A monitoring-based approach for WSN security using IEEE-802.15.4/6LowPAN and DTLS communication

Raul Armando Fuentes Samaniego, Vinh Hoa La, Ana Rosa Cavalli, Juan Arturo Nolazco-Flores, Raúl Valente Ramirez Velarde

To cite this version:

Raul Armando Fuentes Samaniego, Vinh Hoa La, Ana Rosa Cavalli, Juan Arturo Nolazco-Flores, Raúl Valente Ramirez Velarde. A monitoring-based approach for WSN security using IEEE-802.15.4/6LowPAN and DTLS communication. International journal of autonomous and adaptive communications systems, 2019, 12 (3), pp.218-243. 10.1504/IJAACS.2019.10022471 . hal-02184618

HAL Id: hal-02184618

https://hal.science/hal-02184618

Submitted on 17 Oct 2019

A monitoring based approach for WSN security using IEEE-802.15.4/6LowPAN and DTLS communication

Raul A. Fuentes-Samaniego

Inria Saclay, 1 Rue Honor d'Estienne d'Orves, 91120 Palaiseau
SAMOVAR, CNRS, Telecom SudParis, Paris-Saclay University, 9 rue Charles Fourier 91011 EVRY, France
E-mail: raul.fuentes-samaniego@inria.fr

Vinh Hoa La, Ana R.Cavalli

Montimage, 39 rue Bobillot, 75013 Paris, France
SAMOVAR, CNRS, Telecom SudParis, Paris-Saclay University, 9 rue Charles Fourier 91011 EVRY, France
E-mail: vinh_hoa.la@montimage.com, ana.cavalli@telecom-sudparis.eu

Juan A. Nolazco-Flores, Raul V. Ramirez-Velarde

School of Engineering and Science, Tec de Monterrey, Ave. Eugenio Garza Sada No. 2501, Monterrey, Mexico
E-mail: jnolazco@itesm.mx, rramirez@itesm.mx

Abstract: In this paper, we present a monitoring based approach for securing upper layer communications of WSN (Wireless Sensor Networks), the latter using IEEE802.15.4/6LoWPAN stacks and tinyDTLS. The monitoring techniques have been integrated as an extension to the industrial tool MMT (Montimage Monitoring Tool). The MMT-extension verifies that the network is working following a set of security rules that have been defined by ETSI. The security rules check if the protocol stack is working properly. If MMT detects a security rule that was not respected, then it sends an alarm to the system manager so that he can take properly reactive adjustments. We tested each of the security rules in MMT's extension using point-to-point configuration. After all these tests were verified, we tested our MMT-extension using real data gathered from the FIT IoT-LAB platform. The results of these tests shown that our MMT's extension for WSN using IEEE-802.15.4/6LowPAN and DTLS communication is feasible.

Keywords: WSN, IoT, Security Monitoring, Monitoring Tools, Intrusion Detection, 6LoWPAN.

1 Introduction

The Internet of Things (IoT) is a set of interconnected low-resources nodes connected to the Internet to develop a wide range of applications. IoT can be realized using WSN (Wireless Sensor Networks). Depending on the application, the WSN traffic could be sensitive to security (e.g. health telemetry parameters) and these issues are important because many of WSN have not originally been designed to tackle security risks [2, 24, 26].

Network security goals are authentication, privacy, data confidentiality, data integrity and accessibility. Confidentiality means keeping information secret from unauthorized parties. A sensor network should not leak sensor readings to neighboring networks. Data integrity ensures the user that the received data are not altered in transit by an adversary. And availability is the ability to use resources, such as network interfaces, for transmitting packets at the correct location, time and format.

WSN security threats are diverse. In [3] the threats are classified by layer as follows: at physical layer threats are jamming and tampering; at link layer threats are exhaustion, unfairness and flooding. At network layer threats are spoofing, replayed routing information, selective forwarding, sink holes, sybil, worm holes, hello fool and acknowledgment spoofing and others. At transport layer threats are flooding, desynchronization, and others. Jamming and physical layer tampering are relative easy to detect using signal energy, packet repetitions or inclusive measuring energy use in the node. Collisions, exhaustion and unfairness are also easy to

Table 1 Threats defined by the ETSI for M2M communication [5].

ETSI Threat Number (ET#)	Likelihood	Impact	Risk	Security rules
(ET8) Discover Keys by Eavesdropping on Communications Between Entities.	Substantial	High	Critical	SR1, SR2
(ET15) General Eavesdropping on M2M Service-Layer Messaging Between Entities.	Severe	High	Major	SR3, SR4
(ET16) Alteration of M2M Service-Layer Messaging Between Entities.	Substantial	High	Major	SR3, SR4

detect. However, by modifying protocol rules and packet information, an attacker can deliver attacks such as: data integrity, energy exhaustion and spoofing. Detecting frame changes requires high knowledge of protocol frame syntax and protocol state rules. Therefore, it is important to develop monitoring techniques and tools to detect if security rules are not respected when the frame data is modified or protocol rules are changed.

MMT[a] is a monitoring tool that verifies if the frame syntax under a specific protocol scenario is correct. MMT sends a message to the user if it detects that a security rule cannot be verified. At present, MMT has mechanism to detect rule violations for IPv4, IPv6, TCP and UDP, but does not have support for WSNs.

In this paper, we increase the capability of MMT to include WSN. Since nodes in WSN are limited by low energy, low computational power and low storage space, we have modified MMT for IEEE 802.15.4/6LowPAN and tinyDTLS [7, 8, 14]. IEEE 802.15.4 is used for sensing communication capabilities and 6LowPAN for IPv6 addressing capabilities, which are features needed for IoT.

Moreover, IEEE 802.15.4-2006 has a very good packet transmission control in unreliable environments, as well as a native support for the 6LoWPAN. This standard has also the advantage to be supported by the ZigBee products and open source operating systems such as Contiki, RIOT and TinyOS.

MMT verifies if security rules are not respected under a specific context, which is protocol dependent. In our MMT extension, we work with IEEE 802.15.4/6LowPAN using tinyDTLS as the security mechanism. We check also the security rules proposed by ETSI in [8](Table 1). Indeed, we deal with the challenge of secure IoT environments, including the secure communication between sensors (i.e, Machine to Machine (M2M) communication) with very constrained resources.

This paper addresses crucial challenges regarding security for WSN using IEEE-802.15.4/6LowPAN and DTLS communication. Only very few teams worldwide are investigating and developing techniques to guarantee security for these protocols. In addition, another contribution of this paper is that security rules have been tested in real environments and providing an automatic checking of the protocols security requirements. These rules have been integrated as extensions of an industrial tool called MMT.

The organization of this paper is as follows: In Section 2 we discuss relevant related work. In Section 3, the tool MMT is presented together with its main functions. In Section 4, we present the extensions we developed to enable MMT for WSN monitoring. Section 5 illustrates a concrete monitoring scenario with its experimental results. Finally, in Section 6 we present some conclusions and our vision for future work.

2 Related Works

In [25], the authors presented the results of an analysis of TinyDTLS, only for PSK, over beacon-enabled 802.15.4/6LowPAN networks. Each node of the sensor network is implemented using WisMote sensor kit. The nodes are configured to run on Contiki as Operating System. The problem with this configuration is that enabling the beacon makes the DTLS handshake very slow, and they found that DTLS over radio duty-cycled networks, required up to 50 seconds to complete DTLS handshake process, which can make the system useless for real-purpose applications.

Meanwhile, in [9] the authors evaluated IPsec and DTLS in terms of power and memory consumption. Their main contribution was the proposition of an implementation and evaluation of a custom 6LoWPAN dispatch for integrating IPSec to 6LoWPAN, with support for the Authentication Header (AH) and Encapsulating Security Payload (ESP). Their testbed is composed of two TelosB sensors running TinyOS and a Linux host. The TelosB were assisted with the use of TPM chips. The final conclusion is that the use of IPsec or CoAP with DTLS are viable according to the "most appropriate security mode". Still, the approach with DTLS is better suited for scenarios where foreigner clients request information from the sensors.

The work in [11] is another evaluation of DTLS regarding WSN. This work was one of the first adapting DTLS to 802.15.4/6LoWPAN, using public cryptography keys such as RSA. As a testbed it uses TinyOS 2.x, with an integrated TPM chip and the OpenSSL 1.0.0d for the DTLS stack. The hardware used for the sensor is an Opal Sensor. Power was measured with an oscilloscope using a 10 Ohm resistor. The result reported seems to be in accordance with the work of [9]. As the differences between cipher suites are huge, the CoAP specifications avoids the use of RSA with the sensors. In general, this work validates the use of only the cipher suites already supported by tinyDTLs for WSN.

The work in [19] is the adaptation of DTLS as another type of dispatch for 6LoWPAN called Lithe. Wismote sensors running Contiki OS 2.7 and TinyDTLS 0.3.2 were emulated with Cooja for testing Lithe. The main advantage is the reduction of the overhead on the DTLS packets and consequently in the energy consumption of the nodes that can be of at least 12 bytes. Lithe is a very interesting work as it offers an alternative to the use of DTLS and CoAP. However, their work is difficult to upgrade to the last version of Contiki or even tinyDTLS. Given that only nodes supporting the custom 6LoWPAN dispatches can communicate inside of the WSN Lithe, this implementation is more restrictive than the proposed in this paper.

In [4], an analysis of the use of an Intrusion Detection System (IDS) in WSN is provided. The authors propose that each node has a public and private key pair for signing the messages. They perform home network simulations by means of Castalia WSN simulator together with OMNET simulation package. The objective is to evaluate the life time of the system once some intrusions are introduced. Measurements are performed when no IDS is running, when an IDS is eavesdropping constantly and when a Byzantine solution is being used to detect the intrusion. In this work, one obvious conclusion is that prohibiting the nodes to switch into the sleep mode will exhaust the power supply quickly. The difference between the other two scenarios is that it can reach almost to 55% of energy consumption with the maximum number of nodes in the WSN. However, the energy consumption with the IDS based on the Byzantine solution is still very far from life-time requirements.

The work presented in [1] is based on an European project. It presents a denial of service (DoS) detection architecture for 6LoWPAN based on the IDS Suricata. They focused especially on DoS attacks detection by real-time monitoring of various physical parameters. They integrated to the 6LoWPAN some distributed IDS probes acting as sniffers. These IDS probes have the mission to send relevant information to the IDS (e.g., Suricata) through wired connections. Wired connectivity, which is certainly more reliable, allows the framework to be resistant to attacks. However, WSNs are frequently deployed in inaccessible terrains or even hostile environments. A wired connection in that case is seemingly unrealistic. Our MMT is able to play the same role of Suricata in DEMO framework, without the need of decoders (Suricata does not officially support IEEE 802.15.4 and 6LoWPAN yet and not all the rules defined on Section 4.2 are possible to replicate with Suricata)

Regarding intrusion detection systems for IoT in general and for 6LoWPAN-based WSNs in particular, SVELTE [20] has been presented as the most well-known among very few intrusion detection tools working over such small devices. SVELTE consists of three main centralized modules including lightweight modules and mini-firewalls deployed in SNs and central modules called 6Mapper located in BRs. 6Mapper collects the routing information thanks to their "little" collaborators located in SNs. In comparison with our approach, SVELTE is more active and creates additional traffic to achieve the goal. Whilst, our solution attempts to passively monitor the network based on the network's traffic to avoid additional costs which might hamper 6LoWPAN.

Our approach goes beyond the limitations of the related works mentioned in the previous analysis. Moreover, it must be noted that we propose an approach that has been experimented on real environments and that this approach has been integrated as an extension to an industrial monitoring tool, MMT. The results of the experimentations have shown that our MMT's extension for WSN using IEEE-802.15.4/6LowPAN and DTLS communication is feasible and succesful. Our approach also contributes to reduce the costs of evaluation of security issues.

3 Montimage Monitoring Tool

3.1 Overview

MMT facilitates network performance monitoring and operation troubleshooting. MMT captures online, and offline, network traffic to verify if the traffic activates a security or attack rule. With its advanced rules engine, MMT can correlate network and application events in order to detect performance and operational and security incidents. If MMT detects that a security rule is not respected, then it sends a message to the IoT administrator.

MMT uses Deep Packet/Flow Inspection (DPI/DFI) technique for inspecting the packets. MMT detects an attack incident when an attack rule is validated, and detects a secure state when a security rule is validated [27, 16, 13]. MMT consists of three principal modules, as shown in the Figure 1:

- **MMT-Extract** module allows to monitor different observable application, system, or network protocols. This module extracts information from protocols frame for offline and online traffic (e.g., PCAP files) in the node.

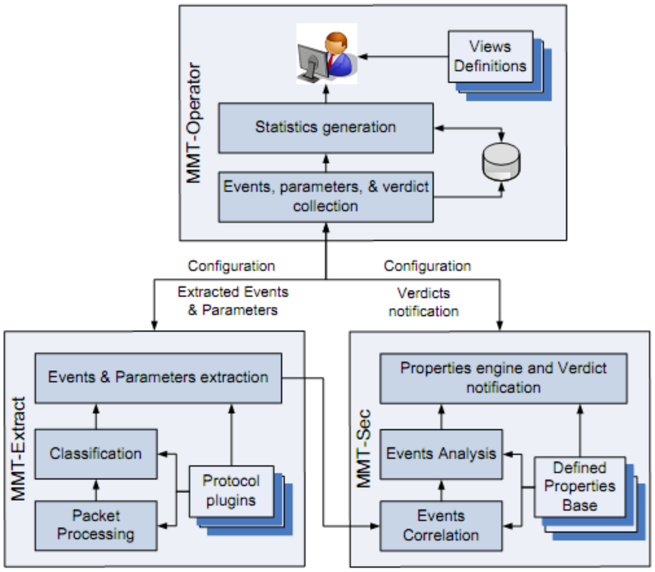


Figure 1: MMT global architecture [27]

- **MMT-Security** contains security rules that refer to both expected and unexpected behaviors. MMT-Security model is inspired from Linear Temporal Logic (LTL). Rules are written in XML, which provides the advantage of simple and straight forward structure verification. A property is an IF- THEN relation, IF $< context >$ THEN IF $< trigger >$ then $property\ is\ satisfied$. The trigger is checked if and only if the context is valid. If the trigger is found valid, then the property is satisfied. Otherwise, the property is violated. Embedded functions can be also added to pre-process the data input before passing to MMT-Security rules.

- **MMT-Operator** collects and aggregates extracted data, generates network and application statistics, and presents them using a graphical user interface.

At present, MMT has mechanisms for IPv4, IPv6, TCP and UDP, but does not have support for WSN. Adding WSN capabilities to MMT is simple because it was designed in such a way that we can add new plugins and modules specifying the structure of the new data input (e.g., the fields of one or more new protocols).

The rest of this section is structured as follow: In Section 3.2 we discuss the main characteristic of all the protocols involved in the infrastructure of the WSN: IEEE 802.15.4, 6LoWPAN, compressed UDP. In Section 3.3 a more detailed discussion is made in relation to the protocols involved in the upper layer as they are the base for the security analysis of the M2M communication. Finally, in Section 3.4 the description of MMT-Security is done.

3.2 MMT-Extract: Extension for 802.15.4/6LoWPAN

In this section, we identify and discuss all the components involved in the design of the proposed infrastructure for a IoT environment and determine which are the key elements to be monitored.

IEEE 802.15.4 is composed of simple unicast and multicast packets. 6LoWPAN can use its source and destiny addresses fields to infer the IPv6 addresses. The field `Field Check Sum` (FCS) is used as the only field to verify the integrity of a packet. These features permit to save resources in the upper protocols.

In its most basic concept, 6LoWPAN aims to compress the IPv6 headers (Figure 2) from their original 40 bytes to a significantly lesser dynamic value. In some cases, the header is compressed to lengths of 3 or 6 bytes [17, 22].

6LoWPAN uses dispatches to achieve compression, which consist of special frames, that precede the IPv6 header and deliver information about the fields of the IPv6 header. Figure 4 shows the standard representation of a dispatch preceding an IPv6 header and its payload.

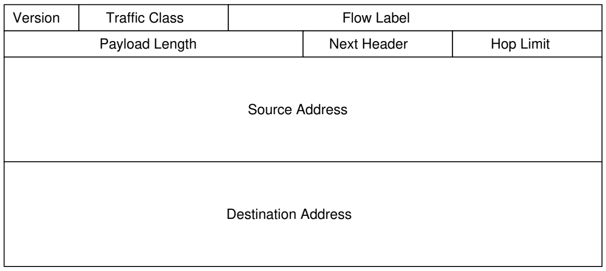


Figure 2: Standard IPv6 header.

Table 2 shows the most common dispatches and for this work the emphasis will be on FRAG1, FRAGN and IPHC (IP Header Compression). These dispatches do not replace IPv6, and the nodes still need to be able to configure their own IPv6 addresses and to know their own scopes.

For some configurations, usually the faulty ones, the dispatch may not be present so the IPv6 header is uncompressed. This will be referred as "raw IPv6" to distinguishing it from the dispatch for uncompressed IPv6. The raw IPv6 packets not only wastes bandwidth, but also the monitoring tools can be misled.

Section 3.2.1 reviews IPHC. Section 3.2.2 reviews the dispatches related to the fragmentation over 6LoWPAN. Finally, Section 3.2.3 reviews NHC (Next Header Compression) for UDP .

3.2.1 6LoWPAN compression

IPHC is a dispatch which consists of seven fields shown in Figure 3 [10]. IPHC can elide or suppress IPv6 address fields from the IPv6 header, saving up to 32 bytes. For

Table 2 List of dispatches headers available for 6LoWPAN [17, p. 8].

Pattern (bits)	Header Type
00 XXXXXX	NALP Not a LoWPAN frame
01 000001	IPv6 Uncompressed IPv6 Addresses
01 000010	LOWPAN_HC1 LOWPAN HC1 compressed IPv6
011X XXXX	LOWPAN_IPHC LOWPAN IPHC compressed IPv6
11 000XXX	FRAG1 Fragmentation Header (first)
11 100XXX	FRAGN Fragmentation Header (subsequent)

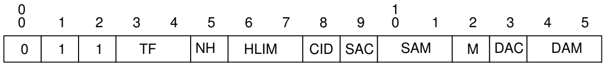


Figure 3: The LoWPAN IPHC Dispatch [10].

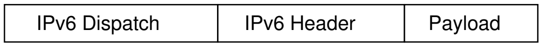


Figure 4: Generic 6LoWPAN Dispatch structure.

multi-cast transmission, IPHC omits transmitting the source address and it can reduce it to either 16, 32 or 48 bits.

For Traffic-class frames, IPHC can infer flow label and Hop-Limit fields based on the values of the fields TF and HLIM. NH (Next header) field will control if a standard or compressed header is following the IPv6 header, such as the compression for UDP.

IPHC always omits `Version` and `Payload Length` IPv6 fields. `Payload Length` is inferred based on the `FCS` field of the IEEE 802.15.4 frame. With a proper configuration, it is possible to compress an IPv6 header from 40 to 3 bytes (2 for IPCH and 1 for NHC).

3.2.2 6LoWPAN fragmentation

MTU (Maximum Transmission Unit) for IEEE 802.15.4 is 127 bytes. Therefore, messages exceeding this value are fragmented, using dispatches FRAG1 and FRAGN. Fragmentation works for compressed and uncompress IPv6 packets. The dispatches are shown at Figure 5 and Figure 6.

FRAG1 dispatch is for the beginning of a new fragmented message, and it uses the ID defined in the field `Datagram_Tag`. It is mandatory for this first message to include all the information concerning the dispatch used in the original message, including the header for the transport layer protocol. If the size of all headers combined exceeds the MTU, then the message to transmit will be the raw IPv6 packet without any type of dispatch preceding it [10].

FRAGN uses the field `Datagram_Offset` to reassemble the fragments of a single message. The field `Datagram_Size` is used in both dispatches to help identify each message and, most importantly, assembling the fragments correctly. The nodes must handle the reception of each FRAGN and calculate if the current amount of frames received fits the datagram size.

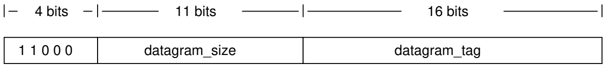


Figure 5: First Fragment (FRAG1).

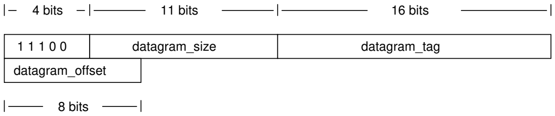


Figure 6: Subsequent Fragments (FRAGN).

It is important to notice that the size that will be in the field `Datagram_Size` will always be the sum of the raw IPv6 header and its payload, even in the case of a packet already compressed. The values in the `Datagram_Offset` field are according to the raw size. This is why the FRAG1 must contain all the possible headers or else a uncompressed IPv6 packet. Otherwise, the nodes are unable to reassemble the rest of the fragments correctly due to the inconsistency with the offset.

3.2.3 Compression mechanism for UDP

The standard UDP header is shown in Figure 8 and the header for NHC in Figure 7. The NHC header is basically made of two variable-length fields, the checksum field and the amount of bits available for port addresses (4, 8 or 16 bits).

Frame compression is achieved when compressing port addresses. By default, the ranges of 0xF0XX and 0xF0BX are used for 8 bits and 4 bits configuration. The sensors must be able to operate accordingly, otherwise, unexpected sensor-error communications may occur.

3.3 MMT-Extract: Extension for DTLS

HTTPS (Hypertext Transfer Protocol Secure) is used for secure M2M communication by using TLS (Transport Layer Secure). For low resources devices, the equivalent of theses protocols are CoAPS (Constrained Application Protocol Secure) and DTLS (Datagram TLS) [23]. Since CoAPS messages are encripted, thus invisible to our monitoring tool, we will monitor DTLS messages.

A DTLS protocol uses connection-oriented services with confirmation, which means that before sending a message a connection must be established. Once the connection is established, the application can configure

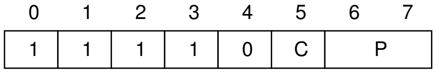


Figure 7: UDP Header Encoding with LoWPAN NHC[10].

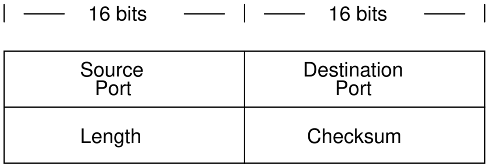


Figure 8: UDP Header.

the time it requires the connection to stay open to send an upper layer message.

The time the connection is open is called DTLS session, which can be configured in one of the following modes [7]:

- `Nosec`. The application messages (CoAP messages) remain in plain text.

- `PreSharedKey` (PSK). All the devices have preloaded symmetric cryptographic keys for all the DTLS sessions. Renewing those keys is outside of DTLS functionality.

- `RawPublicKey` (RPK). This mode allows the use of a pair of asymmetric keys for authentication. However, a third entity is not necessary.

- `Certificate`. This is the equivalent of the infrastructure for HTTPS where the X.509 certificated mode is used. It supports digital certificates and public key encryption.

The four previously mentioned modes can be reached by means of a handshake process in which one or more cipher suites will be enumerated according to the modes that are available to the devices. The handshake is started by the client, with a list of the available cipher suites. The server then will have the final choice, including the possibility to reject the communication.

The full handshake process is challenging for a constrained devices and it will be discussed in two parts: the selection of the cipher suite and the handshake itself.

3.3.1 Identification of the mode by means of cipher suites

Basically, in DTLS what defines each one of the four modes are the cipher suites. Those suites are the stack of four different protocols for establishing a session, sharing secrets and authentication. In DTLS, as in TLS, the client can offer a set of cipher suites list, each one of the ciphers identified with an unique 16-bit ID.

Each cipher suite requires special fields, called “extension”, to comply with specific requirement of the each of the ciphers suites. These fields are highly dynamic in content and size.

3.3.2 Structure of DTLS messages

All the messages related to DTLS are referred as DTLS records and they are classified as handshake, data or alert records. If the MTU has enough space, more than one single DTLS record can be combined inside of a single datagram. In this work, the term jumbo datagram is used for this type of datagrams.

An general architecture of a DTLS record is shown in Figure 9 and is composed by the following fields:

- `Type` - Identify the type of DTLS record received: `ChangeCipherSpec` (0x14), `alert` (0x15), `handshake` (0x16) or `data app` (0x17).

```
struct {
    ContentType type;
    ProtocolVersion version;
    uint16 epoch;
    uint48 sequence_number;
    uint16 length;
    opaque fragment[DTLSPlaintext.length];
  } DTLSPlaintext;
```

Figure 9: DTLS record format [21]

- `Version` - Identifies the current version of DTLS or TLS. The value for DTLS 1.2 is 0xFEFD.

- `Epoch` - New field for DTLS. Helps to identify a DTLS session between the same pair of nodes. Starts at zero and only increases after an `ChangeCipherSpec` record is sent.

- `Sequence Number` - New field for DTLS. Together with `Epoch` helps to guarantee the correct sequence of the record received, mitigating the unreliable nature of UDP.

- `Length` - The length of this DTLS record, including the previous fields. This is used to infer the presence of jumbo datagrams.

- `Fragment` - The body of this record. By example, all the different types of records used in the handshake process.

When two or more records are received with the same `Epoch` and `Sequence Number`, they are duplicated records. If multiples records of consecutive numbers in the `Sequence Number` and same `Type` appears, they are fragmented DTLS records of the same type. Finally, if after processing a specific DTLS record, the size of the UDP datagram does not fit with the `length` field means this is a jumbo datagram, and there is at least another DTLS record available.

The different types of records that compose the handshake are explained in Section 3.3.3 together with the `ChangeCipherSpec` records. The other two types of records are simpler and are already protected by the DTLS session. Alerts are sent when a bad record is received or when the session must be terminated or renegotiated. The `data application` record has the payload of the upper layer, such as the CoAP messages or the request for renegotiating the DTLS session.

3.3.3 DTLS Handshake process

The handshake is divided into six steps which are referred as flights. Flights can be composed of multiple DTLS records. The full process of this handshake is shown in Figure 10 and is described briefly here:

1. The DTLS session is always requested by the client, a `ClientHello` record is used for this flight.

2. The server responds to the client with the HelloVerifyRequest record. This record includes a cookie to mitigate attacks.

3. The client confirms the process with a new ClientHello, this too confirms which version of DTLS is to be used. Also, the client begins to include the cookie.

4. The server answers the client request, selecting one of the cipher suites offered by the client and preparing the exchange of keys. This flight can be composed of multiples DTLS records, but it will always begin with the ServerHello record that carries the cipher suite selected and ends with the ServerHelloDone record.

5. The client will process the selected cipher suite and continue with the process of the exchange of keys with the ClientKeyExchange record. Once the client has processed all its part of this handshake, it will notify the server with the ChangeCipherSpec and the Finished records.

6. The server will use the previous ChangeCipherSpec to finish its part and will notify the client with its owns ChangeCipherSpec and Finished records.

Flights 4 and 5 are dynamic in size due to the extensions that bring the special requirements for the preferred cipher suite of the client and the one selected by the server. Also, cookies are a factor on the dynamic size as their size varies from 0 to 255 bytes.

The Certificate, ServerKeyExchange, CertificateRequest and CertificateVerify records are used in the flights if the cipher suites are using the modes RPK or certificate. In scenarios where the client and server are using different cipher suites the DTLS session cannot be established.

ChangeCipherSpec is not strictly speaking a handshake record. Its function is to indicate that subsequent DTLS records will be protected with a new couple of symmetric keys. Therefore, communications cannot continue until the other part finishes the preparation of the new set of keys. This is indicated by the second ChangeCipherSpec record. The Finished records indicate that the handshake is finished and are already protected by the DTLS session.

3.3.4 TinyDTLS

DTLS is a suitable candidate for securing the M2M communication between the sensors. However, the resources required for establishing a standard DTLS session can be too much for the sensors. A certificate could be more than a single megabyte to transmit and to be processed by the sensors. Even if the PSK mode is used, the size of the keys for the symmetric encryption can drain the power supply of the nodes rather quickly. This is why the specifications of CoAP take in consideration friendlier cipher suites.

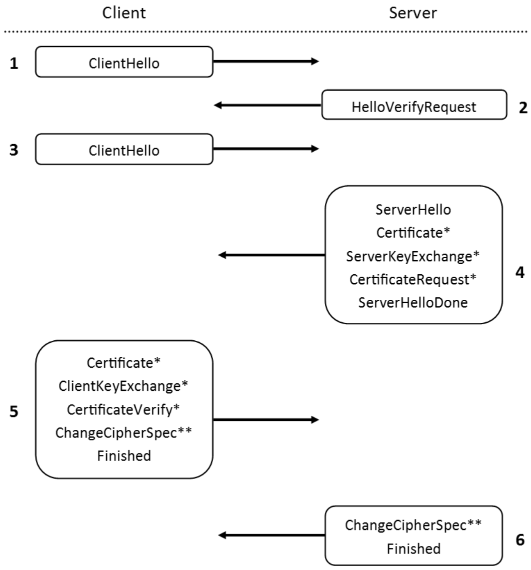


Figure 10: Standard 6 Flights messages for DTLS defined in [21].

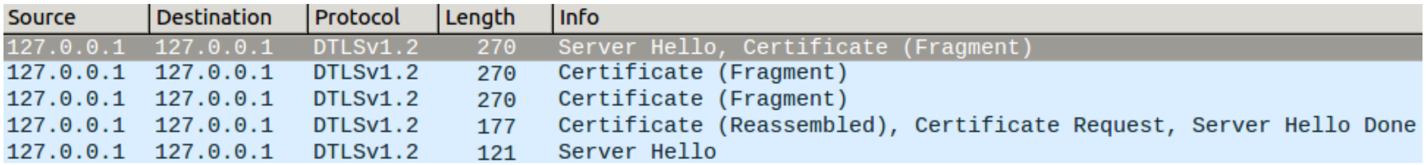
A popular DTLS implementation for WSN is TinyDTLS. The main characteristics of TinyDTLS are:

- The use of the cipher suite TLS_PSK_WITH_AES_128_CCM_8 for the PSK mode. And the cipher suite TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8 for the RPK mode.
- Jumbo datagrams and fragmentation are disabled.
- Cookies are configured to have a size of 16 bytes by default.

At the moment of this work, TinyDTLs is still marked as under development (available version is the 0.8.2). Therefore, changes from a version to another can lead to a low backward compatibility.

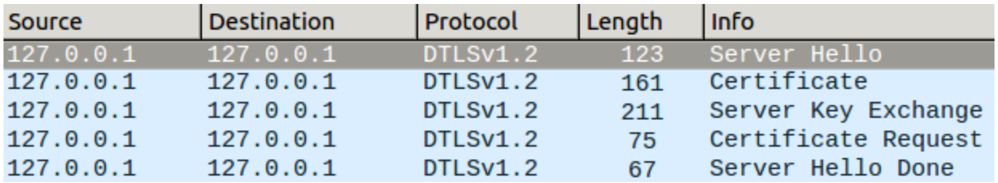
It is important to make the observation that for the next release of TinyDTLS (V. 0.9.0) the ad-hoc ECC library will be changed by the micro-ECC library. Also, it is important to say that the generation of random values used by TinyDTLs is too weak. In addition, the EPOCH value transmitted in the DTLS records related to the handshake process starts at zero (the first of January of 1970). This EPOCH time is used in combination with other 24 random bytes to generate a portion of the master key for the symmetric encryption. Although there is no requirement to use the correct date, it could reduce the resistance of the temporary keys against attacks.

In the example of the fourth flight of Figure 11a, the size of the DTLS header and its payload is near to 886 bytes, once the lower layer protocols are discarded. In comparison, TinyDTLS using TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8 requires only 362 bytes for the same flight as showed in Figure 11b. If the PSK cipher suite is used, the size decreases to 172 bytes.



Source	Destination	Protocol	Length	Info
127.0.0.1	127.0.0.1	DTLSv1.2	270	Server Hello, Certificate (Fragment)
127.0.0.1	127.0.0.1	DTLSv1.2	270	Certificate (Fragment)
127.0.0.1	127.0.0.1	DTLSv1.2	270	Certificate (Fragment)
127.0.0.1	127.0.0.1	DTLSv1.2	177	Certificate (Reassembled), Certificate Request, Server Hello Done
127.0.0.1	127.0.0.1	DTLSv1.2	121	Server Hello

(a) DTLS flight 4 with jumbo datagrams and fragmentation (Over ethernet and IPv4).



Source	Destination	Protocol	Length	Info
127.0.0.1	127.0.0.1	DTLSv1.2	123	Server Hello
127.0.0.1	127.0.0.1	DTLSv1.2	161	Certificate
127.0.0.1	127.0.0.1	DTLSv1.2	211	Server Key Exchange
127.0.0.1	127.0.0.1	DTLSv1.2	75	Certificate Request
127.0.0.1	127.0.0.1	DTLSv1.2	67	Server Hello Done

(b) TinyDTLS flight 4 without jumbo datagram or fragmentation (Over ethernet and IPv4).

Figure 11: DTLS Traffic

3.4 MMT-Security: Security and Attacks rules

There are two types of rules including security rules (SRs) describing expected behaviors and attack rules (ARs) specifying potential attacks, security violations or evasions. Both of them are defined by two set of events: context and trigger. Interestingly, the trigger can be an event or a set of events that occur before, after or at the same time to the context. The trigger is checked only if the context is validated:

- If the trigger is found invalid, a non-respect instance is detected.
  - In the case of SRs, the expected behavior is not respected.
  - In the case of ARs, the trace is attack free.
- If the trigger is found valid, a respect instance is detected.
  - In the case of SRs, the expected behavior is respected.
  - In the case of ARs, an attack has been detected.

An example for detecting ARP poisoning attacks can be found in [18]. The context is any ARP request captured from the medium, and the trigger is the event where two or more ARP replies appear with different MAC addresses in a specific range of time.

The rules are written in Boolean logic and rely on LTL. For complex analysis, it is possible to use embedded functions. The work in [15] presents the utilization of embedded functions to compare HTTP User-Agent strings to signatures coming from prepared databases, as well as to detect specific string patterns in certain fields of the header. Embedded functions play the role as the connector between MMT-Extract and MMT-Security which fulfills the missing in MMT-Extract and helps the MMT-Security works efficiently and conveniently.

4 MMT adaptation for 802.15.4/6LoWPAN and tinyDTLS

In this section, we explain the adaption of MMT-extract and MMT-Security to 802.15.4/6LoWPAN and tinyDTLS.

4.1 MMT-Extract

First, we extended the functionality of MMT-extract to support unicast, broadcast and 802.15.4 acknowledgment messages. Beacon messages will be ignored.

Second, we also extended to support IPHC, FRAG1 and FRAGN for 6LoWPAN messages. For FRAG1 and FRAGN, MMT-extract *is able to reassemble the fragmented packets to get the original packet*, then extract information required for the Security Rules. IPHC dispatch verifies if the received header of the received frame is ICMPv6, UDP or the compression of UDP using NHC. It also verifies that IPv6 addresses are not in the IPv6 message's fields.

Finally, we adapt MMT-extract to identify all different types of DTLS records, including the ones with dynamic sizes records.

Figure 12 shows the adapted plugins for MMT-extract. In a parallel work, Hoa [12] published some of these plugins.

4.2 MMT-Security

Just as briefly discussed in Section I, in [8] we identified threats that could have an impact on communications between sensors. In particular, in the upper layer, where the commands for sensors and the data measured by them are expected to be passed. Also, a list of countermeasures was enumerated with the assumption that a proper configuration with DTLS provides enough mitigation. Said threats are summarized in Table 1 which is composed of the following elements:

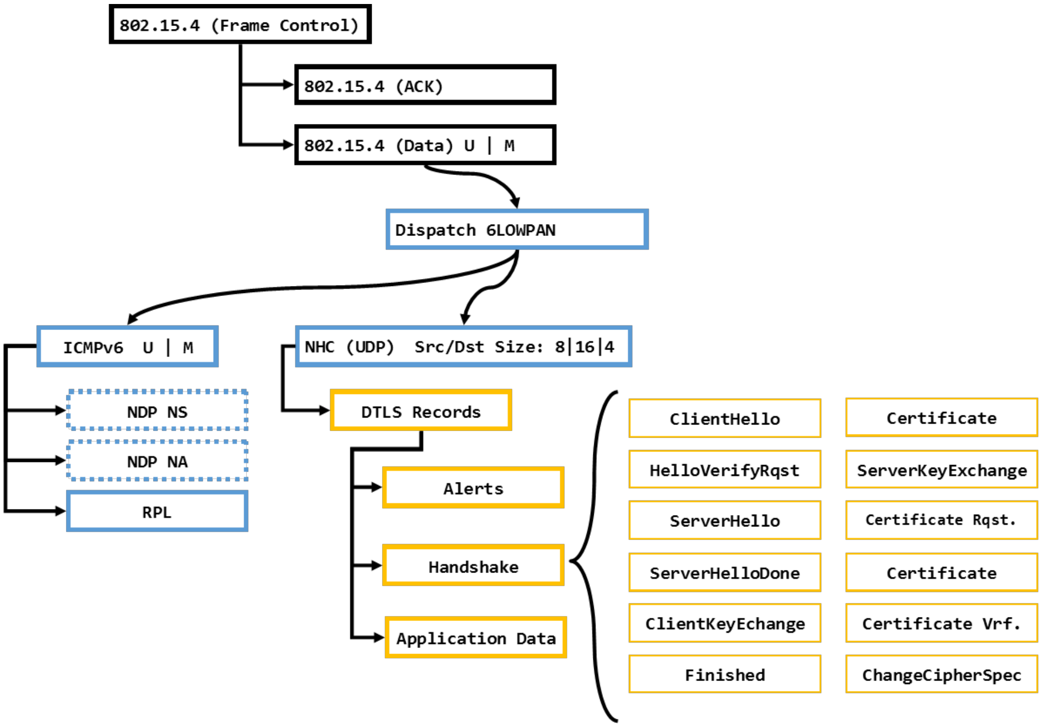


Figure 12: MMT-Extract: Extensions for 802.15.4/6LoWPAN and DTLS

- The likelihood of the threat, measured as low, moderate, substantial or severe.
- The impact over the WSN and the reputation of its provider is measure as low, medium and high.
- The risk is measured as minor, major and critical.
- Security Rules for MMT, which monitors a proper configuration of DTLS able to mitigate the threat.

In this work, we implement Security Rules 1 to 4 using knowledge of the involved protocols, as shown in the next subsections.

4.2.1 SR1 (Security Rule 1)

Since tinyDTLS mitigates ET8 using the cipher `TLS_PSK_WITH_AES_128_CCM_8` and `TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8`, then an alert would be emitted when a node attempts to establish a session using different cipher suites.

The context is defined by the next two events: I) detection of a `ClientHello` record in a DTLS session. And II) detection of a `ServerHello` record for the same DTLS session. Since both events must occur inside of the range of lifetime of the DTLS session, then the context is confirmed if the time is lesser than this time.

The trigger is confirmed if the `Ciphersuites` in `ClientHello` and `ServerHello` are one of the two allowed ciphers.

4.2.2 SR2 (Security Rule 2)

Since tinyDTLS also mitigates ET8 using the size of the cookies, then an alert would be emitted when a a cookie is different than a defined size. For this work, the default size is 16 bytes.

The context is detected by the next two events: the detection of *Hello VerifyRequest* record and the detection of *ClientHello* record. This is because the cookies are used for the first time in the *Hello VerifyRequest* record and for the second time in the *ClientHello* record.

The trigger is confirmed if the size of the cookie inside of the *Hello VerifyRequest* and *ClientHello* records is 16 bytes.

4.2.3 SR3 (Security Rule 3)

The mechanism to mitigate ET15 and ET16 is to use DTLS protocol. Since DTLS is inside the UDP packet, then we first have to verify that a UDP packet has been received.

The context is confirmed if the packet carries a UDP header.

The trigger is confirmed if: I) the port-address is correct for the field destiny or source. And, II) the UDP payload is identified as a DTLS record.

4.2.4 SR4 (Security Rule 4)

Other mitigation mechanism for ET15 and ET16 is by verifying that a client-server DTLS handshake process has been succesfully made before the transmission of data. For TinyDTLS, we can verify this condition by verifying the `ChangeCipherSpec` record, which forces the nodes to renew the DTLS session or to create the first one.

The context is confirmed when `ChangeCipherSpec` record is detected.

The trigger is confirmed if the `ChangeCipherSpec` record is received ten seconds, or less, before `HelloClient` record .

5 Validation of the implementation

5.1 *Testbed definition*

The physical scenario was implemented using FIT IoT-Lab[b], which is *a large-scale open testing infrastructure for systems and applications on wireless and sensor communication networks* [6]. Each node is an M3[c] with tinyDTLS adapted to RIOT operating system[d]. MMT has the capacity to work in real time, but to test the adaptation, forty five minutes of traffic were recorded with `sniffer_aggregator -r` for four scenarios. This data was analyzed off-line by MMT.

Nodes are configured to transmit generic data, which is echoed by the receptor. After that, the DTLS session is terminated. The transmitted data is a set of simple chains of text of small size, no longer than fifteen bytes, which most of them are ICMPv6 packages, routing protocol packages and manually triggered DTLS communication events between nodes.

We tested scenario networks consisting of two, five, ten and twenty nodes[e]. The scenario with two nodes is used to simulate a complete secure environment, in which nodes only support PSK mode and `TLS_PSK_WITH_AES_128_CCM_8` cipher suite. In the scenarios with five, ten and twenty nodes, we configured some nodes with RPK. However, RPK's cipher suite will be used for simulating a unsecured environment, as these scenarios will be used to test if security rules are correctly implemented.

Additionally, sensors are not always able to finish their sessions. There are two identified reasons of this failure: I) sensors are having issues processing the certificates, or II) because a DTLS session can be re-established after a certain time, the registries related to them are not erased fast enough for servicing new requests.

As already discussed in Section 3.3.3, flights 4 and 5 are composed of five DTLS records when `TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8` cipher suite is used in the mode RPK, otherwise, there are only two DTLS records. Fragmentation over 6LoWPAN only happens when the previous cipher suite is used and only in the following DTLS records: `ClientHello`, `Certificate`, `ClientKeyExchange` and `ServerKeyExchange`.

In this work, we do have not implemented node key storage and neither those related to EPOCH starting time value configuration.

Table 3 shows the total number of DTLS sessions triggered manually together with the number of successful connections, whereas Table 4 shows the security rules analyzed by MMT (Section 4.2).

5.2 *Experiment Results*

Table 3 Number of DTLS session started, Number of DTLS completed.

Number of nodes	Started Sessions	Messages
2	1	14
5	7	98
10	16	224
20	52	728

In order to verify the testbed we recorded 45 minutes of traffic under different scenarios with a variable number of nodes. This with the objective of evaluating the number of violated and respected instances detected for the security rules proposed in Section 4.2.

Table 3 shows the number of sessions and the number of messages transmitted for two, five, ten and twenty nodes scenarios. However, not all of them were successful. Table 4 shows the number of successful sessions. We can observe that for the two nodes configuration one session was started and one session was finished. For the five node configuration, we can observe that seven sessions were started, but only four of them were successful. This means that only 57.14% of the session were successful.

One session is not successful for many reasons, some of them are explained below:

- Because some nodes are using the RPK mode, and the RPK mode is not recognized as valid for this testbed.
- Because the sensors are already at their limits for handling DTLS sessions (two sessions for this testbed) and are forced to reject a third one.
- Because the internal timer for the DTLS sessions are not yet expired and still counts against the maximum number. This happens because tinyDTLS is able to recover a DTLS session.
- The current limitation in the resources which is more evident in the RPK mode. This is expected to be a temporary problem as the processing power of the sensors nodes will be increased in future years.

For the ten nodes configuration, we can observe that sixteen sessions were started, but only six of them were successful. This means that only 37.5% of the session were successful. And, for the twenty nodes configuration we can observe that fifty two sessions were started, but only twenty eight of them were successful. This means that only 53.58% of the session were successful.

Although not all sessions were completed we can still use the transmitted messages to test our Security Rules. Table 5 shows the transmitted messages coming from the completed sessions, from the uncompleted sessions and from retransmitted messages. Moreover, it also shows

Table 4 Effectiveness of the sensor network to deal with the DTLS load.

Number of nodes	Started Sessions	Successful Sessions	Effectiveness
5	1	1	100%
5	7	4	57.14%
10	16	6	37.50%
20	52	28	53.85%

Table 5 Relation between number of sessions and number of transmitted messages.

Number of nodes	# Messages Transmitted		Total Transmitted Msg	From Intruders	Total of transmitted msgs.
	Completed	Not-Completed / Retransmitted			
2	14	0	14	0	14
5	56	29	85	27	112
10	84	50	134	0	134
20	392	142	534	0	534

Table 6 Content of the DTLS Database to be tested.

Number of nodes	Messages	Messages from intruders	SR1		SR2		SR3		SR4	
			R.	V.	R.	V.	R.	V.	R.	V.
2	14	0	1	0	1	0	14	0	2	0
5	112	27	4	4	8	0	85	27	8	4
10	134	0	7	3	15	0	134	0	11	3
20	534	0	36	6	47	0	534	0	50	18

messages coming from unregistered nodes, we label them as "intruders".

Note that for each completed session the protocol sends fourteen messages and the number of messages of uncompleted session is uncertain because we do not know what exactly failed. Fourteen messages are for a DTLS session working on PSK mode: ten messages of handshake to start the session, two messages for data transmission and two more to finish the session.

Hence, using this information, Table 5 shows that for the two node configuration, the total number of transmitted messages are fourteen, and all of them are from completed session.

For the five node configuration, the total number of transmitted messages were one hundred and twelve. From these, the number of transmitted messages from completed sessions are fifty two; the number of transmitted messages from intruders are twenty seven; the rest, twenty nine, are from uncompleted or retransmitted sessions.

For the ten node configuration, the total number of transmitted messages were one hundred and thirty four. From these, transmitted messages from completed sessions are eighty four and the rest, fifty, are from the uncompleted or retransmitted sessions. Finally, for the twenty node configuration, the total number of transmitted messages were five hundred and thirty four. From these, the number of transmitted messages from the completed session were three hundred and ninety two; the rest, one hundred and forty two, are from the uncompleted or retransmitted sessions.

Table 6 shows the Security Rules in the Database that have to be verified as Recognized and the ones that have to be verified as violated. For example, for the two nodes configuration, all Security Rules SR1, SR2, SR3 and SR4 in the Database have to be verified has Recognized. For the five nodes configuration, SR1, the Database contains four security Rules that have to be verified as Recognized and four that have to be verified as violated.

The results shown in the Table 6 are also affected by the overhead in the sensors. SR1 for the 5 sensors scenario detects properly the `ClientHello` and the `ServerHello` records. However, for the other scenarios not all of those records end with a fully established DTLS session.

SR2 has as a strong link to SR1 because both of them monitor the same records, although they watch different fields. Their results differ only when the DTLS session

Table 7 Detection percentage of success of the Security Rules.

Number of nodes	Messages	Messages from intruders	SR1		SR2		SR3		SR4	
2	14	0	100%	100%	100%	100%	100%	100%	100%	100%
5	112	27	100%	100%	100%	100%	100%	100%	100%	100%
10	134	0	100%	100%	100%	100%	100%	100%	100%	100%
10	134	0	100%	100%	100%	100%	100%	100%	100%	100%
20	534	0	100%	100%	100%	100%	100%	100%	100%	100%

negotiation is stopped before the ServerHello record is sent.

The final purpose of SR3 is to validate that the traffic at the upper layer is composed only of DTLS records. In the case of the scenario with five sensors, there were two rogue sensors transmitting non-compressed UDP traffic. The presence of rogue nodes is due to the open nature of the wireless channel. This occurrence is not unexpected on real implementations, particularly on environments with a high density of nodes. Therefore, detecting the presence of those nodes is significant and positive.

The results thrown by SR4 can identify the moment when the sensors update their symmetric keys, by sending the ChangeCipherSpec record. The numbers in the first three scenarios double the number of successful connections because the rule takes separately the discovery of each couple of DTLS ChangeCipherSpec records. This in turn is echoed by the servers to the clients. This rule is impacted when sensors are forced to resend packets due to the congestion in the network, which is the case of the scenario with 20 sensors. The cases where the rule was violated was because the lifetime expected was inferior to the real time. Therefore, there are two important conclusions: 1) The rule requires to follow the stream to try to identify the beginning and the end of a data transmission. And 2) With the current configuration for the rules, it is not possible to guarantee the standard lifetime expected by DTLS in bigger scenarios, maybe because of the congestion in the network or maybe because the resources of the sensors.

Finally, Table 7 shows that all the rules were correctly verified as Recognized and Violated.

6 Conclusions and future work

In this paper, advanced monitoring techniques are presented to check secure M2M communications over 802.15.4/6LoWPAN and using TinyDTLS. These techniques have been integrated to the monitoring MMT tool. This implementation has been validated by experiments on real testbeds combining up to 20 sensors. Traffic data for testing were recorded and used as a trace.

Although the use of TinyDTLS on WSN has been validated in other works, especially in terms of energy consumption, more testing on real implementations under real-time conditions are required. Because of parameters such as the number of simultaneous DTLS sessions, the lifetime and the negotiation for resuming previous DTLS sessions have a high impact on the performance of the sensors and WSN bandwidth as noted in our tests.

Also, it is worth emphasizing the difficulties of monitoring native traffic over 6LoWPAN networks, when DPI is preferred. This is because of the high modularity of 6LoWPAN with its dispatches. For this work, the focus was on traffic over UDP using only the IPHC dispatch. This generates five possible ways to transmit messages: Non-compressed UDP and the four compression variants with NHC.

It is challenging because sensors can interchange modes at each point of the transmission. And, if the messages requires to be fragmented, the complexity for preserving the DPI becomes higher. Nevertheless, one advantage of our design is that in normal model DTLS handshake can be completed in very short periods of time.

Although the plugins were developed for a generic DTLS communication, they were tested for TinyDTLS 0.8.2. As a consequence, any support for other stacks that make use of jumbo datagrams and fragmentation over this layer requires further work.

Acknowledgments

The research leading to this paper has received partial funding from the French IDOLE project (Homepage: http://projet-idole.fr/). The views, opinions and findings contained in this paper are those of the authors and should not be construed as official IDOLE's positions, policy or decision.

We also would like to acknowledge Montimage (www.montimage.com) that provided their tool MMT, to perform our research work and experimentation.

References

[1] Denial-of-Service detection in 6LoWPAN based Internet of Things. pages 600–607, Lyon, France, 2013. IEEE.

[2] F. B. Abreu, A. Morais, A. Cavalli, B. Wehbi, and E. Montes de Oca. An Effective Attack Detection Approach in Wireless Mesh Networks. In *27th International Conference on Advanced Information Networking and Applications Workshops*, pages 1450–1455. IEEE, mar 2013.

[3] S. Alam and D. De. Analysis of security threats in wireless sensor network. *arXiv preprint arXiv:1406.0298*, 2014.

[4] K. Doddapaneni, E. Ever, O. Gemikonakli, L. Mostarda, and A. Navarra. Effects of IDSs on the WSNs lifetime: Evidence of the need of new approaches. *Proc. of the 11th IEEE Int. Conference on Trust, Security and Privacy in Computing and Communications, TrustCom-2012 - 11th IEEE Int. Conference on Ubiquitous Computing and Communications, IUCC-2012*, pages 907–912, 2012.

[5] ETSI (European Telecommunications Standards). ETSI TR 103 167 V1.1.1 (2011-08) Machine-to-Machine Communications (M2M); Threat analysis and counter-measures to M2M service layer. 1:1–62, 2011.

[6] E. Fleury, N. Mitton, T. Noel, and C. Adjih. Fit iot-lab: The largest iot open experimental testbed. *ERCIM News*, (101):4, 2015.

[7] R. Fuentes-Samaniego, A. Cavalli, J. Nolazco-Flores, and J. Baliosian. A survey on wireless sensors networks security based on a layered approach. In M. C. Aguayo-Torres, G. Gmez, and J. Poncela, editors, *Wired/Wireless Internet Communications*, volume 9071 of *Lecture Notes in Computer Science*, pages 77–93. Springer International Publishing, 2015.

[8] R. Fuentes-Samaniego, A. R. Cavalli, and J. A. Nolazco-Fores. An analysis of secure m2m communication in wsns using dtls. In *Distributed Computing Systems Workshops (ICDCSW), 2016 IEEE 36th International Conference on*, pages 78–83. IEEE, 2016.

[9] J. Granjal, E. Monteiro, and J. S. Silva. On the Effectiveness of End-to-End Security for Internet-Integrated Sensing Applications. *2012 IEEE International Conference on Green Computing and Communications*, pages 87–93, nov 2012.

[10] J. Hui and P. Thubert. Rfc6282: Compression format for ipv6 datagrams over ieee 802.15.4-based networks. Technical report, IETF, September 2011. accessed 11/11/2015.

[11] 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 *Proceedings - Conference on Local Computer Networks, LCN*, pages 956–963, Clearwater, Florida, 2012. IEEE.

[12] V. H. La and A. R. Cavalli. A misbehavior node detection algorithm for 6LoWPAN Wireless Sensor Networks. In *Proceedings of 36th IEEE International Conference on Distributed Computing Systems (ICDCS 2016), Second IEEE International Workshop on Security Testing and Monitoring (STAM 2016)*, 2016.

[13] V. H. La and A. R. Cavalli. Network Monitoring using MMT: An application based on the User-Agent eld in HTTP headers. In *2016 IEEE 30th International Conference on Advanced Information Networking and Applications (AINA)*, pages 147–154, 2016.

[14] V. H. La, R. Fuentes, and A. R. Cavalli. A Novel Monitoring Solution for 6LoWPAN-based Wireless Sensor Networks. In *2016 IEEE 22nd Asia-Pacific Conference on Communications (APCC)*, August 2016.

[15] V. H. La, R. Fuentes, and A. R. Cavalli. Network monitoring using mmt: An application based on the user-agent field in http headers. In *2016 IEEE 30th International Conference on Advanced Information Networking and Applications (AINA)*, pages 147–154, March 2016.

[16] W. Mallouli, B. Wehbi, and E. M. de Oca. Online Network Traffic Security Inspection Using MMT Tool. In *Systems Testing and Validation Workshop 2012*, pages 23–31, 2012.

[17] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler. Rfc4944: Transmission of ipv6 packets over ieee 802.15.4 networks. Technical report, IETF, September 2007. accessed 11/11/2015.

[18] Montimage. MMT-Security: User Guide, 2013. Last checked: 05.09.2015.

[19] S. Raza, H. Shafagh, K. Hewage, R. Hummen, and T. Voigt. Lithe: Lightweight Secure CoAP for the Internet of Things. *IEEE Sensors Journal*, 13(10):3711–3720, oct 2013.

[20] S. Raza, L. Wallgren, and T. Voigt. SVELTE: Real-time intrusion detection in the Internet of Things. *Ad Hoc Networks*, pages 2661–2674, 2013.

[21] E. Rescorla and N. Modadugu. Rfc6347: Datagram transport layer security version 1.2. Technical report, IETF, January 2012. accessed 11/11/2015.

[22] Z. Shelby, S. Chakrabarti, E. Nordmark, and C. Bormann. Rfc6775: Neighbor discovery optimization for ipv6 over low-power wireless personal area networks (6lowpans). Technical report, IETF, november 2012. accessed 11/11/2015.

[23] A. F. Skarmeta, J. L. Hernandez-Ramos, and M. V. Moreno. A decentralized approach for security and privacy challenges in the Internet of Things. *2014 IEEE World Forum on Internet of Things (WF-IoT)*, pages 67–72, mar 2014.

[24] J. Undercoffer, S. Avancha, A. Joshi, and J. Pinkston. Security for Sensor Networks. *Kluwer Academic Publishers Norwell*, pages 253–275, 2004.

[25] M. Vucinic, B. Tourancheau, T. Watteyne, F. Rousseau, A. Duda, R. Guizzetti, and L. Damon. Dtls performance in duty-cycled networks. *arXiv preprint arXiv:1507.05810*, 2015.

[26] J. P. Walters, Z. Liang, W. D. o. C. S. W. S. U. Shi, and V. D. o. C. S. W. S. U. Chaudhary. Chapter 17 Wireless Sensor Network Security : A Survey . *Security in Distributed, Grid, and Pervasive Computing*, pages 1–50, 2006.

[27] B. Wehbi, E. Montes de Oca, and M. Bourdelles. Events-Based Security Monitoring Using MMT Tool. In *IEEE Fifth International Conference on Software Testing, Verification and Validation (ICST), 2012*, pages 860–863, April 2012.

Notes

[1] http://www.montimage.com
[2] https://www.iot-lab.info/
[3] https://www.iot-lab.info/hardware
[4] https://github.com/RIOT-OS/RIOT/tree/2016.10
[5] The recorded traffic files used for this paper are available online at https://zenodo.org/record/183667.