LICITUS: a lightweight and standard compatible framework for securing layer-2 communications in the IoT

S. Sciancalepore, G. Piro, E. Vogli, G. Boggia, L. A. Grieco, G. Cavone

Dep. of Electrical and Information Engineering (DEI), "Politecnico di Bari", v. Orabona 4, 70125, Bari, Italy; e-mail: {name.surname}@poliba.it.

Abstract

With reference to the IEEE 802.15.4 standard, many solutions have been formulated to face the different facets of layer-2 security. Unfortunately, the opportunities and subtleties arising from their joint adoption has been not investigated, due to the lack of an integrating framework. To this end, hereby a novel standard compatible framework is proposed, which is able to orchestrate several layer-2 security mechanisms with a limited computational footprint. Conceived as a distributed scheme, it covers the following key features: (i) multiple security configurations in homogeneous and heterogeneous scenarios; (ii) adaption to dynamic networks; (iii) lean and scalable initialization functionalities; (iv) lightweight Key Management Protocol; and (v) resilience to several attacks. The robustness against security attacks have been evaluated through a well-known automatic cryptographic protocol verifier, namely ProVerif. Moreover, to further demonstrate its effectiveness, the proposed framework has been implemented within the emerging OpenWSN protocol stack, experimentally evaluated, and compared with respect to the ZigBee IP security architecture, which integrates the Symmetric Key - Key Establishment protocol (SKKE). Results clearly show that, although security features in constrained nodes incur not negligible computational costs (which impair latencies and energy efficiency), the proposed approach always guarantees better performances with respect to the ZigBee IP security architecture. In fact, it speeds up the configuration of

security services (up to 120%), while ensuring relevant energy savings (larger than 50%).

Keywords: IoT, IEEE 802.15.4, Security, Key negotiation, Experimental evaluation.

---

1. Introduction

With the advent of the Internet of Things (IoT), billions of objects will be shattered almost everywhere to enable smart services in relevant application domains, such as health care, logistics, energy management, military, environmental monitoring, and industry-automation, to name a few [1]. However, a globally interconnected network of physical objects inevitably results in a (potentially great) surface that can be easily exploited if not adequately protected [2]. In order to magnify the resilience of protocol architectures to attacks, it is necessary to support data confidentiality, peer authentication, and access control [3].

By observing IoT systems from the bottom, securing communications at layer-2 represents the first important countermeasure to security threats. In this context, the IEEE 802.15.4 standard emerged as the leading enabling technology for short range low rate wireless communications [4]. In fact, its Media Access Control (MAC) and Physical (PHY) layers are embraced in both ZigBee and IETF specifications for Low-power and Lossy Networks (LLNs) [1]. This standard allows to protect MAC packets by means of symmetric-key cryptography techniques and several security options. Nevertheless, it does not directly explain how to handle the initialization of a secure domain, the generation of layer-2 keys, as well as the orchestration of security services in a wide gamut of network configurations.

Of course, such important aspects have been already investigated in literature and standardization bodies [2][5]. However, a deep experimental analysis (with cross comparisons), aimed at evaluating the impact that security features have on latencies and energy consumptions is mostly missing. In addition, to

the best of authors' knowledge, no contribution explored so far the opportunities and subtleties that could arise from a joint integration of existing solutions, in a standard compatible way.

To bridge this gap, this work designs, implements, and experimentally evaluates (for the first time) an integrating framework, namely Lightweight sCheme for IoT secUre communicationS (LICITUS), that harmonizes and orchestrates, the functionalities of consolidated approaches to *Security Configuration*, *Bootstrap*, and *Key Negotiation* phases. The resulting system offers: (i) a rich set of possible security configurations that can be enabled also in heterogeneous scenarios where protected communications may coexist with unsecured ones; (ii) an effective methodology for easing the initialization of secure IEEE 802.15.4 networks; (iii) a lightweight Key Management Protocol (KMP) for negotiating a layer-2 key between a node pair; (iv) resilience to several security attacks, and (v) a full compatibility with IEEE 802.15.4 technology.

LICITUS has been conceived as a distributed framework: starting from a set of cryptographic materials and configuration variables stored by the manufacturer or updated by the system administrator, each node is able to autonomously bootstrap security services and negotiate a link-level key with its neighbors without requiring the intervention of any remote and centralized server. Therefore, it natively promises good levels of scalability, also in high loaded scenarios.

As a preliminary analysis, the resilience of LICITUS against password guessing, replay, and Man-In-The-Middle (MITM) attacks has been proved through the ProVerif tool, i.e., a widely accepted automatic cryptographic protocol verifier[1]. Then, to demonstrate it effectiveness, LICITUS has been implemented within the emerging OpenWSN protocol stack [6] and experimentally evaluated using the TelosB platform[2].

A comparison with respect to the reference ZigBee IP security architecture, which integrates the Symmetric Key - Key Establishment (SKKE) protocol

[1] http://prosecco.gforge.inria.fr/personal/bblanche/proverif/
[2] http://www.memsic.com/userfiles/files/Datasheets/WSN/telosb_datasheet.pdf

for handling the key agreement mechanism [7], has been also provided. Note that the considered ZigBee IP security configuration has been chosen as a valid benchmark strategy for comparison because it shares many features with the proposed framework and, in particular, the ability to establish a secure network domain in which node pairs negotiate symmetric keys. Experimental results show that the introduction of security services generally incurs a not negligible computational effort, which worsens communication latencies and energy efficiency. However, they also demonstrate that LICITUS is able to overcome the SKKE protocol of the ZigBee IP security architecture by speeding up the configuration of security services (up to 120%) and ensuring relevant energy savings (larger than 50%).

It is worth to note that, in authors' humble opinion, this work not only represents an important advancement to the state of the art, but it completely fits within the activity of the IETF IPv6 over the TSCH mode of IEEE 802.15.4e (6TiSCH) Working Group (WG) [8]. In fact, it remarkably extends preliminary works proposed by the same authors of this paper in [9][10] by: (i) detailing components and functionalities of the devised security framework, (ii) evaluating its performance through an extended experimental campaign, and (iii) providing a cross comparison with another well-known solution. In addition, the open source implementation of LICITUS is freely released[3] and shared with all researchers and practitioners working on layer-2 security-related aspects. Therefore, this work can trigger relevant implications in academia, industrial, and standardization fields.

The rest of the paper is structured as it follows: Sec. 2 presents some of the most important techniques nowadays available for securing layer-2 communications in IoT systems. The proposed LICITUS framework is presented in Sec. 3. The experimental evaluation, the comparison with respect to the ZigBee IP security architecture, and the security analysis are discussed in Sec. 4. Finally, Sec. 5 closes the paper and draws future works.

[3]The code is available from http://telematics.poliba.it/security-iot.

2. Layer-2 security in emerging IoT systems

IEEE 802.15.4 specifications [11]-[13] allow to protect MAC packets by means of symmetric-key cryptography techniques, based on the well known AES-CCM* algorithm. Moreover, to support several security options, the standard also specifies eight possible configurations:

- level 0: packets are transmitted in clear and link-level communications are unsecured;
- level 1: packets are only authenticated using the Advanced Encryption Standard (AES) algorithm in Cipher Block Chaining (CBC) mode, with a Message Integrity Code (MIC) that is 4-bytes long;
- level 2: packets are only authenticated using the AES algorithm in CBC mode, with a MIC that is 8-bytes long;
- level 3: packets are only authenticated using the AES algorithm in CBC mode, with a MIC that is 16-bytes long;
- level 4: packets are only encrypted using the AES algorithm in Counter (CTR) mode;
- level 5: packets are both encrypted and authenticated using the AES-Counter with CBC MAC (CCM) algorithm, with a MIC that is 4-bytes long;
- level 6: packets are both encrypted and authenticated using the AES-CCM algorithm, with a MIC that is 8-bytes long;
- level 7: packets are both encrypted and authenticated using the AES-CCM algorithm, with a MIC that is 16-bytes long;

From one side, the standard describes, with a high level of accuracy, procedures and parameters to handle secured MAC frames. From another side, 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, the configuration of security-related parameters at the MAC layer, and the definition of how 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 network.

In what follows, a summary is proposed to draw the main features of the leading approaches proposed so far to cover the empty spaces left by the IEEE 802.15.4 standard.

First of all, the initialization of a secured link and the negotiation of symmetric keys is generally handled through the Key Management Protocol (KMP). In this context, either *centralized* or *distributed* solutions can be used. Centralized approaches, like those presented in [14]-[16], suppose the use of a dedicated entity for generating, distributing, and managing keys. In distributed techniques, instead, each couple of nodes negotiate key materials autonomously, as discussed in [17]-[23]. In particular, in [19], [18] and [20], the link key is calculated by using innovative techniques based on channel measurements. KMP schemes presented in [17] and [23] create a *group key* (also known as *cluster key*), which is shared into the network for different purposes (e.g., node authentication and message encryption). In [17], link keys are generated by means of pseudo-random functions, which receive as input variables a shared secret, i.e., the master key, and other common parameters characterizing the clustering structure of the Wireless Sensor Network. Note that distributed mechanisms presented in [17]-[20], [23], and presented before, are based on symmetric key encryption schemes. Further distributed approaches use asymmetric strategies, based on the usage of public key cryptography [10][14][23][24].

From the standardization perspective, the IETF 6TiSCH WG is particularly active on security-related topics [8]. It targets the definition of: (i) the keying material and authentication mechanism needed by a new node to join an existing network, (ii) a mechanism to allow the secure transfer of application data between neighbor nodes, (iii) a scheme to secure signaling data. In this context, two relevant Internet Drafts have been published. All the relevant elements for

the design of the 6TiSCH security architecture are discussed in [25] Instead, the description of a more complete security architecture for industrial environments, which covers minimal security features for both layer-2 and layer-4, is investigated in [26]. In particular, [26] proposes to adopt Datagram Transport Layer Security (DTLS)-based solutions for enabling secure communications at the application layer.

In the recent past, the ZigBee Alliance introduced a communication stack based on IEEE 802.15.4 radios, namely Zigbee IP [27]. With reference to security issues, Zigbee IP defines key distribution and joining procedures, that are managed, in a centralized fashion, by the *Trust Center* [28]. In particular, it introduces three different keys: the Master key, used to finalize authentication procedures; the Network key, for protecting management messages; and the Link key, adopted to protect the communication between two devices. Generally, both Master and Network keys can be preconfigured or delivered by the *Trust Center*. The Link Key can be either distributed by the *Trust Center* or negotiated between two devices under the supervision of the *Trust Center*. In details, the key agreement is handled by means of the SKKE protocol [7]. It uses a centralized scheme where the *Trust Center* always coordinates the negotiation of keys among any couple of nodes. As a consequence, it may be not really scalable, especially in large IoT systems, and it may suffer fault-tolerance issues because the *Trust Center* is a single point of failure [29].

To solve this important issue, the LICITUS framework presented in this work provides a distributed methodology to initiate and configure security services in a IEEE 802.15.4 network. Specifically, starting from a set of cryptographic materials and configuration variables stored by the manufacturer, each node is able to bootstrap security services (as described in Sec. 3.3) and negotiate link-level keys with its neighbors (as described in Sec. 3.4) without requiring the exchange of information with a remote and trusted node. In contrast with baseline centralized schemes, like ZigBee IP, this distributed approach immediately brings to both energy and airtime savings.

3. LICITUS: The proposed security framework

The Lightweight sCheme for IoT secUre communicationS (LICITUS) framework proposed hereby enables, configures, and manages layer-2 security services in a IEEE 802.15.4 network. As already stated in the Introduction, all of these operations are executed in a distributed manner. Starting from a set of cryptographic materials and configuration variables stored by the manufacturer or updated by the system administrator, in fact, each node is able to autonomously bootstrap security services and negotiate link-level keys with its neighbors without requiring the control of any remote and trusted server. Therefore, specific procedures have been defined for computing keys for protecting broadcast messages (generally used in the bootstrap stage) and keys for protecting unicast packets. In addition, LICITUS covers all the possible security configurations available in both homogeneous networks (i.e., all devices support security features) and heterogeneous ones (i.e., protected communications may coexist with unsecured ones), as well as allows dynamic updates of network settings based on the security capabilities of network devices.

To accomplish these challenging goals, each device belonging to the IEEE 802.15.4 network hosts an instance of the LICITUS framework, made of three main sub-systems, that are *Security Configuration*, *Bootstrap*, and *Key-negotiation managers* (see Figure 1). The *Security Configuration Manager* controls the level of security offered in the network, imposes minimum security requirements, and supports the execution of all the procedures handled by other components of the framework. It stores a set of security parameters and initial credentials, used for initializing the secure domain. The *Bootstrap Manager* executes the bootstrap procedure devoted to the initialization of the secured domain. Finally, the *Key-negotiation Manager* handles the KMP protocol and the algorithm that dynamically updates link keys during the time. In general, LICITUS can integrate different key negotiation schemes, like Diffie-Hellman (DH), Elliptic Curve Diffie Hellman (ECDH), Rivest-Shamir-Adleman (RSA), etc. [30][31]. However, to ease the description of procedures implemented by the *Key-negotiation Man-*

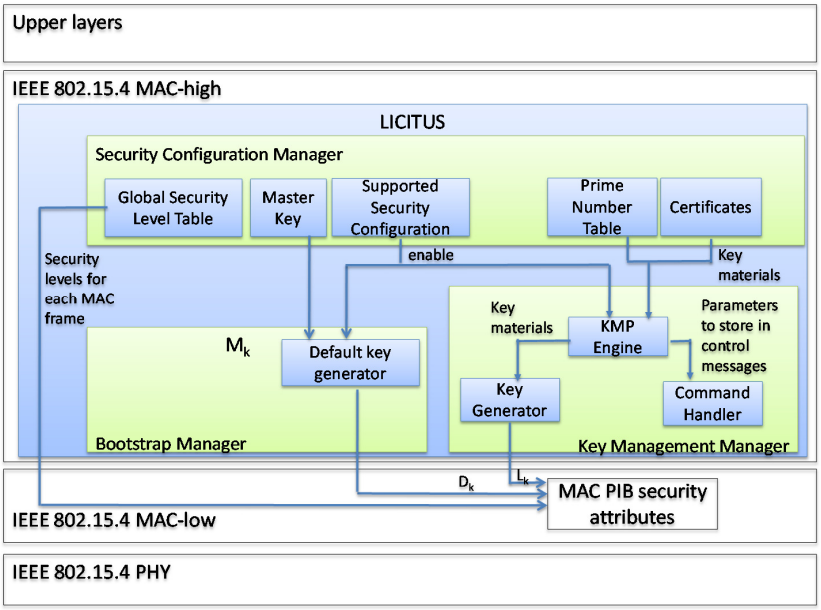


Figure 1: General scheme of LICITUS.

ager, the well-known DH algorithm will be considered from this moment on.

The following subsections provide a thorough description of LICITUS components, their interactions, and resulting security services.

3.1. Security configurations

LICITUS natively supports all the security levels already defined in the IEEE 802.15.4 standard (and summarized in Sec. 2). For instance, it can be used for setting up a network where messages are exchanged in clear (level 0 security option), a network where communications are protected with the highest security level (supported by the level 7 security option), or any other standard-compliant configuration. Also, according to the IEEE 802.15.4 standard, LICITUS allows multiple security options to be concurrently handled within the same network: a given data stream exchanged among a couple of devices may be potentially protected by a security level and a symmetric key that differ from those used for other kinds of broadcast/unicast communications.

From a security perspective, three different types of nodes can be found in

IoT systems: (i) devices that *have* software/hardware components for handling cryptography operations and *are* in possess of initial credentials and parameters set needed to join a secured network, (ii) devices that *have* software/hardware components for handling cryptography operations and *are not* in possess of initial credentials and parameters set needed to join a secured network, and (iii) devices that *have not* software/hardware components for handling cryptography operations. To simplify, it is possible to assume that a node presents *security capabilities* if it has software/hardware components for handling cryptography operations and is in possess of initial credentials and parameters set needed to join the secured network. In this regard, LICITUS also supports heterogeneous scenarios, where nodes with and without security capabilities may coexist within the same network. Differently from homogeneous networks, where all exchanged frames can be protected, heterogeneous scenarios should support both *protected* and *unprotected* communications.

Now, to embrace any possible combination of such security options and configure homogeneous and heterogeneous scenarios, LICITUS introduces four high-level security configurations, that are:

- **Unsecured network**: security services are not enabled and all packets are exchanged in cleartext. This configuration can be natively used in heterogeneous scenario because the possession of security capabilities is irrelevant.

- **Fully Secured network**: all packets are both encrypted and authenticated. This configuration only supports homogeneous scenarios and nodes without security capabilities cannot join the network.

- **Partially Secured network**: only the message integrity is enabled. Also in this case, this configuration only supports homogeneous scenarios and nodes without security capabilities cannot join the network.

- **Hybrid Secured network**: secure and unsecure communications may coexist. This configuration is introduced for better supporting heteroge-

neous scenarios, hosting with and without security capabilities. Since from the beginning, 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 can read the packet contents. Moreover, a couple of nodes with security capabilities can establish a link-level secure communication and exchange unicasts and protected packets.

The relationship between high-level security configurations defined in LICITUS and security options standardized in IEEE 802.15.4 is reported in Tab. 1.

Table 1: Relationship between high-level security configurations and IEEE 802.15.4 security options.

High-Level Security Configuration	**Security option used for broadcast messages (i.e., beacon)**	**Security option used for unicast messages**
Unsecured	Level 0 only	Level 0 only
Fully Secured	From Level 5 to Level 7, according to system administrator choices	From Level 5 to Level 7, according to system administrator choices
Partially Secured	From Level 1 to Level 4, according to system administrator choices	From Level 1 to Level 4, according to system administrator choices
Hybrid Secured	Level 0 only	From Level 1 to Level 7, according to system administrator choices and device capabilities

To provide a further insight, the conceived framework also supports the possibility to dynamically change the security configuration; in the sequel, this will be referred to as *flexibility feature*. In particular, when the *flexibility feature* is enabled, a homogeneous and secured network, that falls in both *Partially Secured* and *Fully Secured* configurations, may move to the *Hybrid Secured* mode. In this way, it is possible to enable the join of external nodes not supporting security capabilities or not in possess of the right initial credentials. In fact, when a node without security capabilities or initial credentials wish to join a

network (that is already operating in a secured fashion), it has just to send a beacon request in clear. Its corresponding coordinator processes the request and switches to the *Hybrid Secured* configuration only if the *flexibility feature* is enabled. From that moment on, the considered node may join the network, all the broadcast messages are transmitted in clear, and nodes supporting security capabilities still continue to exchange protected unicast packets with their parents. Of course, this feature is optional and it can be enabled or disable by the network administration according to the target design criteria.

Note that each of the aforementioned security configuration (including the *flexibility feature*) could be enabled by properly setting the *Security Level Table*, stored at the MAC level of the IEEE 802.15.4 standard [11]-[13].

3.2. Security Configuration Manager

As stated before, the *Security Configuration Manager* controls the level of security, imposes the corresponding minimum requirements, and supports all the procedures handled by other components of the framework. To this aim, the following parameters and initial credentials are stored:

- *Supported Security Configuration*: it represents the security configuration chosen for a given network among those in Sec. 3.1.

- *Master Key*: it is an initial secret shared among all the devices. Note that it is not directly used to encrypt/decrypt messages, but it is used, together with other time-varying parameters (that can be unique in each secured domain and periodically updated during the time) for generating all the required key materials. In particular, starting from the *Master Key*, two different keys are created: the *Default Key*, for protecting *broadcast* messages (i.e., beacon frames) and the *Link Key*, for encrypting and authenticating *unicast* packets (i.e., exchanged between a pair of specific nodes).

- *Global Security Levels Table*: it is a table that stores the minimum security level and the list of allowed security levels that should be adopted for

each kind of MAC frame and security configuration. It should be created considering the type of services offered by the IEEE 802.15.4 network. Annex A presents some examples showing how to set up the Global Security Levels Table in practical use cases, that refer to industrial scenario and smart campus.

- *Prime Numbers Table*: it is a set of N prime numbers and their respective primitive roots used during the *Key Negotiation* phase for generating *Link Keys* according to the DH algorithm. Each available prime number is identified by $N_p = \log_2(N)$ number of bits.

- The *private key* of the device.

- A *Certificate*: it stores the public key of the device in a standardized structure (e.g., X.509 certificate [32]).

- *Public key of a certification authority*: it is used for verifying the authenticity of device certificates.

The *Security Configuration Manager* can be directly configured by the manufacturer or by the administrator before the deployment of the network. To ensure the protection against tampering attacks, specific software-based and/or hardware-based mechanisms may be used for preventing the physical access to all the stored variables [33].

3.3. Bootstrap Manager

The *Bootstrap Manager* is the entity of the LICITUS instance running on a given device that is in charge of initializing security parameters needed for joining a secure domain. Specifically, it processes the list of variables stored in the Security Configuration Manager and sets up security-related parameters at the MAC layer. In addition, it also calculates the *Default Key*, D_k, used for protecting broadcast messages and unicast messages exchanged for negotiating layer-2 keys. The *Default Key*, D_k, is computed starting from the *Master Key*,

M_k, the coordinator MAC address, $MAC_{addr,c}$, and the network ID, PAN_{ID}, by using a 128-bit hash function, $H_{128}\{\cdot\}$:

$$D_k = H_{128}\{PAN_{ID}|MAC_{addr,c}|M_k\}. \quad (1)$$

The security level of the *Default Key* is strictly related to the specific hashing algorithm used to generate it. Without loss of generality, LICITUS may potentially adopt any kind of hash function for generating the 128-bit digests. From one side, it is suggested to use robust hash functions, thus making the entire framework resilient against attacks. From another side, instead, it is even important to adopt hash functions that do not require high computational capabilities (that are generally scarce in constrained nodes forming a IEEE 802.15.4 network).

As a final consideration, we remark that the time instant when the *Default Key* is computed is different for network coordinator and child node. In particular, while the first device can generate the key as soon as it becomes the coordinator of a given portion of the LLN, the child node should firstly receive the beacon messages (i.e., association phase), to extract the parameters needed for the computation of the *Default Key*. Note that this task can be handled without any problem because such parameters are stored in the MAC header and, hence, transmitted in clear.

3.4. Key Negotiation Manager

The *Key Negotiation Manager* implements all the functionalities for the definition of the link key used for layer-2 unicast communications. It is made up of three entities: the *KMP Engine*, which implements the key negotiation scheme to derive a *Pre Link Key*; the *Command Handler* that translates logical messages of the KMP in MAC messages; and the *Key Generator* that generates all layer-2 keys, starting from the *Pre Link Key*.

As well known, resource-constrained devices are unable to perform complex algorithms and protocols in a limited time, so that it is mandatory to implement simple and effective key agreement protocols [34]. For this reason, we develop a lightweight approach with limited computational and bandwidth requirements.

The KMP is implemented in a distributed manner: a couple of communicating nodes can negotiate a layer-2 key without needing to interact with any remote and trusted node. The resulting approach is potentially scalable and it does not brings to huge latencies in complex networks.
Now, without loss of generality, we can consider two devices willing to negotiate a layer-2 key: node A and node B. For instance, A could be a child node directly connected to the network coordinator and B could be the network coordinator. The algorithm is initialized by node A and it is executed by exchanging four different logical messages. The first two messages deliver the key materials to be used by each node involved into the key agreement scheme, for computing the *Pre Link Key* (i.e., a preliminary common secret which all the required link keys are generated from). The last two messages are used, instead, for finalizing the mutual authentication. Whereas the first two messages are protected (i.e., encrypted and/or authenticated) with the D_k, the last are always encrypted and/or authenticated by means of the *Pre Link Key* itself.

From the implementation point of view, these messages are stored within specific *Header Information Elements*, that are fields of the MAC header defined by the IEEE 802.15.4 standard for carrying customized information in a standard compliant way. During the execution of the KMP protocol, the *Command Handler* entity of the LICITUS framework translates high-level commands in specific link-level messages.

The KMP supports both anonymous and certified DH schemes. It consists of six consecutive steps (see Figure 2).

- **Step 1**: The node A sends it public key, $P_{b,A}$, and a nonce, R_A, to the node B. While the former parameter is used for negotiating the key with the remote node, the latter is used, instead, for handling the mutual authentication.

  In the case of adoption of the anonymous DH, a prime number, P, and the corresponding primitive root, g, are firstly extracted from the *Prime Numbers Table* by considering the latest N_p bits from the output of the

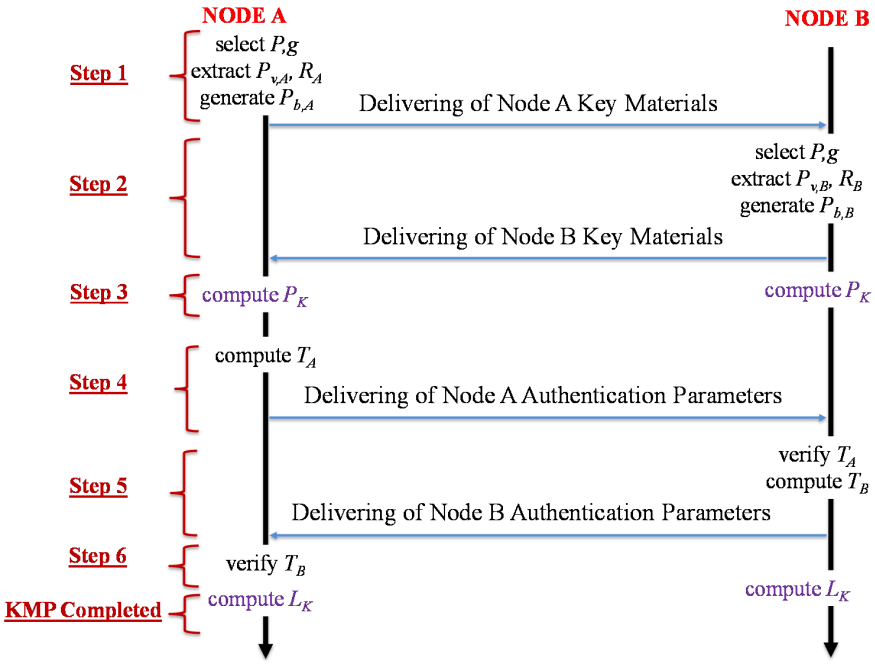


Figure 2: Message sequence chart of the designed KMP procedure.

following hash function:

$$H_{128}\{PAN_{ID}|D_k\}. \tag{2}$$

Then, after having randomly generated the private key, $P_{v,A}$, the public key, $P_{b,A}$, is computed according to the DH algorithm:

$$P_{b,A} = g^{P_{v,A}} \cdot modP. \tag{3}$$

When the certified DH approach is used, the public key is not computed from scratch; it is instead directly delivered by the certificate associated to the child node.

- **Step 2**: Also the node B sends its public key $P_{b,B}$ and a nonce, R_B, to the node A.

  Similarly to the previous case, when the anonymous DH scheme is selected, a prime number, P, and the corresponding primitive root, g, are firstly

extracted from the *Prime Numbers Table* (by following the aforementioned procedure). Then, the public key, $P_{b,B}$, is computed from the randomly generated private key, $P_{v,B}$, according to the DH algorithm:

$$P_{b,B} = g^{P_{v,B}} \cdot modP. \tag{4}$$

Obviously, the public key is directly transmitted within a certificate if the certified DH approach is used.

- **Step 3**: The node A and the node B compute a *Pre Link Key*, P_k, using Eqs. (5) and (6), respectively.

$$P_k = P_{b,B}^{P_{v,A}} \cdot modP, \tag{5}$$

$$P_k = P_{b,A}^{P_{v,B}} \cdot modP. \tag{6}$$

- **Step 4**: The node A computes the authentication parameter, T_A, and sends it to the node B for the mutual authentication:

$$T_A = \begin{cases} H_{128}\{P_k||R_B||R_A\} & \text{for anonymous DH} \\ E(P_k, S_{gn}) & \text{for certified DH} \end{cases} \tag{7}$$

where

$$S_{gn} = S\left(P_{v,A}, H_{128}\{P_k||R_B||R_A\}\right) \ , \tag{8}$$

Note that $E(\cdot)$ and $S(\cdot)$ operators refer to the encryption and the digital sign algorithm, respectively.

- **Step 5**: The node B verifies the validity of the received T_A parameter. In affirmative case, it computes the authentication parameter, T_B, sending it to the node A:

$$T_B = \begin{cases} H_{128}\{P_k||R_A||R_B\} & \text{for anonymous DH} \\ E(P_k, S_{gn}) & \text{for certified DH} \end{cases} \tag{9}$$

where

$$S_{gn} = S\left(P_{v,B}, H_{128}\{P_k||R_A||R_B\}\right) \ , \tag{10}$$

- **Step 6**: The node A verifies the validity of the received T_B parameter.

Once all the steps have been executed, the child node and the coordinator are able to generate the *Link Key*, L_k.

The standard imposes to use the CCM* algorithm and a 128-bit key to protect MAC frames. At the same time, the CCM* algorithm assumes that each key must be used for a specific number of block ciphers (i.e., until the frame counter associated to a given communication reaches its maximum value). Hence, for the i-th group of block ciphers, the *Link Key*, L_k, is computed as:

$$L_k = H_{128}\{i||PAN_{ID}||P_k\}. \tag{11}$$

4. Performance evaluation and security analysis

A thorough evaluation of the LICITUS framework is reported in this section, along with extensive experimental comparisons with respect to the ZigBee IP security architecture.

As a first step, a security analysis is presented for demonstrating the resilience of the proposed framework against different kinds of attacks. Then, the communication overhead incurred during the setup of a secure domain is evaluated. Finally, an experimental analysis is presented for assessing, in different topologies, the time and energy required to establish a secured domain.

4.1. Security analysis

LICITUS, combined with the security capabilities already integrated within the IEEE 802.15.4 technology, is robust with respect to the most important and critical security issues, as cryptanalysis, tamper attacks, password guessing, replay attack, and MITM.

The security analysis discussed hereby starts from the cryptanalysis attack, which can be generally handled for obtaining layer-2 keys (i.e., the *Default Key* and the *Link Key*). The IEEE 802.15.4 technology uses AES-based cryptography algorithms. In general, AES is known as one of the most efficient and secured algorithm offering confidentiality services. Some attacks designed to

force the AES scheme (see for example works presented in [35]) require long times and enormous computational resources, thus they are not effective in real contexts. Another category of attacks that can be successful in breaking the AES algorithm is the side-channel attack category. In this case, attackers use information gained from the physical implementation of the system (e.g., timing information, power consumption, electromagnetic fields, etc.) in order to obtain the key. However, also in this case, the time required to perform such a complex attack is higher than the lifetime of link keys. As a consequence, LICITUS is intrinsically resilient to cryptanalysis issues.

Password-guessing attacks try to extract keys by means of dictionary-attacks [36]. LICITUS is extremely robust with respect to this kind of attacks because nodes do not use passwords for computing layer-2 keys. In fact, *Default* and *Link keys* are obtained from the *Master Key* and DH parameters, respectively.

A malicious user is able to obtain and/or negotiate link keys only accessing security parameters stored into the *Security Configuration Manager* component. In this context, it is very important to remark that tamper attacks are really important in IoT systems, especially when some nodes are physical accessible from an external attacker. As already anticipated in Sec. 3.2, the proposed framework supposes to use specific software-based and/or hardware-based mechanisms (like those proposed in [33]) to prevent the physical access to all variables stored within the *Security Configuration Manager* component.

The adoption of frame counters makes the IEEE 802.15.4 technology resilient against replay attacks. Since the proposed approach fully integrates IEEE 802.15.4 security features, it also inherits robustness to these threats.

Finally, regarding the MITM attack, it could be launched by either an internal device (i.e., a malicious node that knows secured secrets shared among the rest of network devices) or external nodes (i.e., that do not know the value of variables stored within the *Security Configuration Manager* component) for compromising the right execution of the key negotiation phase. The designed KMP uses well-known approaches already adopted in the past for other protocols, as Transport Layer Security (TLS). Hence, its security can be demon-

strated by using existing analysis. In the first part of the protocol, the DH algorithm is used to negotiate the shared key. Here, MITM attacks can be avoided using X.509 certificates, used to uniquely bind the public key to its owner. The mutual authentication scheme implemented in the second part of the protocol, instead, protects the entire process against replay attacks. The aim of the two latest messages is inspired to functionalities provided by *Finished* messages in the TLS protocol [37]. Similarly to TLS, these packets carry an authentication field that is computed by considering all values, included random numbers, exchanged with the first couple of messages. Thus, the security proof related to the proposed protocol is as for the TLS protocol [38][39].

Replay and MITM attacks are generally performed to compromise the KMP protocol and the mutual authentication between communicating nodes. To further validate the resilience of the proposed framework against these issues, the effectiveness of LICITUS has been tested through a widely accepted automatic cryptographic protocol verifier developed at Inria: the ProVerif tool [40].

Several scientific contributions, like [41, 42, 43, 44], already used ProVerif for analyzing the security of their proposals. With ProVerif, it is possible to generate a formal model of a security protocol leveraging different cryptographic primitives, including shared and public key cryptography, hash functions, and Diffie-Hellman key agreements [45]. Security analysis, instead, is done by using the Dolev-Yao model, i.e., a baseline procedure reproducing many operations that could be done during an attack (i.e., capture and modification of the stream of messages exchanged between two devices over an unsecured channel). Starting from a set of initial assumptions (for instance, the attacker only knowns the algorithm, the attacker also knows the master key, and so on), the tool verifies if a malicious node is able to successfully force the KMP procedure, thus compromising the mutual authentication property, or not.

With reference to the KMP depicted in Fig. 2, the formal model developed for the ProVerif tool[4], contains the following main functions:

[4]To allow the reader to verify the validity of the conducted test, the code is freely available

- *begin_Node_A ()*, meaning that the KMP procedure is initiated by node A;
- *end_Node_A ()*, meaning that the KMP procedure is completed by node A with node B;
- *begin_Node_B ()*, meaning that the KMP procedure is initiated by node B;
- *end_Node_B ()*, meaning that the KMP procedure is completed by node B with node A;

Moreover, some important outputs provided by ProVerif appear as reported below:

- *not attacker(MasterKey[]) is true*. It means that the malicious node is not in possess of the master key;
- *not attacker(MasterKey[]) is false*. It means that the malicious node is in possess of the master key;
- *inj-event(last event name ()) ==> inj-event(previous event name ()) is true*. It means that a given operation, i.e, *last event name*, ends after that another operation, i.e., *previous event name*, is really executed. The operations *last event name* and *previous event name* can refer to one of the aforelisted functions.
- *inj-event(last event name ()) ==> inj-event(previous event name ()) is false*. It means that as soon as a given operation, i.e., *last event name*, is completed, it is not possible to ensure that another operation, i.e., *previous event name* has been really executed in the past. This condition identifies a security issue and envisages the possibility to have a successful attack.

The security analysis of the conceived KMP has been conducted by means of three different tests.

at http://telematics.poliba.it/security-iot

The first test assumes that the key agreement protocol is based on the anonymous DH algorithm and the attacker does not know the *Master key*. As reported in Fig. 3(a), ProVerif demonstrates that a malicious node is not able to compromise the right execution of the KMP. Thus, the mutual authentication is always guaranteed. In fact, it is registered that: node B completes the KMP when the procedure is really initiated by node A, and node A completes the KMP when the procedure is really initiated by node B.

The second test assumes that the key agreement protocol is based on the anonymous DH algorithm and the attacker knows the *Master key*. Differently from the previous case, the malicious node can successfully compromise the right execution of the KMP. In fact, as reported in Fig. 3(b), ProVerif realizes that when node B completes the KMP, it is not possible to ensure that the procedure was really initiated by node A. Indeed, it could be initiated by the malicious node. Similar considerations can be argued in the case A is the node that completes the KMP.

The third test assumes that the key agreement protocol is based on the certified DH algorithm and the attacker knows the *Master key*. As reported in Fig. 3(c), ProVerif demonstrates that a malicious node is not able to compromise the right execution of the KMP. Therefore, the mutual authentication is still guaranteed. In fact, it is registered that: node B completes the KMP when the procedure is really initiated by node A, and node A completes the KMP when the procedure is really initiated by node B.

To sum up, when the master key is secret, LICITUS is resilient against any attacks aiming at compromising the mutual authentication (like men-in-the-middle, replay, etc.). Furthermore, when the secrecy of the master key is compromised, the mutual authentication is still guaranteed only in the case the KMP makes use of X.509 certificates.

4.2. Preliminary investigation

The number of logical and MAC messages that a pair of devices have to exchange for establishing a secure communication represents the first term of

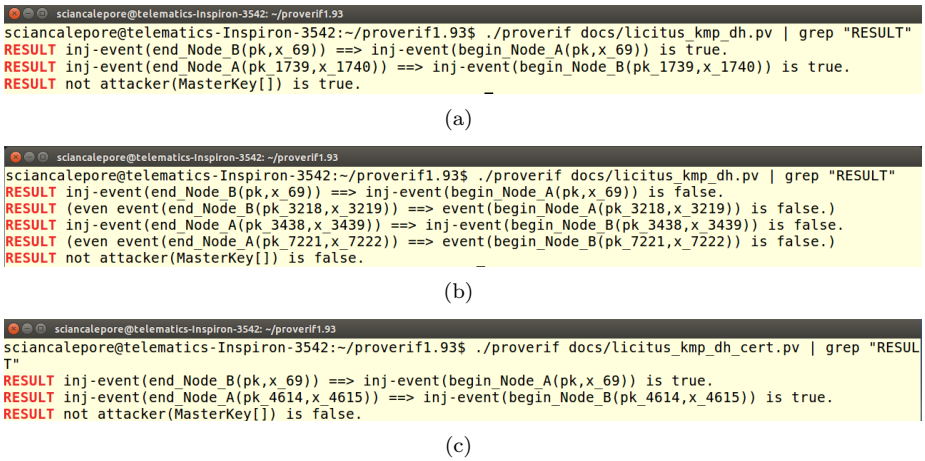


Figure 3: Screen-shots of the output provided by the ProVerif automatic tool a) with the anonymous DH and the Master Key secret b) with the anonymous DH and the Master Key widely known and c) with the certified DH and the Master Key widely known.

comparison between our proposal and the ZigBee IP security architecture.

As depicted in Figure 2, LICITUS always requires 4 logical messages. Considering that IEEE 802.15.4 specifications impose a Maximum Transmission Unit (MTU) equal to 127 bytes, such messages are mapped into 4 MAC packets when the KMP is based on the anonymous DH scheme. Instead, when the certified DH approach is adopted, the number of MAC packets becomes 24. In fact, by storing the public key in a X.509 certificate [32] of 864 bytes [5], the first two logical messages defined in the KMP scheme need to be fragmented in multiple MAC packets.

Regarding the ZigBee IP specifications, the establishment of the secured link is completely coordinated by the *Trust Center* according to the SKKE protocol.

As reported by the message sequence chart in Figure 4, the child node sends a first request to the *Trust Center*, which replies both to child and parent nodes with an initial shared secret (i.e., the *Master Key* in the SKKE protocol language). Then, this secret is used to protect the following 6 messages exchanged

[5] it is supposed to create a X.509 certificate by using the ECC Digital Signature Algorithm (ECDSA) algorithm

between the nodes, for finalizing the key agreement mechanism. Now, considering the size of each logical message and the constraint on the MTU, 9 different MAC messages are required to complete the protocol.

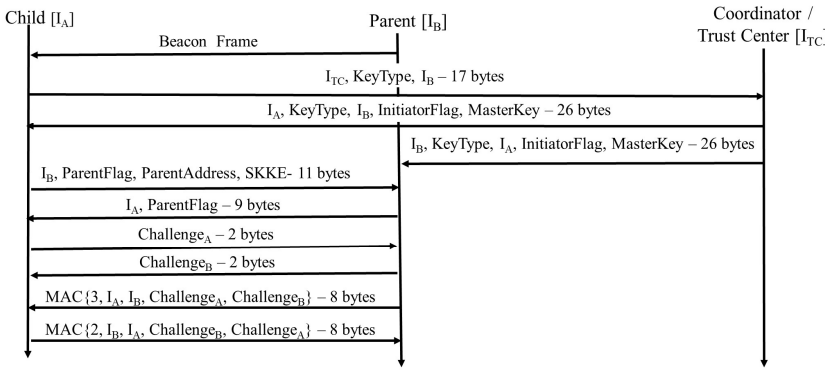


Figure 4: Establishing a secured communication in ZigBee IP specifications.

Indeed, as a first general comment, it is possible to observe that the SKKE protocol integrated in ZigBee IP specifications does not support an authenticated key agreement mechanism. However, supposing to use the anonymous DH scheme as the key negotiation algorithm, it can be immediately noted that the number of MAC packets required to establish a secured communication in ZigBee IP (i.e., 9) is higher than those needed by the proposed solution (i.e., 4). In addition, in Zigbee IP some messages (i.e., the second, the third, and fourth in Figure 4) are exchanged with the *Trust Center* that may be some hops away from the nodes involved in the KMP. As it will be demonstrated in the following section, this will negatively impact on energy consumptions and latencies. It is worth to note that, the designed KMP protocol requires an higher number of MAC messages when the certified DH scheme is used (that are equal to 24 packets as discussed before). This is the cost to pay for offering also the authentication of devices; but, this is an important feature not supported by SKKE.

Anyway, apart these preliminary comments, the time and the amount of energy needed to configure a secure system are strictly influenced by the network load, packet losses, de-synchronization issues, and complexity of each task, as

discussed in the rest of the section.

4.3. Experimental settings

To carry out experimental tests, LICITUS and the security mechanisms defined in ZigBee IP specifications have been implemented in the OpenWSN solution, i.e., a very promising open source protocol stack for LLNs, that integrates Constrained Application Protocol (CoAP), Routing Protocol for LLNs (RPL), IPv6 over Low power Wireless Personal Area Networks (6LoWPAN), Time-Slotted Channel Hopping (TSCH), and the IEEE 802.15.4 physical interface [6]. In particular, components and functionalities of the proposed framework have been developed on top of the security extension presented in [10][46] by some of the same authors of this contribution.

Details about ROM and RAM footprint of the considered security solutions have been summarized in Table 2. From reported values it is possible to observe that the proposed approach guarantees the minimum memory demands, thus becoming very suitable for any kind of constrained device.

Table 2: Memory requirements.

Method	**ROM usage [B]**	**RAM usage [B]**
Zigbee IP	5390	18
LICITUS	5122	6

Experimental testbeds have been set up using TelosB devices, equipped with a 16-bit microcontroller, working at a maximum speed of 8 MHz, 48 kByte Flash Memory, 10 kByte RAM, and a CC2420 radio module. These devices, being very constrained, allow to test LICITUS under very strenuous conditions.

Regarding the hash function, we use in our tests an open-source implementation of the SuperFastHash function[6]. By considering the computational capabilities of the TelosB mote, it ensures a good compromise among simplicity, footprint, security and computational overhead.

Three different network topologies have been considered:

[6] http://www.azillionmonkeys.com/qed/hash.html

- **Star Topology**, composed by one coordinator and up to 10 child devices directly connected to it (i.e., there are only one-hop connections).
- **Chain Topology**, composed by up to 17 devices connected by means of a multi-hop path.
- **Binary Tree Topology**, composed by 7, 15, and 31 devices. In that case, the maximum number of hops from the leaf node and the coordinator is equal to 2, 3, and 4, respectively.

In order to really realize multi-hop connections in both chain and binary tree topologies, devices have been placed sufficiently far, in order to ensure that nodes not directly connected at the layer-2 do not interfere with each other.

Focusing the attention on the MAC layer, and in line with [47], the IEEE 802.15.4 network has been configured to guarantee a slot lasting 15 ms and a superframe made of of 101 slots. Moreover, in each superframe, the first slot is used to exchange advertisement packets; the next five slots are configured in shared mode for sending and receiving keep-alive packets and routing messages, and only one slot is shared between each couple of devices for data transmissions. Instead, at the physical layer the power transmission level has been set to -25 dBm, which brings to a maximum transmission range equal to around 30 cm.

Regarding our proposal, the network has been set in the *Fully Secured* configuration, adopting a KMP protocol based on the anonymous DH scheme (as discussed above, this ensures same operational conditions with respect to the ZigBee IP architecture). For the ZigBee IP security architecture, instead, all the functionalities of the *Trust Center* have been integrated in the network coordinator.

Particular attention has also beed dedicated to the realization of the energy measurement system, used for estimating the amount of the current drained by TelosB batteries during the time. In detail, it is composed by:

- a desktop computer equipped with the LabVIEW tool, i.e., a software providing an easy and programmable interface between the computer itself

and remote measurement equipments.

- A data acquisition card (model 12-bit NI DAQCard-6024E) that enables the interaction between the aforementioned computer and a shielded connector block described below. It is in charge of delivering data sampling instructions from the computer to the probe and data in the reverse path.
- A shielded connector block (model BNC-2120) with signal-labeled BNC connectors, that interface the data acquisition card to the reference resistor through the BNC-to-clip adapter.
- A 1 Ω reference resistor, used to effectively measure the drained current.
- A probe that creates a physical connection between the reference resistor and TelosB batteries.

Starting from the current measurements, the amount of energy consumed by a single device is computed as:

$$E = \sum_i I_i V \Delta T \tag{12}$$

where I_i, V, and ΔT are the i-th sample of the measured current, the voltage of the batteries (i.e., 3 V), and the sampling time interval (i.e., 50 μs), respectively.

Finally, all the tests have been run 100 different times. Moreover, it has been verified that the confidence intervals computed from obtained results are limited. This means that conducted tests are enough for ensuring the goodness of conducted experimental evaluations.

4.4. Time and energy needed for configuring security services

Time and energy consumptions incurred during the configuration of security services have been evaluated in different topologies by varying the number of nodes. All nodes have been configured for starting the key agreement scheme with their parent as soon as they complete the association phase. In the ZigBee

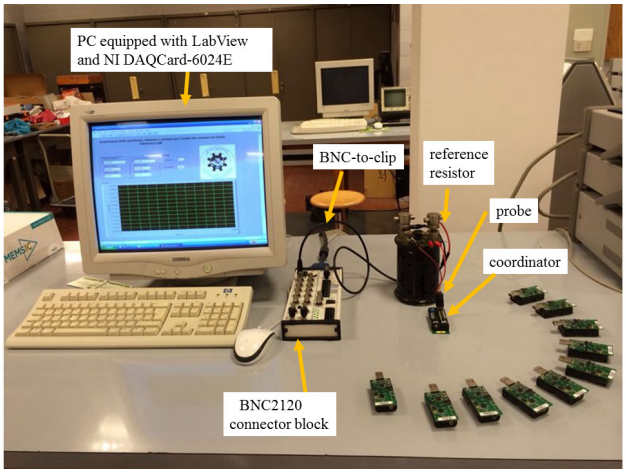


Figure 5: Energy measurement system.

IP security architecture, the SKKE protocol starts immediately after the reception of the beacon message. In the proposed security framework, instead, the KMP is executed only after the generation of the *Default key* by the *Bootstrap Manager*.

The time needed to establish a secured network is reported in Figure 6.

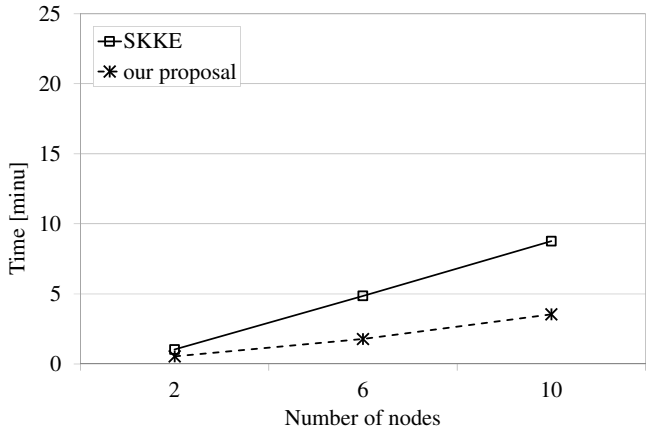
As a general comment, it can be observed that the configuration of security services always brings not negligible computational efforts. In fact, it emerges that the higher the network size, the higher the amount of time required to initialize security services and to negotiate layer-2 keys. However, obtained results clearly demonstrate that the proposed approach always ensures the lower airtime consumption, thus emerging as a promising, efficient, and scalable solution.

SKKE registers worse performance for two reasons. First, the key negotiation scheme requires the exchange of an higher number of messages. Second, the entire procedure is coordinated by a central entity, the Trust Center, that introduces an additional latency. Furthermore, it is evident that SKKE presents serious scalability issues: the amount of time required to configure security services drastically grows when the network size increases (see, for instance, the chain of 16 hop counts and the binary tree with 10 hop counts). As a result, SKKE does not scale with the number of nodes in the network.

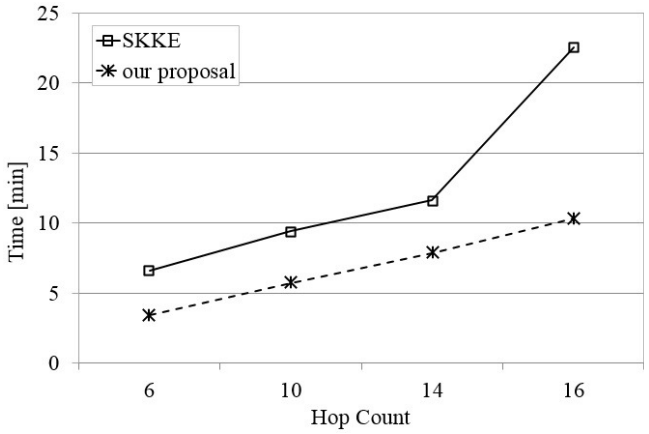
On the contrary, by using distributed and lightweight procedures for network bootstrapping and key negotiation, LICITUS potentially ensures good level of scalability. However, also for the proposed approach, it is possible to observe that the amount of time needed to set up a secure network increases with the number of nodes. Different reasons can be argued for different network topologies. In a star topology, the network coordinator should establish layer-2 keys with many nodes. Unfortunately, constrained nodes cannot execute multiple instances of the KMP at the same time, due to their limited computational and memory capabilities. Accordingly, different KMP processes are serialized and the resulting airtime consumption increases as well. By generalizing, isolated scalability issues can be found when many constrained devices would like to establish, at the same time, a layer-2 key with the same parent node. Of course, such a problem can be fixed when the network hosts more capable devices, able to run several KMP instances in parallel. In network topologies involving multi-hop connections (including both chain and binary tree), the airtime consumption increases because the time needed to set up the entire architecture at both MAC and network layers. For instance, in a chain with 10 hop counts, node at hop 10 can establish a secure communication link with node at hop 9 only after that node at hop 9 completed the same procedure with respect to node at hop 8.

The analysis of the amount of energy consumed by each node of the network, reported in Figure 7, fully confirms all of the comments reported above. In this case, the energy spent by the network coordinator, the leaf child, and the intermediate node (namely *parent* in Figure 7) have been measured. The coordinator, which is in charge of handling the most of tasks in the network, always incurs the highest energy consumption. On the contrary, child devices experience the lowest energy consumptions due to the lower number of operations they manage during the time. However, slightly larger energy consumptions are registered for intermediate nodes that, differently form the leaf child, have to manage key negotiation mechanisms with their parent and child nodes.

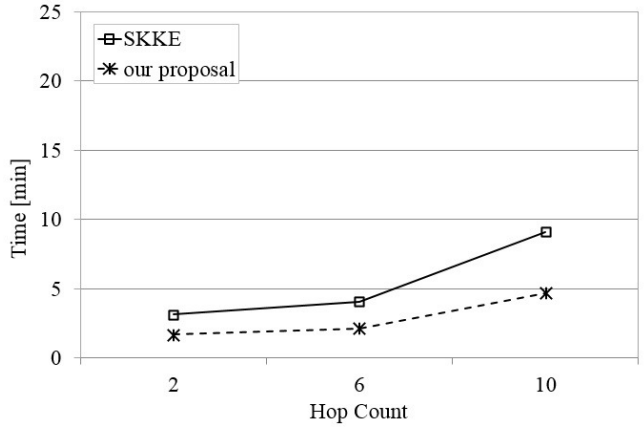
Furthermore, the reported investigation clearly demonstrates, once again,



(a)



(b)



(c)

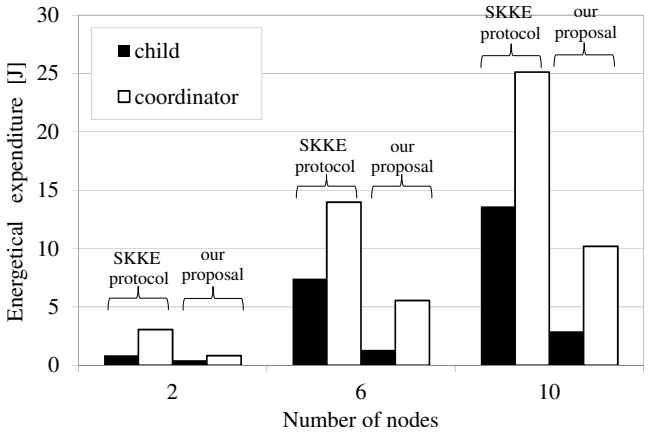
Figure 6: Time required for configuring security services in: (a) star, (b) chain, and (c) binary tree topologies.

that LICITUS is able to configure security services in a consistently less time than the ZigBee IP security architecture, while guaranteeing more than the 50% of energy savings.

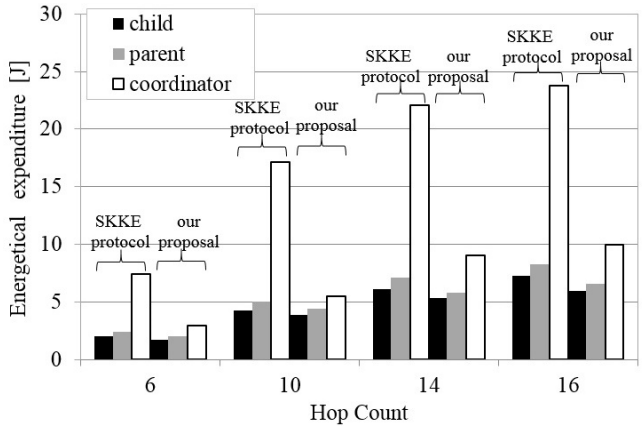
To provide a further insight, we also investigated the impact that such energy consumption has on a network operating in the long run. The study, however, demonstrated that the amount of energy consumed during the initialization of security services when both LICITUS and SKKE are used has a limited impact on the network lifetime. In fact, it is always less than 1% of the whole battery capacity. Nevertheless, in the case the application scenario requires that security services must be periodically renewed, the energy saving reached by LICITUS may further amplify its advantages, thus increasing the network lifetime. Anyway, being out of scope of our contribution, we leave a deep analysis of this specific aspect as a future work.

5. Conclusions

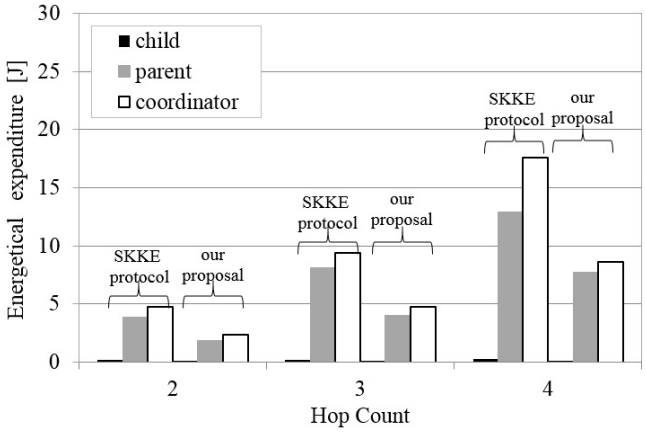
In this paper, a complete security framework for the IEEE 802.15.4 standard, namely LICITUS, has been proposed and its open source implementation has been described as well. Live experiments have been also carried out to demonstrate the real effectiveness of the conceived solution and to scavenge its possible downsides. In particular, it has been shown that all the secured operations provided by the proposed framework (including the set up of a secured link and the encryption/decryption of a MAC packet) require a not negligible computational effort and generate an increment of the energy consumption. At the same time, it has been demonstrated that LICITUS, compared to a benchmark protocol within ZigBee IP specifications, always guarantees lower computational efforts and more than 50% of energy saving. Future research activities will cover also the study of the LICITUS feasibility in more capable devices and the analysis of its impact on the quality of service offered to real applications properly conceived for future IoT systems.



(a)



(b)



(c)

Figure 7: Energy consumptions for configuring security services in: (a) star,(b) chain, and (c) binary tree topologies.

Acknowledgments

This work is supported by the project symbIoTe, which receives funding from the European Union's Horizon 2020 research and innovation programme under grant agreement 688156, and by the project BONVOYAGE, which receives funding from the European Union's Horizon 2020 research and innovation programme under grant agreement 635867.

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

[2] S. Sicari, A. Rizzardi, L. A. Grieco, and A. Coen-Porisini, "Security, privacy & trust in Internet of Things: the road ahead," *Computer Networks (Elsevier)*, vol. 76, pp. 146–164, 2015.

[3] S. L. Keoh, S. Kumar, and H. Tschofenig, "Securing the internet of things: A standardization perspective," *IEEE Internet Things J.*, vol. 1, no. 3, pp. 265–275, June 2014.

[4] M. Khanafer, M. Guennoun, and H. Mouftah, "A Survey of Beacon-Enabled IEEE 802.15.4 MAC Protocols in Wireless Sensor Networks," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 2, pp. 856–876, 2nd Quarter 2014.

[5] J. Granjal, E. Monteiro, and J. Silva, "Security for the Internet of Things: A Survey of Existing Protocols and Open Research issues," *IEEE Commun. Surveys Tuts.*, vol. PP, no. 99, pp. 1–1, 2015.

[6] T. Watteyne, X. Vilajosana, B. Kerkez, F. Chraim, K. Weekly, Q. Wank, S. Glaser, and K. Pister, "OpenWSN: a standards-based low-power wireless development environment," *Tans. on Emerg. Telecom. Technol.*, vol. 23, no. 5, p. 480493, 2012.

[7] C. Wang, T. Jiang, and Q. Zhang, *ZigBee Network Protocols and Applications.* Auerbach Publications, 2014.

[8] T. Watteyne, M. R. Palattella, and L. A. Grieco, *Using IEEE802.15.4e TSCH in an LLN context: Overview, Problem Statement and Goals*, RFC 7554, Internet Engineering Task Force RFC 7554, May 2015.

[9] Piro, G. and Boggia, G. and Grieco, L.A., *A standard compliant security framework for Low-power and Lossy Networks draft-piro-6tisch-security-issues-03 (work in progress)*, IETF 6TiSCH WG, Dec. 2014.

[10] S. Sciancalepore, G. Piro, E. Vogli, G. Boggia, and L. Grieco, "On securing IEEE 802.15.4 networks through a standard compliant framework," in *Proc. of EuroMed Telco Conference (EMTC) 2014*, Naples, IT, November 2014.

[11] *IEEE Standard for Local and Metropolitan Area Networks – Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs)*, IEEE Std., 16 June 2011.

[12] *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.

[13] *P802.15.4 - IEEE Approved Draft Standard for Local and metropolitan area networks - Part 15.4: Low-Rate Wireless Personal Area Networks (WPANs)*, IEEE P802.15.4/Draft Std., 2015.

[14] M. Boujelben, O. Cheikhrouhou, M. Abid, and H. Youssef, "Establishing Pairwise Keys in Heterogeneous Two-Tiered Wireless Sensor Networks," in *Proc. of Int. Conf. on Sensor Technologies and Applications, (SENSORCOMM)*, Jun. 2009, pp. 442–448.

[15] M. Pawlowski, A. Jara, and M. Ogorzalek, "Extending Extensible Authentication Protocol over IEEE 802.15.4 Networks," in *Int. Conf. on Innovative Mob. and Internet Serv. in Ubiquitous Comp. (IMIS)*, Jul. 2014, pp. 340–345.

[16] Y. Li, "Design of a Key Establishment Protocol for Smart Home Energy Management System," in *Int. Conf. on Comput. Intelligence, Commun. Sys. and Netw.(CICSyN)*, Jun. 2013, pp. 88–93.

[17] M. Rahman, S. Sampalli, and S. Hussain, "A robust pair-wise and group key management protocol for wireless sensor network," in *Proc. of IEEE GLOBECOM Workshops (GC Wkshps)*, Dec. 2010, pp. 1528–1532.

[18] R. D. Pietro and G. Oligeri, "COKE Crypto-Less Over-the-Air Key Establishment," *IEEE Trans. on Inf. Forensics and Security*, vol. 8, no. 1, pp. 163–173, Jan. 2013.

[19] B. Tian, S. Han, S. Parvin, and T. S. Dillon, "A Key Management Protocol for Multiphase Hierarchical Wireless Sensor Networks," in *Proc. of IEEE/IFIP Int. Conf. on Emb. and Ubiq. Comp. (EUC)*, Dec. 2010, pp. 617–623.

[20] M. Wilhelm, I. Martinovic, and J. Schmitt, "Secure key generation in sensor networks based on frequency-selective channels," *IEEE Journal on Sel. Areas in Comm. (JSAC)*, vol. 31, no. 9, pp. 1779–1790, September 2013.

[21] S. Raza, D. Trabalza, and T. Voigt, "Lightweight IKEv2: A Key Management Solution for both the Compressed IPsec and the IEEE 802.15.4 Security," *Workshop on Smart Object Security, Paris*, Mar. 2012.

[22] W. Gu, N. Dutta, S. Chellappan, and X. Bai, "Providing End-to-End Secure Communications in Wireless Sensor Networks," *IEEE Trans. on Netw. and Service Manag.*, vol. 8, no. 3, pp. 205–218, Sep. 2011.

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

[24] S. Sciancalepore, A. Capossele, G. Piro, G. Boggia, and G. Bianchi, "Key Management Protocol with Implicit Certificates for IoT Systems," in *Proc. of the 2015 Workshop on IoT Challenges in Mob. and Ind. Sys.*, ser. IoT-Sys '15, 2015, pp. 37–42.

[25] R. Struik, "6TiSCH Security Architectural Elements, Desired Protocol Properties, and Framework draft-struik-6tisch-security-architecture-elements-01," IETF, Internet Draft, Oct. 2014.

[26] M. Richardson, "6tisch secure join using 6top draft-richardson-6tisch–security-6top-05," IETF, Internet Draft, Nov. 2015.

[27] C. Wang, T. Jiang, and Q. Zhang, *ZigBee Network Protocols and Applications*. CRC-Press, Taylor and Francis Group, 2014.

[28] D. Huckelmann, M. Harrison, and F. Michahelles, *Architecting the Internet of Things*. Springer, 2011.

[29] T. Kennedy and R. Hunt, "A Review of WPAN Security: Attacks and Prevention," in *Proc. of Int. Conf. on Mobile Technol., Applications, and Syst.* ACM, 2008.

[30] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Trans. Inf. Theor.*, vol. 22, no. 6, pp. 644–654, Sep. 2006.

[31] S. Galbraith, *Mathematics of Public Key Cryptography*. Cambridge University Press, 2012.

[32] R. Housley, W. Polk, W. Ford, and D. Solo, *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*, RFC 3280, Internet Engineering Task Force RFC 3280, April 2002.

[33] J. P. Walters, Z. Liang, W. Shi, and V. Chaudhary, "Wireless sensor network security: A survey, in book chapter of security," in *in Distr., Grid, and Pervasive Comput., Yang Xiao (Eds.* CRC Press, 2007, pp. 0–849.

[34] D. Altolini, V. Lakkundi, N. Bui, C. Tapparello, and M. Rossi, "Low power link layer security for IoT: Implementation and performance analysis," in *Proc. of Int. Wir. Comm. and Mob. Comp. Conf. (IWCMC)*, 2013, pp. 919–925.

[35] N. Floissac and Y. L'Hyver, "From AES-128 to AES-192 and AES-256, How to Adapt Differential Fault Analysis Attacks on Key Expansion," in *Proc. of Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, Sept. 2011, pp. 43–53.

[36] J. Yan, A. Blackwell, R. Anderson, and A. Grant, "Password memorability and security: empirical results," *Security Privacy, IEEE*, vol. 2, no. 5, pp. 25–31, Sep. 2004.

[37] Dierks, T. and Rescorla, E., *The Transport Layer Security Protocol Version 1.1*, IETF, Apr. 2006.

[38] A. K. Ranjan, V. Kumar, and M. Hussain, "Security analysis of TLS authentication," in *Proc. of Int. Conf. on Contemporary Computing and Informatics (IC3I)*, Nov. 2014, pp. 1356–1360.

[39] A. Ferreira, R. Giustolisi, J.-L. Huynen, V. Koenig, and G. Lenzini, "Studies in Socio-technical Security Analysis: Authentication of Identities with TLS Certificates," in *Proc. of IEEE Int. Conf. on Trust, Security and Privacy in Comput. and Commun. (TrustCom)*, Jul. 2013, pp. 1553–1558.

[40] B. Blanchet, "Automatic Verification of Correspondences for Security Protocols," *Journal of Computer Security*, vol. 17, no. 4, pp. 363–434, Jul. 2009.

[41] M. Abadi, B. Blanchet, and C. Fournet, "Just Fast Keying in the Pi Calculus," *ACM Trans. Inf. Syst. Secur.*, vol. 10, no. 3, Jul. 2007.

[42] R. Ksters and T. Truderung, "Using ProVerif to Analyze Protocols with Diffie-Hellman Exponentiation," in *IEEE Computer Security Foundations Symposium*, July 2009, pp. 157–171.

[43] S. Asadi and H. S. Shahhoseini, "Formal security analysis of authentication in SNMPv3 protocol by an automated tool," in *Int. Symp. on Telecomm. (IST)*, Nov 2012, pp. 1060–1064.

[44] D. Chang, X. Chu, and G. Wei, "Analysis of the Security-Enhanced vTPM Migration Protocol Based on ProVerif," in *Int. Conf. on Comp. and Information Sciences (ICCIS)*, Jun. 2013, pp. 1437–1440.

[45] B. Blanchet, "Using Horn Clauses for Analyzing Security Protocols," in *Formal Models and Techniques for Analyzing Security Protocols*, ser. Cryptology and Information Security Series, V. Cortier and S. Kremer, Eds. IOS Press, Mar. 2011, vol. 5, pp. 86–111.

[46] S. Sciancalepore, G. Piro, G. Boggia, and L. A. Grieco, "Application of IEEE 802.15.4 security procedures in OpenWSN protocol stack," *IEEE Standards Education e-Magazine*, vol. 4, no. 2, Quarter 2014.

[47] X. Villajosana and K. Pister, "Minimal 6TiSCH Configuration, draft-ietf-6tisch-minimal-15," IETF, Internet Draft, Feb. 2016.

Annex A

The configuration of the *Global Security Levels Table* is discussed below through practical examples. In particular, two reference use cases are taken into account: industrial scenario and smart campus.

In an industrial scenario, sensor nodes are spread into the environment for monitoring the temperature of rooms and machines, the movements of motors and components, the humidity, the light, and so on. The security of an industrial plant is generally critical. Data must be kept as more confidential as possible and all the communications should be authenticated. Also, the join of external devices is not admitted. Thus, the *Fully-Secured* network configuration is required. In this way, we can ensure that (i) all packets exchanged in the network are encrypted and authenticated and (ii) nodes that do not support security capabilities cannot be part of the monitoring system.

A *Smart Campus* generally appears as a dynamic and heterogeneous scenario. The system may include fixed sensors deployed by the network administrator and external devices (think for example to mobile terminals of students,

professors, technical workers, etc.) that temporally expose their resources. Fixed sensors may communicate in a secure way. They, in fact, are able to establish layer-2 secure links starting from the set of security parameters stored by the network administrator just before the network deployment. External devices could not be in possession of security capabilities. Therefore, they can just send and receive messages in clear. To handle the coexistence of these heterogeneous nodes, a *Smart Campus* should use the *Hybrid-Secured* network configuration. Broadcast packets (like the beacon message) should be transmitted in clear. Nodes with security capabilities can establish secured link-level communications. Other nodes, instead, can only exchange data in clear.

To conclude, technical details of parameters stored in the *Global Security Levels Table* for both the considered use cases are reported in Tab. 3.

Table 3: Practical configurations of the Globals Security Levels Table

Attribute	**FrameType**	**Use case**	
		Industrial scenario	**Smart Campus**
Security Minimum	Beacon	5	0
	Data	5	0
	Command	5	0
	Ack	5	0
Allowed Security Levels	Beacon	5,6,7	0
	Data	5,6,7	0,1,2,3,4,5,6,7
	Command	5,6,7	0,1,2,3,4,5,6,7
	Ack	5,6,7	0,1,2,3,4,5,6,7