Evaluating the Use of TLS and DTLS Protocols in IoT Middleware Systems Applied to E-health

Ramão Tiago Tiburski, Leonardo Albernaz Amaral, Everton de Matos, Dario F. G. de Azevedo, Fabiano Hessel
Pontifical Catholic University of Rio Grande do Sul (PUCRS) - Porto Alegre - RS - Brazil
(ramao.tiburski, leonardo.amaral, everton.matos.001)@acad.pucrs.br, (dario, fabiano.hessel)@pucrs.br

***Abstract*—The evolution of the Internet of Things (IoT) has brought new security requirements in terms of communication services with respect to data transmitted in mobile networks. Although IoT middleware systems have been used to cope with the most relevant requirements demanded by different IoT applications, security is a special topic that is not mature enough in this kind of technology. E-health is an example of environment that exposes sensitive data. The security challenges regarding e-health applications are concentrated mainly on issues surrounding the communication layer, specially those cases where data are transmitted over insecure networks. TLS and DTLS protocols have been chosen by most of the existing IoT systems in order to protect such communications. However, none of them was designed to be used in IoT situations. In addition, none of the existing works analyzes their suitability to the IoT regarding the usage of mobile networks, which are common in real-world scenarios of e-health. In this paper, we analyze the use of TLS and DTLS protocols in IoT middleware systems applied to the e-health environment regarding performance (i.e., response time), overhead, network latency and packet loss when operating in mobile networks. We evaluated both protocols through a specific e-health scenario. Tests revealed the usage of mobile networks increases response time and overhead of both protocols, on average, when compared to traditional networks.**

I. Introduction

The Internet of Things (IoT) is a computing paradigm that aims to interconnect our everyday life devices or objects using the Internet as the communication medium [1]. IoT ecosystem is based on a layered architecture style and uses this representation to abstract and automate the integration of objects, and also to provide smart services solutions to applications [2]. In IoT, high-level system layers, as the application layer, are composed of IoT applications and middleware system which is an entity that simplifies the development of applications by supporting services to cope with the interoperability requirement of heterogeneous devices [3].

An important application field for IoT middleware is the e-health environment [1]. Many applications for e-health have been created in scenarios like Emergency Medical Services (EMS), in which an ambulance-to-hospital based e-health system is an appropriated example of how IoT technology can help save lives. In this case, by providing patient information to the hospital via mobile networks, this e-health system can enable remote diagnoses and primary care, reducing rescue response time.

Middleware systems technology can be used to abstract the devices integration in an ambulance, and also to allow the proper interaction with hospital systems by the use of mobile networks. In e-health, it is mandatory to have a mobile network able to ensure fast response times between sending and interpreting data to guarantee that all decisions of a physician are based on the current health condition of a patient.

However, IoT environments usually have security issues. Moreover, the deployment of a new security layer may generate an additional overhead and, depending on the constraints of the environmental resources, these costs cannot be allowed [2]. Without security, data transmitted can be targets of attacks that seek to spy or change them. In an EMS scenario, for example, there is a lack for ensuring data confidentiality and integrity as a means of enabling a reliable understanding of a patient's current life state.

Providing solutions able to mitigate the security problems found in IoT middleware architectures is a needful task [4]. In addition, it is important to understand the communication challenges imposed by the IoT when we try to provide security to systems that operate in mobile networks. Basically, challenges are related to performance (i.e., response time), overhead, latency and packet loss. Although most of existing work provide relevant solutions regarding these challenges, the evaluation of TLS and DTLS protocols when operating in mobile networks remains an unreached task [5] [6] [7].

In this paper, we present experiments with TLS and DTLS protocols. Compared with the related work, the main contribution of this work are twofold: (1) evaluate the applicability of TLS and DTLS in IoT middleware applied to a specific e-health scenario; and (2) analyze the performance, overhead, network latency and packet loss of both protocols when operating in mobile networks. We also present related work and a discussion regarding both protocols.

The remainder of this paper is organized as follows: Section II gives an overview of security in IoT middleware. Section III presents the related work. Section IV presents the security protocols and Section V presents the evaluation. Section VI discusses both protocols and related issues. Finally, Section VII concludes the paper.

II. Security in IoT Middleware

IoT middleware is a software layer or a set of sub-layers interposed between IoT technological (perception and transportation layers) and application layers [1]. IoT middleware has received much attention in the last years due to its significant role in simplify the development of applications and the integration of devices.

978-1-5090-6196-9/17/$31.00 ©2017 IEEE

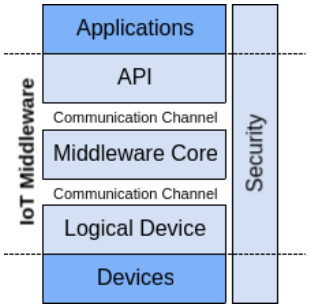

Fig. 1. IoT middleware architecture.

The use of IoT middleware is required due to some reasons: 1) middleware naturally acts as a bond joining IoT heterogeneous components together as a pool of resources to help the development of applications; 2) middleware provides important services for applications allowing an effective management of data; and 3) middleware provides abstraction for physical layer communications and services to applications, hiding all the details of diversity.

An IoT middleware architecture is presented in Fig. 1. According to [4], Applications allow end users to request information services and to interact with the Middleware Core system. The Logical Device system must abstract Devices functions to the Middleware Core. It can be composed of any IoT device, which can connect to the Middleware Core in order to provide services based on its features. Each service provided by the Middleware Core is composed of one or more functions from Devices. The Applications should use an API from the Middleware Core in order to consume the provided services. All the processing activity is generated in the Middleware Core. Finally, Security layer must ensure the protection of all stored/exchanged data. In addition, it must provide security in a non-intrusive manner, ensuring the delivery of data for applications in an acceptable response time.

A secure IoT middleware system should ensure the protection of sensitive data against a lot of attacks. According to [8], systems that use wireless communications are more accessible and exposed than hardwired systems. The main security challenges are related to understand what are the attacks that may happen and also to choose the appropriated security requirements for each attack. We present the most common attacks in the following items [9]:

- *Man-in-the-Middle:* Attacker intercepts the path of communications between two legitimate parties, thereby obtaining authentication credentials and data.
- *Message Modification:* Attacker actively alters a legitimate message by deleting, adding to, changing, or reordering it.
- *Eavesdropping:* Attacker passively monitors the communications network for capturing communicating data and authentication credentials.
- *Denial-of-Service (DoS):* It happens when an attacker can continuously send requests to be processed by specific things and therefore exhaust their resources.

In order to mitigate such attacks in IoT middleware channels, we have to deal with some important security requirements. We give a brief description of them as follows [4]:

- *Authentication:* It includes some features such as credentials and trust management, and the guarantee of correct identity of applications, middleware and devices.
- *Confidentiality:* It ensures the inaccessibility of information for unauthorized users. In addition, confidential messages resist revealing their content to eavesdroppers.
- *Integrity:* It ensures that received data are not altered in transit by an adversary. In addition, the integrity of stored data and content should not be compromised.

III. Related Work

Once security support is essential for IoT middleware systems, we investigate what security requirements have been provided by them. In addition, we analyze what have been proposed regarding the use of TLS and DTLS protocols and how the existing work evaluate their approaches.

The security of IoT middleware systems is the main focus of some relevant works [2] [3] [4]. Most of the middleware systems are focused on protecting transmitted data since they implement authentication, confidentiality, and integrity [10] [11] [12] [13], which are strongly related to information protection. TLS is chosen as the security communication protocol by VIRTUS [10] and SIRENA [11]. On the other hand, SOCRADES [12] and Hydra [13] use WS-Security, which is a standard to protect SOAP web services.

According to [4], WS-Security is focuses on the use of XML Signature and XML Encryption to provide end-to-end security. However, WS-Security is very heavyweight and has issues with key distribution, federated identity and access control. In end-to-end situations, authentication, confidentiality, and integrity can also be enforced on Web services through the use of TLS. In addition, applying TLS can significantly reduce the overhead involved by removing the need to encode keys and message signatures into XML before sending.

In this sense, the work in [2] calls for the needed of non-intrusive security solutions for IoT systems such as key management, confidentiality, integrity, and authentication, which are crucial to have an efficient solution.

Performance and applicability of DTLS to constrained environments have been a discussed issue in IoT research communities. According to [14], many authors studied DTLS in this context and proposed different optimization techniques. Raza et al. [7] focused on reducing the per-datagram overhead and proposed a 6LoWPAN DTLS compression scheme. Another proposal by Raza et al. aims to reduce the communication overhead of the DTLS headers for CoAP through compression [15]. Kumar et al. [16] summarized DTLS memory requirements, high level communication overhead (in terms of number of messages), code size for different DTLS and cryptographic functions.

However, there is a lack in IoT regarding analysis of TLS and DTLS protocols when used with mobile networks. Most of the existing work evaluate the use of TLS/DTLS in the IoT in

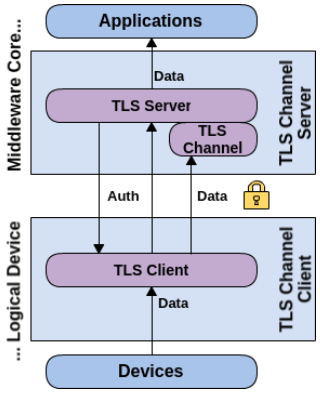

Fig. 2. Schematic overview of the TLS protocol.

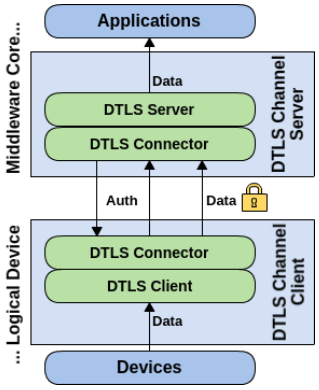

Fig. 3. Schematic overview of the DTLS protocol.

different ways: performance, overhead, memory, and energy consumption [5] [6] [7]. However, none of them analyzes these challenges when operating in mobile networks, which are mandatory in most of IoT environments.

IV. Security Protocols

We implemented both protocols in order to evaluate their applicability in IoT middleware applied to e-health. Although they were designed for traditional networks, both protocols have been used in IoT environments as an alternative to protect communications, mainly the DTLS. The following subsections describe each implemented protocol.

A. TLS

In order to implement the TLS (Transport Layer Security) protocol we have used Java Secure Socket Extension (JSSE) [17], a Java package that enables secure Internet communications using TLS. We used TLS version 1.2 with public-key encryption to ensure the privacy of messages sent over the Internet. Both Middleware Core and Logical Device (see Fig. 1) must have a pair of keys, one public and one private. However, before they can exchange messages, they must generate and store these keys. For the loading of keys we used the Java keytool, which is a key and certificate management tool that manages a keystore (database) of cryptographic keys, X.509 certificate chains, and trusted certificates.

Fig. 2 provides a schematic overview of the TLS protocol that was implemented. It is composed of three main classes: TLSServer, TLSClient and TLSChannel. TLSServer listens for incoming connections on a specified port. Each time a connection comes in, TLSServer creates a new TLSChannel instance to process the connection. Processing a connection means receiving text messages from TLSClient and sending them to the upper layers of the Middleware Core. When TLSClient starts up, it initiates a handshake with TLSServer. TLSClient keeps this connection open throughout the middleware-device session. Each message sent during this connection is protected.

For both authentication, encryption and decryption, the TLS uses a cipher suite to know what algorithms should be employed. We used TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256, which implements Elliptic-curve Diffie-Hellman Ephemeral (ECDHE) key exchange using the Elliptic-curve Digital Signature Algorithm (ECDSA), with AES-128 as the block cipher, GCM for authenticated encryption and SHA-256 HMAC for the authentication hash.

B. DTLS

In order to implement the DTLS (Datagram Transport Layer Security) protocol we have used Scandium (Sc) [9], a part of the Californium Eclipse Project (Java-based implementation of CoAP) that provides security for CoAP [18]. The DTLS version used was 1.2. We chose to implement DTLS protocol based on Scandium because it is able to provide confidentiality, integrity and reliability of transmitted data. In addition, it is similar to JSSE used to implement TLS, once JSSE does not provide support for DTLS.

Although UDP does not provide reliability features as a standard of its protocol, Scandium library provides reliable communications through some adaptions to the DTLS protocol. It provides a DTLS package able to ensure similar characteristics of TCP such as the delivery and ordering of messages [19]. The reliability requirement is extremely important since some IoT environments such as e-health require trust communications between systems in order to provide a reliable result for applications.

In order to achieve authentication, both Middleware Core and Logical Device exchange messages according to the full DTLS handshake protocol [19]. It specifies messages like the Certificate, which contains a certificate chain of X.509 certificates where the first certificate in the chain contains the entity's public key. The loading in entities is done with the Java keytool. In this case, there are two keystore files: the keystore containing a private key and a certificate chain with the first certificate having the corresponding public key, and the truststore that contains all trusted certificates.

Fig. 3 provides a schematic overview of the DTLS protocol that was implemented. It is composed of three main classes: DTLSServer, DTLSClient, and DTLSConnector, that is shared by both sides (i.e., Middleware Core and Logical Device). DTLSServer and DTLSClient open their channels and provide a DatagramSocket to be used during the communication.

DTLSClient is responsible for starting the handshake. On the other hand, DTLSConnector is responsible for all the processing (encrypting, decrypting), key and messages exchange, etc., in both sides. After the handshake, all messages are exchanged in a protected way.

For the DTLS protocol we used TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8, which implements ECDHE key exchange using ECDSA, with AES-128 as the block cipher and CCM for authenticated encryption that uses eight octets (64 bits) for authentication resulting in a ciphertext, which is eight octets longer than the corresponding plaintext.

V. Evaluation

A. E-health Scenario

In order to evaluate both security protocols we implemented them on COMPaaS [20], a SOA-based IoT middleware system that strictly follows the architecture describe in Section II (see Fig. 1). We simulated the deployment of COMPaaS in an EMS scenario, which is a type of emergency service dedicated to provide out-of-hospital acute medical care and uses IoT middleware and mobile networks to transmit data.

To perform the tests, we consider the scenario illustrated in Fig. 4. It allows examining the use of both protocols between Logical Device and Middleware Core. This is the reason why the protocols were implemented only in the lower middleware layers. In addition, this scenario takes us to deal with an essential requirement: to ensure an acceptable response time for applications during data transmission. This scenario highlights the importance of non-intrusive security approaches in IoT environments when there are patients' data involved.

Let's consider an ambulance answers an emergency call in a street. Middleware Core system already knows about the existence of the ambulance (i.e., the ambulance is already registered) and waits for a connection request. The ambulance arrives at the scene and uses a mobile/wireless network infrastructure to establish a connection and authenticate itself with the Middleware Core. Since Logical Device system is already configured in the ambulance, some paramedic has only to start the system when it arrives at the scene. Thus, after the patient is connected to medical devices, they start generating data.

The Logical Device system is connected to three medical devices (i.e., an electrocardiogram machine, a pulse oximeter, and a blood pressure monitor) through their APIs, that compose an IoT gateway. Logical Device receives the devices' data of a patient every 1.5 seconds (i.e., standard generation time of each device) and sends it to the Middleware Core through a mobile/wireless network. The Middleware Core, which is located at the hospital, receives the data through a wired connection and then processes, stores, and sends these data to the corresponding application. The emergency room physicians receive the data and can analyze the current life condition of the patient and also can prepare themselves adequately to receive the patient in an adequate way.

In this scenario, we assume that Application and Middleware Core are communicating through an already protected internal network of the hospital. In this sense, we have to protect only the channel between Middleware Core and Logical Device.

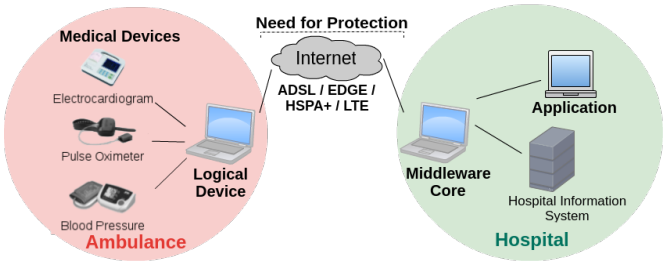

Fig. 4. IoT middleware applied in an EMS scenario.

B. Environment Setup

We analyzed performance, overhead, network latency, and packet loss of both protocols when applied to four distinct networks that were divided into 2 groups. Mobile networks: EDGE (Enhanced Data Rates for GSM Evolution): pre-3G radio technology; HSPA+ (Evolved High Speed Packet Access): HSPA evolution that is noun as 3.5G; and LTE (Long-Term Evolution): standard for high-speed wireless communication for mobile phones and data terminals. And a traditional network (for comparison reasons): ADSL (Asymmetric Digital Subscriber Line), that enables faster data transmission over telephone lines.

To perform the tests we have used an infrastructure composed of three computers with the same software and hardware. Both were configured with Ubuntu 14.04 LTS (64-bit), Intel Core 2 Duo @ 2.20GHz and 4GB of RAM. They were responsible for hosting Application and API, Middleware Core, and Logical Device and Devices, respectively. We used a cell phone to rotate the traffic and send the data from Logical Device through the Internet. The cell phone was configured with Android 4.4, processor Quad Core 1.2 Ghz, 1GB of RAM. Each Logical Device was connected to three medical devices and each message sent had about 1 KB. The goal of each Logical Device was to represent an ambulance in a city infrastructure. According to estimations reported by the World Health Organization (WHO), the optimal amount of ambulances for each city is one unit for every 150 thousand inhabitants. In this sense, the simulation of one ambulance (i.e., Logical Device) is a good approach.

The use of computers in this evaluation is feasible since we are analyzing security protocols in IoT middleware when operating in different networks. Once we are not looking for processing capacity, memory and CPU of the device side, we believe the use of computers is an acceptable way of simulating such scenario. However, we are aware that in a real application this scenario could suffer mobile signal oscillations and intend to evolve our research in future works.

C. Experiments and Results

The security protocols were compared against their respective non-secure approaches in order to verify how much

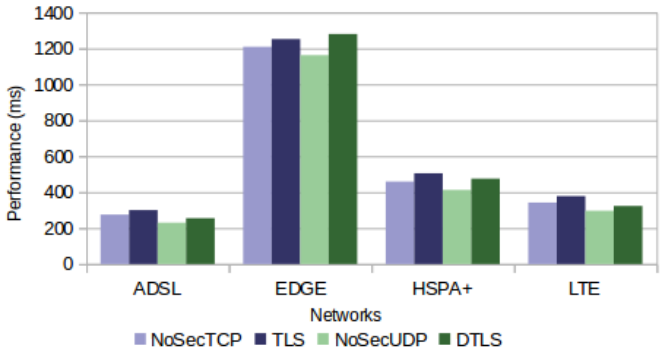

Fig. 5. Performance in different networks (ms).

overhead they create in a transmission. We did not consider the time spent during the handshake between the parts, only the time spent between sending and receiving messages after the handshakes.

We measured the elapsed time for four implemented approaches: NoSecTCP (open channel, only sockets over TCP), TLS (protected channel, TLS over TCP), NoSecUDP (open channel, only datagram sockets over UDP), and DTLS (protected channel, DTLS over UDP). To get comparable results, the process of generation and transmission of data was performed for 10 minutes for each network in each approach.

TABLE I
NETWORKS RESULTS.

Feature/Network	ADSL	EDGE	HSPA+	LTE
Packet Loss (%)	3	7	4	2
Latency (ms)	21.4	959.1	207.3	88.2

Before we start the evaluation, we made some tests with all the networks in order to analyze their behavior in the tested environment. In this way, we can observe the obstacle imposed by each network. Table I presents the behavior of each network related to packet loss and latency. We observed that the latency of the mobile networks is bigger than other networks. Also, packet loss was more frequent in mobile networks. Networks with different percentages of packet loss are good approaches to analyze TLS and DTLS since it impacts in the way they should manage the transmission of messages. In addition, packet loss is common in mobile communications, so this is a relevant way of evaluating.

We also evaluated if TLS and DTLS are able to provide security in a non-intrusive way. It means that Middleware Core and Logical Device should interact with each other in an acceptable time regarding the constrained requirements of the application, which is related to the data generating time of the devices (i.e., in this case, must be lower than 1.5 seconds). We tested the entire flow of information into the IoT middleware system (i.e., before data leaving from Logical Device until they reach the Application.) for different networks.

Fig. 5 presents the performance (i.e., response time for applications) for each approach tested in different networks (in milliseconds). It was observed that the performance of both protocols occurred in accordance with the packet loss rates presented in each network. DTLS was faster than TLS in ADSL, HSPA+ and LTE networks, which had the lowest packet loss and latency results. On the other hand, DTLS was slower than TLS in EDGE network. This happens by the high DTLS packet fragmentation rate existing in Scandium library. Since there are more packets to be sent and the network is slower (as in EDGE network).

We consider the results obtained for performance were acceptable, mainly because the time interval of sending data was 1.5 seconds, and the slower results (observed in EDGE network) are smaller than 1.5s. It means that a generated data is being delivered before a new one is generated, which is extremely important in e-health scenarios as EMS, where patient data need to get as fast as possible to the hospital to be analyzed by physicians.

Table II presents the overhead added by the security approaches (%). Regarding the traditional network (ADSL), TLS increased the response time in 9.2% while DTLS increased in 11.1%. Regarding mobile networks, TLS increased the response time in 6.5%, on average, when compared to NoSecTCP. On the other hand, DTLS increased the response time in 11%, on average, when compared to NoSecUDP. DTLS obtained the best performances, on average, when compared to TLS. However, the overhead obtained by DTLS over NoSecUDP was higher than the overhead obtained by TLS over NoSecTCP, on average.

TABLE II
OVERHEAD ADDED BY THE SECURITY PROTOCOLS (%).

Approach/Network	ADSL	EDGE	HSPA+	LTE
TLS over NoSecTCP	9.2	3.8	10.3	10.7
DTLS over NoSecUDP	11.1	11.5	15.4	9.2

The overhead add by TLS and DTLS is not only related to the security layer, but also with the traffic management (delivering and ordering of messages). Regarding the higher overhead from DTLS, it is related to its fragmentation and reliability functions, which had to deal with packet loss in these tests. Although DTLS does not guarantee reliability as a standard as TLS does, it was able to provide data reliability in the same way that Scandium library provides for CoAP. It uses a “sequence number” field to verify if the messages were coming in an orderly way. Regarding the delivery of data, it uses standard messages as “ACK messages” in order to warn that a message was received. However, this approach spends more time than TLS, which controlled packet loss according to TCP standards.

VI. DISCUSSION

TLS and DTLS have been used as alternative protocols for communications protection in IoT environments. A considerable fact is that TLS is widely more used than DTLS, and the main reason is: TLS is a well-defined standard protocol used to protect TCP communications, which are, in fact,

more common than UDP communications. On the other hand, DTLS emerged based on the need for the protection in UDP connections and inherited the status of "a standard secure protocol for IoT communications". For this reason, the number of existing work focused on optimize TLS are just a few if compared to others which are focused on DTLS optimizations. TLS is widely used in traditional communications. However, although there are some challenges to overcome, it is not seen with a perspective of future in IoT environments.

DTLS performed better when applied to mobile networks with low rates of latency and packet loss. On the other hand, TLS showed stable in relation to performance and overhead, since TCP is the responsible for managing delivery and order packet, and not the security layer, as in DTLS. Even so, the tests demonstrated that both implemented protocols were able to provide security for IoT middleware channels without jeopardizing the system performance when operating in mobile networks.

Regarding the attacks in Section II, TLS and DTLS might contain attacks such as man-in-the-middle, message modification, and eavesdropping. Related to man-in-the-middle, only completely anonymous sessions are inherently vulnerable to such attacks, but we do not negotiate cipher suites that support anonymous sessions. If the server is authenticated using a public key infrastructure, an attacker has no possibility to forge the server's ServerKeyExchange message which is signed with the appropriate private key. Regarding DoS attack, DTLS introduces a stateless cookie in the server's HelloVerifyRequest message that prevents DoS amplification attacks, and therefore protects constrained nodes. However, the server still remains a vulnerable target.

Regarding the used networks, we chose EDGE, HSPA+, and LTE since they are commonly used in real scenarios of IoT. Tests demonstrated that TLS and DTLS protocols had to deal with a considerable latency and packet loss in these networks. However, we understand that it is not a consequence of the security feature since the obtained results point for low percentages when we look for overhead. In this way, although DTLS had to deal with the reliability feature in a specific way, tests demonstrated that the mere addition of the security layer did not mean a significant impact on the performance results.

VII. Conclusions and Future Work

In this paper we evaluated the use of TLS and DTLS protocols in IoT middleware systems applied to a specific e-health scenario when operating in mobile networks (EDGE, HSPA+, and LTE). We analyzed both protocols through tests on COMPaaS middleware, that shown an increase of 6.5% for TLS and 11% for DTLS, on average, regarding the overhead of the security layer when operating in mobile networks. We conclude that the use of TLS and DTLS protocols is suitable for IoT middleware systems applied to e-health scenarios since both protocols protected data and respected the response time requirement of the applications.

In the future, we intend to improve our evaluation analyzing other important IoT issues, such as hardware with limited processing capacity, energy consumption, battery lifetime, etc.

Acknowledgment

Our thanks to CAPES/CNPq for the funding within the scope of the PNPD project.

References

[1] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Computer Networks*, vol. 54, no. 15, pp. 2787 – 2805, 2010.

[2] Q. Jing, A. Vasilakos, J. Wan, J. Lu, and D. Qiu, "Security of the Internet of Things: perspectives and challenges," *Wireless Networks*, vol. 20, no. 8, pp. 2481–2501, 2014.

[3] P. Fremantle and P. Scott, "A security survey of middleware for the Internet of Things," *PeerJ PrePrints*, vol. 3, p. e1521, 2015.

[4] R. Tiburski, L. Amaral, E. Matos, and F. Hessel, "The importance of a standard security architecture for SOA-based IoT middleware," *IEEE Communications Magazine*, vol. 53, no. 12, pp. 20–26, Dec 2015.

[5] S. L. Keoh, S. Kumar, and H. Tschofenig, "Securing the Internet of Things: A standardization perspective," *Internet of Things Journal, IEEE*, vol. 1, no. 3, pp. 265–275, June 2014.

[6] T. Kothmayr, C. Schmitt, W. Hu, M. Brunig, and G. Carle, "A DTLS based end-to-end security architecture for the Internet of Things with two-way authentication," in *37th Conference on Local Computer Networks Workshops*, 2012, pp. 956–963.

[7] S. Raza, H. Shafagh, K. Hewage, R. Hummen, and T. Voigt, "Lithe: Lightweight secure CoAP for the Internet of Things," *Sensors Journal, IEEE*, vol. 13, no. 10, pp. 3711–3720, Oct 2013.

[8] D. Lake, R. Milito, M. Morrow, and R. Vangheese, "Internet of things: Architectural framework for ehealth security," *Journal of ICT*, vol. 3, pp. 301–330, 2014.

[9] S. Jucker, "Securing the constrained application protocol," Ph.D. dissertation, Master's thesis, Department of Computer Science, ETH Zurich, Switzerland, 2012.

[10] D. Conzon, T. Bolognesi, P. Brizzi, A. Lotito, R. Tomasi, and M. Spirito, "The VIRTUS middleware: An XMPP based architecture for secure IoT communications," in *21st International Conference on Computer Communications and Networks*, July 2012, pp. 1–6.

[11] H. Bohn, A. Bobek, and F. Golatowski, "SIRENA - service infrastructure for real-time embedded networked devices: A service-oriented framework for different domains," in *International Conference on Networking*, April 2006, pp. 43–43.

[12] P. Spiess, S. Karnouskos, D. Guinard, D. Savio, O. Baecker, L. Souza, and V. Trifa, "SOA-based integration of the Internet of Things in enterprise services," in *IEEE International Conference on Web Services*, July 2009, pp. 968–975.

[13] A. Badii, J. Khan, M. Crouch, and S. Zickau, "Hydra: Networked embedded system middleware for heterogeneous physical devices in a distributed architecture," in *Final External Developers Workshops Teaching Materials*, April 2010, p. 4.

[14] M. Vucinic, B. Tourancheau, T. Watteyne, F. Rousseau, A. Duda, R. Guizzetti, and L. Damon, "DTLS performance in duty-cycled networks," in *26th International Symposium on Personal, Indoor, and Mobile Radio Communications*. IEEE, 2015, pp. 1333–1338.

[15] S. Raza, D. Trabalza, and T. Voigt, "6LoWPAN compressed DTLS for CoAP," in *8th International Conference on Distributed Computing in Sensor Systems*. IEEE, 2012, pp. 287–289.

[16] S. Kumar, S. Keoh, and H. Tschofenig, "A Hitchhiker's Guide to the (Datagram) Transport Layer Security Protocol for Smart Objects and Constrained Node Networks," 2013.

[17] Oracle, "Java Secure Socket Extension (JSSE) Reference Guide," 2016.

[18] Z. Shelby, K. Hartke, and C. Bormann, "The constrained application protocol (CoAP)," 2014.

[19] E. Rescorla and N. Modadugu, "Datagram transport layer security version 1.2," 2012. [Online]. Available: https://tools.ietf.org/html/rfc6347

[20] L. A. Amaral, R. T. Tiburski, E. de Matos, and F. Hessel, "Cooperative middleware platform as a service for Internet of Things applications," in *30th Symposium on Applied Computing*. ACM, 2015, pp. 488–493.