**TLS 1.2 Handshake**

The TLS handshake is the process in which the client and server negotiate all the parameters for the TLS session. It always follows a predefined scheme with the exception of minor deviations depending on the concrete parameters chosen for the connection.

**Cipher Suites**

In TLS, cipher suites define the cryptographic algorithms used for a connection. That includes the following information:

* The key exchange algorithm
* The method used for authentification
* The encryption algorithm and mode, which provide confidentiality
* The MAC algorithm, which provides integrity protection

As an example, let's have a look at the following TLS 1.2 cipher suite: TLS\_DH\_RSA\_WITH\_AES\_128\_CBC\_SHA256

From the name, we can identify the algorithms used by this cipher suite:

* The key exchange algorithm is Diffie-Hellman (DH)
* Server authentification is performed via RSA
* The encryption is AES-128 in CBC mode
* The MAC algorithm is a SHA256 HMAC

All TLS 1.2 cipher suites follow this naming scheme. The encryption algorithm is always a symmetric algorithm. The symmetric key for this algorithm is exchanged using the key exchange algorithm, which is always an asymmetric algorithm. Thus, TLS encrypts data using a symmetric key due to significant performance advantages compared to asymmetric encryption. The cipher suite used by a specific connection is negotiated in the handshake.

Cipher Suites using the TLS\_DHE and TLS\_ECDHE key exchange algorithms provide Perfect Forward Secrecy (PFS), meaning an attacker is unable to decrypt past messages even after obtaining a future session key. In particular, this protects past communication from leaks potentially occurring in the future. Therefore, PFS cipher suites are preferable if they are supported by the client.

**Handshake Overview**

During the handshake, the client and server establish a connection and negotiate all the required parameters to establish a secure channel for application data. The handshake follows a well-defined schema and varies slightly depending on the cipher suite that is negotiated.

The handshake begins with the client sending the ClientHello message. This message informs the server that the client wants to establish a secure connection. It contains the latest TLS version supported by the client, as well as a list of cipher suites the client supports among other information.

The server responds with a ServerHello message. The server chooses a TLS version that is equal to or lower than the version provided by the client. Additionally, the server chooses one of the cipher suites provided in the ClientHello. This information is included in the ServerHello message.

After agreeing on the TLS version and cryptographic parameters, the server provides a certificate in the Certificate message, thereby proving the server's identity to the client.

If a PFS cipher suite was agreed upon, the server proceeds to share fresh key material in the ServerKeyExchange message. It contains a key share as well as a signature. This is followed by the ServerHelloDone message.

The client responds with the ClientKeyExchange message, containing the client's key share. After this, the key exchange is concluded and both parties share a secret that is used to derive a shared symmetric key. Both parties transmit a ChangeCipherSpec message to indicate that all following messages are encrypted using the computed symmetric key. From here on, all data is encrypted and MAC-protected.



**Analyzing a TLS 1.2 Handshake in Wireshark**

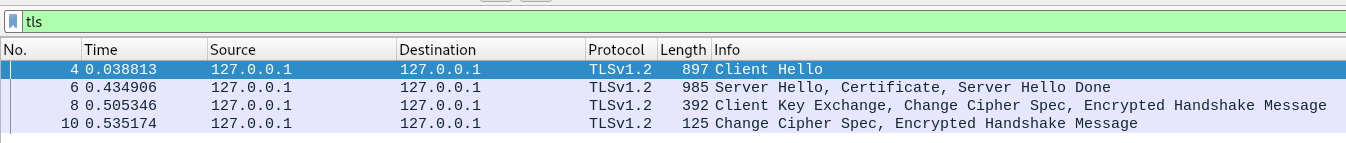
Let's have a look at a TLS 1.2 handshake in [Wireshark](https://www.wireshark.org/), which is a network protocol analyzer. It can typically be installed from the package manager:

yovecio@htb[/htb]$ sudo apt install wireshark

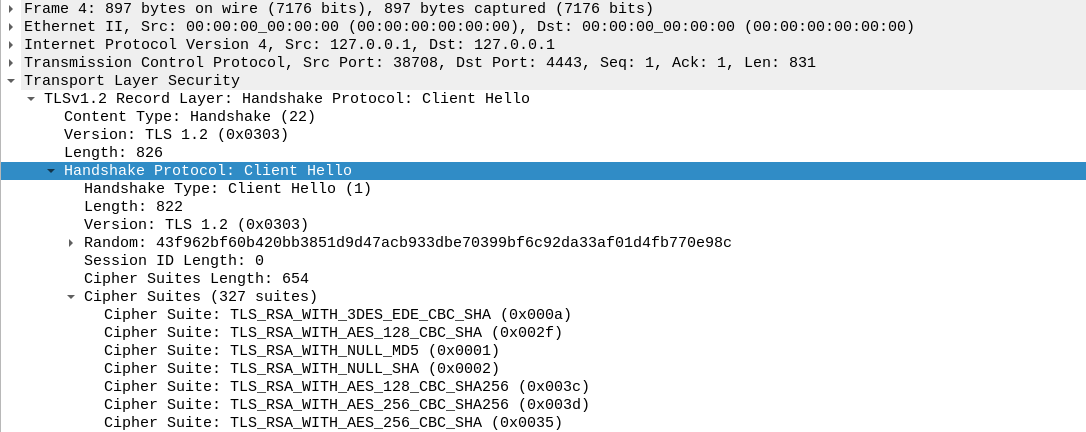
We can then start Wireshark with a path to a packet capture (or pcap) file to analyze the packets:

yovecio@htb[/htb]$ Wireshark /path/to/file.pcap

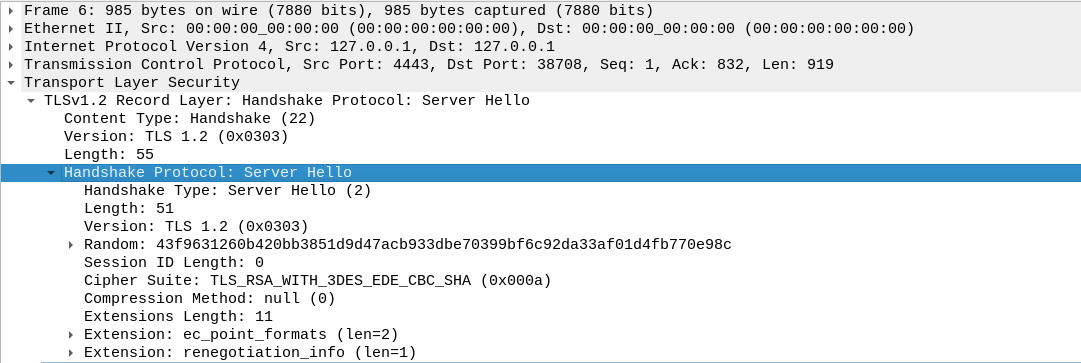
After entering the protocol name tls in the filter bar, we can see the TLS handshake in Wireshark:



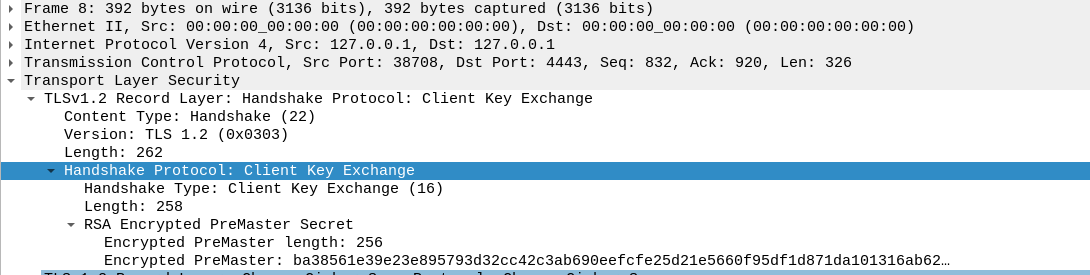
When expanding the ClientHello message, we can inspect the TLS version and supported cipher suites sent by the client:



Doing the same in the ServerHello message reveals the TLS version and cipher suite chosen by the server for this TLS connection:



Lastly, we can inspect the key information sent by the client in the ClientKeyExchange message. In this case, a TLS\_RSA cipher suite was chosen, thus the key information sent by the client is the shared key encrypted with the server's public key.



**Note:** There is no Server Key Exchange message since the cipher suite does not provide PFS.