Project Report

Performance Comparison of TLS 1.2 and TLS 1.3

Abhinav Anand ME CS (2020-2022),
2020H1030130P
Suyash Musalgaonkar ME CSE (2020-2022),
2020H1030127P

June, 2021

1 Abstract

The Transport Layer Security (TLS) protocol is the de facto standard for securing Internet connections. The protocol, which was originally created by Netscape Communications, was adopted by the Internet Engineering Task Force *(IETF)* in the mid-1990s and now serves millions, if not billions, of users on a daily basis.The protocol's widespread use has made it an appealing target for security researchers, particularly in recent years. There has been demand to improve the TLS 1.2 performance which caused the IETF to create TLS 1.3, a new version of the protocol with improved performance and security as compared to TLS 1.2.
In this study we seek to understand the performance differences of TLS 1.2 and TLS 1.3 and demonstrate the improvement of TLS 1.2 over TLS 1.3

2 Introduction

TLS stands for *Transport Layer Security* and is the successor to SSL *(Secure Socket Layer)*.
TLS allows web browsers and servers to communicate in a secure manner since symmetric cryptography is used to encrypt the data transmitted, the connection is safe. The keys are created individually for each connection and are based on a common secret agreed upon at the start of the session, also known as a TLS handshake. TLS is used to encrypt data in several IP-based protocols, including HTTPS, SMTP, POP3, and FTP.

TLS runs over the *Transmission Control Protocol* (TCP), a reliable network protocol that guarantees in-order delivery of network messages.

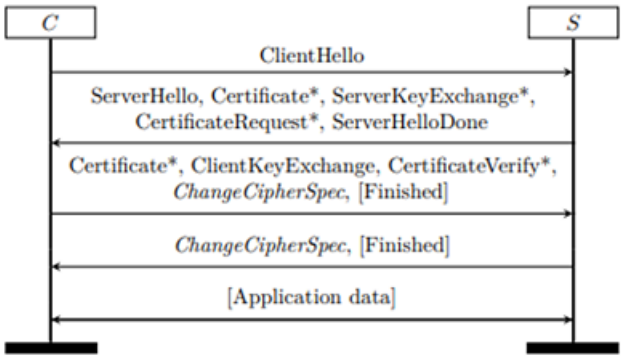
2.1 TLS 1.2 Handshake

1. The client initiates the connection/handshake by sending a "client hello" message to the server. This message contains cryptographic information such as the protocols and cipher suite specifications that are supported.

2. The server replies with a "ServerHello" message in response to the client's "ClientHello" message. This message contains the Cipher Suite that the server selected from among the options presented by the client. Along with the session ID and another random value, the server also sends its certificate.

3. The client now verifies the certificate that the server has sent. After the verification is complete, it sends a random byte string, commonly known as the "pre-master secret," which is encrypted with the server's certificate's public key.

4. The client and server both produce a master key and session keys once the server receives the pre-master secret (ephemeral keys).The data will be symmetrically encrypted using these session keys.



5. The client now sends a "ChangeCipherSpec" message to the server, informing it that it will use session keys to switch to symmetric encryption. It also sends the message *"Client Finished"* along with it.

6. In response to the client's "Change Cipher Spec" communication, the server follows suit and switches to symmetric encryption. The handshake is completed when the server sends the message *"ServerDone"*.

2.2 TLS 1.3 Handshake

The IETF has been working on TLS 1.3, the next version of the protocol, in response to attacks against TLS 1.2 and demand to increase the system's efficiency.
TLS 1.3's key design goals are as follows.

- Encrypting as much of the handshake as possible.
- Reducing Handshake Latency : Introducing a One Round-Trip Time (1 RTT) exchange for full handshakes and a Zero Round-Trip Time (0 RTT) mechanism for repeated handshakes.

We now discuss how TLS 1.3 meets these two requirements

1. **Handshake Encryption:** The goal of handshake encryption is to limit the amount of data that may be seen by both passive and active attackers .TLS 1.3, in contrast to TLS 1.2, which only provides communicating entities with session keys to safeguard application data, allows for the creation of extra session keys for handshake encryption. After the handshake keys have been agreed via a Diffie-Hellman exchange, handshake encryption commences.

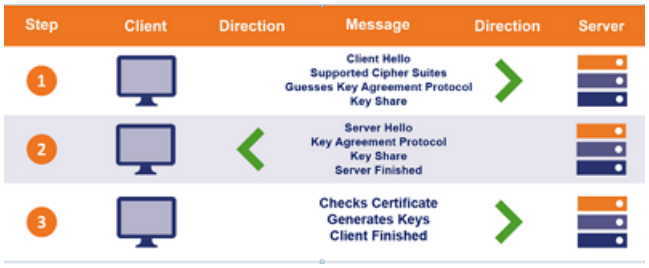
2. **Latency in Handshakes:** Prior to interacting entities being allowed to transfer application data, the TLS 1.2 handshake required a two-round-trip-time (2 RTT) exchange. In TLS 1.3, the handshake has been rewritten to just require 1 RTT if no parameter mismatches occur. TLS 1.3 also includes a 0 RTT option, which allows the client to deliver application data in the first flight of messages, resulting in a significant efficiency gain over TLS 1.2.

2.2.1 Handshake

1. The TLS 1.3 handshake starts with the same "Client Hello" message as the TLS 1.2 handshake, with one major difference. The client transmits the list of supported cipher suites and makes an educated guess as to which key agreement protocol the server will use. In addition, the client submits their key share for that specific key agreement protocol.

2. The server responds to the "Client Hello" message with the key agreement protocol it has chosen. The server's key share, certificate, and the "Server Finished" message are all included in the "Server Hello" message.The "Server Finished" message, which was sent in the 6th step in TLS 1.2 handshake, is sent in the second step. Thereby, saving four steps and one round trip along the way.

3. The client now verifies the server's certificate, produces keys using the server's key share, and delivers the "Client Finished" message. The data is encrypted from this point on.



3 Related Work

We have created two Virtual Machines and configured Apache2 server in both of them; one server uses *TLS v1.2* version and the other one uses *TLS v1.3*.

3.1 Setting Up Servers

3.1.1 Prerequisites

- **TLSv1.2** - The minimum version of the Apache Web Server must be - Apache 2.2.24 and OpenSSL version must be OpenSSL 1.0.1
- **TLSv1.3** - The minimum version of the Apache Web Server must be - Apache 2.4.37 and OpenSSL version must be OpenSSL 1.1.1

3.2 Configuring TLS 1.2 and TLS 1.3

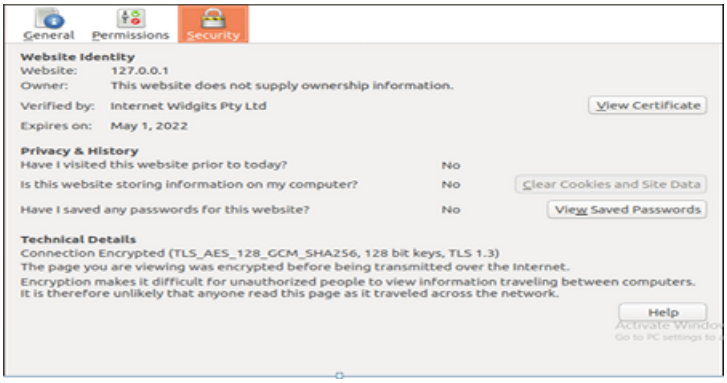
1. Open the *ssl.conf* file using the command `sudonano /etc/apache2/mods-available/ssl.conf`
2. Type the command `SSLProtocol -all +TLSv1.2` for TLSv1.2 and `SSLProtocol -all +TLSv1.3` for TLS1.3
3. Now open the 000-default.conf file using `sudonano /etc/apache2/sites-available/000-default.conf`

4. To generate self-signed *SSL* certificate using *OpenSSL*, type `Opensslreq -x509 -newkey rsa:4096 -keyoutkey.pem -out cert.pem -days 365`

5. Write the following two commands *000-default.conf* file to configure certificate and key.

   - `SSLCertificateFile /etc/apache2/sites-available/cert.pem`
   - `SSLCertificateKeyFile /etc/apache2/sites-available/key.pem`

6. Restart the Apache Server

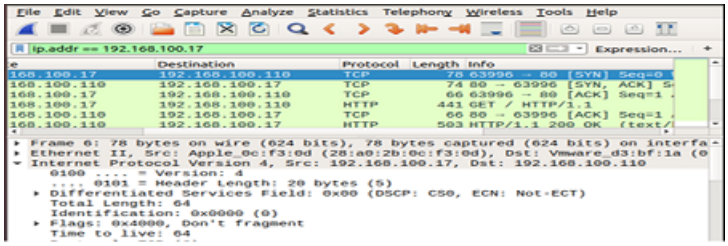


7. Check the IP Address of your server using command `ifconfig`

3.3 Capturing Handshake using Wireshark

First thing you'll need to do is to select an appropriate NIC interface to capture packets. Second capture, locate, and examine packets.

- Capture a web session to the created Server.
- Locate appropriate packets for a web session.
- Examine information within packets.

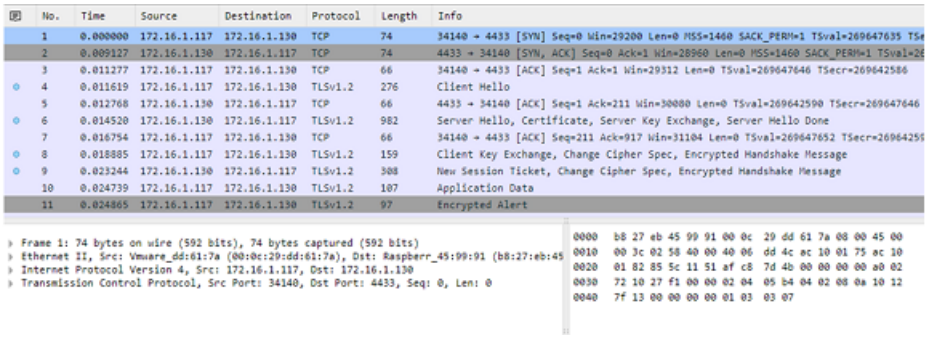


4 Handshake Comparison

4.1 TLS v1.2

The client sends a `Client Hello` with the cipher suites and extensions it supports, which is the initial stage in the handshake procedure.

The Server replies with the suite it wants to use, as well as its certificate and keys. Even when subsequent handshakes are encrypted, some further interactions take place before the secure session is created.

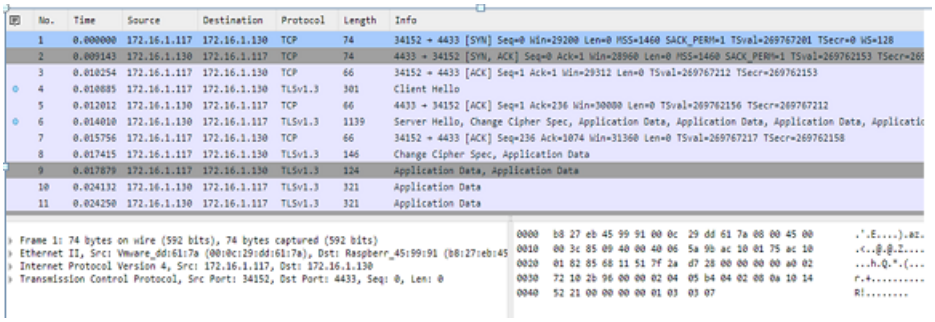


4.2 TLS v1.3

The TLS 1.3 sequences reduce the amount of chatter in this procedure by

1. Limiting the number of cypher suites that can be used in a protocol. Because previous suites have become obsolete, this also has the added benefit of enhancing security

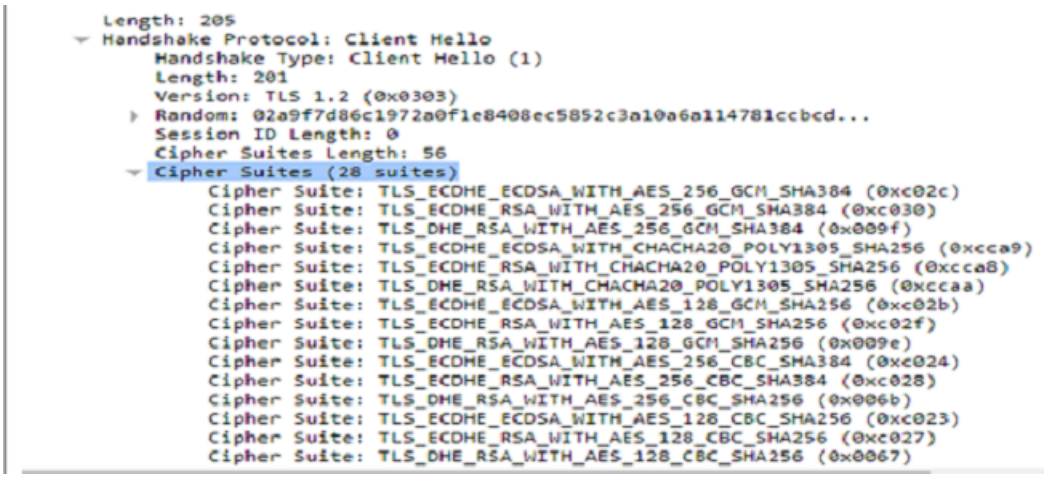
2. Allowing the customer to send his or her key share for the suite that they hope to be accepted. This reduces numerous negotiating stages if it's the one the server will use.



5 Cipher Suite Comparison

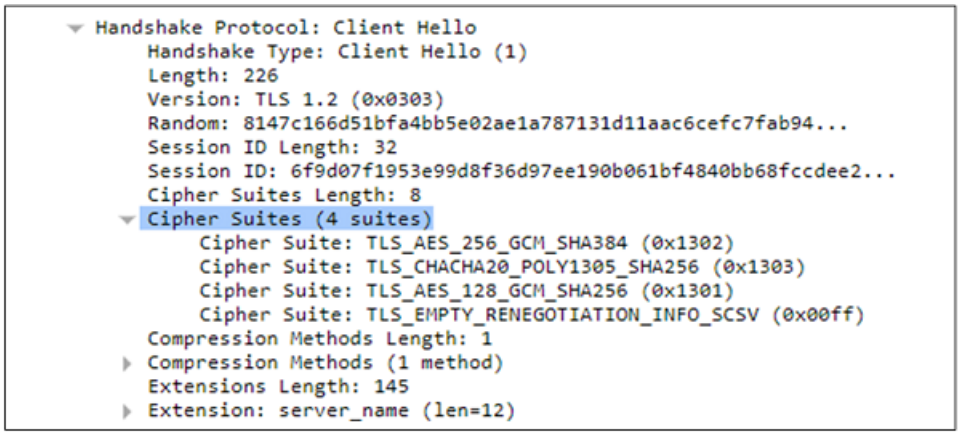
5.1 TLS v1.2

The number of cipher suites used in TLS 1.2 is 28.



5.2 TLS v1.3

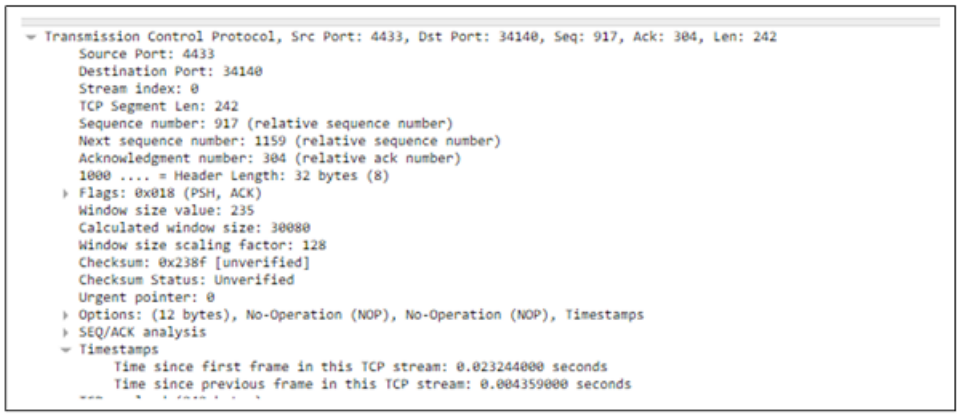
Only 4 cipher suits are used in TLS 1.3.



6 Latency Comparison

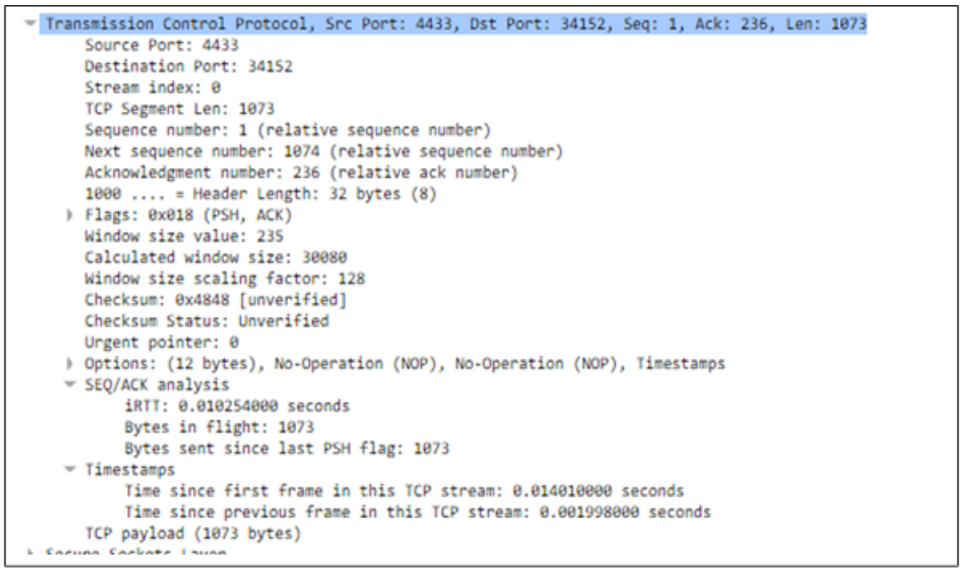
6.1 TLS v1.2

Time since first frame till the last step of handshake is `0.02344000` seconds in TLS1.2.



6.2 TLS v1.3

Time since first frame till last step of handshake is 0.01401000 seconds in TLS1.3 which is lesser in comparison to TLS 1.3



7 Conclusion

TLS 1.3 is an interesting protocol upgrade from which we may expect to benefit for many years to come. Not only will encrypted *(HTTPS)* connections become faster, but they will also become more safe.
We've seen how the handshakes of TLS1.2 and TLS1.3 differ in this work. TLS1.2 handshakes take two round-trip times, whereas TLS1.3 handshakes only take one. There are also variations in the cipher suites used by TLS1.2 and TLS1.3, with TLS1.3 employing far fewer cipher suites. In addition, as compared to TLS1.2, TLS1.3 has a lower latency.
TLS 1.3 is intended to be a solid, safe, resilient, simple, and important foundation for Internet encryption for the foreseeable future. It's also faster, so no one will have a performance reason to not use it.