6TiSCH
INTERNET-DRAFT
Intended Status: Informational
Expires: June 17, 2014

G. Piro
(Politecnico di Bari)
G. Boggia
(Politecnico di Bari)
L. A. Grieco
(Politecnico di Bari)

December 14, 2013

A standard compliant security framework for Low-power and Lossy Networks
draft-piro-6tisch-security-issues-01

Abstract

The aim of this Internet Draft is to define a standard compliant security framework for Low-power and Lossy Networks, in order to enable message encryption and authentication at the MAC layer. The framework is fully compatible with both IEEE 802.15.4 and IEEE 802.15.4e standards and supports a wide range of security features to network architectures developed within the 6TiSCH Working Group. In particular, it defines different kinds of security configurations and, for each of them, proposes lightweight mechanisms for the setting-up of a secure IEEE 802.15.4 domain and the negotiation of link keys among devices.

Status of this Memo

Table of Contents

1 Acronyms

In addition to the acronyms defined in [I-D.palattella-6tisch-terminology], the following acronyms are used in this document:

DH Diffie Hellman

DEX Diet EXchange

DTLS Datagram Transport Layer

HIP Host Identity Protocol

PKI Public Key Infrastructure

2 Introduction

The IEEE 802.15.4 standard [IEEE802154] 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 Medium Access Control (MAC) and physical layers of the protocol stack and supports the possibility to protect MAC packets by means of symmetric-key cryptography techniques with several security options. However, the IEEE 802.15.4 standard does not explain how handling the initialization of a secure IEEE 802.15.4 domain, the generation and the exchange of keys, and the management of joining operations in a secure 802.15.4 network already configured in the past, thus delegating the upper layers to orchestrate, enable, configure, and negotiate security services.

The IEEE 802.15.4e [IEEE802154e] standard introduces some amendments to the IEEE 802.15.4 standard. Among its key features there is the Time-slotted Channel Hopping (TSCH), i.e., a novel MAC protocol, which better supports multi-hop communications in emerging industrial applications.

Since the IEEE 802.15.4e amendment focuses only on link-layer aspects, the 6TiSCH WG was born to define open standards in support of the adoption of IPv6 over the TSCH mode of the IEEE802.15.4e standard, thus covering all facets related to the management of network communications in complex (and eventually distributed) Low-Power and Lossy Networks (LLNs) [I-D.watteyne-6tisch-tsch] [I-D.wang-6tisch-6top].

Security aspects represent an important issue that need to be considered in a 6TiSCH network. TSCH defines mechanisms to encrypt and authenticate MAC frames but it does not define how this keying material is generated [IEEE802154]. For this reason, the 6TiSCH WG needs to (i) define the keying material and authentication mechanism needed by a new mote to join an existing network; (ii) define a mechanism to allow for the secure transfer of application data between neighbor motes; and (iii) define a mechanism to allow for the secure transfer of signaling data between motes and 6TiSCH [I-D.watteyne-6tisch-tsch].

In literature, several security strategies have been proposed for wireless sensor networks. Most of them exploits key negotiation algorithms and key management architectures summarized in [Camtepe2005], [Wang2006], and [Cayirci2007]. However, all of these works focus on a specific issue without embracing all the security features presented in [I-D.watteyne-6tisch-tsch].

Definitively, despite the high number of solutions focusing on security issues for LLNs, there is the lack of a complete and effective framework enabling security services in 6TiSCH compliant networks. Hence, the design of a one-size-fits-all solution 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 sitting on top of the IEEE 802.15.4 MAC [ZIGBEEIP]. ZigBee IP supports end-to-end and link-layer security and a public key infrastructure based on X.509 certificates. It imposes the adoption of the same link-key (shared among all nodes) to protect packets belonging to any kind of services (i.e., only a single security level is allowed). This makes the network highly sensible to the presence of compromised devices. Moreover, the cost needed to update the key in all devices increases with the size of the network. Finally, ZigBee does not support TSCH at the MAC layer. According to that considerations, the security architecture defined in ZigBee IP is not well suitable for 6TiSCH environments.

In the IEFT area, there are three main proposals on this topic: [I-D.garcia-core-security], [I-D.roll-security-framework], and [I-D.ohba-6tisch-security]. In [I-D.garcia-core-security] a set of security profiles in different network environments (e.g., network without security requirements, home, managed home, industrial, and advanced industrial) have been presented. For each of them, a number of security threats, as well as a list of well-known protocols and algorithms able to fixing these issues, have been also identified. In [I-D.roll-security-framework] is presented an accurate analysis on security threats at different point of the proto of a LLN, together

with the presentation of possible countermeasures to these dangers. In [I-D.ohba-6tisch-security] is presented a secure and scalable key management framework to adopt in 6TiSCH networks, as well as a set of requirements that a key management protocols should satisfy in that framework. All these IETF's proposals does not focus the attention on the design of a specific Key Management Protocol (KMP) for LLNs. Indeed, they relay on well-known approaches (i.e., like PANA [RFC5191], HIP DEX [HIPDEX], DTLS [RFC6347]) and to solutions based on the adoption of Public Key Infrastructure (PKI) to manage both node authentication and key negotiation procedures.

Nodes in a LLN have very limited storage, energy, and computational capabilities. For example, the LPC platform, which is the most powerful among those described in [Watteyne2012], have just 64 KB RAM, 512 KB program flash memory, and a 120 MHz ARM Cortex M3 micro-controller. Cryptography operations conducted for a single data packet generate, per se, a not neglectable computational overhead and energy consumption [Altolini2013]. On the other hand, the implementation and the management of complex security protocols (such as PAN, HIP DEX, and DTLS) and PKI infrastructure could not be suitable in LLNs because of the generation of a very large amount of signaling messages and the need for additional (and massive) computational loads and energy consumptions [Riaz2009]. Hence, all the available proposals cannot be easily applied in the 6TiSCH context: the design of simpler solutions, able to cover all the security aspects highlighted in [I-D.watteyne-6tisch-tsch] and able to be fully compatible with IEEE 802.15.4 and IEEE 802.15.4e specifications, is still an uncovered task.

The aim of this Internet Draft is to design a complete, simple, and standard compliant framework supporting a number of security features for the TSCH MAC layer. This work is complementary with respect to [I-D.ohba-6tisch-security] and, at the same time, it is fully integrated within the whole 6TiSCH architecture. Moreover, is has been designed in order to ensure an easy and effective implementation on real devices.

The main goals of this proposal have been summarized in the sequel:

- identify potential security configurations that could be supported by a 6TiSCH network;

- define a framework covering all the security issues (i.e., confidentiality and integrity protection, mote bootstrap, key negotiation and maintenance) presented in [I-D.watteyne-6tisch-tsch];

- propose an efficient mechanism, handled by the 6top adaptation layer, to configure a secured 6TiSCH network;

- propose a lightweight KMP to generate link keys that will be used to protect, at the MAC layer, unicast communications;

- explain the interaction between 6top and TSCH MAC layer during the configuration of the secured 6TiSCH network.

3 Security in IEEE 802.15.4

This section summarizes security features defined within the standard [IEEE802154], thus simplifying the comprehension of the remaining part of this Internet Draft.

The IEEE 802.15.4 standard defines eight security levels to protect MAC frames, as summarized in Fig. 1. The standard imposes the adoption of the CCM* algorithm to perform encryption and description procedures. It requires a key of 128 bit.

Security level	Security attribute	Data Integrity	Data Confidentiality
0	None	No	No
1	MIC-32	Yes	No
2	MIC-64	Yes	No
3	MIC-128	Yes	No
4	ENC	No	Yes
5	ENC-MIC-32	Yes	Yes
6	ENC-MIC-64	Yes	Yes
7	ENC-MIC-128	Yes	Yes

Figure 1. Security levels available for a IEEE 802.15.4 network.

At the MAC layer, encryption and decryption functionalities are implemented within the "outgoing frame security" and the "incoming frame security" procedures, respectively. They exploits a number of security attributes, summarized in what follows:

- macKeyTable: it is composed by a set of KeyDescriptor elements.

A specific KeyDescriptor element is created for each key, composed by (see Tab. 61 of the IEEE 802.15.4 standard for more details [IEEE802154]):

- The KeyIdLookupList, which is a list of KeyIdLookupDescriptor entries. A KeyIdLookupDescriptor is composed by a set of parameters (see Tab. 65 of the IEEE 802.15.4 standard for more details [IEEE802154]), i.e., KeyIdMode, KeySource, KeyIndex, DeviceAddMode, DevicePANId, and DeviceAddress, that are used to identify the key within the macKeyTable.

- The DeviceDescriptorHandleList, which contains pointers to DeviceDescriptor elements stored within the macDeviceTable. It is used to identify which devices may use the key.

- The KeyUsageList, which is a list of KeyUsageDescriptor elements. A KeyUsageDescriptor is composed by the FrameType and the CommandFrameIdentifies fields that indicate the frame type with which the considered key may be used (see Tab. 62 of the IEEE 802.15.4 standard for more details [IEEE802154]).

- The Key.

- macDeviceTable: it is composed by a set of DeviceDescriptor elements, providing some information about remote devices which the node can establish a secure communication with. A dedicated DeviceDescriptor element is associated to each remote device. It is composed by a number of fields, i.e., PANId, ShortAddress, ExtAddress, FrameCounter, and Extemp, which collect information related to a specific remote device (see Tab. 64 of the IEEE 802.15.4 standard for more details [IEEE802154]).

- macSecurityLevelTable: it is made by a set of SecurityLevelDescriptor elements, which store details about the security level required for each MAC frame type and subtype. Fields belonging to the SecurityLevelDescriptor data structure are: FrameType, ComamndFrameIdentifier, SecurityMinimum, DeviceOverrideSecurityMinimum, and AllowedSecurityLevels (see Tab. 63 of the IEEE 802.15.4 standard for more details [IEEE802154]).

- macFrameCounter: it is an integer value storing the outgoing frame counter for the considered device.

- macAutoRequestSecurityLevel: it is an integer value providing the security level used for automatic data requests.

- macAutoRequestKeyIdMode: it is an integer value indicating the key identifier mode used for automatic data requests. It is not valid if the macAutoRequestSecurityLevel attribute is set to 0x00.

- macAutoRequestKeySource: it represents a short or extended IEEE 802.15.4 MAC address, indicating the originator of the key used for automatic data requests. This attribute is not valid if the macAutoRequestKeyIdMode element is not valid or set to 0x00.

- macAutoRequestKeyIndex: it is an integer value storing the index of the key used for automatic data requests. It is not valid if the macAutoRequestKeyIdMode attribute is not valid or set to 0x00.

- macDefaultKeySource: it is the extended IEEE 802.15.4 MAC address of the originator of the default key used for key identifier mode 0x01.

During the outgoing security procedure, the high layer uses the KeyIdMode parameter to select a specific key in the macKeyTable to be used for protecting the MAC frame.

The KeyIdMode is set to 00, 01, 10, and 11 in the case the key can be derived implicitly by both sender and the receiver and its is not specified in the message, the key is determined explicitly by the KeyIndex parameter stored into the MAC header and the macDefaultKeySource, the key can be derived by considering KeyIndex and KeySource fields stored into the MAC header (with KeySource representing the short address of the device that has generated the key), and the key can be derived by considering KeyIndex and KeySource fields stored into the MAC header (with KeySource representing the IEEE extended address of the device that has generated the key), respectively.

Both IEEE 802.15.4 and IEEE 802.15.4e standards do not provide any guideline to create (and or negotiate) keys, as well as to configure the aforementioned security MAC attributes.

4 Definition of the secured domain

In this document, the "secured domain" concept refers to the portion of a 6TiSCH network where procedures and techniques described in this proposal must be performed in order to configure and maintain secured communications.

In a 6TiSCH network, nodes can be arranged in both peer-to-peer and star

topologies. In general, they can be organized through a Direction Oriented Directed Acyclic Graph (DODAG), which is initialized by the PAN coordinator. From the beginning, the PAN coordinator is in charge of generating initial advertising messages and handling joining procedures of neighbors. During the construction of the DODAG, a mote can become a reference node for its neighbors and, for this reason, it can handle the generation of advertisement and the management of joining procedure [PalattellaSurvey].

A cluster represents the portion of a 6TiSCH network where the distribution of radio resources and the management of TSCH frames is handled by a given mote, namely cluster head or coordinator.

A secured domain coincides with the cluster’s concept.

This means that the secured framework presented in this draft is in charge of configuring and managing security capabilities in each cluster in a 6TiSCH network. It is hence highly scalable because its complexity does not depend from the network size but only by the number of motes forming a specific cluster.

Just to simplify the description of all the security procedures and mechanisms, the rest of the document will focus the attention on a 6TiSCH network with only one cluster, which is composed by a PAN coordinator and a number of remote motes.

5 Security Configurations

The following network configurations are defined to support different security features within a 6TiSCH network.

- Fully Secured network: all the IEEE 802.15.4 devices forming the network are configured to fully support security services. It represents the most secured configuration: all packets, independently from the message they carry, are encrypted and authenticated. Nodes that do not support security capabilities (or that are not in posses of all the information to joining the network, such as key materials and encryption and decryption algorithms) are not allowed to join the network.

- Unsecured network: security services are not supported. Even if in possession of security capabilities, any pair of nodes is not allowed to establish a secured communication. Differently for the Fully Secured scheme, this is the lowest security level. Since the data encryption, the message integrity, and the peer authentication are not implemented, all the MAC frames are

exchanged in clear. Hence, the setup and the maintaining of the network are described by the standard and no further upgrades are required.

- Partial Secured network: only the integrity of message is supported.

- Hybrid Secured network: the network can be composed by heterogeneous nodes that could or could not support security features. As default, the network is created in an unsecured manner. All the non-unicast control messages sent by the coordinator should be transmitted in clear. In this way, in fact, it is ensured that all the devices are able to read the content of packets. A RFD node with security capabilities, that intends to exchange encrypted and/or authenticated packets with the coordinator, could negotiate a set of link key with its reference FFD.

- Flexible Secured network: as default, the network is setup with the Fully Secured configuration and all packets are encrypted and authenticated. If there is at least one node that have not security capabilities, the coordinator could decides to switch to the Hybrid Secured configuration.

The implementation of each of these network configurations is fully supported by the secured architecture devised within the IEEE 802.15.4 and the IEEE 802.15.4e standard [IEEE802154]. This means that the proposal presented in this Internet Draft does not introduces any changes to the standard but it just introduces techniques and procedures able to accomplish those aspects that have been left opened by IEEE specifications.

5.1 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 the DeviceDescriptor element (see Sec. 3) to define the minimum security level (i.e., one of those reported in Fig.1), 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 secure network configurations presented in Sec. 3.1, these parameters must be set as reported in Fig. 2.

Attribute	Secured Network Configurations				
	Unsecured	Fully	Partial	Hybrid	Flexible
SecurityMinimum	0	from 5 to 7	from 1 to 4	0	from 1 to 7
DeviceOverride-SecurityMinimum	FALSE	FALSE	FALSE	FALSE	TRUE
AllowedSecuri-tyLevelsvels	0	from 5 to 7	from 1 to 4	from 0 to 7	from 0 to 7

Figure 2. Setting of security attributes of the DeviceDescriptor element in each proposed secure network configuration.

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.

If the 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.

6 Initialization of a secured domain in a 6TiSCH network

The secured framework builds a secured domain (the definition of a secured domain is provided in Sec. 4) through the execution of three consecutive phases: Setting-up, Bootstrapping, and Key Negotiation (see the framework architecture in Fig. 3.

The Setting-up Phase is used to store into the device all the secrets required to initialize a secured domain. The Bootstrap Phase is used to initialize the secured domain and to compute a key that will be adopted to protect broadcast messages at the MAC layer. The Key Negotiation Phase handles the KMP and it is used to negotiate the key between a couple of nodes in a cluster.

These phases are not always mandatory for all the secured network configurations listed in Sec. 5. In particular, when both Unsecured and Partial Secured Configurations are used, it is not necessary to implement none of aforementioned phases because there is not the need to compute any key. On the contrary, the Fully Secured Configuration, and as a consequence also the Flexible Secured Configuration, requires the implementation of all the phases. For the Hybrid Secured Configuration, instead, the Bootstrap Phase is not mandatory because broadcast messages are always sent in clear.

The need to implement or not Setting-up, Bootstrap, and Key Negotiation Phases in each envisaged secured configuration is highlighted in Fig. 4.

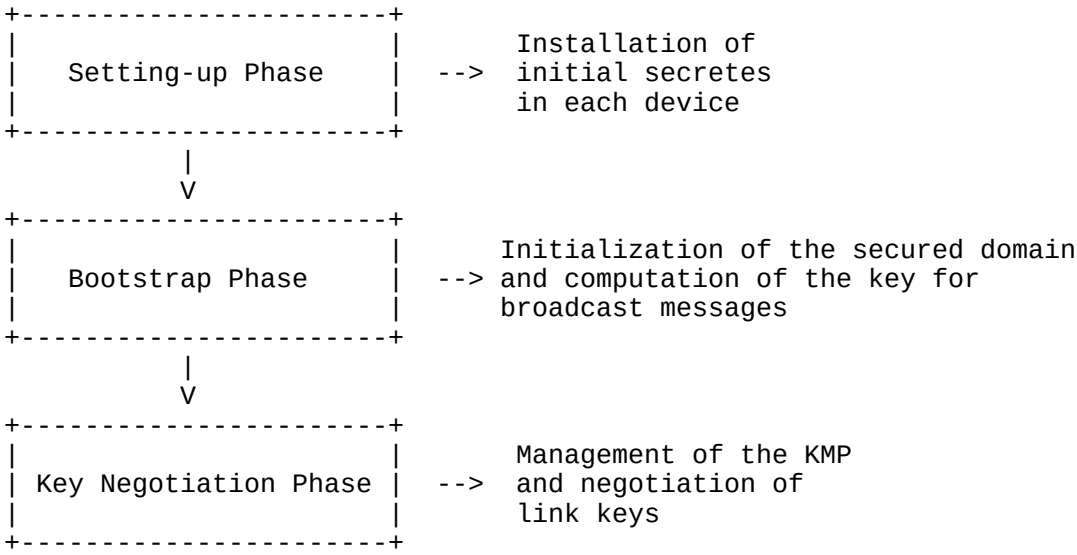

Figure 3. Summary of the proposed framework.

Phase	Secured Network Configurations				
	Unsecured	Fully	Partial	Hybrid	Flexible
Setting-up	NO	YES	NO	YES	YES
Bootstrap	NO	YES	NO	NO	YES
Key Negotiation	NO	YES	NO	YES	YES

Figure 4. Implementation of Setting-up, Bootstrap, and Ken Negotiation Phases for each envisaged secured configuration.

6.1 Setting-up phase

The setting-up phase is used to properly configure the device that will join to a secured 6TiSCH networks. It consists in storing, within the device, parameters and initial secrets (i.e., the masterKey), which will be used by secure algorithms and procedure to setup the secure domain. They include. (i) the MasterKey, (ii) the PrimeNumbersTable, (iii) the GlobalSecurityLevelsTable, and (iv) the public key of a certification authority.

This operation may be performed by the manufacturer or by the network

administrator.

6.1.1 The role of the MasterKey

The MasterKey is an initial secret, which is shared among all the motes.

The MasterKey is used to generate the DefaultKey that will be exploited to to protect broadcast messages (such as the beacon frame) in a given cluster.

A security approach based on preloaded key is widely adopted, especially in the case motes need to compute a common secret without relaying on preliminary negotiation steps [Camtepe2005], [Wang2006], [Cayirci2007].

This is the case of the Fully Secured Configuration. When this configuration is enabled, in fact, all the packets, including the beacon message, must be encrypted from the beginning. This means that any mote that wants to join to the network must compute the key adopted to protect the beacon message autonomously. Otherwise it will not be able to complete the join process because it will not be able to correctly extract information from the beacon message.

The MasterKey is not directly used to encrypt and decrypt broadcast messages. Another key, i.e., the DefaultKey, is instead computed starting from the MasterKey and other time-varying parameters

From one side, the MasterKey is unique for the whole 6TiSCH network and it may potentially remain the same for the whole life of the network. From another hand, each cluster will compute its specific the DefaultKey, which may potentially have a limited life time.

By using a time-varying key, we improve the resilience of the network to known-plaintext attack [StallingsSecurityBook], through which an attacker computes a key starting from the knowledge of group of clear/encrypted messages. In fact, even if an attacker will be able to obtain, through cryptanalysis techniques, the DefaultKey, it will be able to compromise only a specific cluster of the network and for a limited amount of time.

A mote can be subjected to any kind of tamper attacks. Without any further shrewdness, an attacker that may physically access to the mote could extract the MasterKey, thus compromising the security of the whole 6TiSCH network. Hence, it is very important to ensure the protection to that tampering attacks by using specific software-based and/or hardware-based mechanisms [Walters07][Becher2006].

6.1.2 The role of the GlobalSecurityLevelsTable

The GlobalSecurityLevelsTable, that has been reported in Fig. 5, is used to store the minimum security level and the list of allowed security levels that must be adopted for each kind of MAC frame and for each security configuration defined in Sec. 5. The table reported in Fig. 5 does not consider ACK messages because they must not be protected (as imposed by the IEEE 802.15.4 standard [IEEE802154]).

Both the minimum security level and the list of allowed security levels must be chosen by the manufacturer or by the network administrator, according to restrictions reported in Fig. 2.

Attribute	Frame Type	Secured Network Configurations			
		Fully	Partial	Hybrid	Flexible
Security Minimum	Beacon				
Security Minimum	Data				
Security Minimum	Command MAC				
AllowedSecurityLevels	Beacon				
AllowedSecurityLevels	Data				
AllowedSecurityLevels	Command MAC				

Figure 5. Structure of the GlobalSecurityLevelsTable.

6.1.3 The role of the PrimeNumbersTable

The PrimeNumbersTable stores a set of N prime numbers and their respective primitive roots, which are used during the Key Negotiation Phase. Its implementation is reported in Fig. 6.

These prime numbers are not the keys to protect MAC frames but are just numbers that will be exploited to generate keys according to the DH algorithm [DH].

PrimeNumberId	Prime Number	Primitive Root
1	p1	g1
2	p2	g2
:		:
N	pN	gN

Figure 6. Structure of the PimeNumbersTable.

As described in Appendix A.1, the security level of the proposed approach does not depend from the size of the PrimeNumbersTable but it is only influenced by the length of each prime number.

6.1.4 The role of the public key of the authority

The Key Negotiation Phase exploits a KMP based on the DH algorithm. The possibility to authenticate motes through public certificates is also supported.

6.2 Bootstrap phase

The implementation of the Bootstrap Phase is different for both PAN coordinator and remote mote. Moreover, for a remote mote, two different procedure has been conceived for both the not-beacon-enabled and the beacon-enabled networks.

6.2.1 Bootstrap phase for the PAN coordinator

A 6TiSCH network is initialized by a FFD node that will become the PAN coordinator. As described in [IEEE802154], at the end of this phase the PAN coordinator has chosen both the identification number for the PAN, i.e., the PAN_ID, and the short MAC address of the IEEE 802.15.4 network, i.e., shortMACaddress.

Once such task has been finished, the 6top adaptation layer computes the DefaultKey, D_k, and configures a set of security MAC attributes through specific commands (they have been introduced in [I-D.wang-6tisch-6top]).

In particular, it executes the following operations:

a) A CONFIGURE.security command is generated by the 6top layer and sent to the MAC entity to initialize security attributes (as discussed in [I-D.wang-6tisch-6top], in Sec. 2.4.9.1). The set of parameters handled by this command are set as in the sequel:

a.1) enable = true;

a.2) macAutoRequestSecurityLevel = security level expected for the beacon message and stored within the GlobalSecurityLevelsTable;

a.3) macAutoRequestKeyIdMode = 0x03;

a.4) macAutoRequestKeySource = MAC address of the PAN coordinator;

a.5) macAutoRequestKeyIndex = 1;

a.6) macDefaultKeySource = MAC address of the PAN coordinator;

b) CONFIGURE.security.macSecurityLevelTable command is generated by the 6top layer and sent to the MAC entity to initialize macSecurityLevelTable (as discussed in [I-D.wang-6tisch-6top], in Sec. 2.4.9.3). Parameters stored into this command are taken from the GlobalSecurityLevelsTable.

c) The DefaultKey, D_k, is obtained from the MasterKey, M_k, by using a 128-bit hash function, H_128(.)

D_k= H_128(PAN_ID | shortMACaddress | M_k).

d) A new KeyIdLookupList data structure is created. A KeyIdLookupDescriptor is generated and stored into the KeyIdLookupList data structure. The KeyIdMode, the KeySource, and the KeyIndex variables of this KeyIdLookupDescriptor are set to 0x03, the MAC address of the device, and 1, respectively. Instead, DeviceAddrMode, DevicePANId, and DeviceAddress are not set due to the selected KeyIdMode (see Tab. 65 of the IEEE 802.15.4 standard for more details [IEEE802154]).

e) A KeyUsageList data structure is created. One KeyUsageDescriptor for each kind of broadcast messages is create and stored into the KeyUsageList data structure.

f) An empty DeviceDescriptorHandleList is created. No data are stored within this list because the PAN coordinator does not yet know the list of devices that may use this key.

Then, the 6top layer deliver the DefaultKey, the KeyIdLookupList, the KeyUsageList, and the DeviceDescriptorHandleList to the MAC layer by using the CONFIGURE.security.macKeyTable primitive (as discussed in [I-D.wang-6tisch-6top], in Sec. 2.4.9.2).

Triggered by the CONFIGURE.security.macKeyTable command, the MAC layer will create a KeyDescriptor associated to the DefaultKey, D_k,in which storing all the parameters received by the 6top layer, and and will store it within the macKeyTable.

The Bootstrap phase for the PAN coordinator has been summarized in Fig. 7.

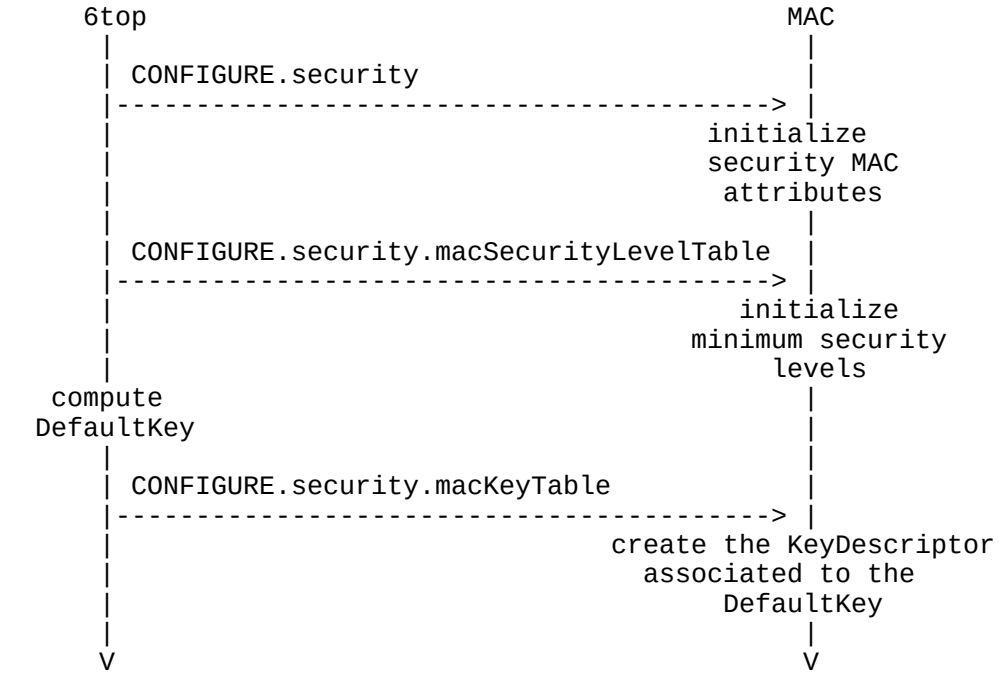

Figure 7. Bootstrap Phase for the PAN coordinator.

6.2.2 Bootstrap phase for a mote in a Beacon-enabled network

To join the network, a mote should associate with the coordinator. The Next Higher Layer sends to the MAC entity the MLME-ASSOCIATE.request primitive, starting the association procedure.

As for the PAN coordinator, in this phase the 6top adaptation layer should initialize security MAC attributes, compute the DefaultKey, D_k, and updates MAC security attributes accordingly.

To this end, after the reception of the beacon message, the following operations are executed:

a) The PAN_ID, the MAC address of the coordinator, and the FrameCounter are extracted from the header of the beacon message.

b) A CONFIGURE.security primitive is generated by the 6top layer and sent to the MAC entity to initialize security attributes (as discussed in [I-D.wang-6tisch-6top], in Sec. 2.4.9.1). The set of parameters handled by this primitive are set as in the sequel:

b.1) enable = true;

b.2) macAutoRequestSecurityLevel = security level expected for the beacon message and stored within the GlobalSecurityLevelsTable;

b.3) macAutoRequestKeyIdMode = 0x03;

b.4) macAutoRequestKeySource = MAC address of the PAN coordinator;

b.5) macAutoRequestKeyIndex = 1;

b.6) macDefaultKeySource = MAC address of the PAN coordinator;

c) CONFIGURE.security.macSecurityLevelTable primitive is generated by the 6top layer and sent to the MAC entity to initialize macSecurityLevelTable (as discussed in [I-D.wang-6tisch-6top], in Sec. 2.4.9.3). Parameters stored into this command are taken from the GlobalSecurityLevelsTable.

d) The DefaultKey, D_k, is obtained from the MasterKey, M_k, by using a 128-bit hash function, H_128(.)

D_k= H_128(PAN_ID | shortMACaddress | M_k).

e) A new KeyIdLookupList data structure is created. A KeyIdLookupDescriptor is generated and stored into the KeyIdLookupList data structure. The KeyIdMode, the KeySource, and the KeyIndex variables of this KeyIdLookupDescriptor are set to 0x03, the MAC address of the PAN coordinator, and 1, respectively. Instead, DeviceAddrMode, DevicePANId, and DeviceAddress are not set due to the selected KeyIdMode (see Tab. 65 of the IEEE 802.15.4 standard for more details [IEEE802154]).

f) A KeyUsageList data structure is created. One KeyUsageDescriptor for each kind of broadcast messages is create and stored into the KeyUsageList data structure.

g) A new DeviceDescriptor element, associated to the PAN coordinator (i.e., the FFD node that sent the Beacon message) is created and stored into the macDeviceTable. It is built considering these specifications (see Tab. 64 of the IEEE 802.15.4 standard [IEEE802154] for more details):

g.1) The PANId variable is associated to the PAN_ID value extracted from the Beacon message.

g.2) The ShortAddress is set to the MAC address of the coordinator whenever the short addressing mode is used. This parameter is set to 0xfffe if only the extended addressing mode is used. If its value is unknown, the ShortAddress parameter is set to 0xfff.

g.3) The ExtAddress is set to the IEEE MAC address of the coordinator.

g.4) The FrameCounter parameter is set to the FrameCounter value extracted from the Beacon message.

g.5) The Extempt boolean flag is set to the allowed value of the DeviceOverriddeSecurityMinimum variable described in Fig. 2.

h) The DeviceDescriptorHandleList is created and populated with the DeviceDescriptor created at the previous step.

i) A KeyUsageList data structure is created and stored within the KeyDescriptor element. One KeyUsageDescriptor for each broadcast message is create and stored into the KeyUsageList data structure.

Then, the 6top layer deliver the DefaultKey, the KeyIdLookupList, the KeyUsageList, and the DeviceDescriptorHandleList to the MAC layer by using the CONFIGURE.security.macKeyTable primitive (as discussed in [I-D.wang-6tisch-6top], in Sec. 2.4.9.2).

Triggered by the CONFIGURE.security.macKeyTable primitive, the MAC layer will create a KeyDescriptor associated to the DefaultKey, D_k, in which storing all the parameters received by the 6top layer, and and will store it within the macKeyTable.

The Bootstrap Phase for a remote mote in a beacon-enabled network has been summarized in Fig. 8.

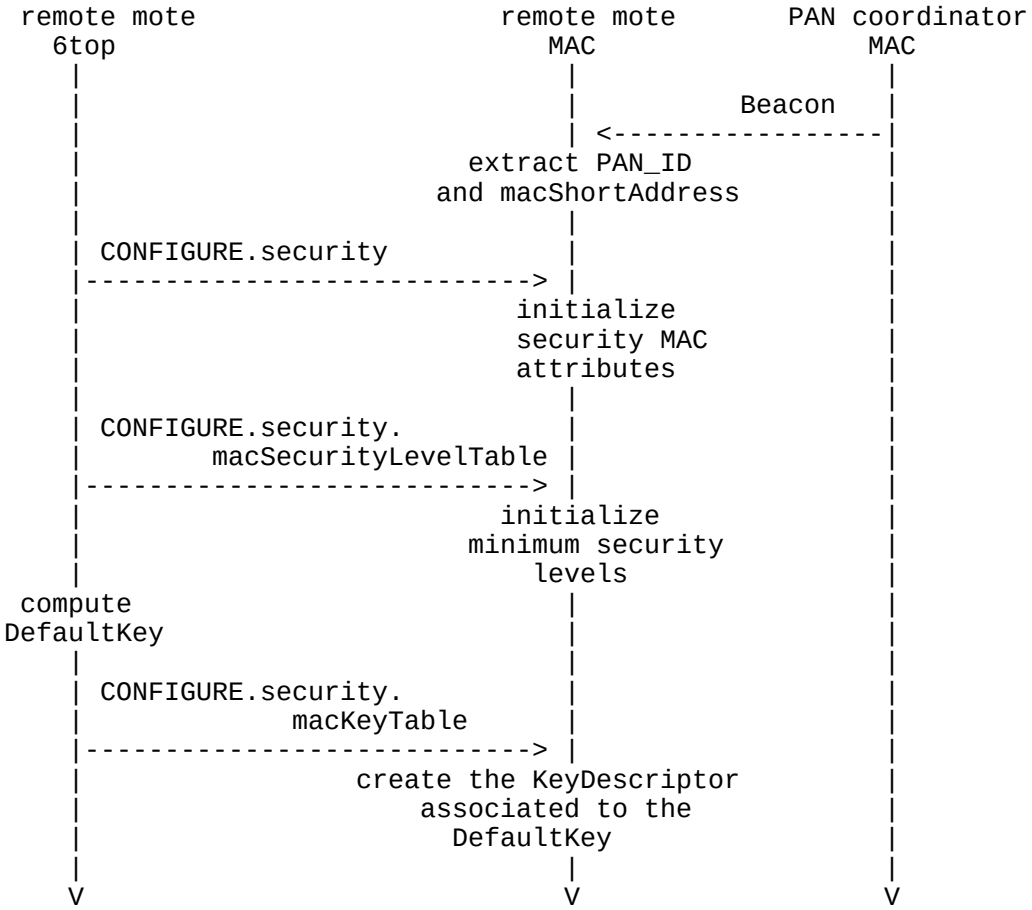

Figure 8. Bootstrap Phase for the remote mote in an enabled-beacon network.

6.2.3 Bootstrap phase for a mote in a not-Beacon-enabled network

In the case the not-beacon-enabled scheme is enabled, the mote must explicitly requests its generation to the coordinator. The payload of the Beacon Request packet must be protected using an ephemeral key, phi_k, obtained from the MasterKey, M_k, and the source address of the device, srcMACaddrMote, as

$$phi\_k = H\_128(srcMACaddrMote \mid M\_K).$$

The KeyIdMode of the Beacon Request packet is set to 00, thus enabling the PAN coordinator to implicitly obtain the ephemeral key.

Once received the Beacon frame, the remote mote will execute all the steps described in Sec. 6.3.

The Bootstrap Phase for a remote mote in a not-beacon-enabled network has been summarized in Fig. 9.

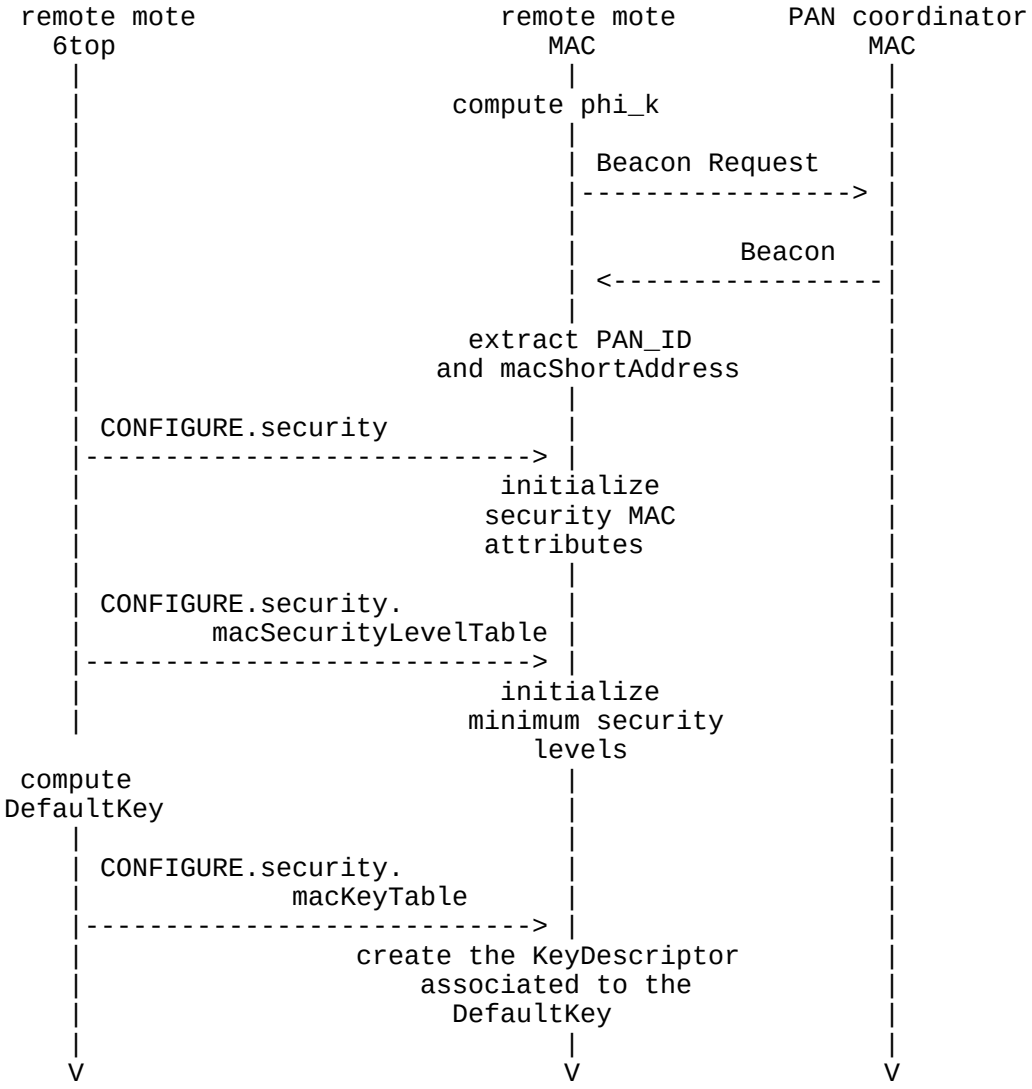

Figure 9. Bootstrap Phase for the remote mote in an not-enabled-beacon network.

6.3 Key Negotiation Phase

Since resource-constrained devices are unable to perform complex algorithms and protocols [Altolini2013][Riaz2009], a simple key agreement protocol is adopted during the execution of the key negotiation phase.

The KMP adopted by the secured framework presented in this draft is based on both DH algorithm [DH] and Station-To-Station protocol [StsProtocol].

Both anonymous and certified DH schemes are supported. The former one does not require that a mote will deliver its public key through a certificate and, for this reason, it doe snot support the node authentication. The latter one, instead, supports the use of certificates to authenticate the public key of each mote. With the anonymous DH scheme, a mote is able to deliver its public key by means of only one packet. When the certified DH scheme is used, instead, the public key will be delivered through multiple packets because of the high size of certificate (i.e., the key material generated by the mote needs to be fragmented).

To handle the Key Negotiation Phase, a new command MAC frame and two new 6top commands have been defined.

6.3.1 The new command MAC frame

A new command MAC frame, which is identified with a CommandFrameIdentifier set to 0xAA, has been introduced.

It is composed by four different fields: KeyGenControlField, Rand, KeyMaterial, and AuthenticationField.

The structure of the new command MAC frame has been reported in Fig. 10. The structure of the KeyGenControlField, instead, are shown in Fig. 11. The introduction of these new fields respects the constraints imposed by the standard about the maximum packet size.

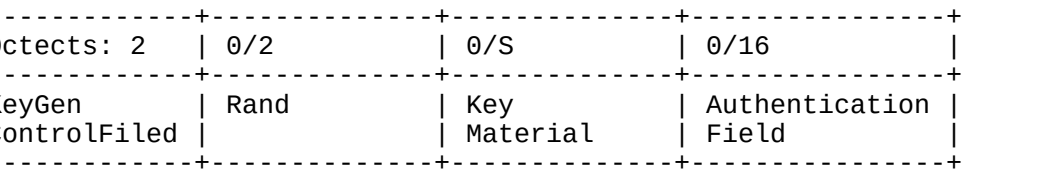

Figure 10. A new command MAC frame adopted during the key negotiation phase.

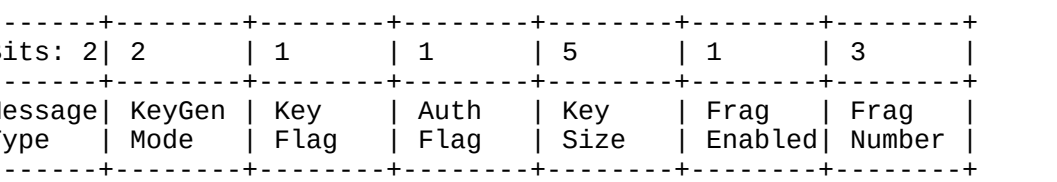

Figure 11. KeyGenControlField of the new command MAC frame adopted during the key negotiation phase.

The KeyGenControlField (2 bytes long) stores details about the content of the message. It is composed by the following fields:

- the MessageType (2 bits long), which identifies the type of message exchanged during the procedure. It may assume these values:

  - MessageType=00 identifies a message storing key materials (i.e., DH parameters).

  - MessageType=01 identifies final messages belonging to the Key Negotiation Phase that are used to verify the mutual authentication of nodes.

  - MessageType=10 and MessageType=11 are reserved for future upgrades.

- the KeyGenMode (2 bits long), which describes the algorithm adopted for key generation. It is set to 00 and 01 when the key is computed through the anonymous DH and the certified DH algorithms, respectively. Other values, i.e., 10 and 11, are reserved and can be used for future upgrades.

- the boolean KeyFlag (1 bit long), which is set to TRUE in the case the message delivers key materials or to FALSE otherwise.

- the boolean AuthFlag (1 bit long)), which is set to TRUE in the case the message delivers an authentication field or to FALSE otherwise.

- the KeySize (5 bits long), which indicates the size of the transported key material, expressed in bytes. Its value is set to 0 in the case the message does not contain any key materials.

- the boolean FragEnabled (1 bit long), which is set to TRUE when the key material contains a fragment of the certificate storing the public key of a mote;

- the FragNumber (3 bit long), which indicates the fragment number associated to the key material field.

The Rand field (0/2 bytes long) contains a random value used for generating the PreLinkKey, P_k, and for verifying the authenticity of the remote device. It is present only if MessageType is equal to 00 or 01.

The KeyMaterial field (0/S bytes long, where S is the size of the prime number) contains key materials, such as DH parameters. It is present only if MessageType is equal to 00 or 01.

AuthenticationField field (0/16 bytes long) is used to verify the authenticity of the remote device. It is present only if MessageType is equal to 10.

6.3.2 New 6top commands

The following 6top commands have been designed to perform the KMP:

- CONFIGURE.security.startKMP: it is used to send the initial key material that will be exploited by the DH protocol to generate the key. The command requires:
  - KeyGenMode, which represents the algorithm adopted to negotiate the key;
  - KeyMaterial, which represent the public key or a certificate storing the public key;
  - Rand, which is a random number exploited during the mutual authentication process.
- CONFIGURE.security.completeKMP: it is used to complete the KMP by handling the mutual authentication between motes according to the Station-To-Station protocol. The command requires:
  - KeyGenMode, which represents the algorithm adopted to negotiate the key;

- AuthenticationField, which is used to verify the mutual authentication.

6.3.3 KMP implementation when the anonymous DH scheme is used

The KPM is initialized by the 6top adaptation layer of the remote device connected to the coordinator, that has already completed the joining procedure and that wants to establish a secured link with the PAN coordinator.

The procedure assumes that both motes store into the PrimeNumbersTable the same set of N prime numbers and their primitive roots, each one having size equal to S (see Sec. 6.1 for more details).

The number of bits needed to identify each prime number of the PrimeNumbersTable, i.e., P_bits, is equal to

$$P\_bits = \log2 (N).$$

We note that the security level of the proposed approach does not depend from P_bits, but it is only influenced by the length of each prime number, S. See Appendix A.1 for more details.

The KMP is performed through the execution of these operations:

a) The 6top adaptation layer of the remote mote performs the following steps:

a.1) a prime number, P, and the corresponding primitive root, g, are identified in the PrimeNumbersTableby by extracting the latest P_bits bits from the output of the following hash function:

$$H\_128(PAN\_ID \mid D\_k).$$

a.2) two random numbers are generated: the former one represents its private key, i.e., PVK_remoteMote; the latter one is used for the mutual authentication, i.e., RAND_remoteMote.

a.3) the public key, i.e., PBK_remoteMote, is generated according to the DH algorithm:

$$PBK\_remoteMote = g^{PVK\_remoteMote} * \bmod P$$

a.4) A CONFIGURE.security.startKMP command is released and delivered to the MAC layer. KeyGenMode, KeyMaterial, and Rand parameters of this command are ser to 00, PBK_remoteMote, and RAND_remoteMote, respectively.

b) Triggered by the CONFIGURE.security.startKMP command, the MAC layer of the remote mote generates a command MAC frame, which is composed by the following fields: MessageType=00, KeyGenMode=00, KeyFalg=TRUE, AuthFlag=FALSE, KeySize=S, Rand=RAND_remoteMote, and KeyMaterial=PBK_remoteMote. In the case the Fully Secured Configuration is enabled, this message is encrypted with the DefaultKey, D_k. Otherwise it is sent in clear.

c) The 6top adaptation layer of the PAN coordinator performs the following steps:

c.1) a prime number, P, and the corresponding primitive root, g, are identified in the PrimeNumbersTableby by extracting the latest P_bits bits from the output of the following hash function:

H_128(PAN_ID | D_k).

c.2) two random numbers are generated: the former one represents its private key, i.e., PVK_coordinator; the latter one is used for the mutual authentication, i.e., RAND_coordinator.

c.3) the public key, i.e., PBK_coordinator, is generated according to the DH algorithm:

PBK_coordinator = g^PVK_coordinator * mod P

c.4) A CONFIGURE.security.startKMP command is released and delivered to the MAC layer. KeyGenMode, KeyMaterial, and Rand parameters of this command are ser to 00, PBK_coordinator, and RAND_coordinator, respectively.

d) Triggered by the CONFIGURE.security.startKMP command, the MAC layer of the remote mote generates a command MAC frame, which is composed by the following fields: MessageType=00, KeyGenMode=00, KeyFalg=TRUE, AuthFlag=FALSE, KeySize=S, Rand=RAND_coordinator, and KeyMaterial=PBK_coordinator. In the case the Fully Secured Configuration is enabled, this message is encrypted with the DefaultKey, D_k. Otherwise it is sent in clear.

e) The 6top adaptation layer of the remote mote computes the PreLinkKey, P_k

P_k = PBK_coordinator^PVK_remoteMote * mod P.

f) The 6top adaptation layer of the PAN coordinator computes the PreLinkKey, P_k,

P_k = PBK_remoteMote^PVK_coordinator * mod P.

g) Both remote motes and PAN coordinator compute the LinkKey by using the procedure described in Sec. 6.3.5.

h) The 6top adaptation layer of the remote mote computes the authentication parameter, AUTH_remoteMote, through the 128-bit hash function, H_128(.), as in the sequel

AUTH_remoteMote=H_128(P_k || RAND_coordinator || RAND_remoteMote).

Then, it releases a CONFIGURE.security.completeKMP command with KeyGenMode and AuthenticationField set to 00 and AUTH_remoteMote, respectively.

i) The MAC layer of the remote mote sends to the coordinator a new MAC command message to complete the mutual authentication. This message is composed by the following fields: MessageType=10, KeyGenMode=00, KeyFalg=FALSE, AuthFlag=TRUE, KeySize=0, and AuthenticationField=AUTH_remoteMote. This message is protected by using the LinkKey computed before.

j) The 6top adaptation layer of the PAN coordinator verifies the validity of the received AUTH_remoteMote parameter. In affirmative case, it computes the authentication parameter, AUTH_coordinator, through the 128-bit hash function, H_128(.), as in the sequel:

AUTH_coordinator=H_128(P_k || RAND_remoteMote || RAND_coordinator).

Then, it releases a CONFIGURE.security.completeKMP command with KeyGenMode and AuthenticationField set to 00 and AUTH_coordinator, respectively.

k) The MAC layer of the PAN coordinator sends to the remote mote a new MAC command message to complete the mutual authentication. This message is composed by the following fields: MessageType=10, KeyGenMode=00, KeyFalg=FALSE, AuthFlag=TRUE, KeySize=0, and AuthenticationField=AUTH_coordinator. This message is protected by using the LinkKey computed before.

l) The remote motes verifies the validity of the received AUTH_coordinator parameter.

m) PAN coordinator and remote mote update MAC security attributes according to procedures described in Sec. 6.3.6 and Sec 6.3.7, respectively.

The aforediscussed KMP has been summarized in Fig. 12.

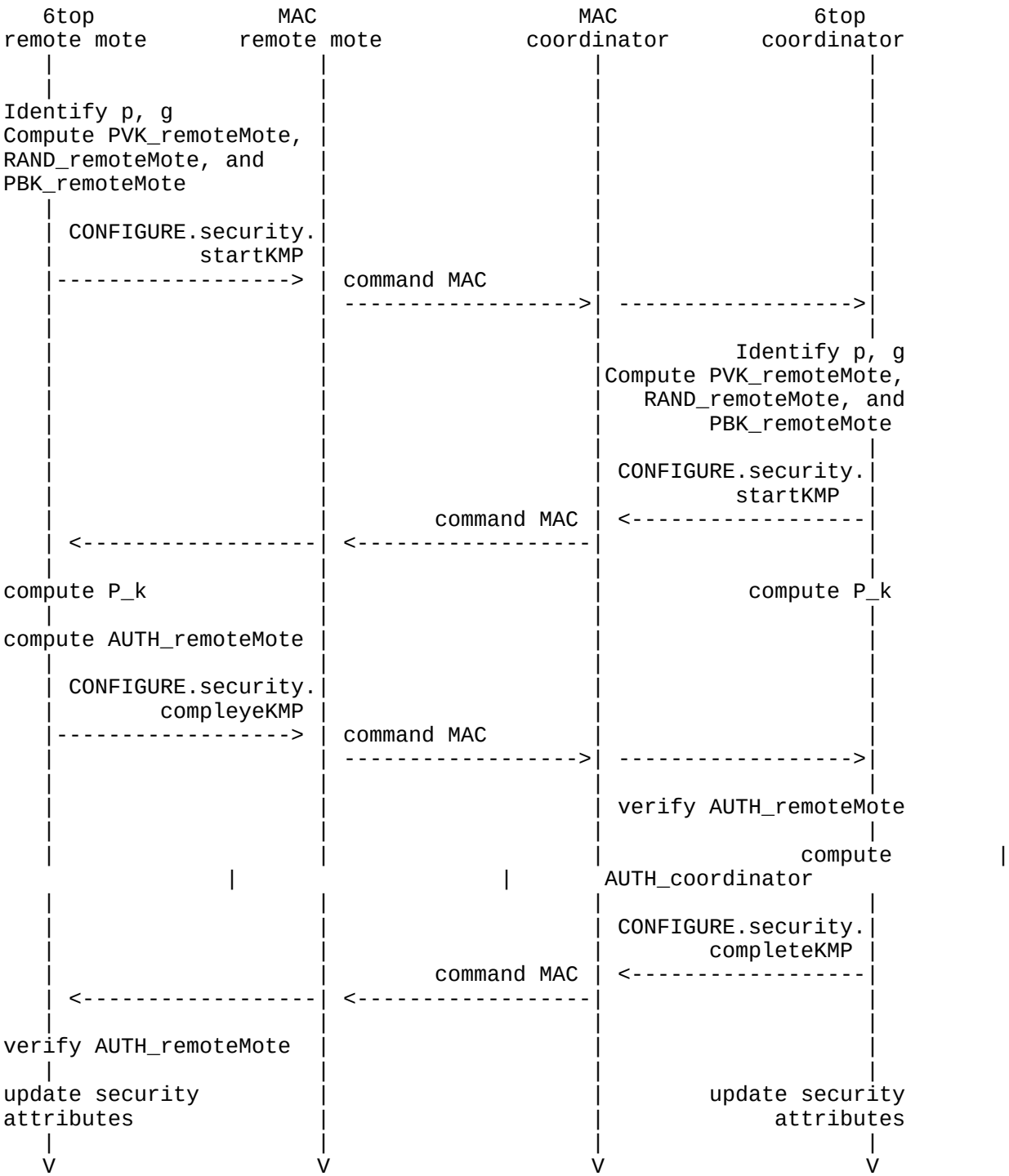
6top
remote mote
MAC
remote mote
MAC
coordinator
6top
coordinator
Identify p, g
Compute PVK_remoteMote,
RAND_remoteMote, and
PBK_remoteMote
CONFIGURE.security.
startKMP
command MAC
Identify p, g
Compute PVK_remoteMote,
RAND_remoteMote, and
PBK_remoteMote
CONFIGURE.security.
startKMP
command MAC
compute P_k
compute P_k
compute AUTH_remoteMote
CONFIGURE.security.
compleyeKMP
command MAC
verify AUTH_remoteMote
compute
AUTH_coordinator
CONFIGURE.security.
completeKMP
command MAC
verify AUTH_remoteMote
update security
attributes
update security
attributes

Figure 12. KMP implementation when the anonymous DH scheme is used

6.3.4 KMP implementation when the certified DH scheme is used

When the certified DH scheme is used, is is assumed that all devices have its certificate in which is stored its public key.

Let CERT_remoteMote, CERT_coordinator, PBK_remoteMote, PBK_coordinator be the certificate of the remote mote, the certificate of the coordinator, the public key of the remote mote, and the public key of the coordinator, respectively.

The KMP is performed through the execution of these operations:

a) The 6top adaptation layer of the remote mote extracts a random number, RAND_remoteMote, that will be used to complete the mutual authentication procedure. Then, it releases a CONFIGURE.security.startKMP command and delivers it to the MAC layer. KeyGenMode, KeyMaterial, and Rand parameters of this command are ser to 00, CERT_remoteMote, and RAND_remoteMote, respectively.

b) Triggered by the CONFIGURE.security.startKMP command, the MAC layer generates Z fragments of the certificate, with size T. For each i-th fragment, it generates a command MAC frame, which is composed by the following fields: MessageType=00, KeyGenMode=01, KeyFalg=TRUE, AuthFlag=FALSE, KeySize=T,, FlagEnabled=TRUE, FragNumber=i, Rand=RAND_remoteMote, and KeyMaterial=fragment_i. In the case the Fully Secured Configuration is enabled, this message is encrypted with the DefaultKey, D_k. Otherwise it is sent in clear.

c) The 6top adaptation layer of the PAN coordinator extracts a random number, RAND_coordinator, that will be used to complete the mutual authentication procedure. Then, it releases a CONFIGURE.security.startKMP command and delivers it to the MAC layer. KeyGenMode, KeyMaterial, and Rand parameters of this command are ser to 00, CERT_coordinator, and RAND_coordinator, respectively.

d) Triggered by the CONFIGURE.security.startKMP command, the MAC layer generates Z fragments of the certificate, with size T. For each i-th fragment, it generates a command MAC frame, which is composed by the following fields: MessageType=00, KeyGenMode=01, KeyFalg=TRUE, AuthFlag=FALSE, KeySize=T, FlagEnabled=TRUE, FragNumber=i, Rand=RAND_coordinator, and KeyMaterial=fragment_i.

In the case the Fully Secured Configuration is enabled, this message is encrypted with the DefaultKey, D_k. Otherwise it is sent in clear.

e) The 6top adaptation layer of the remote mote extracts the public key of the PAN coordinator from its certificate and computes the PreLinkKey, P_k

P_k = PBK_coordinator^PVK_remoteMote * mod P.

f) The 6top adaptation layer of the PAN coordinator extracts the public key of the remote mote and computes the PreLinkKey, P_k,

P_k = PBK_remoteMote^PVK_coordinator * mod P.

g) Both remote motes and PAN coordinator compute the LinkKey by using the procedure described in Sec. 6.3.4.

h) The 6top adaptation layer of the remote mote computes the authentication parameter, AUTH_remoteMote, by following these steps:

h.1) generate an authenticator message:
authMsg_remoteMote = P_k || RAND_coordinator || RAND_remoteMote

h.2) sign with the private key the 128-hash function of the authenticator message:

sign=S(PVK_remoteMote, H_128(authMsg_remoteMote))

h.3) obtain AUTH_remoteMote by encrypting with the preLinkKey, P_k, the sign computed before:

AUTH_remoteMote=E(P_k, sign)

Then, it releases a CONFIGURE.security.completeKMP command with KeyGenMode and AuthenticationField set to 00 and AUTH_remoteMote, respectively.

i) The MAC layer of the remote mote sends to the coordinator a new MAC command message to complete the mutual authentication. This message is composed by the following fields: MessageType=10, KeyGenMode=01, KeyFalg=FALSE, AuthFlag=TRUE, KeySize=0, and AuthenticationField=AUTH_remoteMote. This message is protected by using the LinkKey computed before.

j) The 6top adaptation layer of the PAN coordinator verifies the

validity of the received AUTH_remoteMote parameter. In affirmative case, it computes the authentication parameter, AUTH_coordinator, by following these steps:

j.1) generate an authenticator message:

authMsg = P_k || RAND_remoteMote || RAND_coordinator

j.2) sign with the private key the 128-hash function of the authenticator message:

sign=S(PVK_remoteMote, H_128(authMsg))

j.3) obtain AUTH_remoteMote by encrypting with the preLinkKey, P_k, the sign computed before:

AUTH_coordinator=E(P_k, sign)

Then, it releases a CONFIGURE.security.completeKMP command with KeyGenMode and AuthenticationField set to 00 and AUTH_coordinator, respectively.

k) The MAC layer of the PAN coordinator sends to the remote mote a new MAC command message to complete the mutual authentication. This message is composed by the following fields: MessageType=10, KeyGenMode=01, KeyFalg=FALSE, AuthFlag=TRUE, KeySize=0, and AuthenticationField=AUTH_coordinator. This message is protected by using the LinkKey computed before.

l) The remote motes verifies the validity of the received AUTH_coordinator parameter.

m) PAN coordinator and remote mote update MAC security attributes according to procedures described in Sec. 6.3.6 and Sec 6.3.7, respectively.

The aforediscussed KMP has been summarized in Fig. 13.

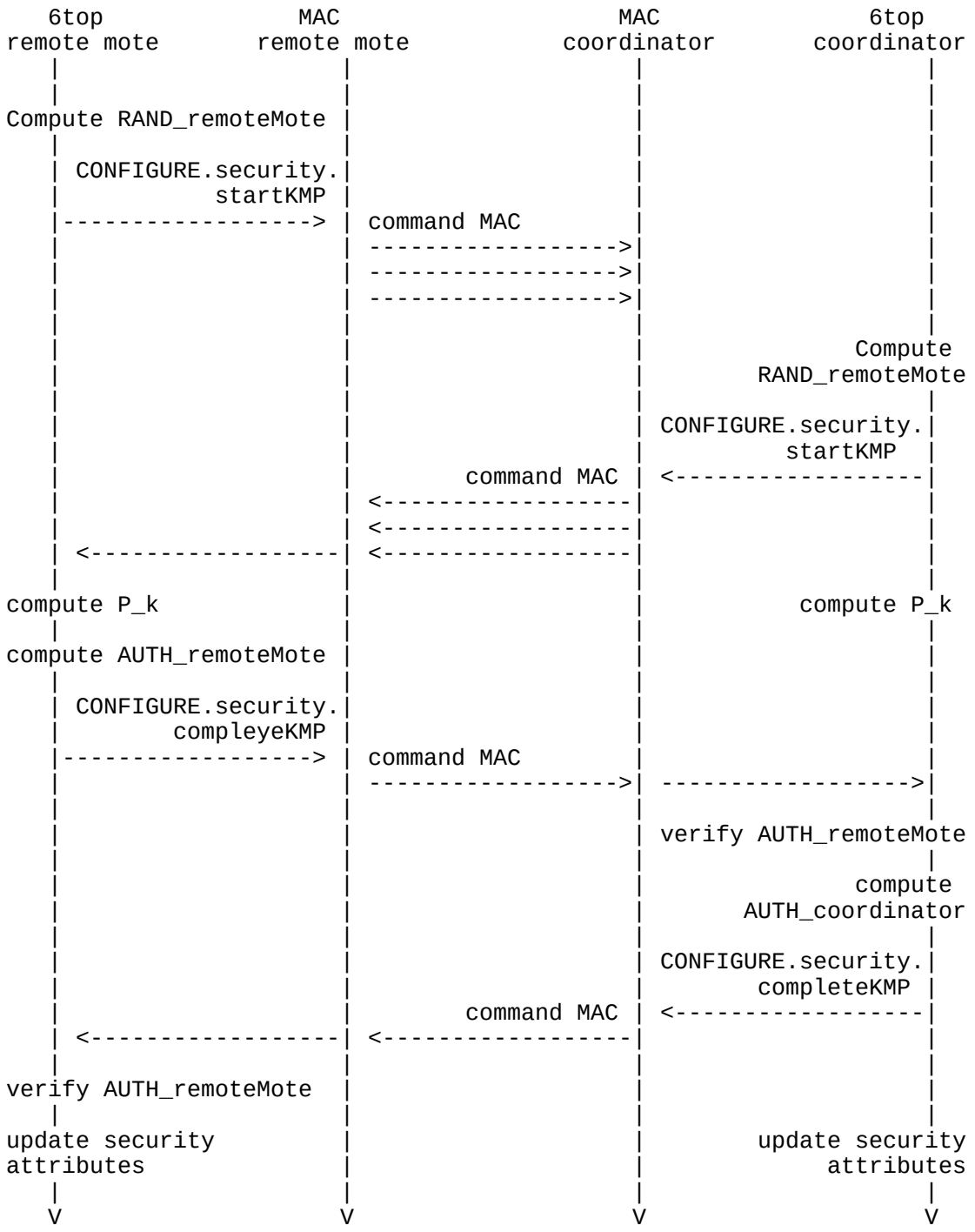
6top
remote mote
MAC
remote mote
MAC
coordinator
6top
coordinator
Compute RAND_remoteMote
CONFIGURE.security.
startKMP
command MAC
Compute
RAND_remoteMote
CONFIGURE.security.
startKMP
command MAC
compute P_k
compute P_k
compute AUTH_remoteMote
CONFIGURE.security.
compleyeKMP
command MAC
verify AUTH_remoteMote
compute
AUTH_coordinator
CONFIGURE.security.
completeKMP
command MAC
verify AUTH_remoteMote
update security
attributes
update security
attributes

Figure 13. KMP implementation when the certified DH scheme is used

6.3.5 Generation of the LinkKey

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 [IEEE802154].

For each i-th group of block ciphers, the LinkKey, L_k, is computed as in the following:

$$L_k = H_{128} (i \mid PAN\_ID \mid P_k).$$

6.3.6 Update of MAC security attributes for the PAN coordinator after the generation of the LinkKey

After the calculation of the i-th LinkKey, the 6top adaptation layer of the PAN coordinator updates its MAC security attributes as described in what follows.

a) If i=1, a new DeviceDescriptor element, associated to the remote mote with which it has negotiated a link key, is created. It is composed by:

a.1) the PANId, which is set to the PAN_ID value.

a.2) The ShortAddress, which is set to the MAC address of the RFD node whenever the short addressing mode is used. This parameter is set to 0xfffe if only the extended addressing mode is used. In the case its value is unknown, this parameter is set to 0xfff.

a.3) The ExtAddress, which is set to the IEEE MAC address of the RFD node.

a.4) The FrameCounter, which is set to the FrameCounter value extracted from the latest packet received by the RFD node.

a.5) The Extempt boolean flag, which is set to the allowed value of the DeviceOverriddeSecurityMinimum variable described in Fig. 2.

b) A new KeyIdLookupList data structure is created. A KeyIdLookupDescriptor is generated and stored into the KeyIdLookupList data structure. The KeyIdMode, the KeySource, and the KeyIndex variables of this KeyIdLookupDescriptor are set to 0x03, the MAC address of the remote mote that initialized the Key Negotiation Phase, and 1, respectively. DeviceAddrMode, DevicePANId, and DeviceAddress are not set because of the selected KeyIdMode (see Tab. 65 of the IEEE 802.15.4 standard for more details [IEEE802154]).

c) A KeyUsageList data structure is created and stored within the KeyDescriptor element. One KeyUsageDescriptor associated to data MAC frames is created and stored into the KeyUsageList data structure.

d) A DeviceDescriptorHandleList is created and populated with the pointer to the DeviceDescriptor created at the point a).

Then, the 6top layer delivers the LinkKey, the KeyIdLookupList, the KeyUsageList, and the DeviceDescriptorHandleList to the MAC layer by using the CONFIGURE.security.macKeyTable command (as discussed in [I-D.wang-6tisch-6top], in Sec. 2.4.9.2).

Triggered by the CONFIGURE.security.macKeyTable command, the MAC layer will create a KeyDescriptor associated to the LinkKey, L_k,in which storing all the parameters received by the 6top layer, and and will store it within the macKeyTable.

6.3.7 Update of MAC security attributes for the remote mote after the generation of the LinkKey

After the calculation of the i-th LinkKey, the 6top adaptation layer of the remote mote updates its MAC security attributes as described in what follows.

a) A new KeyIdLookupList data structure is created. A KeyIdLookupDescriptor is generated and stored into the KeyIdLookupList data structure. The KeyIdMode, the KeySource, and the KeyIndex variables of this KeyIdLookupDescriptor are set to 0x03, the MAC address of the PAN coordinator that initialized the Key Negotiation Phase, and 1, respectively. DeviceAddrMode, DevicePANId, and DeviceAddress are not set because of the selected KeyIdMode (see Tab. 65 of the IEEE 802.15.4 standard for more details [IEEE802154]).

b) A KeyUsageList data structure is created and stored within the

KeyDescriptor element. One KeyUsageDescriptor associated to data MAC frames is created and stored into the KeyUsageList data structure.

c) A DeviceDescriptorHandleList is created and populated with the pointer to the DeviceDescriptor associated to the PAN coordinator and created during the Bootstrap Phase.

Then, the 6top layer delivers the LinkKey, the KeyIdLookupList, the KeyUsageList, and the DeviceDescriptorHandleList to the MAC layer by using the CONFIGURE.security.macKeyTable command (as discussed in [I-D.wang-6tisch-6top], in Sec. 2.4.9.2).

Triggered by the CONFIGURE.security.macKeyTable command, the MAC layer will create a KeyDescriptor associated to the LinkKey, L_k,in which storing all the parameters received by the 6top layer, and and will store it within the macKeyTable.

7 Additional features

There is the possibility to switch from the Flexible Secured to the Hybrid Secure configuration.

To this aim, during the join process, a mote without security capabilities sends to the PAN coordinator a Beacon Request message with the SecurityEnabled flag set to FALSE.

The PAN coordinator, if properly configures, switches to the Hybrid Secure configuration and update all the MAC security attributes accordingly.

From this moment on, the coordinator will send broadcast messages in clear.

8 Security Considerations

There are no security considerations for this document.

9 IANA Considerations

There is no IANA action required for this document.

10 References

10.1 Normative References

[I-D.watteyne-6tisch-tsch] Watteyne, T., "Using IEEE802.15.4e TSCH in an LLN context: Overview, Problem Statement and Goals", Internet-Draft draft-watteyne-6tisch-tsch-00, (work in progress) October 2013.

[I-D.wang-6tisch-6top] Wang, Q., Vilajosana, X. and T. Watteyne, "6TiSCH Operation Sublayer (6top)", Internet-Draft draft-wang-6tisch-6top-00, (work in progress) October 2013.

[I-D.draft-palattella-6tisch-terminology] Palattella, MR., Ed., Thubert, P., Watteyne, T., and Q. Wang, "Terminology in IPv6 over Time Slotted Channel Hopping". Internet Draft draft-palattella-6tisch-terminology-00, (work in progress) October 2013.

[DH] W. Diffie and M. Hellman, "New directions in cryptography," IEEE Trans. Inf. Theor. 22, 6 Sep., 2006.

[StsProtocol] Whitfield Diffie, Paul C. van Oorschot and Michael J, "Wiener, Authentication and authenticated key exchange", Designs, Codes, and Cryptography, 1987.

10.2 Informative References

[IEEE802154e] IEEE standard for Information Technology, "IEEE std. 802.15.4e, Part. 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs) Amendament 1: MAC sublayer", April 2012.

[IEEE802154] IEEE standard for Information Technology, "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", June 2011.

[ZIGBEEIP] ZigBee Public Document 15-002r00, "ZigBee IP Specification", 2013.

[Camtepe2005] Seyit A. Camtepe and Bulent Yener, "Key Distribution Mechanisms for Wireless Sensor Networks: a Survey", Technical Report 2005.

[Walters07] John Paul Walters, Zhengqiang Liang, Weisong Shi, and Vipin Chaudhary, "Wireless sensor network security: A survey," in book chapter of Security", Proc. of Distributed, Grid, and Pervasive Computing, CRC Press, 2007.

[Wang2006] Yong Wang, Garhan Attebury, and Byrav Ramamurthy, "A survey of security issues in wireless sensor networks", IEEE Communications Surveys & Tutorials, 2006

[Cayirci2007] Security in Wireless Ad Hoc and Sensor Networks. John Wiley & Sons, 2007.

[I-D.roll-security-framework] Tzeta Tsao, Roger Alexander, Mischa Dohler, Vanesa Daza, and Angel Lozano, "A Security Framework for Routing over Low Power and Lossy Networks", Internet Draft draft-ietf-roll-security-framework-07, Jan 2013.

[I-D.garcia-core-security] O. Garcia-Morchon, S. Keoh, S. Kumar, R. Hummen, and R. Struik, "Security Considerations in the IP-based Internet of Things," IETF, Internet Draft, Sep. 2013.

[RFC5191] Forsberg, D., Ohba, Y., Patil, B., Tschofenig, H., and A. Yegin, "Protocol for Carrying Authentication for Network Access (PANA)", RFC 5191, May 2008.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, January 2012.

[HIPDEX] Moskowitz, R., "HIP Diet EXchange (DEX)", draft-moskowitzhip-rg-dex-06 (work in progress), May 2012.

[PalattellaSurvey] Maria Rita Palattella, Nicola Accettura, Xavier Vilajosana, Thomas Watteyne, Luigi Alfredo Grieco, Gennaro Boggia, and Mischa Dohler," Standardized Protocol Stack For The Internet Of (Important) Things", IEEE Communications Surveys & Tutorials, December, 2012

[StallingsSecurityBooks] William Stallings: Cryptography and network

security - principles and practice. Prentice Hall 2010.

[Becher2006] Alexander Becher, Zinaida Benenson, and Maximillian Dornseif, "Tampering with motes: real-world physical attacks on wireless sensor networks", In Proc. of conf. on Security in Pervasive Computing (SPC), Berlin, 2006

[TELOSB] "Crossbow Technology, TelosB Datasheet." [Online]. Available: http://www.willow.co.uk/TelosB_Datasheet.pdf

[Riaz2009] Riaz, R.; Ki-Hyung Kim; Ahmed, H.F., "Security analysis survey and framework design for IP connected LoWPANs," Autonomous Decentralized Systems, 2009. ISADS '09. International Symposium on , vol., no., pp.1,6, 23-25 March 2009

[Altolini2013] Altolini, D.; Lakkundi, V.; Bui, N.; Tapparello, C.; Rossi, M., "Low power link layer security for IoT: Implementation and performance analysis," Wireless Communications and Mobile Computing Conference (IWCMC), 2013 9th International , vol., no., pp.919,925, 1-5 July 2013

[Watteyne2012] Thomas Watteyne, Xavier Vilajosana, Branko Kerkez, Fabien Chraim, Kevin Weekly, Qin Wang, Steven D. Glaser, Kris Pister: OpenWSN: a standards-based low-power wireless development environment. Trans. Emerging Telecommunications Technologies 23(5): 480-493 (2012)

Appendix A. DH protocol

A.1 Security considerations about the DH protocol

As discussed in Sec. 6.5.5, the CCM* transformation requires a 128-bit key.

According to the DH algorithm, and considering properties of the modular arithmetic [DH], the length of both the public key of a mote and the prime number used for its generation must be at least equal to 128 bits.

The security level of the proposed approach does not depend from the number of prime numbers stored into the PrimeNumberTable (because these numbers are not the keys), but it coincides with the security level of the DH protocol. hence, it is strictly related to length of prime numbers, S.

In particular, the total number of keys that a mote can use during the Key Negotiation Phase are equal to 2^S. Supposing to have S >= 128, the total number of keys is higher than 3*10^38. This should guarantee a very high resilience to any kind of brute force attack.

Authors' Addresses

G. Piro
DEI, Dep. of Electrical and Information Engineering
Politecnico di Bari
Via Orabona 4, 70125, Bari, ITALY
Phone: +39 0805963301

Email: g.piro@poliba.it

G. Boggia
DEI, Dep. of Electrical and Information Engineering
Politecnico di Bari
Via Orabona 4, 70125, Bari, ITALY
Phone: +39 0805963913

Email: g.boggia@poliba.it

L.A. Grieco
DEI, Dep. of Electrical and Information Engineering
Politecnico di Bari
Via Orabona 4, 70125, Bari, ITALY
Phone: +39 0805963911

Email: a.grieco@poliba.it