Table of Contents

II

Transport Layer Security Protocol For Internet Of Things

Illya Gerasymchuk
illya.gerasymchuk@tecnico.uliboa.pt,
WWW home page: https://iluxonchik.github.io/

Instituto Superior Tcnico
Supervisors: Ricardo Chaves, Aleksandar Ilic

Abstract. The abstract should summarize the contents of the paper using at least 70 and at most 150 words. It will be set in 9-point font size and be inset 1.0 cm from the right and left margins. There will be two blank lines before and after the Abstract. . . .

Keywords: TLS, IoT, cryptography, protocol, lightweight cryptography

1 Intoduction

TODO: Intruduce the topic: explain what is IoT; what is TLS; what are the issues with using RAW TLS with IoT(power, computation, limited resources).

1.1 Goals

2 Related Work

TODO: Tell that first I describe the parts of TLS that are common to both and then specialize for TLS 1.2 and TLS 1.3

2.1 The TLS Protocol

TLS stands for Transport Layer Security, it's a **client-server** protocol that runs on top a **connection-oriented and reliable transport protocol**, such as **TCP**. Its main goal is to provide **privacy** and **integrity** between the two communicating peers. Privacy implies that a third party will not be able to read the data, while integrity means that a third party will not be able to alter the data.

In the TCP/IP Protocol Stack, Transport Layer Security (TLS) is placed between the **Transport** and **Application** layers. It's designed to make the application developer's life easier: all the developer has to do is create a "secure" connection, instead of a "normal" one.

SSL vs TLS: What's The Difference? You will find the names Secure Sockets Layer (SSL) and TLS used interchangeably in the literature, so I think it's important to distinguish both. TLS is an evolution of the SSL protocol. The protocol changed its name from SSL to TLS when it was standardized by the Internet Engineering Task Force (IETF).SSL was a proprietary protocol owned by Netscape Communications, and The IETF decided that it was a good idea to standarize it, which resulted in `RFC 2246` [5], specifying TLS 1.0, which was nothing more than a new version SSL 3.0, very few changes were made. In this document, I'll be concentrating on TLS 1.2 and TLS 1.3 protocols. The first one is the most recently standardized version of TLS and the latter is currently and in-draft version with many improvements and optimizations relevant for the topic of this dissertation. Despite the protocol name not suggesting it TLS 1.3 is very different from TLS 1.2, in fact, it should've probably been called TLS 2.0 instead. For this reason, I will first describe what is common to both protocols and then go into the relevant details about each one.

TODO: Explain what RFCs are?

2.2 Security Services

TLS provides the following 3 security services:

- **authentication** - both, **peer entity** and **data origin** (or **integrity**) authentication.
  - **peer entity authentication** - we can be sure that were talking to certain entity, for example, `www.google.com`. This is achieved thought the use of **asymmetrical** or Public Key Cryptography (PKC) (for example, `RSA` and `DSA`) or **symmetric key cryptography**, using a Pre-Shared Key (PSK).
- **confidentiality** - the data transmitted between the communicating entities (the client and the server) is encrypted. Symmetric cryptography is used of data encryption (for exmaple, `AES`).
- **integrity** (also called **data origin authentication**) - we can be sure that the data was not modified or forged, *i.e.*, be sure that the data that were receiving is coming from the expected entity (for example, we can be sure that the `index.html` file sent to us when we connected to `www.google.com` in fact came from `www.google.com` and that it was not modified (i.e tampered with) en route by an attacker (**data integrity**). This is achieved through the use of a keyed Message Authentication Code (MAC) or an Authenticated Encryption With Associated Data (AEAD) cipher.

Despite using PKC, TLS does **not** provide **non-repudiation services**: neither **non-repudiation with proof of origin**, which addresses the user denying having sent a message, not **non-repudiation with proof of delivery**, which addresses the user denying the receipt of a message. This is due to the fact, that instead of using **digital signatures**, either a keyed MAC or an AEAD cipher is used, both of which require a **shared secret** to be used.

You are not required to use all of the 3 security services in every situation. You can think of TLS as a framework that allows you to select which security services you want to use for a communication session. As an example, you might ignore certificate validation, which means you're ignoring the **authentication** guarantee. There are some differences regarding this claim between TLS 1.2 and TLS 1.3, for example, while in the first you have a `null` cipher (no authentication, no confidentiality, no integrity), in the latter this is not true, since it deprecated all non-AEAD ciphers in favor of AEAD ones.

Cipher Spec vs Cipher Suite The meaning of these terms differs in TLS 1.2 and TLS 1.3. For TLS 1.2, **cipher spec** is the message encryption algorithm and the message authentication algorithm, while the **cipher suite** is the **cipher spec**, as well as the **key exchange** algorithm. In TLS 1.3, the **cipher spec** has been removed altogether, since the **ChangeCipherSpec** protocol has been removed. The concept of **cipher suite** has been updated to define the pair of AEAD algorithm and hash function to be used with HMAC-based Extract-and-Expand Key Derivation Function (hkdf): in TLS 1.3 the **key exchange** algorithm is negotiated via extensions. You'll find more details on this below.

2.3 TLS (Sub)Protocols

In reality TLS is composed of several protocols, a brief description of each one of which follows:

- **TLS Record Protocol** - the lowest layer in TLS. It's the layer that runs directly on top of **TCP/IP** and it serves as an **encapsulation for the remaining sub-protocols** (4 in case of TLS 1.2 and 3 in case of TLS 1.3). To the **Record Protocol**, the remaining sub-protocols are what `TCP/IP` is to `HTTP`.
- **TLS Handshake Protocol** - the core protocol of TLS. Allows the communicating peers to **authenticate** one to another and negotiate a **cipher suite** (**cipher suite** and key exchange algorithm in case of TLS 1.3) which will be used to provide the security services. For TLS 1.2, **compression** method is also negotiated here.
- **TLS Alert Protocol** - allows the communicating peers to signal potential problems.
- **TLS Application Data Protocol** - used to transmit data securely.
- **TLS Change Cipher Spec Protocol** (removed in TLS 1.3) - used to activate the initial **cipher spec** or change it during the connection.

Figure 1 shows the subprotocols composing tls.

TLS Record Processing A TLS record must go through some processing before it can tbe sent over the netwrok. This processing involves the following steps (4 for TLS 1.2 and 3 for TLS 1.3):

1. **Fragmentation** - the `\gls{tls} Record Layer` takes arbitrary-length data and **fragments** it into manageable pieces: each one of the resulting fragments is called a `TLS Plaintext`.

2. **Compression** (removed in TLS 1.3) - the `TLS Record Layer` compresses the `TLSPlaintext` structure according to the negotiated compression method, outputting `TLSCompressed`. Compression is optional. If the negotiated compression method is `null`, `TLSCompressed` is the same as `TLSPlaintext`.
3. **Cryptographic Protection** - in case of TLS 1.2, either an AEAD cipher or a separate encryption and MAC functions transform a `TLSCompressed` fragment into a `TLSCipherText` fragment. In case of TLS 1.3, the `TLSPlaintext` fragment is transformed into a `TLSCipherText` by applying an AEAD cipher.
4. Append the **TLS Record Header** - encapsulate `TLSCipherText` in a `TLS Record`.

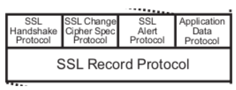

Fig. 1. TLS (Sub)protocols and Layers

The process described above, as well as the structure names are depicted in figure 2. Step 2 is not present in TLS 1.3. The structure names are exactly as the appear in the TLS specifications.

With this the common description of the TLS of protocols ends and we'll jump into the specifics of the two verions. I'll be mostly concentrating on the **Handshake Protocol**, since this is where my work will be concentrated and it's the main part, where the most interesting and important things happen.

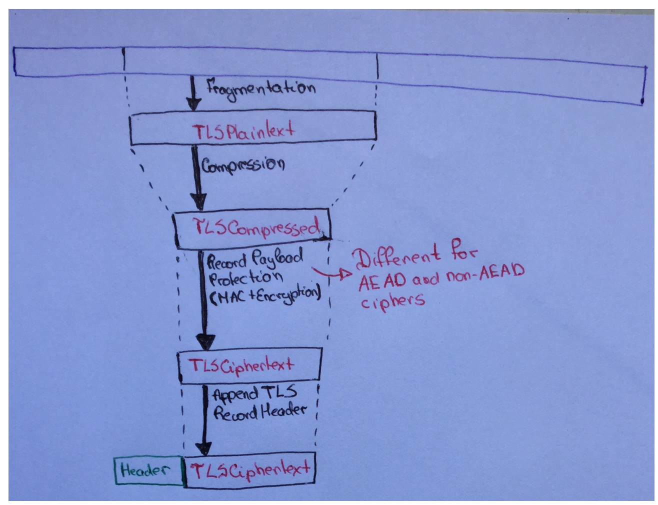

Fig. 2. TLS Record Processing

2.4 TLS 1.2

The latest standardized version of TLS is 1.2.

Notes and Comments. This is an example of a paragraph. Note the styling.

2.5 TLS 1.3

Despite the protocol name not suggesting it TLS 1.3 is very different from TLS 1.2, in fact, it should've probably been called TLS 2.0 instead.

How Do Peers Distinguish Different TLS Versions? TODO: Talk about version numbers

2.6 TLS Extension Mechanism

TODO: Describe the Extended ClientHello/ServerHello. Use one description for both, TLS 1.2 and TLS 1.3

References

1. Badra, M., Hajjeh, I.: ECDHE$_{P}SKCipherSuitesforTransportLayerSecurity(TLS).RFC5489,RFCEditor(March$
2. Blake-Wilson, S., Bolyard, N., Gupta, V., Hawk, C., Moeller, B.: Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS). RFC 4492, RFC Editor (May 2006), http://www.rfc-editor.org/rfc/rfc4492.txt, http://www.rfc-editor.org/rfc/rfc4492.txt
3. Blumenthal, U., Goel, P.: Pre-Shared Key (PSK) Ciphersuites with NULL Encryption for Transport Layer Security (TLS). RFC 4785, RFC Editor (January 2007)
4. Dierks, T., Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246, RFC Editor (August 2008), http://www.rfc-editor.org/rfc/rfc5246.txt, http://www.rfc-editor.org/rfc/rfc5246.txt
5. Dierks, T., Allen, C.: The TLS Protocol Version 1.0. RFC 2246, RFC Editor (January 1999), http://www.rfc-editor.org/rfc/rfc2246.txt, http://www.rfc-editor.org/rfc/rfc2246.txt
6. Eronen, P., Tschofenig, H.: Pre-Shared Key Ciphersuites for Transport Layer Security (TLS). RFC 4279, RFC Editor (December 2005), http://www.rfc-editor.org/rfc/rfc4279.txt, http://www.rfc-editor.org/rfc/rfc4279.txt
7. Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.3. Internet-Draft draft-ietf-tls-tls13-21, IETF Secretariat (July 2017), http://www.ietf.org/internet-drafts/draft-ietf-tls-tls13-21.txt, http://www.ietf.org/internet-drafts/draft-ietf-tls-tls13-21.txt
8. Tschofenig, H., Fossati, T.: Transport Layer Security (TLS) / Datagram Transport Layer Security (DTLS) Profiles for the Internet of Things. RFC 7925, RFC Editor (July 2016)
9. Wouters, P., Tschofenig, H., Gilmore, J., Weiler, S., Kivinen, T.: Using Raw Public Keys in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS). RFC 7250, RFC Editor (June 2014)

Glossary

AEAD Authenticated Encryption With Associated Data. 2–4

hkdf HMAC-based Extract-and-Expand Key Derivation Function. 3

IETF Internet Engineering Task Force. 2

MAC Message Authentication Code. 2, 4

PKC Public Key Cryptography. 2

PSK Pre-Shared Key. 2

SSL Secure Sockets Layer. 2

TLS Transport Layer Security. 1–4