A DTLS-based Security Architecture for the Internet of Things

Gléderson Lessa dos Santos, Vinícius Tavares Guimarães, Guilherme da Cunha Rodrigues,
Lisandro Zambenedetti Granville, Liane Margarida Rockenbach Tarouco
Institute of Informatics
Federal University of Rio Grande do Sul (UFRGS)
Porto Alegre - RS - Brazil
Email: {glsantos,vtguimaraes,gcrodrigues,granville}@inf.ufrgs.br, liane@penta.ufrgs.br

***Abstract*— The Internet of Things (IoT) is part of the Future Internet. IoT comprises a huge amount of devices (hereinafter called as constrained devices) able to interact with the environment and to communicate over the Internet. Among other challenges that prevents the growth of IoT, the IoT is challenged for security issues. In this work, we are mainly interested in secure communication concerns for constrained devices. In essence, constrained devices are devices operating under low-power, and with limited computational and network resources. For such characteristics, they do not support standard security protocols and, consequently, they become a potential target for traditional Internet attacks (e.g., Denial of Service and man-in-the-middle). Thus, we introduce an architecture to enable constrained devices to use Datagram Transport Layer Security (DTLS) with mutual authentication to communicate with Internet devices. Briefly, we propose a third part device called Internet of Things Security Support Provider (IoTSSP) and two main mechanisms: (i) the Optional Handshaking Delegation, and (ii) the Transfer of Session. Experimental results show the proposal feasibility and its additional benefits.**

Keywords*—*Internet of Things; DTLS; CoAP; mutual authentication; handshaking delegation.

I. Introduction

The Internet of Things (IoT) is an extension of the Internet, where real-world objects/devices (hereinafter called as constrained devices [1]) are able to both interact with the environment (through sensors and actuators), and exchange data with users or other devices by using the Internet infrastructure. Nowadays, the growth of IoT depends, mainly, on the improvement of security aspects, privacy policies, data and wireless standards, as well as the perception that the IoT requires the formation of a topology of services, applications, and infrastructure connections [2].

In general, constrained devices collect a significant amount of information from the environment and send them over the network. Accordingly the RFC 7228[1], several types of constrained devices have limited capabilities because they need gathering many design characteristics that lead to these constraints, including cost, size, weight and other scaling factors. The different combinations of these constraints can limit several characteristics of constrained devices, such as the code complexity (ROM, Flash), size of constrained device state and buffers (RAM), processing power, power avaliability, among others.

Such characteristics make difficult the use of traditional approaches (e.g., asymmetric keys) for secure communications because they are too complex for constrained devices. Historically, in Wireless Sensor Networks (WSN), such limitation led to the use of symmetric key techniques, where usually the keys are loaded in sensor nodes prior to deployment (Key Predistribution Scheme) in a provisioning phase [3]. WSNs are a suitable reference because, like in IoT, they are composed of similar constrained devices communicating over an IEEE 802.15.4 network.

In order to address this issue for the IoT, the IETF has started the Constrained RESTful Environments (CoRE) working group, which aims at standardising the integration of constrained devices with the Internet at service level. The CoRE proposal aims to allow the integration of constrained devices with the Internet, at service level. CoRE proposes the use of Constrained Application Protocol (CoAP) in constrained devices, a specialized RESTful Web transfer protocol [4]. In the same context, the DTLS In Constrained Environments (DICE) working group focuses on supporting the use of DTLS Transport-Layer Security in these environments. Actually, the first task of DICE is a work in progress that aims to define a suitable DTLS profile for the IoT [5].

Constrained devices usually employ the DTLS only with symmetric cryptography in Pre-Shared Key (PSK) security mode, as in WSN. This approach is feasible for WSN scenarios where communication occurs among a limited and known set of nodes. However, it is not enough for the IoT assumptions. For instance, *a priori*, it is not possible to predict which nodes a constrained device will communicate with. Thus, an approach to provide an end-to-end secure communication between a constrained device and an Internet device remains an open issue.

In this paper, we introduce an architecture to enable constrained devices to perform an end-to-end secure communication with Internet devices by using DTLS based on certificates with mutual authentication. To do that, we propose a third part device called Internet of Things Security Support Provider (IoTSSP) and two main mechanisms: (i) the Optional Handshaking Delegation, and (ii) a new extension of DTLS, named Transfer of Session . These mechanisms enable the use of the IoTSSP to establish a secure session to constrained devices. The main results obtained from our proposal are:

- Transparency: the architecture is transparent for non-constrained devices or constrained devices without critical data.

978-1-4673-7194-0/15/$31.00 ©2015 IEEE

- Scalability: the architecture supports several instances of the IoTSSP, each one responsible for a subset of constrained devices. For this reason, the architecture design is prepared to deal with a large number of constrained devices with suitable performance.
- Efficiency: The architecture minimizes the computational processing and the network consumption of constrained devices, increasing their lifetime.
- Adaptability: IoTSSPs are able to employ additional security mechanisms in order to deal with potential threats, such as Denial of Service attacks (DoS).

The remainder of this paper is organized as follows. In Section II, we start with an overview of works related to secure end-to-end communications in the IoT context and the problem statement. In Section III, we present the background on the IETF proposed standards and IoT network. In Section IV, we explain our proposed security architecture, while in Section V we evaluate it. Finally, in Section VI we conclude our paper and provide indications for future work.

II. Related Work

Asymmetric cryptography mechanisms, especially the RSA algorithm, are prohibitive for constrained devices in three-fold: (*i)* the processing cost to calculate the keying material; (*ii)* the memory usage to store a large number of keys and certificates, and (*iii)* the energy consumption to handle this algorithm. Thus, symmetric cryptography based solutions with PSK are a feasible approach and are commonly used in both WSN and IoT [6].

The use of PSK is reinforced because the IEEE 802.15.4 standard recommends the use of cryptography AES/CCM to provide security at the data link layer. Thereby, devices that communicate over a 802.15.4 network usually implement this encryption in hardware. Brachmann [7] proposes a TLS-DTLS mapping with Pre-Shared Keys to secure the IoT. Such a mechanism requires a trusted 6LoWPAN Border Router (6LBR). Lithe [8] is another proposal based on DTLS PSK, which uses the compression of DTLS header integrated with 6LoWPAN compression mechanisms. However, the PSK-based DTLS approaches do not achieve the IoT requirements, once these approaches presuppose that a constrained device must already know every entity that will communicate with it.

Several investigations have been carried to provide an end-to-end secure communication in the IoT context. We divided them in three main alternatives, as follows: (*i)* by means of ECC cryptography; (*ii)* by means of specific hardware; or (*iii)* by means of a trusted third party. We present the characteristics and related work of each alternative.

ECC is a feasible approach to support asymmetric cryptography in IoT, since it is more efficient than RSA in several aspects such as: energy consumption, memory usage, processing time, and communication costs [9] [10]. These advantages make the ECC the current method proposed to support the CoAP security [4]. The use of ECC is enhanced due to the optimizations in its algorithm to use in IoT. In this context we highlight the proposals of Marin *et al.* [11] [12] that obtains an optimization up to 42.8% with the Shifting Primes method when compared to other ECC algorithms.

Sizzle [13] is a historical initiative which presents a compact web server stack that provides secure HTTP communication by means of SSL with 160-bit ECC keys. However, Sizzle requires a reliable transport layer communication, which impacts in low-power communication performance, and mutual authentication is not supported. Similarly, SSNAIL [14] proposes an end-to-end communication with ECC-enabled handshaking and mutual authentication support, but it also requires a reliable transport layer protocol. Nevertheless, most of constrained devices are not ready to provide ECC support as demostrated in [6] and [15]. Specifically, the amount of required memory to store and to interpret certificates and public keys may exhaust the resources of more constrained devices.

The second alternative is based on the use of a dedicated hardware to enable security in constrained devices. It is also an interesting alternative, since security tasks are delegated to a specific hardware, which enables the employment of more sophisticated security mechanisms. Although feasible in some contexts, this approach has three main weaknesses: *(i)* it impacts in the design complexity, which may hinder the miniaturization of the constrained device, *(ii)* it increases production costs and, consequently, become expensive in a system with a large number of constrained devices, and *(iii)* it does not solve the overhead generated by certificates communication through a constrained network. Kothmayr [2] proposes the use of trusted-platform modules (TPM) added to each constrained device. This hardware is designed to support RSA encryption and private keys storage. However, this proposal is incompatible with the CoAP recommendation of using ECC cryptography. Furthermore, the work does not address issues associated with the overhead generated by the RSA handshaking with certificates in a 802.15.4 network.

The third alternative is based on delegation, where some stages of the authentication process and the mechanism of key establishment are delegated to a trusted third party. In essence, this approach assumes the following assumption: the trusted third-party devices have both computational power and energy supply to perform the most expensive stages to establish an end-to-end secure communication. Normally these proposals uses the 6LoWPAN Border Router (6LBR) as a trusted device. Yu [16] implements this approach using two categories of trusted devices: The Service Provider and a trusted gateway (the 6LBR). The first one is responsible for managing the access control to the constrained devices, using authentication by means of certificates. The trusted 6LBR is responsible for managing the public keys of users and constrained devices, and for negotiating a shared session key to establish the communication between them. However, this work is only focused on communication between a human internet user and constrained devices, without evaluating a Machine-to-Machine (M2M) communication. Moreover, this approach uses the TLS security protocol, which is not suitable for low-power networks.

Granjal [17] proposes a DTLS handshake mediated by the 6LBR device. In short, the 6LBR intercepts and forwards packets in the transport layer and executes the DTLS handshaking with ECC public-key cryptography. The communication between the 6LBR and constrained devices is performed based on pre-shared keys. Additionally, it uses an Access Control Server (AC) in the 6LoWPAN domain to support both the

authentication and the trust operation between the 6LBR and the constrained devices. The operation of DTLS with pre-shared keys was modified to enable the transmission of the pre-master secret. However, the AC needs communicating with the 6LBR without the limitations of the 802.15.4 network. The AC exchanges security information with both the 6LBR and the CoAP server by means of a proposed protocol named as PMSK.

Both proposals [16] [17] delegate the management of public keys and certificates to a 6LBR device. This is feasible by assuming that the 6LBR has enough resources to perform these mechanisms. However, this might be too complex for large infrastructures, since a single device manages the handshaking process for all constrained devices, and also the adaptation and routing of IPv6 packets to the IEEE 802.15.4 network. Thus, the overload generated by the handshaking process may decrease the system performance and scalability. Besides, both proposals do not take into account the existence of multiple 6LBRs. These proposals also do not regard the network heterogeneity, i.e., the 6LBR is responsible for performing DTLS tasks for all devices within the network. For this reason, even powerful devices delegate such tasks for the 6LBR. This behaviour decreases the network performance and its scalability, once the 6LBR can be overwhelmed with unnecessary security tasks.

Hummen *et al.* [18] propose to delegate the handshaking process to a trusted server named as Delegation Server (DS). The DS controls the bootstrapping of new connections for local constrained devices. To do that, the DS and constrained devices are administered by a common operator. This is an interesting approach, since enables an end-to-end secure communication among constrained devices and internet devices without overloading the 6LBR. However, the secure communication depends on a common operator, since to afford a new connection between a constrained device and an Internet device, the common operator configures the DS prior to handshaking. In essence, a secure communication between a constrained device and an Internet device can be started only after the human intervention. Such a behaviour hinders a traditional M2M communication, which characterizes an important gap in the proposal. In addition, constrained devices needs to record sessions tickets for future communications, with can also exhaust its resources faster.

Following the third alternative, we propose an architecture based on delegation, which allows to delegate authentication steps to a trusted third-party device (the IoTSSP) only for constrained devices. Thus, the architecture is transparent for devices with enough capabilities to execute entire DTLS communication process. Moreover, the architecture simplifies the session management in constrained devices, since they can establish new connections without previously store session data. However, this feature does not preclude the use of session resumption mechanisms. This approach is possible because communication requests between an Internet device and a constrained device are redirected from the 6LBR to the IoTSSP on-the-fly. Thereby, the IoTSSP controls the connections between Internet devices and constrained devices, the processing of public keys, and the certificate management without involving both a trusted 6LBR or human intervention.

III. Background

CoAP [4] defines four security modes: *NoSec*, *PreSharedKey*, *RawPublicKey* and *Certificate*. In the first one, the DTLS is disabled and security mechanisms are implemented by lower layer protocols. In *PreSharedKey* mode, the device has a list of connecting nodes and their respective keys. The exchange of these keys occurs in a previous phase, named Provisioning Phase, where all devices are configured. The Provisioning Phase is not defined by CoAP, although it instantiates the existence of this phase. In *RawPublicKey* mode, the device has an asymmetric key pair, an identity calculated from the public key, and a list of identities of the nodes that can communicate with it. This list is also configured in the Provisioning Phase. Finally, in *Certificate* mode, the device has an asymmetric key pair with a X.509 certificate. The CoAP defines the TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8 ciphersuite when using DTLS with Certificates. The certificates must be signed with ECDSA using secp256r1, and the signature must use SHA-256.

The IoT network consists of 6LowPAN domains connected to the Internet through 6LBRs. Each 6LoWPAN domain is composed by a set of heterogeneous constrained devices communicating through IEEE 802.15.4. The 6LoWPAN domain uses the 6LoWPAN adaptation layer to send and receive IPv6 packets. Each constrained device implements the CoAP protocol to enable a RESTful communication and uses one of security modes defined by CoAP to establish a communication. Furthermore, a constrained device may assume the function of CoAP server or client.

IV. Proposed Architecture

The main function of the architecture is to delegate the establishment of a DTLS session with a constrained device to the IoTSSP. In essence, the IoTSSP is responsible for: (*i)* manage certificates and cryptographic keys of constrained devices, (*ii)* analyze the Internet device certificate, (*iii)* authenticate the constrained device and the Internet device, (*iv)* establish a secure session between them, and (*v)* transfer this session to the contrained device. When an Internet device wants to communicate with a constrained device, the 6LBR redirects the DTLS handshaking to the IoTSSP. We propose two main mechanisms to implement this proposal:

- Optional Delegation of Handshaking: delegates the handshaking process to the IoTSSP.
- Transfer of Session: a DTLS extension that transfers a secure communication session to the constrained device.

The configuration of these mechanisms is executed in the provisioning phase, according to the resources and the security mode required for each constrained device. The infrastructure could be composed by one or more IoTSSPs, each one responsible for a set of constrained devices. The IoTSSP may be also responsible for the management of the access control policies on small infrastructures. Otherwise, it must communicate with the Access Control (AC) to verify the access permission. Additionally, the IoTSSP communicates with a *Certification Authority* (CA) to perform the certificate analysis as usual in a Public Key Infrastructe (PKI). Figure 1 depicts the architecture and its additional modules.

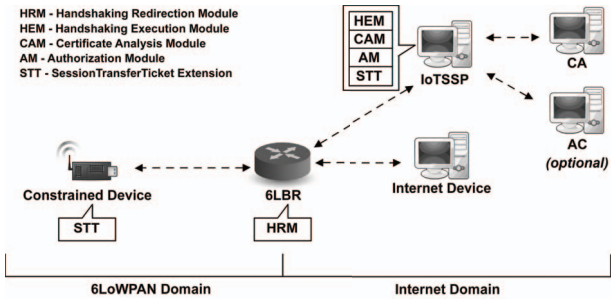

Figure 1. Proposed architecture and its additional modules

A. Provisioning Phase

In this phase the nodes exchange the keying material of both *RawPublicKey* and *PreSharedKey* operation modes. In the proposed architecture, a constrained device that needs to execute asymmetric cryptography exchanges the keying material with a trusted IoTSSP. It enables the constrained device to establish a secure channel with the IoTSSP by means of *PreSharedKey* mode. Furthermore, the IoTSSP receives the required information to perform the handshaking, such as the certificate, the supported ciphersuites, and the keying material of each constrained device associated to it.

In turn, the 6LBR records the associations between each constrained device and IoTSSP. As said before, unlike other proposals, the 6LBR does not execute the handshaking process, redirecting the handshaking messages to the IoTSSP. It is executed in the 6LBR by the Handshaking Redirection Module (HRM). In essence, the 6LBR verifies if the DTLS handshaking messages are addressed to a constrained device. If it is true, the 6LBR redirects the message to the respective IoTSSP. This is possible because the 6LBR is able to read the CONTENT_FIELD of DTLS record layer header. If the value of this field is 23 (application data) the 6LBR forwards the message to the constrained device. Otherwise, it redirects the message to IoTSSP. We place the IoTSSP in the Internet domain only to do not overwhelm the 802.15.4 network domain.

B. Optional Handshaking Delegation

The Optional Handshaking Delegation is depicted in Figure 2. In the IoTSSP, this process is executed by the Handshaking Execution Module (HEM). The HEM is responsible for both managing of the handshaking process and creating the session parameters. First, an Internet device (the client) sends a *ClientHello* message to start a communication with the constrained device (the CoAP server). As aforementioned, the 6LBR executes the HRM to redirect this message and all other handshaking messages to the IoTSSP. The HRM also tags each message with the identification of the real destination (the constrained device). The IoTSSP replies to the Internet device with a *VerifyRequest* message. This message contains a cookie generated by the IoTSSP to prevent Denial of Service (DoS) attacks. This protection is important because DoS attacks should be executed to exhaust the resources of constrained devices, such as their battery. Thus, the process continues only if the Internet device replies with another *ClientHello* containing the same cookie.

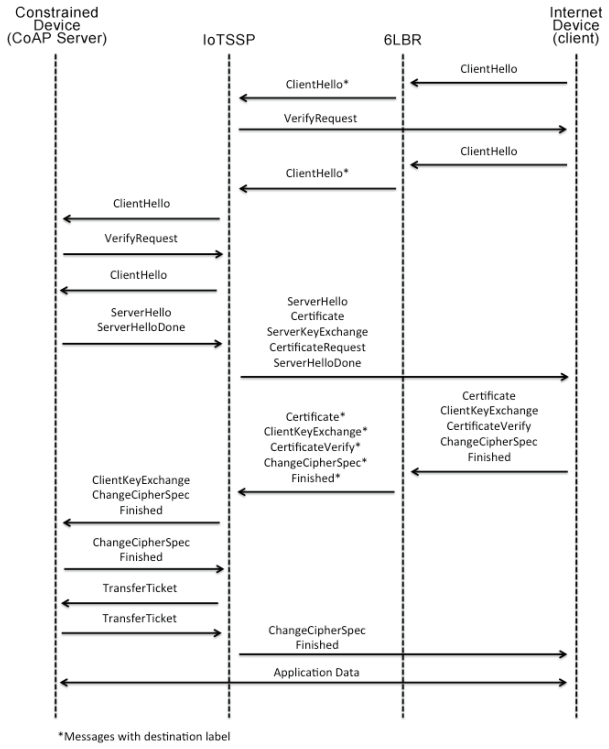

Figure 2. DTLS Handshaking modified to Constrained Devices

After that, the IoTSSP sends a *ClientHello* message to the constrained device. This message has the *Client Random Value*, that will be used to create the keying material of session between the Internet device and the constrained device. In this flight, the IoTSSP indicates to the constrained device the use of a new DTLS extension proposed in this work, the *SessionTransferTicket*. This extension enables the use of the Transfer of Session mechanism. The Transfer of Session Mechanism is detailed later.

Additionally, the *ClientHello* message checks the availability of the constrained device, since it may be busy due to its limited number of simultaneous connections. So, this message allows either the IoTSSP interrupts the handshaking process, or the constrained device allocates resources for this communication.

The handshaking between the IoTSSP and the constrained device executes the control to prevent DoS attacks as described previously. Then, the constrained device sends the *ServerHello* message to the IoTSSP. This message carries the *Session Id* and the *Server Random Value*. These values and the *Client Random Value* form the session keying material. The IoTSSP sends a *ServerHello* message to the Internet device with the same values. The IoTSSP continues the handshaking sending the *ServerKeyExchange* and the *ServerHelloDone* messages to the Internet device. The *ServerKeyExchange* carries the ephemeral ECDH public key to create the premaster secret key, as defined in [19]. Besides, both *ClientHello* and *ServerHello* messages carry, respectively, the list of ciphersuites that the Internet device and the constrained device supports. Although our proposal analyzes the establishment of a DTLS session with TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8,

other ciphersuites should be establised with this architecture transparently to the constrained device, such as TLS_DHE_RSA_WITH_AES_128_CCM_8 and TLS_RSA_WITH_AES_128_CCM_8. To do that, both the IoTSSP and the Internet device need to support the desired ciphersuite.

Furthermore, the IoTSSP validates the certificates by means of Certificate Analisys Module (CAM). The IoTSSP sends the constrained device certificate to the Internet device and requests the Internet device certificate. These messages allow mutual authentication. This is a hard process for the constrained device, in twofold: *(i)* the amount of resources required to validate a certificate; *(ii)* the size of the certificates. In this sense, an improvement in the architecture is to enable the IoTSSP to accomplish only the certificate analysis. It is applicable to constrained devices with enough resources to implement asymmetric cryptography but not the certificate validation.

The Internet device replies with its certificate and a *CertificateVerify* message to authenticate itself. In this point, the Authorization Module (AM) communicate with the Access Control to check if the Internet device has permission to access the constrained device. The Internet device also sends to the IoTSSP the *ClientKeyExchange*, which carries the premaster secret key. This enables both devices to calculate the master secret. The session is established with this secret. So, the Internet device sends both the *ChangeCipherVerify* and the *Finished* messages, and waits for the reply of these messages to communicate over the established session. However, before finishing the handshaking, the IoTSSP needs to transfer the session parameters to the constrained device.

C. Transfer of Session

As said previously, the transfer of session starts when the IoTSSP receives the Finished message from the Internet device. To transfer the session parameters, we implemented a DTLS extension called *SessionTransferTicket*. It is based on the *SessionTicket* extension, defined in [20]. The *SessionTicket* is used to reestablish a previous session, without maintaining a server-side state. In this extension, the client stores all parameters of a previous session in an encrypted ticket and sends it to the server. On the other hand, the *SessionTransferTicket* proposes to send the parameters of an active session, defined in a third device (the IoTSSP) to the constrained device. Unlike other proposals, this process is transparent to the Internet device.

Continuing the handshaking between the IoTSSP and the constrained device, after the second message *ClientHello*, the constrained device replies with a *ServerHello*. Both messages contain an empty *SessionTransferTicket* record. This record identifies that the IoTSSP wishes to use this mechanism and indicates that the constrained device supports it. The constrained device then sends a *ServerHelloDone* message.

The IoTSSP sends the *ClientKeyExchange* to define the preshared key used in this communication. Additionally, it sends both the *ChangeCipherVerify* and the *Finished* messages to the constrained device. The constrained device uses the *Finished* message to authenticate the handshaking as usual. Then, the IoTSSP sends both the *ChangeCipherVerify* and the *Finished* messages indicating the establishiment of a secure connection between them. The IoTSSP uses this session to send to the constrained device the parameters established with the Internet device, i.e., the premaster secret and the client id.

The constrained device creates the connection state of the session by using these parameters. Afterwards, it sends another *SessionTransferTicket* to the IoTSSP, which is composed by a nonce (with the same size of the premaster secret and the server id), confirming that it has received the message.

The IoTSSP then sends the *ChangeCipherVerify* and the *Finished* messages to the Internet device, enabling the session. It also deletes the keying material and the current status about the session with the Internet device. The constrained device deletes the session data created with the IoTSSP. Thus, the constrained device is ready to communicate with the Internet device by means of the transferred session.

The execution of this handshaking in the constrained device is similar to a handshaking in DTLS PSK mode. The difference is the new *TransferTicket* messages sent after the *Finished* message. For security reasons, the constrained device enables the transfer of session mechanism only for its respective IoTSSP.

V. Evaluation

In this section, we demonstrate the evaluation of the proposed architecture. First, we show the experimental scenario. Afterwards, we discuss about three aspects: feasibility, performance, and efficiency.

A. Experimental scenario

Figure 3 depicts the experimental scenario. We use the following devices to perform the evaluation:

- Constrained Device with IoTSSP support (CDSSP): it is a constrained device that performs a secure communication using the proposed architecture. It is based on an 8-bit ATmega2560 microprocessor (16MHz), 8Kbytes of RAM, 128Kbytes of flash memory for storing code and 4Kbytes of EEPROM for storing non-volatile data, in a Harvard architecture. It supports communication on 2.4GHz, achieving a bandwidth around 250Kbps. The 6LoWPAN stack is based on Arduino pIPv6 library [21], a port of Contiki OS.
- Constrained Device in DTLS PSK mode (CDPSK): it is a constrained device with the same features of the CDSSP. However, the CDPSK executes the traditional PSK mode to secure communications. In essence, this device is used to compare a constrained device that uses a traditional PSK mode (CDPSK) to another that uses Certificates mode by means of the proposed architecture (CDSSP).
- Device with DTLS in Certificate mode (DCert): it is a powerful Linux-based device that performs the usual DTLS in certificate mode and does not use the architecture to establish a secure communications. In a nutshell, this device simulates a constrained device that is capable to execute the entire DTLS protocol.

Additionally, three Linux servers were used to perform, respectively, the Internet client, the 6LBR and the IoTSSP.

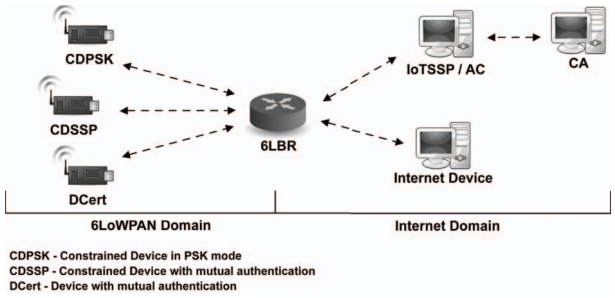

Figure 3. Experimental Scenario

B. Feasibility

In order to analyze the feasibility of the proposed architecture, we used the CDSSP to perform tests and mesure two metrics: memory usage and processing.

In a traditional approach, the memory capacity of this constrained device is not enough to perform the following tasks: managing of X.509 certificates, generating public keys, and executing the buffer control for messages fragmentation. When using the proposed architecture, we verified that the CDSSP is both able to establish a secure session and to achieve a suitable memory usage. Table I shows the memory footprint of the CDSSP.

Table I. CDSSP MEMORY FOOTPRINT

	Available (Kbytes)	Usage (Kbytes)	Usage (%)
Flash Memory	128.0	28.2	22.03
EEPROM	4.0	1.8	45.00
RAM	8.0	5.2	65.00

In the same way, the constrained device processing capabilities are too constrained to calculate public keys, even using ECC criptography. For instance, this device spent about 2532ms to generate ECDSA messages (by using the Relic library [22]). This processing time may be identified as a packet loss, generating a message retransmission and increasing the power consumption of nodes. With the proposed architecture, the processing is transferred to the IoTSSP removing this overhead from the constrained device.

C. Performance

In this point, we analyzed the network latency introduced by performing a DTLS handshaking. We measured the time from the beginning of the handshaking (the first *ClientHello* message) until the *Finished* message received by the Internet device. Experiments were repeated 30 times for each device. Figure 4 shows the average time to perform the handshaking comparing the CDSSP, the DCert, and the CDPSK. Each bar shows the average and the standard deviation over these measurements. Due to low transmission rates of such network, we configured all devices with two seconds of retransmission latency. The obtained values shows that the proposed architecture (CDSSP) establishes a DTLS connection with mutual authentication in a time about 26% bigger than a traditional DTLS PSK connection (CDPSK). It is expected, since the proposed architecture executes the Transfer of Session mechanism and needs both to communicate with the IoTSSP, and to validate certificates, to establish a new handshaking. However, it is acceptable since enables a communication between the constrained device and the Internet device with assymetric keys.

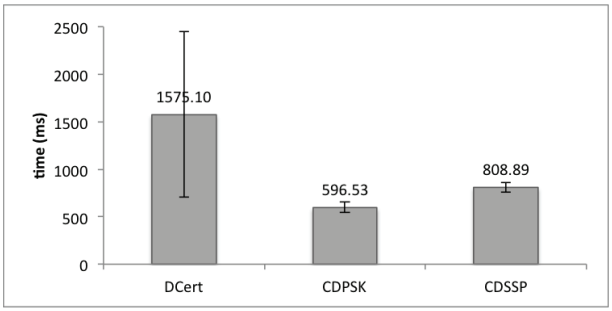

Figure 4. Handshaking Latency for Different DTLS Configurations

The number of message fragments traveling through the 6LoWPAN domain impacts in the results of the previous evaluation. The package payload is a scarse resource in an IEEE 802.15.4 network. This protocol provides only 102 bytes of payload and 25 bytes for link layer addressing, assuming that the security is supported only by DTLS. This way, optimize the protocol headers and the control information is an essential requirement. Moreover, the DTLS fragmentation requires buffers to control these fragments. If a fragment is lost, the entire message is retransmitted, i.e., the network latency is increased. In this sense, it is also important to minimize the number of handshaking messages in the 6LoWPAN Domain, reducing the occurrence of retransmissions. Figure 5 depicts the number of handshaking fragments in the 802.15.4 network, for each device. This behaviour explains the deviation displayed in Figure 4 when a constrained device tries to establish a traditional DTLS with mutual authentication (DCert). Although with resources enough to execute entire DTLS handshaking, the DCert needs to exchange Certificate messages through the 802.15.4 network. These messages are very fragmented, increasing the occurence of packet loss and retransmissions. We highlights that the use of architecture reduces the number of handshaking fragments in 70.20%, when compared to traditional approaches that use DTLS in Certificates mode.

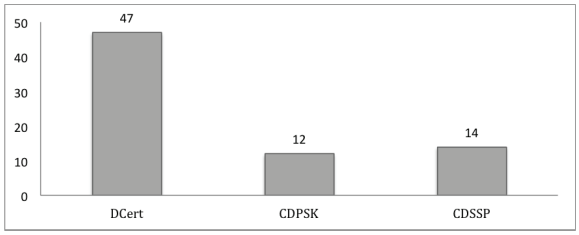

Figure 5. Number of Handshaking 6LoWPAN Fragments

D. Efficiency

In this point, we verified the energy consumption to establish a secure session with mutual authentication. This metric is very important, once it impacts directly on the lifetime of the constrained device.

In order to conduct the experiments, we placed a resistor in series with the battery of the constrained device. By doing that,

we measure the voltage during the entire DTLS handshaking process in two ways: *(i)* without the proposed architecture, operating in PSK mode (CDPSK); and *(ii)* with the proposed architecture (CDSSP). Table II shows the obtained results.

Table II. ENERGY CONSUMPTION

Approach	Energy Consumption
CDPSK	370.8 mJ
CDSSP	397.5 mJ

The CDSSP uses 397.5mJ to execute the DTLS with certificates handshaking, while the CDPSK requires 370.8mJ to establish a DTLS in PSK mode. We understand that the increase of around 8% in energy consumption during the handshaking process is pretty acceptable to enable a DTLS with mutual authentication in a constrained device.

VI. CONCLUSIONS AND FUTURE WORK

The use of DTLS with certificates may be prohibitive for several constrained devices. Moreover, the infrastructure heterogeneity demands an architecture able to work with different types of devices.

In this sense, the proposed architecture introduces a set of mechanisms that enables constrained devices to implement the DTLS based on certificates with mutual authentication. Based on an experimental evaluation, we observe that the proposed architecture provides suitable results in terms of feasibility, performance, and efficiency. For instance, the energy consumption is increased only in 8% when compared with DTLS in PSK mode.

Differently from other proposals, our mechanisms are transparent to devices with resources enough to implement the entire DTLS protocol. Moreover, the architecture minimizes the number of handshaking fragments in 70.20% when compared to traditional approaches that use DTLS in Certificates mode, minimizing the occurrence of message retransmission in the 802.15.4 network domain. Moreover, our architecture does not overload the 6LoWPAN Border Router with security processes. When the number of constrained devices increases, it is possible to insert new IoTSSPs, distributing the security management of constrained devices and increasing the architecture scalability. Finally, the architecture mitigates the risk of Denial of Service attacks to the constrained devices.

As future work, we will analyze the compression of DTLS messages sent to the constrained devices as well as mechanisms to enable the session resumption. Additionally, we will investigate Access Control mechanisms specifics for the proposed architecture.

REFERENCES

[1] C. Bormann, M. Ersue, and A. Keranen, "Terminology for Constrained-Node Networks," RFC 7228, Internet Engineering Task Force, May 2014.

[2] T. Kothmayr, C. Schmitt, W. Hu, M. Brünig, and G. Carle, "Dtls based security and two-way authentication for the internet of things," *Ad Hoc Networks*, vol. 11, no. 8, pp. 2710 – 2723, 2013.

[3] M. A. S. Jr., P. S. Barreto, C. B. Margi, and T. C. Carvalho, "A survey on key management mechanisms for distributed wireless sensor networks," *Computer Networks*, vol. 54, no. 15, pp. 2591 – 2612, 2010.

[4] Z. Shelby, K. Hartke, and C. Bormann, "The constrained application protocol (coap)," RFC 7252, Internet Engineering Task Force, Jun. 2014.

[5] H. Tschofenig and T. Fossati, "A tls/dtls 1.2 profile for the internet of things," Working Draft, IETF Secretariat, Internet-Draft draft-ietf-dice-profile-09, January 2015, http://www.ietf.org/internet-drafts/draft-ietf-dice-profile-09.txt. [Online]. Available: http://www.ietf.org/internet-drafts/draft-ietf-dice-profile-09.txt

[6] R. Roman, C. Alcaraz, J. Lopez, and N. Sklavos, "Key management systems for sensor networks in the context of the internet of things," *Computers and Electrical Engineering*, vol. 37, no. 2, pp. 147 – 159, 2011, modern Trends in Applied Security: Architectures, Implementations and Applications.

[7] M. Brachmann, S. L. Keoh, O. Morchon, and S. Kumar, "End-to-end transport security in the ip-based internet of things," in *Computer Communications and Networks (ICCCN), 2012 21st International Conference on*, July 2012, pp. 1–5.

[8] 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.

[9] A. Wander, N. Gura, H. Eberle, V. Gupta, and S. Shantz, "Energy analysis of public-key cryptography for wireless sensor networks," in *Pervasive Computing and Communications, 2005. PerCom 2005. Third IEEE International Conference on*, March 2005, pp. 324–328.

[10] M. Sethi, J. Arkko, and A. Keranen, "End-to-end security for sleepy smart object networks," in *Local Computer Networks Workshops (LCN Workshops), 2012 IEEE 37th Conference on*, Oct 2012, pp. 964–972.

[11] L. Marin, A. Jara, and A. Skarmeta, "Shifting primes: Extension of pseudo-mersenne primes to optimize ecc for msp430-based future internet of things devices," in *Availability, Reliability and Security for Business, Enterprise and Health Information Systems*, ser. Lecture Notes in Computer Science, A. Tjoa, G. Quirchmayr, I. You, and L. Xu, Eds., vol. 6908. Springer Berlin Heidelberg, 2011, pp. 205–219. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-23300-5_16

[12] L. Marin, A. Jara, and A. S. Gomez, "Shifting primes: Optimizing elliptic curve cryptography for 16-bit devices without hardware multiplier," *Mathematical and Computer Modelling*, vol. 58, no. 5-6, pp. 1155 – 1174, 2013.

[13] V. Gupta, M. Millard, S. Fung, Y. Zhu, N. Gura, H. Eberle, and S. Shantz, "Sizzle: a standards-based end-to-end security architecture for the embedded internet," in *Pervasive Computing and Communications, 2005. PerCom 2005. Third IEEE International Conference on*, March 2005, pp. 247–256.

[14] W. Jung, S. Hong, M. Ha, Y.-J. Kim, and D. Kim, "Ssl-based lightweight security of ip-based wireless sensor networks," in *Advanced Information Networking and Applications Workshops, 2009. WAINA '09. International Conference on*, May 2009, pp. 1112–1117.

[15] J. Granjal, E. Monteiro, and J. Silva, "On the effectiveness of end-to-end security for internet-integrated sensing applications," in *Green Computing and Communications (GreenCom), 2012 IEEE International Conference on*, Nov 2012, pp. 87–93.

[16] H. Yu, J. He, T. Zhang, P. Xiao, and Y. Zhang, "Enabling end-to-end secure communication between wireless sensor networks and the internet," *World Wide Web*, vol. 16, no. 4, pp. 515–540, 2013.

[17] J. Granjal, E. Monteiro, and J. Sa Silva, "End-to-end transport-layer security for internet-integrated sensing applications with mutual and delegated ecc public-key authentication," in *IFIP Networking Conference, 2013*, May 2013, pp. 1–9.

[18] R. Hummen, H. Shafagh, S. Raza, T. Voig, and K. Wehrle, "Delegation-based authentication and authorization for the ip-based internet of things," in *Sensing, Communication, and Networking (SECON), 2014 Eleventh Annual IEEE International Conference on*, June 2014, pp. 284–292.

[19] S. Blake-Wilson, N. Bolyard, V. Gupta, C. Hawk, and B. Moeller, "Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS)," RFC 4492, Internet Engineering Task Force, May 2006.

[20] J. Salowey, H. Zhou, P. Eronen, and H. Tschofenig, "Transport Layer Security (TLS) Session Resumption without Server-Side State," RFC 5077, Internet Engineering Task Force, Jan. 2008.

[21] R. Navas, B. Gaultier, and L. Toutain, "Arduino pico ipv6 stack," https://github.com/telecombretagne/Arduino-pIPv6Stack.

[22] D. F. Aranha and C. P. L. Gouvêa, "RELIC is an Efficient LIbrary for Cryptography," http://code.google.com/p/relic-toolkit/.