Application of IEEE 802.15.4 Security Procedures in OpenWSN protocol stack

Article · December 2014

CITATIONS
9

READS
505

4 authors, including:

Savio Sciancalepore
Hamad bin Khalifa University
39 PUBLICATIONS **176** CITATIONS
SEE PROFILE

Giuseppe Piro
Politecnico di Bari
84 PUBLICATIONS **2,421** CITATIONS
SEE PROFILE

Luigi Alfredo Grieco
Politecnico di Bari
206 PUBLICATIONS **6,070** CITATIONS
SEE PROFILE

Application of IEEE 802.15.4 Security Procedures in OpenWSN Protocol Stack

Savio Sciancalepore, *Student Member, IEEE,*
Giuseppe Piro, *Member, IEEE,*
Gennaro Boggia, *Senior Member, IEEE* and
Luigi Alfredo Grieco, *Senior Member, IEEE*
Department of Electrical and Electronical Engineering (DEI), Politecnico di Bari,
v. Orabona 4, 70125 Bari, Italy; Email: {name.surname}@poliba.it

***Abstract*—With the diffusion of the Internet of Things paradigm, even more researchers and industries worldwide are focusing their attention on the definition of optimized protocol architectures, able to offer a wide range of services in Low-power and Lossy Networks. In this context, the OpenWSN project emerges as one of the most promising open-source protocol stack for IoT devices; it is based on the IEEE 802.15.4 radio and is particularly suitable for constrained devices. Unfortunately, at the time of this writing, it does not support security features defined by IEEE 802.15.4 specifications for the MAC layer. To bridge this gap, we present in this contribution a freeware and open-source implementation of security procedures for the OpenWSN protocol suite, which embraces security attributes stored into the MAC entity, security functions operating at the MAC layer, and cryptographic techniques used to execute encryption and decryption functionalities. To provide a further insight, we also evaluated, through real experiments conducted with the TelosB hardware platform, the impact that the adoption of security features has on both computational load and communication latencies. Our findings demonstrate that computational capabilities of constrained motes drastically influence the network operation, thus requiring the design of optimized and enhanced security services. We believe that the open source nature of the developed module could widely support all people involved in this research area.**

I. Introduction

The emerging Internet of Things (IoT) paradigm introduces the possibility to create a capillary networking infrastructure, able to provide several ICT services, spanning several different contexts like transportation, logistics, healthcare, smart environment (home, office, plant), and personal/social domains [1]. This idea leads to the definition of a Low Power and Lossy Network (LLN), which is mainly composed by a large number of low-power nodes that can establish short-range wireless connections among them and that can be connected to the Internet through gateway servers [2]. Such devices have several constraints in terms of computational capabilities, energy consumptions, and storage resources. As a consequence, the definition of suitable protocol stacks (that cover all aspects, from the application layer to the physical interface) is a key topic, which is currently attracting the attention of researchers, industries, and standardization bodies worldwide [3], [4], [5].

In this context, the IEEE 802.15.4 standard emerges as the leading enabling technology for short range low rate wireless communications [6]. In fact, it defines (i) physical and the Medium Access Control (MAC) layers for LLNs, (ii) two types of network nodes, i.e., Fully Function Device (FFD) and Reduced Function Device (RFD), that can build peer-to-peer or star networks, and (iii) advanced security features at the MAC layer. More recently, instead, the IEEE 802.15.4e specification has been published for introducing some amendments to the IEEE 802.15.4 standard [7]. Among its key features, the Time Synchronized Channel Hopping (TSCH) emerges as a novel MAC protocol, which better supports multi-hop communications in industrial applications.

To actualize the IoT vision and easy plug and play operations of smart devices in IPv6 networks, the ZigBee alliance and the Internet Engineering Task Force (IETF) have recently proposed and standardized novel solutions, based on the aforementioned IEEE 802.15.4 radio, at different layers of the protocol stack. At the same time, researchers and industries are looking at designing and developing innovative algorithms and approaches that optimize and/or extend what have been conceived within standardization efforts. In this context, the availability of sophisticated research instruments, which model and implement in real devices LLN-related protocols, can be very useful for better supporting research activities aiming at evaluating pros and cons of existing and novel solutions.

At the time of this writing, one of the most promising freeware and open-source implementation of IoT-compliant protocol stack for constrained devices has been developed within the OpenWSN project [8]. Nevertheless, despite it already offers a large number of protocols, including PHY and MAC layers defined in IEEE 802.15.4 and IEEE 802.15.4e specifications, as well as other high level protocols devised by the IETF, i.e., IPv6 over Low Power and Lossy Networks (6LoWPAN), Routing Protocol for Low Power and Lossy Network (RPL), and Constrained Application Protocol (CoAP), some important features are still not fully available. Among them, one of the most important lacks is the unavailability of security procedures and services defined by the IEEE 802.15.4 standard for the MAC layer.

The present contribution intends to overcome this weakness by proposing an open source implementation of IEEE 802.15.4 security features within the OpenWSN project. To this end, we properly extended the OpenWSN protocol stack by developing:

- MAC PIB attributes related to security aspects, i.e., tables and variables storing all the information needed to provide security services (including keys, devices' authorization, security levels, and so on);
- functions operating at the MAC-high layer, handling the inserting and the retrieving of security-related parameters and attributes introduced at the previous point;
- cryptographic techniques used to execute encryption and decryption functionalities, based on the Advanced Encryption Standard (AES) scheme.

We remark that the developed code is fully compliant with IEEE 802.15.4 specifications and it is integrated with the entire OpenWSN protocol stack (and, hence, it could be used for managing security services at the MAC layer independently from the set of configured upper layers). Moreover, the code is freely available at: *http://telematics.poliba.it/openwsn_ieee802154_security*.

To provide a further insight, we also evaluated, through real experiments, the impact that the provisioning of security features has on communication latencies. In particular, we set up a simple testbed composed by a couple of TelosB motes [9], that exchange a number of packets, which are protected at the MAC layer according to IEEE 802.15.4 specifications. Obtained results clearly show that the enabling of security features in constrained nodes requires additional computational efforts, which involves a not negligible growth of communication latencies.

The rest of the paper is organized as follows: Sec. II describes the IEEE 802.15.4 standard and focuses on security mechanisms it proposes; the description of the implemented security module within the OpenWSN project is provided in Sec. III; Sec. IV shows experimental results; finally, Sec. V draws conclusions and future works.

II. IEEE 802.15.4

A. General Description

The IEEE 802.15.4 standard is widely recognized as one of the most successful low-power technologies for short range and low-rate wireless communication, and defines both MAC and Physical (PHY) layers of the protocol stack [3].

Two types of network nodes, i.e., the FFD and the RFD, can be found in an IEEE 802.15.4 network. They could be arranged in both peer-to-peer and star topologies. A FFD has the highest computational capabilities and it works as the coordinator of the network (also called PAN coordinator), thus being a reference node for a group of others RFD devices. Instead, a RFD has lower resource and communication capabilities and it is able to communicate only with its reference FFD. Whereas RFDs must be connected only to a single coordinator, FFDs can connect among them forming more complex meshed network architectures.

At the physical layer, the IEEE 802.15.4 network operates in the $2.4 - 2.485$ GHz frequency band, i.e., a worldwide and unlicensed frequency range, and exploits the Offset-Quadrature Phase Shift Keying (O-QPSK) modulation scheme and the Driect Sequence Spread Spectrum (DSSS) transmission technique. The maximum physical data rate is equal to 2 Mbps. However, due to the presence of enhanced coding algorithms, which make the transmission robust against interferences, the effective bit rate from the users' perspective is equal to 250 kbps.

The channel access is regulated by a specific *Super Frame* structure, as illustrated in Fig. 1, composed by 16 consecutive slots (all with the same duration). In order to communicate some details about the PAN and the medium utilization, as well as enabling synchronization procedures for all the attached devices, the coordinator periodically releases a control message, namely *Beacon*, at the beginning of the *Super Frame*. Moreover, as depicted in Fig. 1, the transmission of messages is done during the *active period* (i.e., a first part of the *Super Frame*). The remaining unused slots form the inactive period, during which devices can turn in low-power mode for energy-saving purposes.

Before the packet transmission, a physical preamble of $128\mu s$ is exploited to execute synchronization operations. Then, a Start of Frame Delimiter (SFD) is sent to indicate the start of the physical payload, whose size is up to 128 bytes. However, since the first byte of the physical payload is used to indicate the packet length, the overall amount of useful bytes, i.e., the Maximum Transmission Unit (MTU), is equal to 127 bytes.

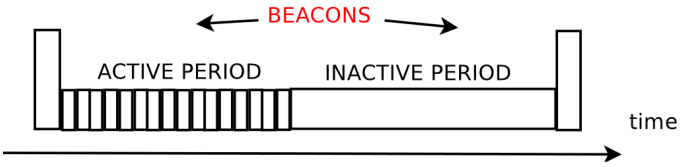


Fig. 1. The SuperFrame Structure

B. Security in IEEE802.15.4

1) Security services: As highlighted by the general description of the standard, an IEEE 802.15.4 network is vulnerable to a number of security threats/attack, which include active tampering and eavesdropping. As a consequence, to make the network robust against these kind of hazards, it is important to provide security services as data confidentiality, message integrity, and the protection to *replay attacks*.

The data confidentiality refers to the possibility to limit the access to information stored within packets to only authorized nodes. It is generally achieved by encrypting part of the message (i.e., in our case the MAC payload) through the adoption of key and algorithm common to both sender and receiver. While not knowing secrets exploited by the encryption process, a given attacker could modify the content of the message while it transits over the network for reaching the destination node. The message integrity protection must ensure that the packet arrives at the receiver side without any tampers and alterations. This is done thanks to the use of a one-way hash function that, by combining all the bytes in the message with a secret key, is able to verify the integrity of the packet itself. Finally, an encrypted and authenticated message, which has

been exchanged between two authorized node, may be stored and sent again into the network by a fraudulent device (i.e., the so called *replay attack*). To prevent this kind of attack, the sender typically assigns a monotonically increasing sequence number to each packet and the receiver rejects packets with smaller sequence numbers than it has already seen.

To offer these security services, the IEEE 802.15.4 specification introduces procedures and mechanisms for protecting MAC frames, through symmetric-key cryptography techniques based on the *AES-CCM** algorithm. In the case security features are supported by a given device, the *macSecurityEnabled* attribute, stored at the MAC layer, is set to TRUE.

2) Security levels: Eight security levels are defined to protect the frame generated at the MAC layer in different manners. As summarized in Tab. I, they include *unsecured*, *only encrypted*, *only authenticated*, and *encryption with authentication* configurations. When the *unsecured* level is enabled, nor data confidentiality neither message integrity are provided. In other cases, instead, the data encryption and the authentication of messages are provided by means of AES and AES-CBC techniques, respectively. It is possible to offer a specific service to each kind of packet. However, the selection of the security level and the definition of other parameters required for performing security procedures have to be handled by an upper layer and then communicated to the MAC entity through dedicated primitives.

TABLE I. DIFFERENT SECURITY LEVELS PROVIDED BY THE IEEE802.15.4 STANDARD

Security Level	Security Level Field b2, b1, b0	Security Attributes	Data Confidentiality	Data Authenticity	Authentication tag length (bytes)
0	000	none	OFF	NO	0
1	001	MIC-32	OFF	YES	4
2	010	MIC-64	OFF	YES	8
3	011	MIC-128	OFF	YES	16
4	100	ENC	ON	NO	0
5	101	ENC-MIC-32	ON	YES	4
6	110	ENC-MIC-64	ON	YES	8
7	111	ENC-MIC-128	ON	YES	16

3) IEEE 802.15.4 header structure: The IEEE 802.15.4 MAC frame, which has been pictured in Fig 2, is composed by a MAC header, a payload and a Frame Check Sequence (FCS) footer. Security parameters are included within the *Frame Control* and the *Auxiliary Security Control* fields.

If the *Security Enabled* flag of the *Frame Control* field is set to 1, it means that the current MAC frame is protected and the node transmitting the packet supports at least one of the security services discussed above. If the flag is set to 0, instead, it means that the device sending the packet does not support any security capabilities and, for this reason, it is not able to send and receive encrypted and/or authenticated messages.

The *Auxiliary Security Control* field, which is present into the MAC header only if the *Security Enabled* flag is equal to 1, stores some parameters adopted for protecting the frame. They will be exploited by the destination node for performing the reverse security procedure (i.e, decryption and/or integrity check). This security header is composed by the *Security Control*, the *Frame Counter*, and the *Key Identifier* fields. The first one explains the security level and the key identification mode chosen by the sender. The counter stored into the second field is generated by the source in order to protect the message from replay attacks. Finally, the last field, i.e., the *Key Identifier*, is optional and it stores information (*KeySource* and *KeyIndex*) needed to determine the key exploited for the encryption of the message.

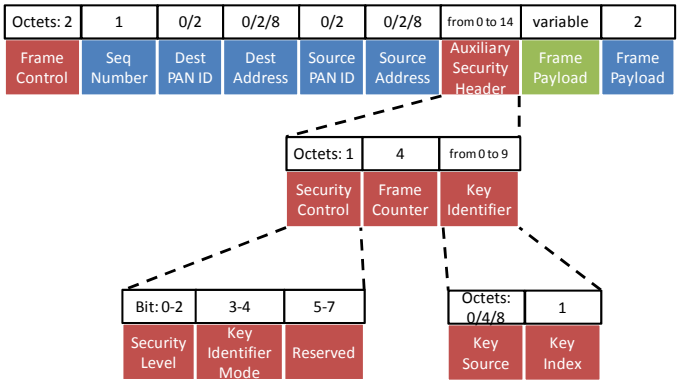


Fig. 2. The Auxiliary Security Header Structure

4) Security procedures and MAC PIB attributes: At the MAC layer, encryption and decryption functionalities are implemented within the *outgoing frame security* and the *incoming frame security* procedures, respectively. They exploit a number of security attributes, which have been summarized in Tab II.

It is very important to remark that the standard allows the possibility to use a dedicated key for each remote device and for each type of MAC frame (i.e., beacon, command frame, data packet, and ACK). Moreover, it is necessary to define a specific security service to be guaranteed for each kind of message. The related information is stored in the *macSecurityLevelTable*. It is composed by a set of *SecurityLevelDescriptor* elements, which provide information about the frame type which it refers to, the minimal expected/required *security level*, the set of allowed *security levels*, and a boolean flag indicating if the minimal security service may be overridden by a given device.

A node stores into the *macDeviceTable* the list of devices with which it can setup a secure communication. For each of them, a dedicated *DeviceDescriptor* is created. It contains the PAN ID, its short MAC address, its extended MAC addresses, as well as the counter of the latest packet received from the remote device and a boolean flag indicating if the considered node may override the minimum security level settings.

Without any doubts, the most important attribute is the *macKeyTable* where all the keys are organized in. A *keyDescriptor*, i.e., the single element of the aforementioned table, contains the key, the set of devices that can use it, a list of *KeyUsageDescriptor* indicating which frame may be protected with this key, and other parameters (e.g., *KeySource*, *keyIndex*) used for uniquely identifying the key.

A node that intends to secure a packet (i.e., by encrypting and/or authenticating the MAC payload) has to execute the

TABLE II. SECURITY-RELATED ATTRIBUTES DEFINED IN IEEE 802.15.4

Attribute	Type	Description
macKeyTable	Set of *KeyDescriptors*	It stores keys and other specific information, useful for protecting a MAC frame. A *keyDescriptor* element is created for each key, and contains: the key, the set of devices that can use it, a list of *KeyUsageDescriptor* indicating which frame may be protected with this key, and *KeySource* and *keyIndex*) parameters adopted to uniquely identify the key.
macDeviceTable	Set of *DeviceDescriptors*	It provides some information about remote devices which the node can establish a secure communication with. A dedicated *DeviceDescriptor* is associated to each remote device.
macSecurityLevelTable	Set of *SecurityLevelDescriptors*	It provides information about the security level required for each MAC frame type and subtype. In particular, each *SecurityLevelDescriptors* stores information about the frame type which it refers to, the minimal expected/required *security level*, the set of allowed *security levels*, and a boolean flag indicating if the minimal security service may be overridden by a given device.
macFrameCounter	Integer	The outgoing frame counter for the considered device.
macAutoRequestSecurityLevel	Integer	The security level used for automatic data requests.
macAutoRequestKeyIdMode	Integer	The key identifier mode used for automatic data requests. It is invalid if the *macAutoRequestSecurityLevel* attribute is set to 0x00.
macAutoRequestKeySource	Short or extended IEEE 802.15.4 MAC address	The originator of the key used for automatic data requests. This attribute is invalid if the *macAutoRequestKeyIdMode* element is invalid or set to 0x00.
macAutoRequestKeyIndex	Integer	The index of the key used for automatic data requests. It is invalid if the *macAutoRequestKeyIdMode* attribute is invalid or set to 0x00.
macDefaultKeySource	Extended IEEE 802.15.4 MAC address	The originator of the default key used for key identifier mode 0x01.

outgoing frame security procedure, i.e., the following steps:

1) identify the *Security Level* that has to applied to the current MAC frame among those listed in Tab I;
2) make sure that the size of the frame does not exceed the maximum allowed value (i.e., 127 bytes [6]);
3) identify the key to use during the encryption process. It will be selected among those available into the *macKetTable* by taking into account the *Key Identify Mode* announced by the upper layer;
4) protect the MAC payload according to the selected *Security Level*, by using the CCM* algorithm;
5) create the *Auxiliary Security Control* field and include it within the protected frame;
6) generate the FCS;
7) reassemble the whole packet;

When a device receives a MAC frame, it should firstly verify if it has been protected by the sender (i.e., if the *Security Enabled* flag is set to 1). In affirmative case, it will run the *incoming frame security* procedure, i.e., the following operations:

1) verify the packet integrity through the check of the FCS;
2) identify the key to exploit during decryption process;
3) verify that the *Security Level* chosen by the sender is allowed for the message the packet contains;
4) decrypt the payload;
5) verify that all security constraints (e.g., allowed *Security Level*, allowed key, frame counter) do not generate any security conflicts;
6) delivery the message to the upper layer;

To rightly complete both aforementioned procedures, each device uses other subroutines for different purposes. They include: (i) the *KeyDescriptorLookup* procedure, exploited to find the key in the *MacKeyTable*; (ii) the *DeviceDescriptorLookup* procedure, used to find the right entry in the *MacDeviceTable*; (iii) the *SecurityLevelDescriptorLookup* procedure, used to detect the type of frame to be unsecured, (iv) the *IncomingSecurityLevelChecking* procedure, exploited to check if the security level of the incoming frame conforms to the standard security levels for the receiving node; and (v) the *IncomingKeyUsagePolicyChecking*, which identifies if the key is properly used.

5) Cryptographic operations: The *CCM** cryptographic process is based on the AES-128 block cipher, with the use of a key with either 32, 64, or 128 bits (note that the generation of this key is outside the scope of the standard).

The *CCM* mode forward procedure*, adopted to protect a MAC message, involves the execution of three subroutines:

1) the *Input Transformation*, which is in charge of creating two data strings, i.e., *AuthData* and *PlainTextData*, that represent initialization vectors in next steps;
2) the *Authentication Transformation* procedure, which generates the authentication tag. As shown in Fig. 3, it is based on the CBC-MAC mode;
3) the *Encryption Transformation*, that generates the encrypted message through a AES- Counter Mode (AES-CTR) scheme (see Fig. 4).

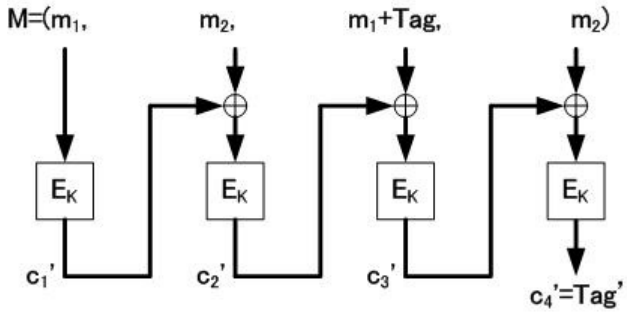


Fig. 3. CBC-MAC scheme. In that figure, M is the plaintext divided in a number of m_i strings; E_k is the encryption primitive (i.e., AES-128), and c_i is the encrypted version of the m_i string.

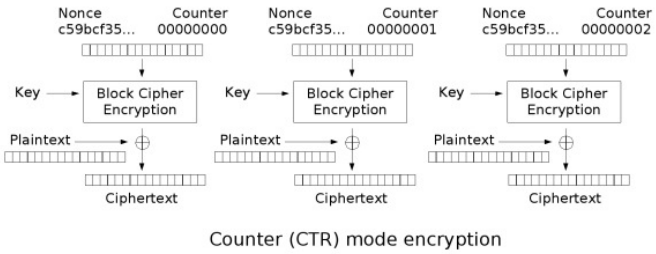


Fig. 4. AES-CTR scheme. The *Nonce* string is generated taking into account the Source Address, the Frame Counter and the Security Level of the current frame.

Finally the *CCM* mode inverse transformation* handles the decryption and the message integrity checking through the *Decryption Transformation* and the *Authentication Checking Transformation* function, respectively. Without loss of generality, we note that these functions execute the same subroutines described for the *CCM* mode forward procedure*.

III. Implementation: The Standard Applied

In this contribution, we present an open-source and freely available module implementing IEEE 802.15.4 security procedures within the emerging OpenWSN protocol stack.

6) The OpenWSN project: The OpenWSN project provides an open-source implementation of a standard-based protocol stack for constrained devices. It integrates several IEEE and IETF protocols, such as Time Synchronized Channel Hopping (TSCH), CoAP, RPL and 6LoWPAN, and allows the creation of ultra-low power and highly reliable mesh networks connected among them through Internet [8].

At the time of this writing, OpenWSN is supported by a large number of hardware platforms (they have been summarized in Tab. III) and embraces debugging functionalities and additional application tools. As a consequence, it is able to sustain any extensive study and experimental evaluation of novel solutions for LLN.

A comprehensive scheme of the OpenWSN protocol stack has been reported in Fig. 5. It is important to underline that despite the presence of a huge number of protocols at different layers of the stack, the OpenWSN project does not implement security features described within the IEEE 802.15.4 specification.

7) Implementation details: In order to overcome the aforementioned lack, we properly extended the original version of the OpenWSN protocol suite by developing (i) MAC PIB attributes related to security aspects (ii) security functions operating at the MAC-high layer, and (iii) cryptographic techniques. A clear description of entities forming the developed module, including their relationship with the existing OpenWSN protocol stack, is depicted in Fig. 6.

First of all, in line with the IEEE 802.15.4 standard, a number of data structures and variables, which refer to MAC PIB security attributes, have been developed. The most important implementation details can be found in Tab. IV. In addition, to enable security aspects we also extended the

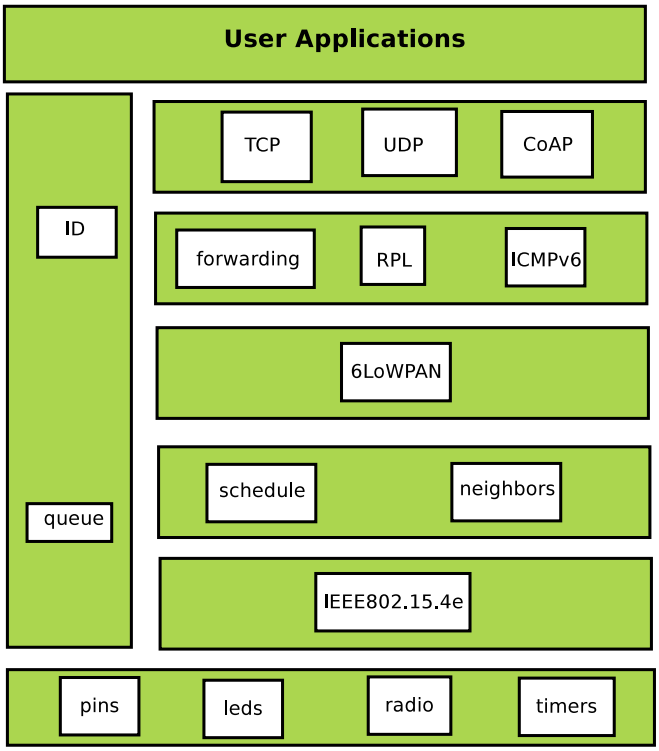


Fig. 5. The OpenWSN default protocol stack

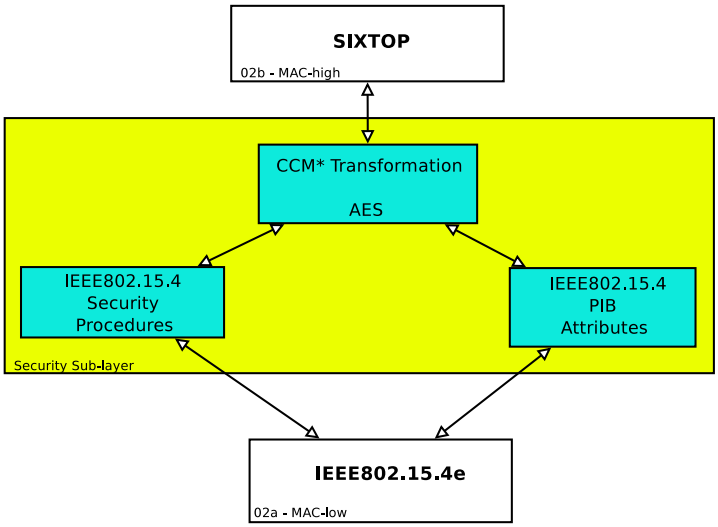


Fig. 6. The OpenWSN modified portion of the stack

TABLE III. PLATFORM RUNNING THE OPENWSN PROTOCOL STACK

Hardware Platform	Manufacturer	Architecture	Maximum Speed	Flash	RAM	Radio Module
TelosB	Texas Instruments	16-bit	8 MHz	48 kB	10 kB	CC2420
GINA	Texas Instruments	16-bit	16 MHz	116 kB	8 kB	AT86RF231
WSN430	SensLab	16-bit	8 MHz	48 kB	10 kB	CC1101 or CC2420
Z1	Zolertia	16-bit	16 MHz	92 kB	10 kB	CC2420
OpenMote STM	Texas Instruments	32 bit	72 MHz	256 or 512 kB	up to 64 kB	AT86RF231
OpenMoteCC2538	Texas Instruments	16-bit	32 MHz	up to 512 kB	up to 32 kB	CC2538
STM32F103RE	ST Microelectronics	32-bit	72 MHz	512 kB	64 kB	AT86RF231
K20	FreeScale	32-bit	72 MHz	256 kB	64 kB	AT86RF231
MC1321x	FreeScale	8-bit	8 MHz	up to 60 kB	up to 4 kB	689S08A
EZ430-RF2500	Texas Instruments	16-bit	16 MHz	up to 32 kB	2 kB	CC2500

OpenQueueEntry_t structure, which models the data packet, by adding the following variables:

- *l2_security*, reporting if security features are enabled or not for that packet (data type = *boolean*);
- *l2_securityLevel*, which specifies the security level of the frame (data type = *uint8_t*);
- *l2_keyIdMode*, indicating the *KeyIdMode* value to store into the MAC header (data type = *uint8_t*);
- *l2_keyIndex*, indicating the *KeyIndex* value to store into the MAC header (data type = *uint8_t*);
- *l2_frameCounter*, storing the frame counter (data type = *uint32_t*);
- *l2_keySource*, indicating the *KeySource* value to store into the MAC header (data type = *open_addr_t*);
- *l2_authenticationLength*, storing the length of the authentication field to append to the packet (data type = *uint8_t*);

To enable security features at the MAC layer, *Outgoing Frame Security*, *Incoming Frame Security*, *Key Descriptor Lookup*, *Device Descriptor Lookup*, *Security Level Descriptor*, *Incoming Security Level Checking*, and *Incoming Key Usage Policy Checking* procedures have been implemented as imposed by the IEEE 802.15.4 standard and described in sec. II. They are handled at the MAC-high layer of the protocol stack (in particular within the *SIXTOP* module, as depicted in Fig. 6), and are also in charge to built the *Auxiliary Security Header* and retrieve data from it (see Sec. II for more details). Encryption and decryption operations have been implemented in *Input Transformation*, *Authentication Transformation*, *Encryption Transformation*, *Decryption Transformation*, and *Authentication Checking Transformation* functions.

Finally, with respect to cryptographic primitives, we included in our module a free and open-source implementation of the AES-128 algorithm[1].

8) Usage of the developed module: The developed module is perfectly integrated within the overall OpenWSN protocol stack.

When the *SIXTOP* module of the MAC-high layer receives from upper layers a packet with the *l2_security* flag set to TRUE, it invokes, as described in [6], the implemented *OutgoingFrameSecurity* procedure by providing the *l2_securityLevel*, the *l2_keySource*, the *l2_keyIndex* and the Next Hop address.

In the case the message is received from the physical interface and the *l2_security* flag is set to TRUE, instead, the *taskSixtop_NotifReceive* function (that is called during the decapsulation process) delivers the message to the *IncomingFrameSecurity* procedure.

Finally, the initialization of MAC PIB attributes reported in Tab. IV should be statically handled by the user by setting their values in the *security_init* function, which is executed into the OpenWSN stack boot-up process (i.e., *OpenWSN_init* function).

IV. EXPERIMENTAL TESTS

We evaluated, through real experiments, the impact that the enabling of security features has on both computational requirements and communication latencies. To this end, we configured a real testbed made up of 2 motes, i.e., a coordinator and a child node (see Fig. 7), that exchange messages with variable size ranging from 117 Bytes (i.e., the minimum allowed value) and 127 Bytes (i.e., the maximum allowed value).

In our experiments, we used the TelosB hardware platform [9]. Despite its very limited capabilities (16-bit microcontroller working at a maximum speed of 8 MHz, 48 kB Flash Memory, 10 kB RAM, and CC2420 radio module), it is highly used in todays' research to evaluate protocols and algorithms in LLN environments with extreme constraints.

We configured the *security_ init* function in order to:

- select the 5-th security level, which provides both encryption and authentication services with a key of 32 bit;
- store a *Pre-Shared Key*;
- create a *KeyDescriptor* associated to the aforementioned key. It will be composed by (i) a *m_keyIdLookupDescriptor*, in which *keyIdMode* is set to

[1]It is available at *http://comp.ist.utl.pt/ec-csc/Code/Ciphers/AES_Encrypt.cpp*

TABLE IV. DESCRIPTION OF IEEE 802.15.4 MAC SECURITY ATTRIBUTES IMPLEMENTED IN OPENWSN

Implemented Parameter	Description	Composition
m_deviceDescriptor	structure modeling the *DeviceDescriptor* attribute	device address (data type = *open_addr_t*), Frame Counter (data type = *integer*), Exempt (data type = *boolean*)
m_keyIdLookupDescriptor	structure modeling the *KeyIdLookupDescriptor* attribute	Key Identifier Mode (data type = *uint8_t*), Key Index (data type = *uint8_t*), Key Source (data type = *open_addr_t*), PAN ID (data type = *open_addr_t*) and device address (data type = *open_addr_t*)
m_securityLevelDescriptor	table modeling the *SecurityLevelDescriptor* attribute	*DeviceOverrideSecurityMinimum* (data type = *boolean*), Frame Type (data type = *uint8_t*), Command Frame Identifier (data type = *uint8_t*), Security Minimum (data type = *uint8_t*), and Allowed Security Levels (data type = *uint8_t*)
m_keyUsageDescriptor	structure modeling the *KeyUsageDescriptor* table	*FrameType* (data type = *uint8_t*) and *CommandFrameIdentifier* (data type = *uint8_t*)
m_macDeviceTable	table modeling the *MacDeviceTable* attribute	vector of *m_deviceDescriptor* elements
m_keyDescriptor	structure modeling the *KeyDescriptor* attribute	*m_keyIdLookupDescriptor* element, *DeviceDescriptorHandleList* (i.e., pointer to the *m_macDeviceTable*), *m_keyUsageDescriptor* element, and key (data type = vector of *uint8_t*)
m_macKeyTable	table modeling the *MacKeyTable* attribute	vector of *m_keyDescriptor*, elements
m_macSecurityLevelTable	table modeling the *MacSecurityLevelTable*	vector of *m_securityLevelDescriptor* elements
m_macFrameCounter	variable modeling the *MacFrameCounter*	single variable (data type = *uint32_t*)
m_macDefaultKeySource	variable modeling the *MacDefaultKeySource*	single variable (data type = open_addr_t)

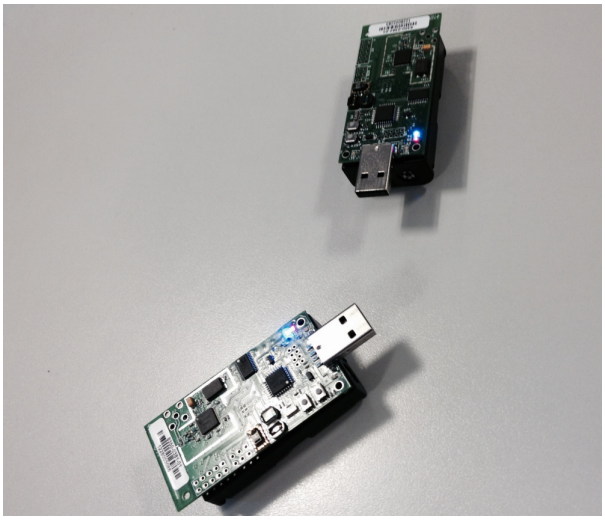


Fig. 7. Analyzed Testbed

$0x03$, *keyIndex* is set to $0x01$, *keySource* is set to the 64-bit address of the PAN Coordinator, *PANId* is set to the value of the PAN ID of the network, and *DeviceAddress* is set to NULL, (ii) a *m_keyUsageDescriptor* object, which has the *FrameType* set to $0x01$, and the *CommandFrameIdentifier* set to NULL, and (iii) and the pointer to the *m_macDeviceTable*. Such *KeyDescriptor* will be stored within the *MacKeyTable*.

- create a *m_deviceDescriptor* object storing information about the remote node (i.e., its MAC address, a *FrameCounter* initialized to 0, and the *Exempt* flag set to FALSE). It will be stored in the first entry of the *MacDeviceTable*;
- generate a *m_securityLevelDescriptor* object to append to the *MacSecurityLevelTable*. That entry will have the *FrameType* set to $0x01$ (i.e., Data Frame), the *CommandFrameIdentifier* set to NULL, the *SecurityMinimum* set to 5, the *DeviceOverrideSecurityMinimum* flag set to FALSE, and the *AllowedSecurityLevels* attribute set to 5;

In order to understand the processing cost of the cryptographic operations, we investigated the computational load introduced by encryption and decryption operations. The tests have been executed 100 times and average results are reported in Fig. 8. As expected, the time required to encrypt/decrypt a MAC packet increases with the packet size: when the MAC payload length increases, the duration of the cryptographic operations increases too, because of the higher amount of information to process. Anyway, encryption procedures always requires a time interval included in the range $[320, 380]$ ms.

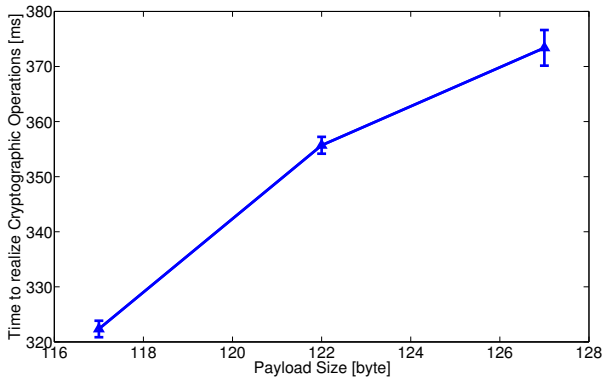
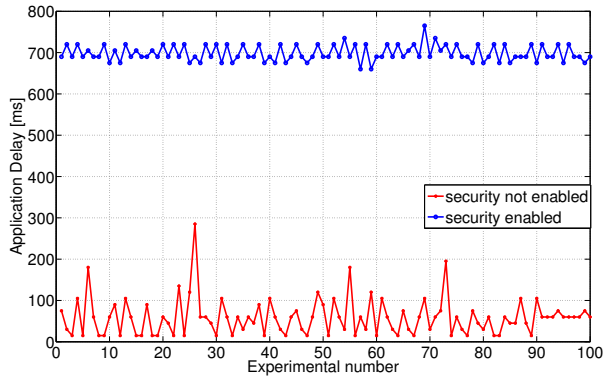


Fig. 8. Duration of cryptographic operations for different MAC payload size

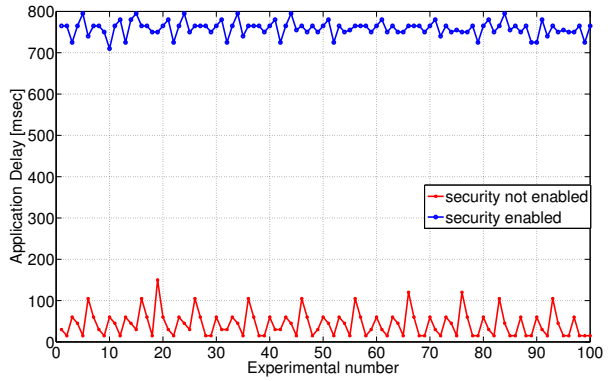
In order to evaluate the impact that security services have on communication latencies, we installed the *UDPLatency* application (i.e., a well-known application layer of the OpenWSN protocol suite) on the child node, which generate a fixed number of 100 packets to send to the coordinator. The packet generation time is set to 0.33 packets/s, i.e., a single packet every 3 seconds. In line with previous tests, we varied the payload size in the range [117-127] bytes.

We reported in Fig. 9 and Fig. 10 application end-to-end packet delays and its average value, measured in both secured and unsecured networks. As expected, communication

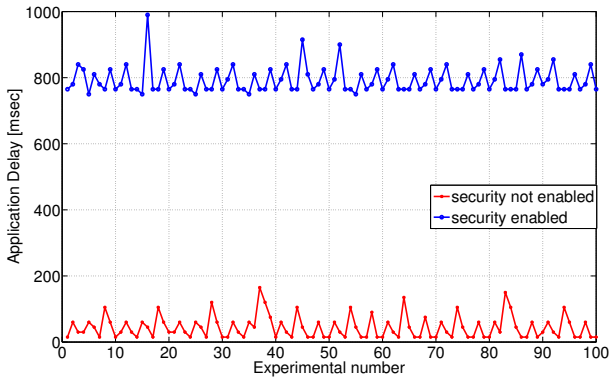
latencies increase with the packet size when security features are enabled. It is very important to remark that the highest impact on the end-to-end delays is provided by encryption and decryption operations (whose average value can be observed in Fig. 8).



(a)



(b)



(c)

Fig. 9. Communication latencies with MAC payload size of 117b(a), 122 b (b) and 127b (c) respectively.

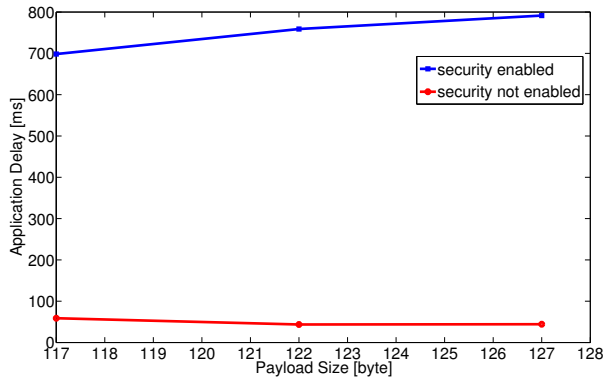


Fig. 10. Communication latencies for different payload size.

V. Conclusions

In this work we presented an open source and freeware implementation of IEEE 802.15.4 security features within the emerging OpenWSN protocols stack. Moreover, in order to investigate the impact of the developed module on normal operations, we also evaluated, through real experiments, the computational load and communication latencies. Our analysis demonstrated that computational capabilities of motes drastically influence the network operation. Without any doubts, these studies represent an important starting point for all researchers and industries interested to design and implement optimized solutions aimed at offering enhanced security services in Low Power and Lossy Network based on IEEE 802.15.4 radios. In this context, open source nature of the developed module motivates its adoption in this kind of research activities. In the future, we plan to evaluate the performances of security services in more complex networks, composed by a higher number of nodes arranged in various topologies, as well as to implement cryptographic procedures directly in hardware (thus leading to a strong reduction of the cryptographic time). In addition, using the developed module, we also plan the design of efficient Key Management Protocols, able to initialize and configure security attributes inside the network in a dynamic fashion.

References

[1] G. Piro, G. Boggia, and L. A. Grieco, "A standard compliant security framework for ieee 802.15.4 networks," in *Proc. of IEEE World Forum on Internet of Things (WF-IoT)*, Seoul, South Korea, Mar. 2014.

[2] O. Hersent, D. Boswarthick, and O. Elloumi, *The Internet of Things: Key Applications and Protocols*. Wiley, 2012.

[3] M. Palattella, N. Accettura, X. Vilajosana, T. Watteyne, L. Grieco, G. Boggia, and M. Dohler, "Standardized Protocol Stack for the Internet of (Important) Things," *Communications Surveys & Tutorials, IEEE*, 2012.

[4] D. Miorandi, S. Sicari, F. D. Pellegrini, and I. Chlamtac, "Internet of Things: Vision, Applications & Research Challenges," *Ad Hoc Networks*, 2012.

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

[6] IEEE std. 802.15.4, *Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs)*, Standard for Information Technology Std., 16 June 2011.

[7] *802.15.4e-2012: IEEE Standard for Local and Metropolitan Area Networks – Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs) Amendment 1: MAC Sublayer*, IEEE Std., 16 April 2012.

[8] T. Watteyne, X. Vilajosana, B. Kerkez, F. Chraim, K. Weekly, Q. Wang, S. D. Glaser, and K. S. J. Pister, "OpenWSN: a Standards-Based Low-Power Wireless Development Environment," *Transactions on Emerging Telecommunications Technologies*, vol. 23, no. 5, pp. 480–493, 2012.

[9] Telosb datasheet. [Online]. Available: http://www.willow.co.uk/TelosB_Datasheet.pdf