Report on TLS 1.3

Aarav Varshney[1]

[1]**Ashoka University**, `aarav.varshney@alumni.ashoka.edu.in`

Abstract

The Internet Engineering Task Force (IETF) formed the Transport Layer Security (TLS) working group to develop a standardized SSL-like protocol. In 1999, the group released the specification for the TLS 1.0 protocol [1], which is a minor variation of SSL 3.0 and is sometimes referred to as SSL version 3.1. In 2006 and 2008, minor updates were made to TLS 1.0, leading to the development of TLS version 1.2. However, due to numerous security vulnerabilities, TLS 1.2 was overhauled in 2017, resulting in the much stronger TLS version 1.3 [2]. TLS has become widely used in software systems worldwide. This report will mostly focus on TLS 1.3.

1. Introduction

Application
Transport
Network
Physical
Link Layer

Figure 1. The TCP/IP Reference Model [3]

When two computers want to send data to each other, they may use the **Internet Protocol** (IP) [4] directly. IP fragments data into blocks of data called *packets* (also called datagrams) and transmits them from the source to the destination. The sources and destinations are computers identified by fixed length addresses. However, IP is a low-level protocol that does not interact directly with application data and may cause unreliable data transmission.

In the TCP/IP Reference Model [3], the IP protocol works in the network layer, which is two layers below the application layer (see Figure 1). This layer provides an unreliable, connectionless delivery system (there is no direct connection between two hosts) which does not provide any functionality for error recovery for datagrams that are either duplicated, lost or arrive at the remote host in another order than they were sent. The transport layer, which is the layer above the network layer, is responsible for fixing the unreliable data transmission with transport protocols like **Transmission Control Protocol (TCP)** [1]. TCP is the most commonly used transport protocol on top of IP and includes strategies for packet ordering, retransmission, and maintaining data integrity.

TCP is a connection-oriented protocol, which means that before any data can be sent, a connection must be established between the two hosts. This connection is established by a three-way handshake between the two hosts. The first computer sends a packet with the SYN (a field in the

TCP header) bit set to 1. The second computer responds with a packet with the SYN and ACK (another field in the TCP header) bits set to 1. The first computer then sends a packet with the ACK bit set to 1. Once the connection is established, the two hosts can start sending data to each other.

Now that we have established a connection between two hosts and have the ability to reliably send data across, we can further improve the transmission by using **Transport Layer Security (TLS)** [2] (original [5]). TLS protocol allows two hosts to communicate in a way that is designed to prevent eavesdropping, tampering, or message forgery. While TCP provides reliable data transmission, the data in the packets is unencrypted and can be read by anyone who has access to the network. This is acceptable when the data is not sensitive, but when the data is sensitive, TLS is used to *encrypt* the data. TLS also provides *authentication* of the remote host, which means that the remote host can be trusted to be the host it claims to be.

TLS is primarily used in a client/server setting where the client initiates the connection and the server responds. Setting up a TLS connection requires a handshake between the client and the server. At the end of the handshake, the client and the server have agreed upon a shared secret key which is used to encrypt the data, the encryption schemes and their parameters, and the client is also assured of the server's identity (not without issues [6, 7]).

In this report, we discuss TLS 1.3 [2], which is the latest version of TLS in two sections. Section 1 covers the *handshake protocol* in TLS 1.3. Section 2 discusses the *record protocol* that uses the parameters established by the handshake protocol to protect traffic between the communicating hosts.

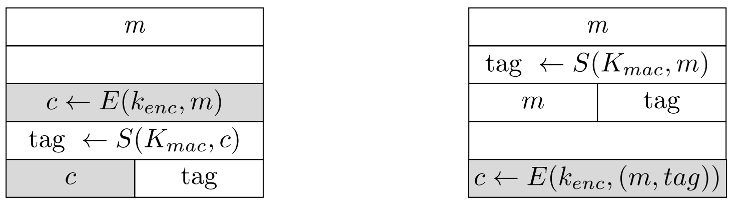
2. Background

2.1. Authenticated Encryption with Associated Data (AEAD)

Cryptographic applications commonly require both confidentiality and message authentication. Confidentiality ensures that data is available only to those authorized to obtain it; usually it is realized through encryption. Message authentication is the service that ensures that data has not been altered or forged by unauthorized entities; it can be achieved by using a Message Authentication Code (MAC). This service is also called data integrity. **Authenticated Encryption (AE)** [8] schemes ensure both data secrecy (confidentiality) and data integrity. These schemes can be constructed with either a **generic composition** that combines a CPA-secure cipher with a secure MAC, or to build them directly from a block cipher or a PRF without first constructing either a standalone cipher or MAC [9]. The latter schemes are called **integrated schemes**.

Let (E, D) be a cipher and (S, V) be a MAC. Let k_{enc} be a cipher key and k_{mac} be a MAC key. In a generic composition, these two primitives can be combined using two commonly used options: **Encrypt-then-MAC** and **MAC-then-Encrypt**. In the first option, the plaintext is encrypted using the cipher and the resulting ciphertext is authenticated using the MAC. In the second option, the plaintext is authenticated using the MAC and the resulting MAC tag is encrypted using the cipher (see Figure 2a and Figure 2b). Only the first method is secure for every combination of CPA-secure cipher and secure MAC. The intuition is that the MAC on the ciphertext prevents any tampering with the ciphertext. The second method is known to have attacks in some cases [9].

TLS uses a nonce-based **Authenticated Encryption with Associated Data (AEAD)** [10] cipher to encrypt and authenticate packets. AEAD schemes are an extension of AE schemes (discussed above) that allow the sender to send additional data along with the message. This additional data is called the **associated data** (AD) and its integrity is protected by the ciphertext, but its secrecy is not. The associated data is required to set context; for example in a networking protocol, authenticated encryption protects the packet body, but the header must be transmitted in the clear so that the network can route the packet to its intended destination [9]. Nonetheless, we want to ensure header integrity so that a malicious adversary can't get away with tampering the header. This header is provided as the associated data input to the AEAD encryption scheme. Secondly, the encryption scheme takes an additional input nonce $\mathcal{N}$, which is a random value that is used only once. This nonce ensures that the same plaintext is never encrypted to the same ciphertext.


m
$c \leftarrow E(k_{enc}, m)$
tag $\leftarrow S(K_{mac}, c)$
c
tag
m
tag $\leftarrow S(K_{mac}, m)$
m
tag
$c \leftarrow E(k_{enc}, (m, tag))$

(a) Encrypt-then-MAC **(b)** MAC-then-Encrypt

Figure 2. Two different ways to combine a cipher and a MAC.

3. TLS Handshake Protocol

For consistency with the notation, we let P play the role of the client and Q play the role of the server. P and Q wish to setup a secure session.

TLS 1.3 supports both one-sided and mutual authentication. In most cases, authentication for the client is optional. In the notation, (E_s, D_s) is a symmetric encryption scheme that provides authenticated encryption, such as AES-128 in GCM mode. Algorithm S refers to a MAC signing algorithm, such as HMAC-SHA256. Algorithms $Sig_P(\cdot)$ and $Sig_Q(\cdot)$ sign the provided data using P's or Q's signing keys. Finally, the hash functions H_1, H_2 are used to derive symmetric keys. They are built from HKDF (Apppendix I) with hash functions such as SHA-256. The notations are taken from [9].

The cipher-suites which determine the symmetric encryption scheme, the hash function in the MAC signing algorithm and HKDF are negotiated during the handshake. The cipher-suites are defined in [2]. They are specified in Table 1.

Description	Value
TLS_AES_128_GCM_SHA256	$\{0 \times 13, 0 \times 01\}$
TLS_AES_256_GCM_SHA384	$\{0 \times 13, 0 \times 02\}$
TLS_CHACHA20_POLY1305_SHA256	$\{0 \times 13, 0 \times 03\}$
TLS_AES_128_CCM_SHA256	$\{0 \times 13, 0 \times 04\}$
TLS_AES_128_CCM_8_SHA256	$\{0 \times 13, 0 \times 05\}$

Table 1: Cipher-Suites [2]. Cipher suite names follow the naming convention: CipherSuite TLS_AEAD_HASH = VALUE;

The protocol. The handshake protocol

References

[1] "Transmission Control Protocol," RFC 793, Sep. 1981. [Online]. Available: https://www.rfc-editor.org/info/rfc793

[2] E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3," RFC 8446, Aug. 2018. [Online]. Available: https://www.rfc-editor.org/info/rfc8446

[3] A. S. Tanenbaum and D. J. Wetherall, *Computer Networks*, 5th ed. USA: Prentice Hall Press, 2010.

[4] "Internet Protocol," RFC 791, Sep. 1981. [Online]. Available: https://www.rfc-editor.org/info/rfc791

[5] C. Allen and T. Dierks, "The TLS Protocol Version 1.0," RFC 2246, Jan. 1999. [Online]. Available: https://www.rfc-editor.org/info/rfc2246

[6] B. Laurie, A. Langley, and E. Kasper, "Certificate Transparency," RFC 6962, Jun. 2013. [Online]. Available: https://www.rfc-editor.org/info/rfc6962

[7] A. Parsovs, "Practical issues with tls client certificate authentication," 01 2014.

[8] M. Bellare and C. Namprempre, "Authenticated encryption: Relations among notions and analysis of the generic composition paradigm," vol. 21, 12 2000, pp. 531–545.

[9] D. Boneh and V. Shoup, *A Graduate Course in Applied Cryptography*, 2015. [Online]. Available: https://crypto.stanford.edu/~dabo/cryptobook/draft_0_2.pdf

[10] P. Rogaway, "Authenticated-encryption with associated-data," in *Proceedings of the 9th ACM Conference on Computer and Communications Security*, ser. CCS '02. New York, NY, USA: Association for Computing Machinery, 2002, p. 98–107. [Online]. Available: https://doi.org/10.1145/586110.586125