A standard compliant security framework for IEEE 802.15.4 networks

G. Piro, G. Boggia, and L. A. Grieco
Department of Electrical and Information Engineering (DEI)
Politecnico di Bari, Italy
Email: {g.piro, g.boggia, a.grieco}@poliba.it

***Abstract*—The IEEE 802.15.4 standard is widely recognized as one of the most successful enabling technologies for short range low rate wireless communications. It covers all the details related to the MAC and PHY layers of the protocol stack. In addition, it supports the possibility to protect MAC packets by using symmetric-key cryptography techniques and it offers several security options. But, at the same time, the standard relies on upper layers to orchestrate the usage of the plethora of security profiles and configuration settings it makes available, as well as to handle the creation and the exchange of encryption keys. In support of this functionality, this work describes a standard compliant security framework aimed at proposing: (i) different kind of security architectures, (ii) an efficient mechanism for initializing a secure IEEE 802.15.4 domain, and (iii) a lightweight mechanism to negotiate link keys among devices.**

***Index Terms*—IEEE 802.15.4, security framework, key management protocol.**

I. Introduction

The IEEE 802.15.4 standard defines the Physical (PHY) and the Media Access Control (MAC) layers for low-rate wireless personal area networks (LR-WPANs). It considers two types of network nodes that can build peer-to-peer or star networks: Full-Function Devices (FFDs) and Reduced-Function Devices (RFDs). A FFD works as the coordinator of the network and it is the reference device for all the other RFD nodes that have lower resource and communication capabilities [1][2]. The IEEE 802.15.4e specification introduces some amendments to the IEEE 802.15.4 standard. Among its key features there is the Time Synchronized Channel Hopping (TSCH), i.e., a novel MAC protocol, which better supports multi-hop communications in emerging industrial applications [3][1].

In its original version, the IEEE 802.15.4 standard already offer security services in low-power and lossy networks. In particular, it defines at the MAC layer a set of procedures and parameters (i.e., single variables and tables stored at this layer) that can be exploited to protect and authenticate packets by adopting techniques based on symmetric-key cryptography [2]. Moreover, all of these specifications remain unchanged for the IEEE 802.15.4e amendment. At the same time, both standards rely on upper layers to orchestrate, enable, configure, and negotiate security services, as well as to handle the creation and the exchange of encryption keys. For this reason, several research papers have investigated specific security aspects related to IEEE 802.15.4 network, including: lightweight authentication routines, optimized symmetric encryption algorithms, key agreement mechanisms, and protocols offering security features at the network layer (see for example solutions conceived in [4]-[7]).

However, the design of a one-size-fits-all solution, which embrace all the security facets of an IEEE 802.15.4 network (including the initialization and the dynamic management of a secure domain, as well as the contemporaneous support of different security configurations) is still an ambitious goal for researchers working in this area.

A first step in this direction has been moved by ZigBee IP specifications, i.e., a suite of high level communication protocols (such as 6LoWPAN, IPv6, PANA, RPL, TCP, TLS, and UDP) sitting on top of the IEEE 802.15.4 MAC, where both end-to-end and link-layer security, as well as a public key infrastructure based on X.509 certificates, is supported [8]. Unfortunately, ZigBee IP presents three main limitations: (i) the same link-key is shared among all nodes, thus making the network highly sensible to the presence of compromised devices, (ii) the same security level is used for all the services, and (iii) the cost needed to update the key in all devices increases with the size of the network

More recently, a new IETF WG, namely IPv6 over the TSCH mode of IEEE 802.15.4e (6TiSCH), has been defined to integrate the powerful MAC proposed within the IEEE 802.15.4e standard with all the IPv6-based upper layer protocols conceived for Low Power and Lossy Networks (LLNs). With respect to security aspects, this WG intends to define (i) the keying material and authentication mechanism needed by a new mote to join an existing network; (ii) a mechanism to allow for the secure transfer of application data between neighbor motes; and (iii) a mechanism to allow for the secure transfer of signaling data [9]. Furthermore, at the time of this writing, these challenging goals are investigating within three different Internet Drafts, i.e., [10], [11], and [12].
In line with these research activities, herein we present a security framework for the IEEE 802.15.4 MAC in low power lossy networks. networks. While maintaining a full compatibility with the IEEE 802.15.4 standard, the conceived framework supports five security configurations (i.e., *Fully Secured*, *Unsecured*, *Partial Secured*, *Hybrid Secured*, and *Flexible Secured*), an efficient mechanism for initializing a secure IEEE 802.15.4 domain, and a lightweight mechanism to negotiate link keys among devices.

The rest of the paper is organized as it follows: Sec. II provides a review of procedures and methodologies introduced within the IEEE 802.15.4 specifications to enable security services in Wireless Sensor Networks (WSNs); Sec. III describes the conceived framework; finally, Sec. IV draws conclusions and forecasts future research activities.

II. Security in IEEE 802.15.4 networks

To handle security features, the IEEE 802.15.4 standard defines a specific *Auxiliary Security Control* field within the MAC header. It is made by three sub-fields: the *Security Control*, which explains the security level and the key identification mode chosen by the sender; the *Frame Counter*, used to protect the message from replay attacks; and the *Key Identifier*, to identify the key for encrypting and authenticating the packet by using the *KeySource* and *KeyIndex* variables. Moreover, this information is present into the MAC frame whenever the *Security Enabled* flag in the *Frame Control* field of the MAC header is set to TRUE.

When a device intends to send or receive a protected MAC frame, it executes the *outgoing frame security* or the *incoming frame security* procedures, respectively, by exploiting security-related attributes stored at the MAC layer. In what follows, particular attention will be devoted only to a sub set of them, i.e., those that will be exploited for the design of the presented security framework, that is, the *macSecurityLevelTable*, the *macKeyTable*, and the *macDeviceTable*.

First of all, it is important to remark that 8 security levels have been defined by considering the following configurations: “unsecured” (i.e., level 0), “only authenticated” (i.e., from level 1 to level 3), “only encrypted” (i.e., level 4), and “encryption with authentication” (i.e., from level 5 to level 7). A specific security level should be guaranteed for each kind of message (i.e., beacon, command frame, data packet, and ACK). The related information is stored in the *macSecurityLevelTable*, made by a set of *SecurityLevelDescriptor* elements providing 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 dedicated key can be used for each remote device and for each type of MAC frame. To this aim, the *macDeviceTable* is created to store information about devices which a given node can interact with a secure communication. Finally, all keys are organized in the *macKeyTable*, where each *keyDescriptor* element 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* and *keyIndex*) which uniquely identify the key.

During the *outgoing frame security* procedure, the node identifies the key to use during the encryption process (this is done considering the *Key Identifier* announced by the upper layer), protects the MAC payload according to the selected *Security Level*, creates the *Auxiliary Security Control* field, and reassemble the whole packet. Instead, when a device receives a protected MAC frame, it identifies the key to exploit during decryption process, verifies that the *Security Level* chosen by the sender is correct, and decrypt the payload.

Unfortunately, despite the standard describes, with a high level of accuracy, procedures and parameters to be adopted for handling secured MAC frames, it does not clarify some crucial aspects, such as the initialization of a secure IEEE 802.15.4 domain, the generation and the exchange of keys, how to build the *macKeyTable*, and the definition of the way the entire system may react when a new device (that does not support security capabilities, or is not able to synchronize itself with the existing secure domain) wants to join the IEEE 802.15.4 network.

III. The proposed security framework

To resolve all the challenges described before, we conceived a complete and efficient security framework, which is fully compatible with the IEEE 802.15.4 specifications [2].

A. Envisaged security configurations

We envisaged five different security configurations:

- **Fully Secured network**: all packets are both encrypted and authenticated. Nodes that do not support security capabilities are not allowed to take part to the network.
- **Unsecured network**: security services are not supported.
- **Partial Secured network**: only the message integrity is enabled.
- **Hybrid Secured network**: the network can be composed by heterogeneous nodes that could or could not support security features. At the bootstrap phase, the network is created in an unsecured manner. All the non-unicast control messages sent by the coordinator should be transmitted in clear, thus ensuring that all devices are able to read the content of packets. A RFD node with security capabilities could negotiate link keys with its coordinator by following the same procedure as in *Fully Secured* mode.
- **Flexible Secured network**: as default, the network is set up with the *Fully Secured* configuration and all packets are encrypted and authenticated. If there is at least one node that does not support security capabilities, the coordinator could decide to switch to the *Hybrid Secured* configuration.

B. Minimum security requirements

The IEEE 802.15.4 standard imposes to specify, for each kind of MAC packet, minimum security levels that should be guaranteed. These restrictions must be detailed for each remote device. To this end, *SecurityMinimum*, *DeviceOverrideSecurityMinimum*, and *AllowedSecurityLevels* parameters are stored into each entry of the *macDeviceTable* (see Sec. II) to define the minimum security level, the possibility to override the minimum security level (i.e., *DeviceOverrideSecurityMinimum* is just a boolean flag), and the list of allowed security levels in the case the minimum one could be overridden, respectively. With reference to aforelisted secure network configurations, these parameters must be set as reported in Tab. I.

TABLE I

SECURITY-RELATED ATTRIBUTES TO SET IN EACH PROPOSED SECURE NETWORK CONFIGURATION.

Attribute	**Secure Network configuration**				
	Unsecured	**Fully Secured**	**Partial Secured**	**Hybrid Secured**	**Flexible Secured**
SecurityMinimum (minimum security level to be guaranteed)	0	from 5 to 7	from 1 to 4	0	from 1 to 7
DeviceOverrideSecurityMinimum (boolean flag: possibility to override minimum security level)	FALSE	FALSE	FALSE	FALSE	TRUE
AllowedSecurityLevels (allowed security levels when minimum lev. could be overridden)	0	from 5 to 7 It must be equal to the *SecurityMinimum* value.	from 1 to 4 It must be equal to the *SecurityMinimum* value.	from 0 to 7 All values are allowed.	from 0 to 7 All values are allowed.

The *Unsecured* network configuration does not support any security features. Hence, both minimum and allowable security levels are set to 0 for all the MAC frames and the possibility to override such constraints is disabled for all devices.
If the *Fully Secured* configuration is enabled, the minimum security level must be chosen in the range [5,7], thus allowing the possibility to support the encryption and the authentication of messages. The manufacturer must set the default value to 7; it can be updated by the network administrator. The minimum security level must not be overridden by any devices and, as a consequence, the field *AllowedSecurityLevels* should contain only one value, equal to the minimum security level.
If the *Partial Secured configuration* is enabled, the minimum security level must be chosen in the range [1,4], thus allowing the possibility to support the authentication of messages. The manufacturer must set the default value to 4; it can be updated by the network administrator. The minimum security level must not be overridden by any devices and, as a consequence, the field *AllowedSecurityLevels* should contain only one value, equal to the minimum security level.
If the *Hybrid Secured* configuration is enabled, the minimum security level must be set to 0, thus supporting the joining of devices having different security capabilities. All the security levels could be allowed and the network administrator could decide to enable only a subset of them according to the network design.
Finally, in the casethe *Flexible Secured* configuration is enabled, the minimum security level must be set to 1. The joining of nodes without (or with limited) security capabilities is permitted by setting the *DeviceOverrideSecurityMinimum* variable to TRUE and by including lower security levels in the list of *AllowedSecurityLevels*.

C. Initialization of a secured communication

The conceived framework builds a secured domain through the execution of three consecutive phases: Setting-up, Bootstrapping, and Key Negotiation. The implementation of the Bootstrap Phase is different for both FFD and RFDdevices. Moreover, for a remote mote, two different procedure has been conceived for both the not-beacon-enabled and the beacon-enabled networks.

1) Setting-up phase: During this phase the device is configured to properly operate in a IEEE 802.15.4 network. As suggested in [13], we assume that all nodes know algorithms, procedures, and initial secrets (i.e., the *masterKey*) that will be used to set up the secure domain. These information can be directly configured by the manufacturer or updated by the network administrator. In particular, the *Master Key* can be used to generate two different keys: the *DefaultKey*, exploited to protect *broadcast* messages (i.e., the beacon frame) and the *LinkKey*, used to encrypt and authenticate *unicast* packets (i.e., those exchanged between only two specific nodes).

2) Bootstrap phase for a FFD: A Personal Area Network (PAN) is initialized by a FFD node [2] by following the procedure summarized in Fig. 1. In particular, the MAC entity starts scanning the channel (with the aim of discovering the presence of other active coordinators and identifying the portion of the spectrum which it could operate in) after the reception of the *MLME-START.request* primitive, generated by the so called *Next Higher Layer*. Then, it will answer with a *MLME-START.confirm* primitive reporting a SUCCESS status. From this moment on, the device behaves as the PAN coordinator and it is able to select the identification number for the PAN, i.e., the PAN_{ID}, and the short MAC address of the IEEE 802.5.4 network, as well as defining initial secret materials.

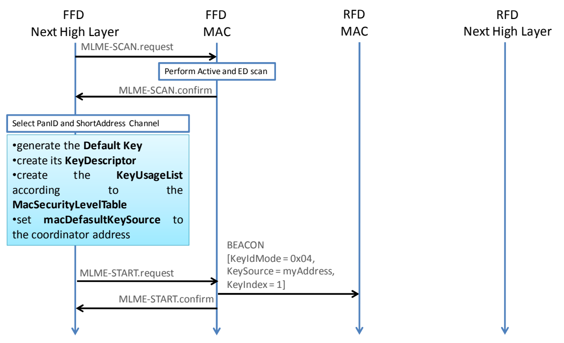
FFD
Next High Layer
FFD
MAC
RFD
MAC
RFD
Next High Layer
MLME-SCAN.request
Perform Active and ED scan
MLME-SCAN.confirm
Select PanID and ShortAddress Channel
•generate the **Default Key**
•create its **KeyDescriptor**
•create the **KeyUsageList** according to the **MacSecurityLevelTable**
•set **macDefasultKeySource** to the coordinator address
BEACON
[KeyIdMode = 0x04,
KeySource = myAddress,
KeyIndex = 1]
MLME-START.request
MLME-START.confirm

Fig. 1. Bootstrap phase of a secured FFD device.

The *DefaultKey*, D_k, is generated starting from the *MasterKey*, M_k, by using a 128-bit hash function, $H_{128}\{\cdot\}$[1]:

$$D_k = H_{128}\{PAN_{ID}|shortMACaddress|M_k\}.$$

All the parameters explored by the hash function are available as plaintext into the MAC header of the beacon message;

[1]Note that this solution is just an example of a possible way for generating D_k. It can be obviously personalized by the network administrator, making sure to store the formula in all devices.

thus, they can be extracted by any other device. Then, the FFD creates the *keyDescriptor* associated to the *DefaultKey*, D_k, putting it into the *keyMacTable*. The $KeyIdMode$, the $KeySource$, and the $KeyIndex$ variables of this *KeyDescriptor* element are set to $0x03$, the address of the device, and 1, respectively. Taking into account restrictions imposed by the *macSecurityLevelTable*, the *KeyUsageList* should be generated in order to use this key for all control packets. Finally the *macDefaultKeySource* is set equal to the MAC address of the device. From this moment on, the FFD device is able to encrypt and/or decrypt packets.

3) Bootstrap phase for a RFD: To join the network, the RFD device should associate itself with the coordinator. The *Next Higher Layer* sends to the MAC entity the *MLME-ASSOCIATE.request* primitive, starting the association phase (see Fig. 2). The RFD should generate the *DefaultKey* by extracting some parameters from the beacon message sent by the coordinator, such as the PAN ID, the short MAC address, and so on. Note that in the presence of the not-beacon-enabled scheme, the RFD device has to explicitly request its generation to the coordinator. The Beacon Request packet could be protected using an ephemeral key, ϕ_k, obtained from the *MasterKey*, M_k and the source address of the RFD node, as:

$$\phi_k = H_{128}\{MACaddress|M_K\}.$$

The $KeyIdMode$ of the Beacon Request packet is set to $0x00$ for enabling the coordinator to implicitly obtain the ephemeral key. After receiving the Beacon from the coordinator, the RFD node reads the MAC header and generates the *DefaultKey*, D_k. The associated *keyDescriptor* and the *macDefaultKeySource* variable are set as for the FFD device.

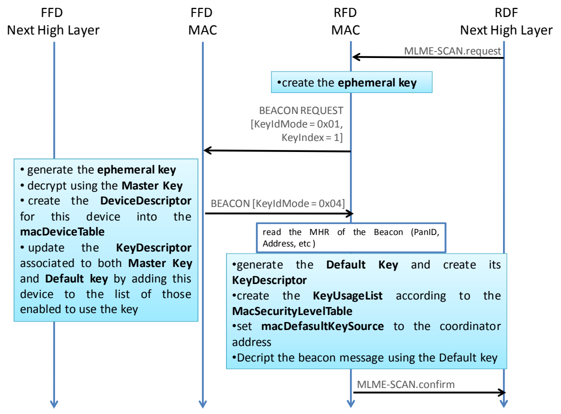
FFD
Next High Layer
FFD
MAC
RFD
MAC
RDF
Next High Layer
MLME-SCAN.request
•create the ephemeral key
BEACON REQUEST
[KeyIdMode = 0x01,
KeyIndex = 1]
• generate the ephemeral key
• decrypt using the Master Key
• create the DeviceDescriptor for this device into the macDeviceTable
• update the KeyDescriptor associated to both Master Key and Default key by adding this device to the list of those enabled to use the key
BEACON [KeyIdMode = 0x04]
read the MHR of the Beacon (PanID, Address, etc)
•generate the Default Key and create its KeyDescriptor
•create the KeyUsageList according to the MacSecurityLevelTable
•set macDefasultKeySource to the coordinator address
•Decript the beacon message using the Default key
MLME-SCAN.confirm

Fig. 2. Bootstrap phase of a secured RFD device.

4) Key Negotiation phase: Since resource-constrained devices are unable to perform complex algorithms and protocols, a simple key agreement protocol has been designed. It makes use of a set of new command messages (that do not exceed the maximum packet size defined into the standard), which are identified with a *FrameType* set equal to $0xAA$ and are composed by four different fields: *KeyGenControlField*, *R*, *KeyMaterial*, and *AuthenticationField*. The *KeyGenControlField* (2 bytes long) stores details about the content of the message:

- the $KeyGenMode$ (2 bits long) describes the algorithm adopted for key generation. It is set to 00, 01, and 10 if the procedure does not generate any *LinkKeys* (i.e., it just verifies the mutual authentication between devices), exploits the Rivest-Shamir-Adleman (RSA) and the Diffie Hellman (DH) algorithm, respectively. The value 11 is reserved and can be used for future upgrades.
- The $MessageType$ (2 bits long) identifyes the type of message exchanged during the procedure. $MessageType = 00$ is used for initializing the procedure; $MessageType = 01$ is used by the coordinator to inform the remote device to change the key generation algorithm; $MessageType = 10$ is used to deliver key or authentication materials; $MessageType = 11$ is reserved for future upgrades.
- The boolean $KeyFlag$ (1 bit long) indicates the presence of the key materials into the message.
- The boolean $AuthFlag$ (1 bit long) indicates the presence of the authentication field.
- The $KeySize$ (10 bits long) reports the size of the transported key material.

The *R* field (2 bytes long) contains a random value used for generating the *LinkKey*, L_k, and for verifying the authenticity of the remote device. The*KeyMaterial* field (0/23/64 bytes long) contains RSA or DH parameters, and *AuthenticationField* field (0/16 bytes long) is used for verifying the authenticity of the remote device.

An example of the negotiation procedure is reported in Fig. 3. Note that the last two messages are encrypted with the *LinkKey* to ensure the mutual authentication.

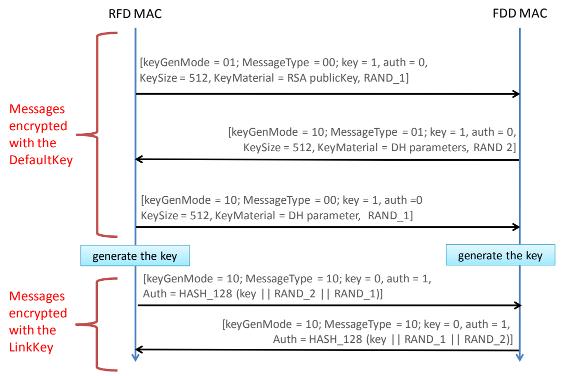
RFD MAC
FDD MAC
[keyGenMode = 01; MessageType = 00; key = 1, auth = 0,
KeySize = 512, KeyMaterial = RSA publicKey, RAND_1]
Messages encrypted with the DefaultKey
[keyGenMode = 10; MessageType = 01; key = 1, auth = 0,
KeySize = 512, KeyMaterial = DH parameters, RAND 2]
[keyGenMode = 10; MessageType = 00; key = 1, auth =0
KeySize = 512, KeyMaterial = DH parameter, RAND_1]
generate the key
generate the key
[keyGenMode = 10; MessageType = 10; key = 0, auth = 1,
Auth = HASH_128 (key || RAND_2 || RAND_1)]
Messages encrypted with the LinkKey
[keyGenMode = 10; MessageType = 10; key = 0, auth = 1,
Auth = HASH_128 (key || RAND_1 || RAND_2)]

Fig. 3. Example of the key negotiation procedure.

The RSA or the DH algorithm generates the *PreLinkKey*, P_k, which is used to generate key material (i.e., the list of *LinkKeys*) as described in the following:

$$\begin{aligned} \textit{LinkKeys} \quad = \quad & H_{128}\{``AAA''|PAN_{ID}|P_k\}|| \\ & H_{128}\{``BBB''|PAN_{ID}|P_k\}|| \\ & H_{128}\{``CCC''|PAN_{ID}|P_k\}\ldots \end{aligned}$$

5) Switching from the Flexible Secured to the Hybrid Secure configuration: The switching from the *Flexible Secured* to the *Hybrid Secure* configuration could be performed following the procedure in Fig. 4. During the association phase, a device without security capabilities sends to the coordinator a Beacon Request message with the *SecurityEnabled* flag set to FALSE. The FFD device then switches to the *Hybrid Secure* configuration and update all the MAC security attributes accordingly. From this moment on, the coordinator sends control messages in clear.

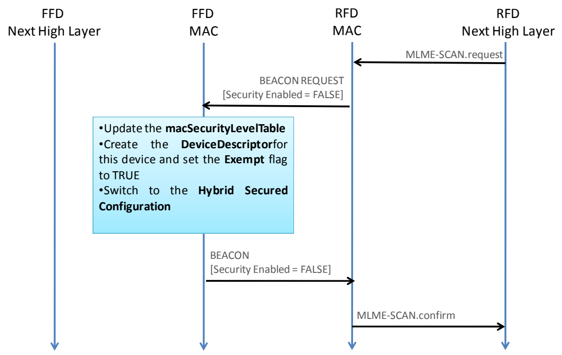
FFD Next High Layer
FFD MAC
RFD MAC
RFD Next High Layer
MLME-SCAN.request
BEACON REQUEST [Security Enabled = FALSE]
•Update the macSecurityLevelTable
•Create the DeviceDescriptor for this device and set the Exempt flag to TRUE
•Switch to the Hybrid Secured Configuration
BEACON [Security Enabled = FALSE]
MLME-SCAN.confirm

Fig. 4. Switching from *Flexible Secured* to the *Hybrid Secure* configuration.

IV. Conclusion and Future works

In this paper we presented a standard compliant security framework aimed at initializing and maintaining secure IEEE 802.15.4 networks. We firstly envisaged some possible secure configurations that can be setup in a low-power and lossy network. Then, we deeply described how each of them can be setup and how nodes can generate and exchange keys that have to be used for protecting MAC packets. In the near future, we will implement the devised framework in real devices, thus being able to evaluate its performance in terms of computational and energy requirements. In addition, we will also investigate the possibility to extend our proposal also to any other upper layers (e.g., routing protocol), thus designing a complete and optimized security architectures for the emerging Internet of Thing paradigm.

V. Acknowledgments

This work was supported by the PON projects (RES NOVAE, ERMES-01-03113, DSS-01-02499 and EURO6-01-02238) funded by the Italian MIUR and by the European Union (European Social Fund).

References

[1] M. R. Palattella, N. Accettura, X. Vilajosana, T. Watteyne, L. A. Grieco, G. Boggia, and M. Dohler, "Standardized Protocol Stack For The Internet Of (Important) Things," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 3, pp. 1389–1406.

[2] IEEE std. 802.15.4, *Part. 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks(LR-WPANs)*, Standard for Information Technology Std., Jun. 2011.

[3] ——, *IEEE Standard for Local and Metropolitan Area Networks – Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs) Amendment 1: MAC Sublayer*, Institute of Electrical and Electronics Engineers Std., Apr. 2012.

[4] R. Daidone, G. Dini, and M. Tiloca, "On experimentally evaluating the impact of security on IEEE 802.15.4 networks," in *Proc. of IEEE Int. Conf. on Distributed Computing in Sensor Systems and Workshops, DCOSS*, Barcelona, Spain, Jun. 2011, pp. 1–6.

[5] J. Mišic, "Cost of secure sensing in IEEE 802.15.4 networks," *IEEE Trans. Wireless Comm.*, vol. 8, no. 5, pp. 2494–2504, May 2009.

[6] W. Bechkit, Y. Challal, A. Bouabdallah, and V. Tarokh, "A highly scalable key pre-distribution scheme for wireless sensor networks," *IEEE Tran. on Wireless Commun.*, vol. 12, no. 2, pp. 948–959, 2013.

[7] L. B. Oliveira, A. Kansal, C. P. Gouvêa, D. F. Aranha, J. López, B. Priyantha, M. Goraczko, and F. Zhao, "Secure-tws," *Comput. J.*, vol. 55, no. 4, pp. 384–396, Apr. 2012.

[8] ZigBee IP Specification, Available online: http://www.zigbee.org/Specifications/ZigBeeIP/.

[9] Watteyne, T. and Palattella, M.R. and Grieco, L.A., *Using IEEE802.15.4e TSCH in an LLN context: Overview, Problem Statement and Goals draft-ietf-6tisch-tsch-00 (work in progress)*, IETF 6TiSCH WG, Nov. 2013.

[10] S. Chasco, S. Das, R. Marin-Lopez, Y. Ohba, P. Thubert, and A. Yegin, *Security Framework and Key Management Protocol Requirements for 6TSCH draft-ohba-6tisch-security-00 (work in progress)*, IETF 6TiSCH WG, Jul. 2013.

[11] G. Piro and G. Boggia and L. A. Grieco, *A standard compliant security framework for Low-power and Lossy Networks draft-piro-6tisch-security-issues-01 (work in progress)*, IETF 6TiSCH WG, Oct. 2013.

[12] M. Richardson, *Security architecture for 6top: requirements and structure draft-richardson-6tisch-security-architecture-00 (work in progress)*, IETF 6TiSCH WG, Dec. 2013.

[13] O. Garcia-Morchon, S. Keoh, S. Kumar, R. Hummen, and R. Struik, "Security Considerations in the IP-based Internet of Things," IETF, Internet Draft, Mar. 2012.