

TLS For IoT

Illya Gerasymchuk
Instituto Superior Tecnico
Lisbon, Portugal
illya@iluxonchik.me

Abstract—Transport Layer Security (TLS) is, one of the most used communication security protocols, it is however, not suitable in the context of Internet Of Things (IoT). The resource-limited nature of a big part of IoT devices does not allow for the use of computationally complex and memory demanding operations present in a standard TLS implementation. Most of previous work focused entirely on Datagram TLS (DTLS) and can not be easily integrated with existing deployments. This work focuses on how TLS and the extension mechanism can be used to define a framework to adapt the protocol to specific needs. This is the approach that will be followed in the second part of the work. Having an adaptable and easy to use solution is crucial for its adaptation in IoT, where security might have been completely foregone otherwise.

Index Terms—TLS, DTLS, SSL, IoT, cryptography, protocol, lightweight cryptography

I. INTRODUCTION

The Internet of Things (IoT) is a network of devices, from simple sensors to smartphones and wearables which are connected together. In fact, it can be any other object that has an assigned IP address and is provided with the ability to transfer data over a network. Even a salt shaker [1] can now be part of the global network.

The IoT technology provides many benefits, from personal comfort to transforming entire industries, mainly due to increased connectivity and new sources for data analysis. The technological development, however, tends to focus on innovative design rather than on privacy and security. IoT devices frequently connect to networks using inadequate security and are hard to update when vulnerabilities are found.

This lack of security in the IoT ecosystem has been exploited by the the *Mirai* botnet [2] when it overwhelmed several high-profile targets with massive Distributed Denial-Of-Service (DDoS) attacks. This is the most devastating attack involving IoT devices done to date. However, the *Reaper* botnet [3] could be even worse if it is ever put to malicious use. Similar attacks will inadvertently come in the future.

TLS is one of the most used security protocols in the world, allowing two peers to communicate securely. It is designed to run on top of a reliable, connection-oriented protocol, such as TCP. Datagram TLS (DTLS) is the version of TLS that runs on top of an unreliable transport protocol, such as UDP. Most IoT devices have very limited processing power, storage and energy. Moreover, the performance of TCP is known to be inefficient in wireless networks, due to its congestion control algorithm. This situation is worsened with the use of low-power radios and lossy links found in sensor networks.

Therefore, the use of TCP with IoT is usually not the best option. For this reason, DTLS, which runs on top of UDP, is used more frequently in such devices. The work that will be done in the context of this dissertation, can however, be applied to either one of them, so even though mostly TLS will be mentioned, almost everything can also be applied to DTLS. This is a consequence of DTLS being just an adaption of TLS over unreliable transport protocols, with no changes done to the core protocol.

The problem in using (D)TLS in IoT is that it is not lightweight, since it has not been designed for such environments. An IoT device may only have 256 KB of RAM and needs to conserve the battery, while sending and receiving a large amount of small information constantly. For example, consider the case of a temperature sensor that sends temperature measures every 30 seconds to a server. In this case it just needs to send a few bytes of data and do it with minimal overhead, to conserve RAM and battery. If that sensor is going to use (D)TLS 1.2, it will need two extra roundtrips before it can send any data. This can result in an overhead of several hundreds of milliseconds. Besides that, it will need to perform heavy mathematical operations involved in cryptography, using even more energy and taking even more time. Given this, there is a clear need for a more lightweight (D)TLS for the IoT. This is the problem that we will be addressing in this work.

In the process of the work on this dissertation, we have already made several contributions to the TLS 1.3 specification, being recognized as contributors [4].

The document is organized as follows: Section II describes the TLS protocol. Section III summarizes the related work. Section IV describes the goal of this work. In Section V, the work being currently done is described. Finally, the conclusion is done in Section VI.

II. THE TLS PROTOCOL

TLS is 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, TLS is placed between the **Transport** and **Application** layers. It is designed to simplify the establishment and use of secure communications from the application developer's standpoint. The developer's task is

reduced to creating a "secure" connection (*i.e.* socket), instead of a "normal" one.

A secure communication established using TLS has two phases.

- 1) The communicating peers authenticate one to another and negotiate security parameters, such as the secret keys and the encryption algorithm. This occurs under the Handshake Protocol.
- 2) The communicating peers exchange cryptographically protected data using the security parameters negotiated during the Handshake Protocol. This occurs under the Record Protocol.

In order to achieve its goals, during the Handshake Protocol the client and the server exchange various messages. The message flow is depicted in Figure 1. * indicates situation-dependent messages that are not always sent.

TLS provides the following **security services**:

- **authentication** - both, **peer entity** and **data origin** (or **integrity**) authentication.
 - peer entity authentication** - a peer has a guarantee that it is talking to certain entity, for example, www.google.com. This is achieved through the use of Asymmetrical Cryptography (AC), also known as Public Key Cryptography (PKC), (*e.g.* 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 for data encryption (*e.g.*, AES).
- **integrity** (also called **data origin authentication**) - a peer can be sure that the data was not modified or forged, *i.e.*, there is a guarantee that the received data is coming from the expected entity. For example, a peer can be sure that the `index.html` file that was sent to when it connected to www.google.com did, in fact, come from www.google.com and it was not tampered with by an attacker (**data integrity**). This is achieved either through the use of a keyed Message Authentication Code (MAC) or an Authenticated Encryption With Associated Data (AEAD) cipher.
- **replay protection** (also known as **freshness**) - a peer can be sure that a message has not been replayed. This is achieved through the use of sequence numbers. Each TLS record has a different sequence number, which is incremented. If a non-AEAD cipher is used, the sequence number is a direct input of the MAC function. If an AEAD cipher is used, a nonce derived from the sequence number is used as input to that cipher.
- **perfect forward secrecy (PFS)** - the confidentiality of the past interactions is preserved even if the long-term secret is compromised.

Despite using PKC, TLS does **not** provide **non-repudiation services**: neither **non-repudiation with proof of origin**, which addresses the peer denying the sending of a message,

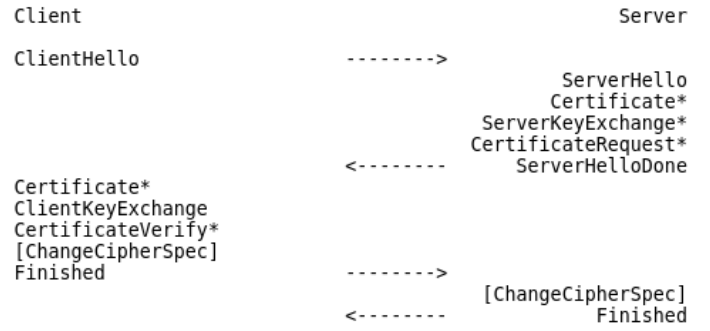


Fig. 1. TLS 1.2 message flow for a full handshake

nor **non-repudiation with proof of delivery**, which addresses the peer 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 secret to be **shared** between the peers.

It is not required to use all of the three security services every situation. In this sense, TLS is like a framework that allows to select which security services should be used for a communication session. As an example, certificate validation might be skipped, which means that the **authentication** guarantee is not provided. There are some differences regarding this claim between TLS 1.2 [5] and TLS 1.3 [6]. For example, while in the first there is a null cipher (no authentication, no confidentiality, no integrity), in the latter this is not true.

III. RELATED WORK

Most of the work has been centered around DTLS, even though the majority of solutions can be applied to TLS as well. Herein we want to further explore TLS optimization. There is clearly a need for that, specially with Constrained Application Protocol (CoAP) over TCP and TLS standard being currently developed. The mentioned standard does not explore any TLS optimizations, and since any IoT device using it in the future would benefit from them, this is an important area to explore. CoAP [7] is often referred to as the "HTTP protocol for constrained devices".

None of the related work explored (D)TLS 1.3, mainly because the protocol is still in draft stages, however, major design changes are not expected at this point.

The majority of the work done in the area proposes a solution that is either tied to a specific protocol, such as CoAP, or requires an introduction of a third-party entity, such as the trust anchor in the case of the S3K system [8] or even both. This has two main issues. First, a protocol-specific solution cannot be easily used in an environment where (D)TLS is not used with that protocol. Second, the requirement of a third-party introduces additional cost and complexity, which will be a big resistance factor in adopting the technology. This is specially true for developers working on personal projects or projects for small businesses, leaving the communications insecure in the worse case scenario.

IV. PROPOSED SOLUTION

The goal of this work is to develop a lightweight version of (D)TLS that is fully backwards compatible and does not require any third-party entities, in order to simplify its deployment process. The solution will be developed for (D)TLS version 1.2, while also bearing in mind the new 1.3 version. The idea is to make it customizable, depending on the security requirements and the context of its usage.

To achieve this goal, a thorough evaluation of every part of (D)TLS is needed. The main parts that will be evaluated are power consumption and the number CPU cycles used.

We would like to associate a quantitative cost with every (D)TLS feature. For example, we would like to answer questions such as "*How much does Perfect Forward Secrecy cost, in terms of CPU cycles?*". This will help the application developers in making decisions related to the security/performance trade off.

Furthermore, there is currently no tool which can suggest a (D)TLS configuration, based on the provided environment's limitations (e.g. available memory, power, processing speed) and security requirements (e.g. PFS, data privacy). We would like to develop it.

V. CURRENT WORK

In order to understand which features of (D)TLS should be used, as an answer to the limitations and requirements of the context, it's important to associate a cost (e.g. number of CPU cycles used) with each one of those features. For this reason, we are currently working on profiling the various parts of the TLS protocol.

All of the evaluation work is being done on the TLS implementation of the *MBEDTLS* library [9].

VI. CONCLUSION

The lack of security in IoT is a serious issue that can lead to a high monetary costs, when botnets infect the devices. Recent attacks clearly show that serious damage can be caused. An old saying attributed to the US National Security Agency (NSA) states that "Attacks always get better; they never get worse". Combined with the fact that the number of IoT devices is growing at a high pace, without any major improvements to their security, makes it clear that it is fundamental for this issue to be addressed.

While there are well established security solutions, not all of them can be used with IoT devices, due their constrained nature. One such example is the (D)TLS protocol, that because of its heavyweight nature is not suitable for a large part of IoT devices. With the proposed work, we want to contribute to this area, by designing a solution that is suitable for the IoT devices, transparent to the programmer and provides security services adaptable to the specific context needs.

ACKNOWLEDGMENTS

I would like to thank my supervisors, Ricardo Chaves and Aleksandar Ilic for their continuous support, ideas and help that they have given me during this work.

REFERENCES

- [1] SMART - The Smart Home Device That Elevates Your Dining Experience. Electronic Gadget, 2017. (Accessed on 01/05/2018).
- [2] Manos Antonakakis, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J Alex Halderman, Luca Invernizzi, Michalis Kallitsis, et al. Understanding the mirai botnet. 2017.
- [3] Brian Krebs. Reaper: Calm before the iot security storm? krebs on security. <https://krebsonsecurity.com/2017/10/reaper-calm-before-the-iot-security-storm/>, 10 2017. (Accessed on 01/04/2018).
- [4] Addition to tls 1.3 contributors list. <https://github.com/tlswg/tls13-spec/commit/43461876882a60251ecf24fb097f0ce2d7be4745>. (Accessed on 01/11/2018).
- [5] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246, RFC Editor, August 2008. <http://www.rfc-editor.org/rfc/rfc5246.txt>.
- [6] Eric Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3. Internet-Draft draft-ietf-tls-tls13-26, IETF Secretariat, March 2018. <http://www.ietf.org/internet-drafts/draft-ietf-tls-tls13-26.txt>.
- [7] C. Bormann and Z. Shelby. Block-Wise Transfers in the Constrained Application Protocol (CoAP). RFC 7959, RFC Editor, August 2016.
- [8] Shahid Raza, Ludwig Seitz, Denis Sitenkov, and Göran Selander. S3k: Scalable security with symmetric keydtls key establishment for the internet of things. *IEEE Transactions on Automation Science and Engineering*, 13(3):1270–1280, 2016.
- [9] mbed TLS (formerly PolarSSL): SSL/TLS Library For Embedded Devices. (Accessed on 12/19/2017).