Innovative TLS/DTLS Security Modules for IoT Applications: Concepts and Experiments

Pascal Urien(✉)

Telecom ParisTech, Paris Saclay University,
23 avenue d'italie, 75015 Paris, France
Pascal.Urien@Telecom-ParisTech.fr

Abstract. The Internet of Things is a new technological step in the anytime, everywhere, anything IP connectivity context. Things (sensors, wearable objects, connected cars…) are equipped with computers and various communication resources. IoT devices deal with Wireless Local Area Network, Wireless Personal Area Network, Near Field Communication, or new operated radio networks with low throughput such as SIGFOX or LoRA. In this context security and trust are very critical topics, both for users and service providers. In this paper we present new and innovative security modules based on ISO7816 chips, which have been recently introduced by an IETF draft. These low cost, low power, tamper resistant devices, run TLS and DTLS stacks. DTLS is the datagram adaptation of the well known TLS protocol, which is de facto standard for the internet security. It is the security layer of the Constrained Application Protocol (CoAP) targeting sensors networks in a context of smart energy and building automation. We shortly recall TLS and DTLS features, and introduce the flights concept. We present the TLS/DTLS security module interface, which is based on previous work dealing with the EAP-TLS protocol, widely used for authentication in wireless networks and VPNs. We describe our prototype platform based on a java framework that implement a software bridge with the TLS/DTLS security module and which is compatible with the popular Raspberry Pi board. Finally we detail the experimental performances, compatible with the constraints of IoT, observed for an implementation running in a javacard.

Keywords: IoT · Security · TLS · DTLS · Secure element

1 Introduction

The internet of things is a new technological step in the anytime, everywhere, anything IP connectivity context. Things (sensors, wearable objects, connected cars…) are equipped with computers and various communications resources. IoT devices could deal with Wireless LAN (such as legacy IPv4, 6LoWPAN), Wireless Personal Area Network (WPAN, Zigbee, Bluetooth Low Power…) Near Field Communication [3] (NFC), or new operated radio networks with low throughput such as SIGFOX [1] or LoRA [2]. Even in the case of operated networks whose access control is usually managed by symmetric credentials (for example the LoRA network security is based on device AES

B. Mandler et al. (Eds.) IoT 360° 2015, Part I, LNICST 169, pp. 3–15, 2016.
DOI: 10.1007/978-3-319-47063-4_1

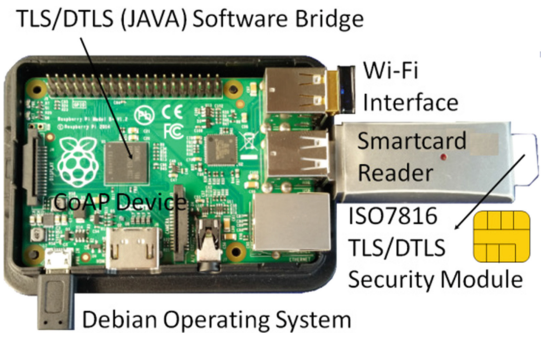

Fig. 1. Illustration of a TLS/DTLS security module in a Raspberry Pi hardware platform

keys, named Application Key - *AppKey*) there is a need to enforce a strong security for information exchange between objects and their authorized data aggregator.

As an illustration according to [4] "the Constrained Application Protocol (CoAP) is a specialized web transfer protocol for use with constrained nodes and constrained (e.g., low-power, lossy) networks. The nodes often have 8-bit microcontrollers with small amounts of ROM and RAM". CoAP messaging model provides reliable (i.e. acknowledged) and unreliable transport mechanisms.

CoAP targets sensors networks in a context of smart energy and building automation. Its security is enforced by the DTLS [7] protocol, working over the UDP datagram layer.

The DTLS protocol [7, 9] is an adaptation of the TLS [5, 6, 8] protocol, which is the de facto standard for secure information exchange over the Internet. TLS works over a reliable transport mode (TCP), while DTLS is designed in order to deal with (UDP) packets lost.

The DTLS/TLS protocols could enable identity models for IoT nodes based on *Public Keys Infrastructures* (PKI), i.e. these devices could embed X509 certificates performing strong mutual authentication with remote entities.

In this paper we define and test TLS/DTLS security modules dedicated to IoT platforms. These devices have been recently detailed by an IETF draft [10] (Fig. 1).

In this context the security module is an ISO7816 compliant and low cost chip. It fully processes the TLS and DTLS protocols, without any IP flavors (i.e. no IP resources such as TCP or UDP are available in the die), in a trustworthy computing environment.

For example the popular Raspberry Pi [11] platform based on a Debian Linux distribution, natively supports interfaces to ISO7816 chips thanks to the *pcsc-lite* library.

The basic idea behind the TLS/DTLS security module is to provide secure channels to objects, which are setup after a strong mutual authentication with remote servers. Furthermore many networks use protocols based on TLS (for example EAP-TLS [13]) for authentication and access control purposes.

This paper is constructed according to the following outline. Section 2 recalls TCP and DTLS protocols main features. Section 3 presents EAP-TLS, a standard used for authentication, based on TLS, which transports TLS without TCP/IP flavors. We introduce a similar but not standardized protocol, EAP-DTLS transporting DTLS

without UDP/IP flavors. Section 4 introduces the TLS/DTLS security modules and their associated software bridges. Section 5 details the prototype platform, the basic cryptographic figures of the ISO7816 security module, and experimental performances. Finally Sect. 6 concludes this paper.

2 About TLS and DTLS

This section briefly recall TCP and DTLS protocols main features.

2.1 TLS

TLS is built over five logical blocks, the Record protocol, the Handshake protocol, the Alert protocol, the Change Cipher Spec (CCS) protocol and the Application data layer.

All TLS data are transported in Record packets, including a five bytes header in clear text, and a payload, which may be encrypted and HMACed.

The Handshake entity manages the booting of TLS sessions, it performs the key exchange operations according to anonymous, one way, or mutual authentication procedures. Ephemeral session keys are generated for record packets privacy and integrity.

The CCS entity generates a message that notifies the switching of the record protocol from clear text to encrypted and HMACed payload.

The Alert entity notifies particular events such as protocols errors or end of TLS sessions.

The application data layer, for example COAP or HTTP, delivers information transported by the record protocol.

A TLS session occurs in two logical stages, first a cryptographic context is negotiated by the handshake protocol, second encrypted data are exchanged over the record protocol.

During the setup of a session TLS messages are grouped into a series of flights, (four for the TLS full mode, and three for the TLS Session Resumption), as illustrated by Fig. 2.

TLS flights are the key concept for DTLS design; there are also the cornerstone for the definition of TLS/DTLS security modules.

2.2 DTLS

The DTLS 1.0 [7] protocol is based on TLS 1.1 [6], while DTLS 1.2 [9] is based on TLS 1.2 [8].

The DTLS protocol [12] provides three new features to TLS, in order to be compatible with a datagram and unreliable transport layer:

- A segmentation/reassembly service for the Handshake entity (see Fig. 3).
- A modified record protocol header (see Fig. 3), including a sequence number used both for clear and ciphered operations.

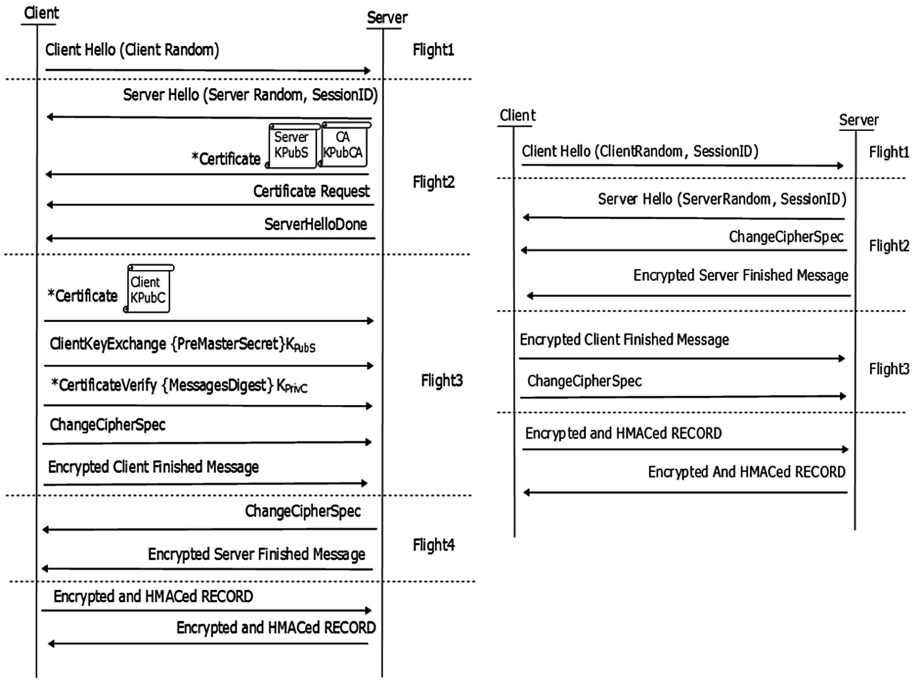

Fig. 2. TLS fights for full and abbreviated mode.

Handshake Message	
Type	1B
Length	3B
Message Sequence	2B
Fragment Offset	3B
Fragment Length	3B
Total length	12B

Record Packet	
Type	1B
Version	2B
Epoch	2B
Sequence Number	6B
Length	2B
Total Length	15B

Fig. 3. Structure of DTLS handshake messages and DTLS record packets

– Two new optional flights, DTLS-HelloVerifyRequest and DTLS-ClientHello with cookie. They manage a cookie mechanism in order to prevent some denial of service attacks. The server delivers a cookie that is thereafter included in the next ClientHello message.

Handshake cryptographic calculations are insensitive to fragmentation operations. According to [7, 12] finished messages (either client or server) have no sensitivity to fragmentation. There are computed as if each handshake message had been sent as a single fragment, i.e. with *Fragment-Length* set to Length, and *Fragment-Offset* set to zero (see Fig. 3); the *Message-Sequence* field is not used in these procedures. It also should be noticed that the DTLS-HelloVerifyRequest message and the previous

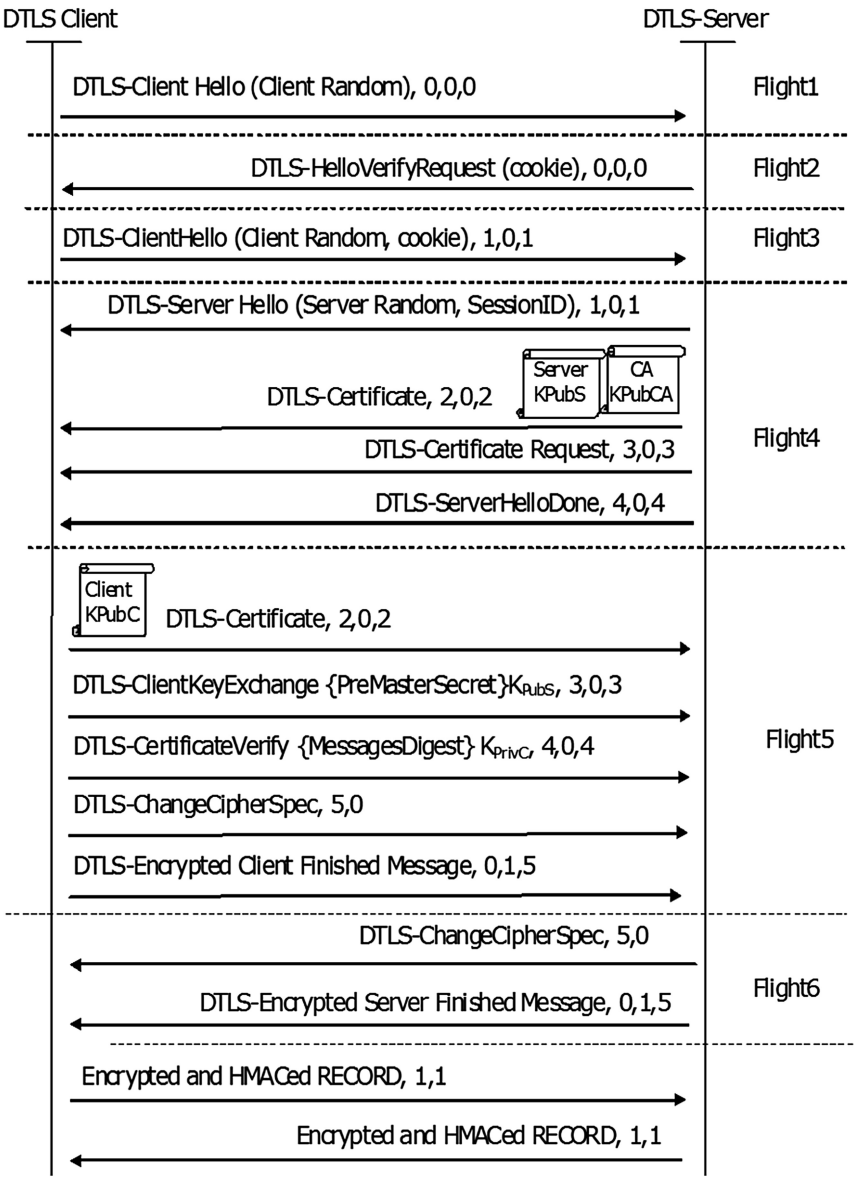

Fig. 4. DTLS flights, in the full mode

associated DTLS-ClientHello are not taken into account by the Handshake cryptographic calculation.

According to [7, 12] the DTLS HMAC computed by the Record protocol is the same as that of TLS 1.1. However, rather than using TLS implicit sequence number, the sequence number used to compute the MAC is the 64-bit value formed by

concatenating the epoch and the sequence number in the order they appear on the wire. TLS MAC calculation is parameterized on the protocol version number, which, in the case of DTLS, is the on-the-wire version, i.e., {254, 255} for DTLS 1.0.

The Fig. 4 illustrates the setup of a DTLS full session, in which both end entities (Client and Server) are equipped with certificates and private keys. No segmentation/reassembly operations are performed by Handshake layers. Each message is transported by a record packet. The two first number are respectively the record sequence number and the *epoch* field. The optional third number is the message sequence used by a handshake message. The *epoch* field indicates the number of delivered CCS packets. A session is started by the DTLS-ClientHello message (flight 1), it is opened after the exchange of DTLS finished messages; these latter are the first encrypted record packets, the sequence number is reset and the *epoch* attribute is set to one.

3 From EAP-TLS to EAP-DTLS

This section presents EAP-TLS [13], a standard used for authentication based on TLS, which enables the use of TLS without TCP/IP flavors. It introduces a similar but not standardized protocol, EAP-DTLS for the use of DTLS without UDP/IP flavors.

3.1 EAP-TLS

EAP-TLS [13] is an authentication protocol widely used in Wi-Fi networks (IEEE 802.11i), for VPN setup (IKEv2, PPTP) or in broadband over power line networks (such as IEEE Std 1901).

EAP-TLS packets are transported by EAP (*Extensible Authentication Protocol*) messages, according to a classical request and response scheme. EAP-TLS provides a transparent encapsulation of TLS (see Fig. 6) until the exchange of finished messages, both for server and client. It supports segmentation and reassembly operations managed via the "Flags" byte, which is detailed by Fig. 5.

- The L bit (length included) is set to indicate the presence of the four-octet TLS flight length field, and is set for the first fragment of a fragmented TLS message or set of messages.
- The M bit (more fragments) is set on all but the last fragment.
- The S bit (EAP-TLS start) is set in an EAP-TLS Start message.

b0	b1	b2	b3	b4	b5	b6	b7
L	M	S	R	R	R	R	R

Fig. 5. The EAP-TLS flags byte.

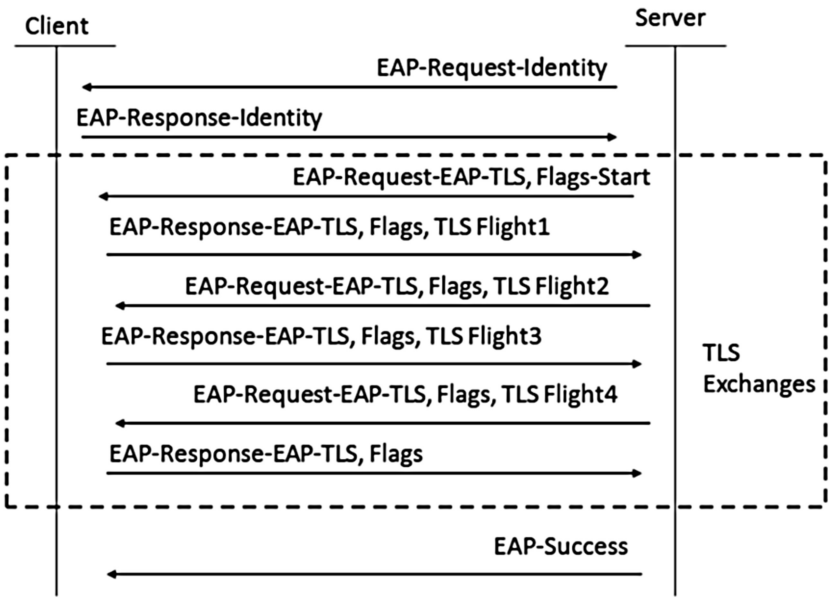

Fig. 6. EAP-TLS choreography for the transport of TLS flights

When an EAP-TLS peer receives an EAP-Request packet with the M bit set, it responds with an EAP-Response with EAP-Type = EAP-TLS and no data. This serves as a fragment acknowledgment (ACK).

Although not defined/used by the EAP-TLS protocol, decryption and encryption of record packets may be provided by dedicated EAP-Request and EAP-Response messages. This principle is used by the TLS/DTLS security module.

3.2 EAP-DTLS

The non standardized EAP-DTLS is very similar to EAP-TLS, excepted that it transports DTLS flights in spite of TLS flights.

Figure 7 illustrates the boot of a DTLS session (as described in Sect. 2.2) including six flights. The EAP-Request-Identity and EAP-Success are omitted. Two new EAP-Request and EAP-Response commands, detailed in Sect. 4 performed the following operations:

- Generation of encrypted and HMACed record packet from clear *Application Data*.
- Recovery of clear *Application Data* from encrypted and HMACed record packet.

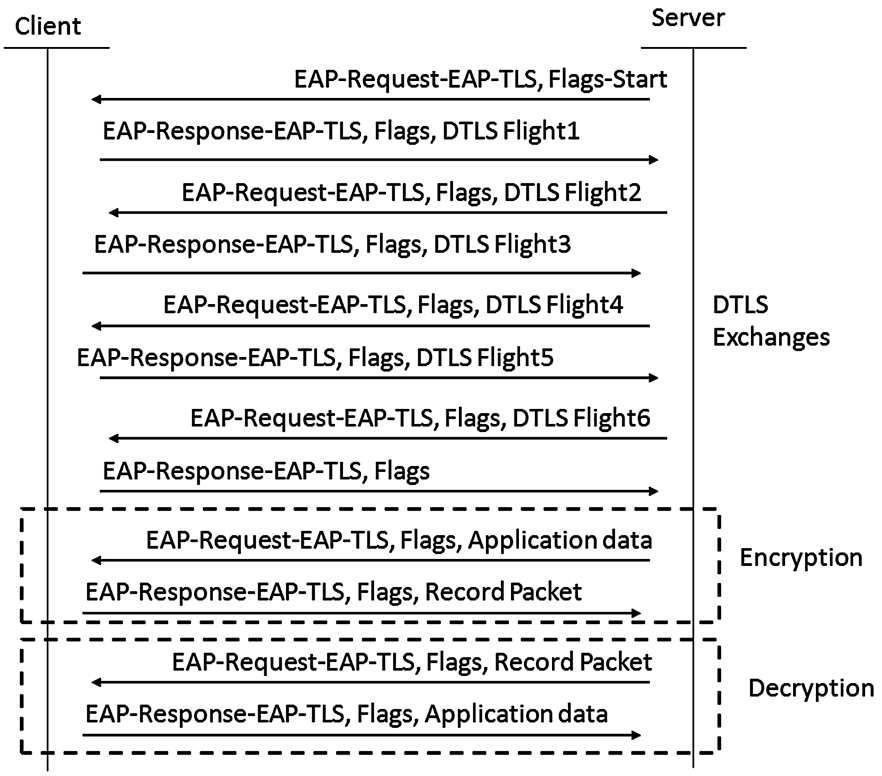

Fig. 7. EAP-DTLS: re-use of EAP-TLS for the transport of DTLS flights

4 TLS and DTLS Security Modules

TLS/DTLS security modules are secure elements that run TLS/DTLS. They are based on EAP-TLS stacks for smartcard [14, 15] designed for trustworthy computing of the EAP-TLS protocol.

4.1 About Secure Elements

Secure Elements [16, 17] are tamper resistant microcontrollers, whose security is enforced by multiple hardware and software countermeasures. Their security level is ranked by evaluations performed according to the Common Criteria standards, whose level range from one to seven. The chip area is typically 25 mm^2 (5 mm × 5 mm). The power consumption is low [18], as an illustration for SIM module 1.8 V–0,2 mA (3.6 mw) in idle state and no more than 1.8 V–60 mA (108 mW) in pike activity.

According to ISO7816 standards, secure elements exchange messages, over serial or USB IO links, whose maximum size is about 256 bytes, and which are named APDU. Request comprises a five byte header (CLA INS P1 P1 P3), the fifth byte (P3)

indicating either the length of outgoing data or the length of incoming data. Response comprises up to 256 bytes and a two byte word status (SW1, SW2).

Secure microcontrollers comprise a few hundred KB of ROM, about one hundred KB of non volatile memory (E^2PROM, Flash) and a few KB of RAM. Most of them include a Java Virtual Machine and therefore run applications written in the Javacard language, a subset of the java language.

A TLS/DTLS stack is an application, typically a javacard application, stored and executed in a secure element. Its logical interface is a set of APDUs exchanged over the IO link.

We previously designed EAP-TLS smartcards, which compute TLS flights encapsulated in EAP-TLS messages, until the generation of server and client finished messages. A full EAP-TLS exchange is detailed in [10], Sect. 17 Annex 6. EAP-TLS devices are identity oriented, i.e. they may store different PKI profiles (Certification Authority, client certificate and associated private keys…).

TLS/DTLS security modules extend from EAP-TLS smartcards [14, 15]. EAP-TLS devices support a double fragmentation mechanism, the size of an EAP fragment is about thousand bytes, which is thereafter segmented in several ISO7816 APDUs. TLS/DTLS devices only deal with small EAP fragments, whose size is about 200 bytes.

Two main ISO7816 commands are used by the security module:

- RESET (CLA = xx, INS = 19, P1 = 10, P2 = 00, P3 = 00), resets the DTLS/TLS session state machine
- Process-EAP (CLA = xx, INS = 80, P1 = 00, P2, P3 = LC) forwards an EAP-TLS packet and returns an EAP-TLS packet (either an acknowledgement or an EAP fragment)

A TLS/DTLS session always starts by the RESET request. Afterwards an EAP-TLS request, with the start indicator set, is sent whose induced response contains the TLS/DTLS ClientHello. TLS/DTLS flights exchanges are performed until the reception of TLS/DTLS client and server finished messages.

At this step the security module is ready to produce encrypted TLS/DTLS record packets or to check and decrypt ciphered record packets.

- The Process-EAP request with P2 set to (80 h or Type) is used to generate a TLS/DTLS record with a given type (see Fig. 8).
- The Process-EAP request with P2 set to zero is used for integrity checking and decryption of a ciphered record packet (see Fig. 9).

Process-EAP, type=17h, 97h= 80h or 17h, payload = 313233340D0A ("1234CrLf")
>> *A08000970C* 0111000C 0D00 313233340D0A
Encrypted TLS Record packet in EAP-Response
<< 0211002F 0D8000000025
1703010020 1506B77D1F1F3514A8E703CAEB2EFEFD045A71E3F68
92AF0C09C79197F7C2E6 *9000*

Fig. 8. Encryption of a clear text ("1234CrLf"), associated to a TLS protocol (type = 17 h). ISO7816 headers are in italics. EAP-TLS headers are underlined. The record packet is in bold.

Process-EAP-Decrypt
>> *A080000043* 01140043 0D00 **15FFF00010000000000000020030 6B4A48869288953CD90D7BCD9E947B93025C75FEC1253 E5 B0D998D1306A33D3612CDF91B230BCE6E55E1B19F39 18FA10**
DTLS Record Clear Payload in EAP-Response= 0100h
<< 021400C 0D8000000002 0100 *9000*

Fig. 9. Decryption of a record layer packet. ISO7816 headers are in italics. EAP-TLS headers are underlined. The record packet is in bold. The return clear text is 0100 h.

4.2 TLS Security Module

TLS security modules are managed by a TLS software bridge (see Fig. 10). This entity starts TLS sessions, performs TLS flights sending and receiving operations over TCP/IP. It exchanges TLS packets with the device encapsulated in EAP-TLS messages. It also manages the interface with the application that needs a secure access to the network.

DTLS Security Module. DTLS security modules are managed by a DTLS software bridge (see Fig. 11). This entity starts DTLS sessions, performs DTLS flights sending and receiving operations over UDP/IP. It exchanges DTLS packets with the DTLS device, encapsulated in EAP-TLS messages. It performs segmentation and reassembly of handshake messages, according to network requirements. DTLS handshake messages exchanged with the device are not fragmented; the EAP layer is in charge of the fragmentation required by ISO7816 constraints. It also provides the interface with the application, such a sensor, which needs a secure access to the network.

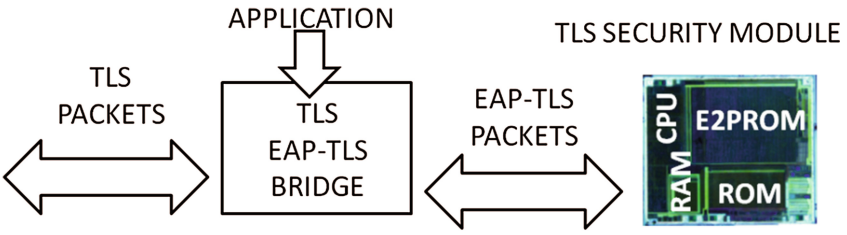

Fig. 10. A TLS security module and its associated software bridge.

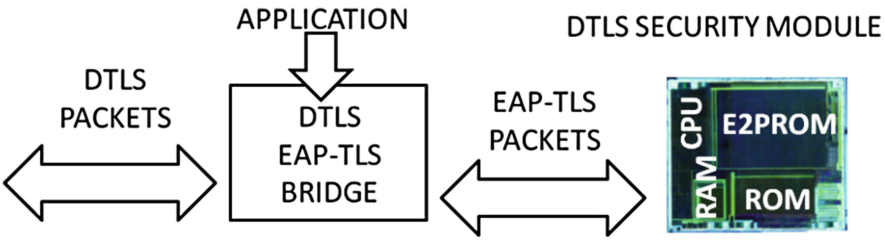

Fig. 11. A DTLS security module and its associated software bridge

5 Performances

5.1 Experimental Platform

We implemented the TLS/DTLS application on a TOP-IM_GX4 [21] javacard [20] manufactured by the GEMALTO company. The cipher suite is AES128 and SHA1. The application size is about 25 KB.

DTLS/TLS software bridges are written in Java and run in a java environment. DTLS and TLS server are based on the popular OPENSSL tool [21], which supports TLS 1.0, TLS 1.1 and DTLS 1.0.

The cryptographic module is based on the Samsung S3CC9TC chip. It includes:

- a 16 bits CPU
- 72 KB of EEPROM
- 384 KB of ROM
- 8 KB of RAM for the CPU
- 2 KB of RAM for the crypto processor

The chip manages various security sensors (glitch, temperature, voltage…) and hardware protections (bus scramble, shield, MMU) and includes a crypto processor for triple DES and PKI computing.

The DTLS/TLS application works with EAP-TLS packets whose maximum size is 128 bytes. It comprises X509 certificates dealing with 1024 bits RSA keys (both client and server and authenticated by their certificate), and uses AES and SHA1 algorithms for the ciphered and HMACed record layer.

5.2 Basic Parameters

Cryptographic performances are illustrated by Fig. 12. According to these figures the processing of encrypted record packets, with a 1024 bytes size, should require about 143 ms, according to the following relations:

- 135 ms (64 × 2,1) for the encryption/decryption of 64 blocks of data.
- 18 ms (20 × 0,9) for the HMAC (SHA1) processing of 20 (16 + 4) blocks of data

The booting of a TLS/DTLS session (until the delivering of finished messages) should cost about 878 ms consumed by the following operations:

- 556 ms for RSA procedures, one RSA private key encryption and two public key decryption (510 + 23 + 24)
- 322 ms for hash procedures, requiring the computing of 230 MD5 et 230 SHA1 blocks.

MD5 ms/block 64B	SHA1 ms/block 64B	3xDES ms/block 8B	AES ms/block 16B	RSA Pub ms 128B	RSA priv ms 128B	IO ms/B
0,50	0,90	1,8	2,1	23	510	0,1

Fig. 12. Basic performances of the TOP-IM_GX4 javacard.

Fig. 13. COAP electronic lock prototype.

The IO attribute (0,1 ms/B) in Fig. 12 is the observed time to send/receive an ISO7816 request/response via a smartcard reader to/from an application running in the secure element. It means that 10 ms are required to transfer 100 bytes to/from the secure module. The experimental results, detailed in the next session, are in concordance with these basic cryptographic parameters.

5.3 Experimental Results

A DTLS/TLS full session is opened in 1400 ms, it is split in two parts 250 ms for information transfer (about 2500 bytes are exchanged, i.e. the throughput is around 0,1 ms/B) and 1150 ms consumed by cryptographic calculations. A DTLS/TLS resumed session is opened in 360 ms, with only 250 bytes of exchanged data. The generation of a 1024 record packet costs 400 ms, 240 ms are consumed by IO operations and 160 ms by cryptographic calculations. The checking and decryption of a 1024 record packet costs 430 ms, 240 ms are consumed by IO operations and 190 ms by cryptographic calculations.

6 Conclusion

In this paper we introduced the TLS/DTLS security module for Internet of Things applications. Future work could target the popular Raspberry Pi platforms, that are powered by an open operating system based on the Debian Linux distribution. These devices are natively compatible with secure element, thanks to the pcsc-lite library. They may interface DTLS/TLS security modules by several programming environments such as C language, Java, or Python. As illustrated by Fig. 13, we are currently working on a prototype dealing with COAP [4] electronic lock, in which two TLS/DTLS javacard applications running in a SIM card, manage secure key downloading and secure operations with an electronic lock.

References

1. LoRa Alliance: LoRaWAN™ Specification, Version: V1.0, January 2015
2. SigFox: One network A billion dreams: M2 M and IoT redefined through cost effective and energy optimized connectivity. White paper (2015)

3. ISO/IEC 18092: Near Field Communication - Interface and Protocol (NFCIP-1), April 2004
4. Shelby, Z., Hartke, K., Bormann, C.: The Constrained Application Protocol (CoAP). RFC 7252, June 2014
5. Dierks, T., Allen, C.: The TLS Protocol Version 1.0. RFC 2246, January 1999
6. Dierks, T., Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.1. RFC 4346, April 2006
7. Rescorla, E., Modadugu, N.: Datagram Transport Layer Security. RFC 4347, April 2006
8. Dierks, T., Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246, August 2008
9. Rescorla, E., Modadugu, N.: Datagram Transport Layer Security Version 1.2. RFC 6347, January 2012
10. TLS and DTLS Security Modules, draft-urien-uta-tls-dtls-security-module-00.txt, June 2015
11. https://www.raspberrypi.org/
12. Modadugu, N., Rescorla, E.: The design and implementation of datagram TLS. In: The 11th Annual Network and Distributed System Security Symposium, San Diego, CA, USA, February 2004
13. Simon, D., Aboba, B., Hurst, R.: The EAP-TLS Authentication Protocol. RFC 5216, March 2008
14. Urien, P.: EAP Support in Smartcard, draft-urien-eap-smartcard-29.txt, July 2015
15. Urien, P.: Collaboration of SSL smart cards within the WEB2 landscape. In: International Symposium on Collaborative Technologies and Systems, CTS 2009, 18–22 May 2009, pp. 187–194 (2009)
16. ISO 7816: Cards Identification - Integrated Circuit Cards with Contacts. The International Organization for Standardization (ISO)
17. Jurgensen, T.M., et al.: Smart Cards: The Developer's Toolkit. Prentice Hall PTR, Upper Saddle River (2002). ISBN 0130937304
18. ETSI: Specification of the 1.8 Volt Subscriber Identity Module - Mobile Equipment (SIM - ME) interface. ETSI TS 101 116 V7.0.1
19. Chen, Z.: Java Card™ Technology for Smart Cards: Architecture and Programmer's (The Java Series). Addison-Wesley, Boston (2002). ISBN 020170329
20. GEMALTO: GemXpresso R4 E36/E72 PK - MultiApp ID 36 K/72 K - TOP IM GX4 Security Policy (2009)
21. Seggelmann, R., Tuexen, M.: DTLS Documentation, version 1.0. http://sctp.fh-muenster.de/index.html