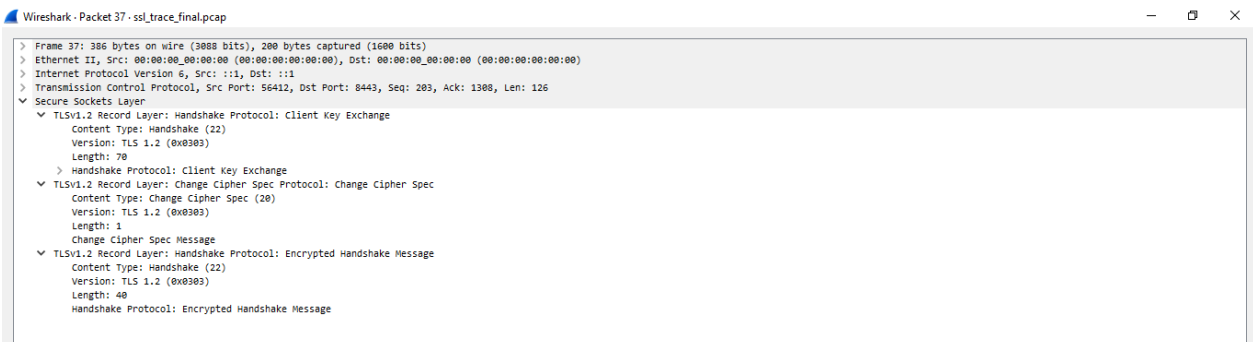


Figure. Client Key Exchange message

### **SSL/TLS trivia:**

The SSL protocol ensures:

- a) Confidentiality and data integrity
- b) Authenticity
- c) Man in the middle attacks
- d) Confidentiality, integrity and authenticity

The difference between SSL and TLS is:

- a) The physical level in which each protocol is used
- b) The volume of data they can protect
- c) TLS is better than SSL in protecting against several attacks
- d) They are exactly the same

The SSL Handshake Protocol enables:

- a) The definition of how to encapsulate the data transmitted between client and server
- b) The configuration of keys for a one time pad encryption
- c) The negotiation of the security parameters required to establish a secure communication between client and server
- d) Web clients to use vulnerable encryption algorithms

The SSL/TLS Record Protocol enables:

- a) The exchange of session keys for encrypting a communication
- b) The definition of how to encapsulate the data transmitted between client and server
- c) The storage of session keys used to ensure the integrity of the sent data
- d) The SSL handshake protocol to use 1024-bit RSA keys

SSL/TLS protects against MITM attacks on online banking access:

- a) Provided that a Web client has the guarantee that the digital certificate received from the Web server is valid
- b) Regardless of the nature of the certificate that identifies the server
- c) SSL doesn't protect against MITM attacks
- d) Whenever a user enters the URL of the bank in a web browser to confirm that it's the correct URL

## Exercise

Choose one of the following options to encrypt in the transport layer:

- 1) Remote connection using SSH and Telnet.
- 2) Transfer files via FTP and SFTP.
- 3) Web server to create client/server requests.
- 4) Encrypt with SSL the python sockets applications implemented in the first part of the course.
- 5) Create a sketch in Arduino to connect a WiFi network and make an HTTPS request.
- 6) Connect to an existing HTTP/HTTPS web server, in addition write an 1-2 pages report about TLSv1.3.

For any of the implementations you choose, you should show the packets without encryption and after encryption to see the difference and the protocol handshake. You can refer to the following material for the implementations:

- <https://www.arduino.cc/en/Tutorial/WiFiNINAWiFiSSLClient>
- [https://github.com/xliu59/SSL-TLS\\_SOCKET](https://github.com/xliu59/SSL-TLS_SOCKET)
- <https://tomcat.apache.org/tomcat-9.0-doc/ssl-howto.html>
- <https://websiteforstudents.com/create-ssl-tls-self-signed-certificates-on-ubuntu-16-04-18-04-18-10/>

**The submission is in form of a report explaining your results, wireshark captures, code and relevant information by 10.4.2019**

## References:

- [1] <https://kevincurran.org/com320/labs/wireshark/lab-ssl.pdf> (Accessed on 26.3.2019)
- [2] The Transport Layer Security (TLS) Protocol Version 1.2 RFC specification. T. Dierks, E. Rescorla. Aug. 2008. <https://tools.ietf.org/html/rfc5246#page-42>
- [3] <http://www.cs.kent.edu/~mallouzi/ccn%20Spring%202014/lab-ssl.pdf> (Accessed on 26.3.2019)



[4] James F. Kurose and Keith W. Ross. Computer Networking – A Top-Down Approach. Pearson.

[5] <http://www.criptored.upm.es/intypedia/docs/en/video9/ExercisesIntypedia009.pdf>. Author: PhD Alfonso Muñoz Muñoz (Accessed on 26.3.2019)