ZigBee-2007 Security Essentials

Ender Yüksel Hanne Riis Nielson Flemming Nielson
Informatics and Mathematical Modelling, Technical University of Denmark
Richard Petersens Plads bldg 321, DK-2800 Kongens Lyngby, Denmark

{ey,riis,nielson}@imm.dtu.dk

Abstract

ZigBee is a fairly new but promising standard for wireless networks due to its low resource requirements. As in other wireless network standards, security is an important issue and each new version of the ZigBee Specification enhances the level of the ZigBee security.

In this paper, we present the security essentials of the latest ZigBee Specification, *ZigBee-2007*. We explain the key concepts, protocols, and computations. In addition, we formulate the protocols using standard protocol narrations. Finally, we identify the key challenges to be considered for consolidating ZigBee.

1 Introduction

Low-cost and low-power wireless networks have an enormous potential for many areas such as health, security, telecom, agriculture, energy, etc. ZigBee is an emerging standard for *low-rate* (in terms of cost, power consumption, range, data) wireless networks and the use of the ZigBee technology is limited only by imagination [33, 43]. However, wireless networks are generally susceptible to attacks and security is clearly very important. As a low-rate standard, ZigBee has very limited resources (memory, processor, storage, power, etc.), therefore implementing security guarantees is a great challenge and the security specification is of paramount importance.

The ZigBee Specification is difficult to read with hundreds of pages and refers to many other standards. In this study, we present a high level self-contained overview and explain the key components of the latest ZigBee Specification, *ZigBee-2007*. We explain the important points in key protocols, algorithms, and computations. We derive the protocol narrations from the specification, which will be useful for both implementations and security analyses. In addition, we survey the key studies related to ZigBee security. Therefore, this paper goes beyond a mere survey, aiming to facilitate the use and study of ZigBee in both industry and academia.

The rest of this paper is organized as follows. We present the structure and the development of the *ZigBee Specification* in Section 2. We give an overview of the *ZigBee Stack Architecture* in Section 3. We explain the *ZigBee Security Services Specification*, the core part of the ZigBee security, in Section 4. We explain the *ZigBee Security Procedures*, the security protocols for ZigBee networks, in Section 5. We discuss the literature on the *ZigBee Security*, and present latest related work in Section 6. We conclude in Section 7, and leave the *protocol narrations* to the appendix.

2 The ZigBee Specification

ZigBee is a very low-cost, very low-power-consumption, two-way, wireless communication standard that is being developed by the *ZigBee Alliance*, an independent nonprofit organization. ZigBee is targeted at radio frequency (**RF**) applications that require low data rate, long battery life, and secure networking. ZigBee has a very wide application area that covers consumer electronics, home and building automation, industrial controls, PC peripherals, medical sensor applications, toys and games, etc.

The ZigBee Specification provides a definitive description of the ZigBee protocol standard as a basis for the implementations. The specification is a kind of *dynamic document*, which is currently called *ZigBee-2007* [5]. The history of the specification begins with the ZigBee v.1.0 draft that was published in December 2004 and now referred to as *ZigBee-2004* [7]. The first version of the ZigBee Specification was published in February 2006; the successor ZigBee Specification was published in October 2006 and called *ZigBee-2006* [6]. From February 2006 to present, there have been seven releases. The development is so rapid that sometimes the interval between two specifications is less than one month (*i.e. the last version of ZigBee-2006 and the first version of ZigBee-2007*). However, in the latest specification (ZigBee-2007) it is stated that there exists three main specifications: ZigBee-2004, ZigBee-2006 and ZigBee-2007. The current specification, ZigBee-2007, is required to be backward compatible with ZigBee-2006 but not necessarily with ZigBee-2004. In addition, it is stated that ZigBee-2006 is not required to have backwards compatibility with ZigBee-2004.

The ZigBee-2007 Specification [5] is composed of a short protocol overview followed by specifications of the Application Layer, Network Layer, and Security Services. In addition, the specification covers necessary details that are required in ZigBee implementations.

ZigBee-2007 contains two feature sets that interoperate network-wise: *ZigBee* and *ZigBee* PRO. A *feature set* is an agreement of configuration parameters, network settings and policies. The ZigBee PRO feature set has two security modes (Standard Security, High Security), and three types of security keys (Network Key, Link Key, Master Key), whereas the ZigBee feature set has only Standard Security mode and two types of security keys (Network Key, Application Link Key). Both feature sets use symmetric encryption, the *Advanced Encryption Standard* (**AES**-128) [3], and apply authentication/encryption on Network and Application layers.

ZigBee-2007 also has different application profiles. An *application profile* is a common application level language for exchanging data in a given application domain. The ZigBee public application profiles are *Commercial Building Automation*, *Home Automation*, *Personal Home and Hospital Care*, *Smart Energy*, *Telecom Applications*, and *Wireless Sensor Applications*.

Related to the device types, we used two different classifications: hardware and logical. The hardware device type distinguishes the type of the hardware platform and it may be either *Full Function Device* (**FFD**), or *Reduced Function Device* (**RFD**) according to the IEEE 802.15.4 standard [1]. A logical device type distinguishes devices in a specific network and it may be *Coordinator*, *Router*, or *End Device* in a ZigBee network.

Throughout this paper, ZigBee refers to the ZigBee-2007 Specification unless otherwise stated. In addition, we typed the abbreviations in boldface font, where they are first defined.

3 The ZigBee Stack Architecture

ZigBee is built on the Physical layer (**PHY**) and the Medium Access Control layer (**MAC**), both defined by the IEEE 802.15.4-2003 standard [1]. The PHY layer can operate in two separate frequency ranges: lower 868(European)/915(United States, Australia, etc.) MHz and higher 2.4 GHz (worldwide). The MAC layer controls access to the radio channel using a CSMA-CA mechanism. Upon this structure, ZigBee builds the Network layer (**NWK**) and the Application layer (**APL**) which consists of the Application Support sublayer (**APS**) and the ZigBee Device Object (**ZDO**). Fig. 1 shows the ZigBee stack architecture, including the end manufacturer defined part in dashed box. This study focuses on the Security Service Provider part of the ZigBee Specification, which interacts with the NWK and APS layers.

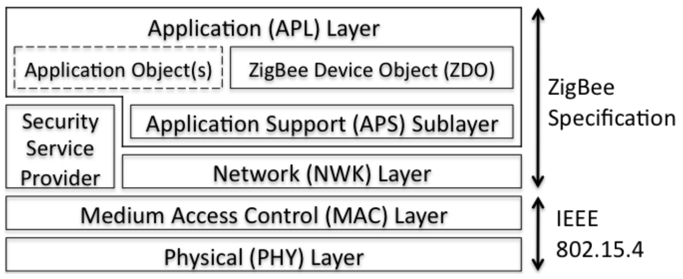
Application (APL) Layer
Application Object(s)
ZigBee Device Object (ZDO)
Security Service Provider
Application Support (APS) Sublayer
Network (NWK) Layer
Medium Access Control (MAC) Layer
Physical (PHY) Layer
ZigBee Specification
IEEE 802.15.4

Figure 1: ZigBee Stack Architecture

4 The Security Service Provider

The Security Services Specification is a chapter in the ZigBee Specification [5] which specifies the security services available within the ZigBee stack. These services include methods for key establishment, key transport, frame protection and device management. As shown in Fig. 1, ZigBee provides security mechanisms for the NWK and APS layers. Each layer is responsible for securing their frames. In addition, the APS sublayer provides services for security relationship establishment and maintenance whereas ZDO manages the security policies and the configuration of ZigBee devices. Throughout this section, we present the overall security architecture and the security mechanisms in different layers.

4.1 Open Trust Model

The ZigBee security architecture depends on assumptions such as safekeeping the symmetric keys, proper implementation of the cryptographic mechanisms and the associated security policies involved. Besides, ZigBee assumes that different applications using the same radio are not logically separated (*e.g. by firewall*). Also, a device cannot verify whether cryptographic separation between different applications/layers on another device is properly implemented. Therefore, it must be assumed that separate applications using the same radio trust each other, which means that there is no cryptographic task separation. In addition, the lower layers (i.e. APS, NWK, MAC) are fully accessible to the applications. As a result, ZigBee has the notion of an *open trust model* that is derived from these assumptions. This model states that the security services only protect the interfaces between different devices, but not the interfaces between different layers nor the different applications on the same device. Here, protection means cryptographic protection, and for the separation of the interfaces between the different stack layers on the same device non-cryptographic mechanisms should be employed in the designs. As stated in the specification, this model allows the reuse of the same keying material among different layers on the same ZigBee device. The model also allows end-to-end security to be realized on a device-to-device basis instead of layer-to-layer basis.

4.2 Architectural Design Choices

In this subsection we focus on the design choices listed in the current ZigBee Specification. The main point in these choices is that any malevolent device should not be able to use the network to transport frames across the network without permission.

1. *The layer that originates a frame is responsible for initially securing it.* A simple example from the specification: If a NWK command frame needs protection, then NWK layer security must be used.

2. *If protection from theft of service is required, then NWK layer security shall be used for all frames.* Only a device that joined the network and authenticated (i.e. received the active network key) will be able to communicate using the network. However, it is not possible to

Table 1: Key Acquirement Schemes

Method \ Key Type	**NK**	**MK**	**LK**
Key Transport	YES	YES	YES
Key Establishment	NO	NO	YES
Pre-installation	YES	YES	YES

Table 2: Availability of Key Types

Keys	**Layers**		**Modes**	
	NWK	**APL**	**SS**	**HS**
NK	YES	YES	YES	YES
MK	NO	YES	NO	YES (O)
LK	NO	YES	YES (O)	YES (O)

apply NWK layer security between a router and a joined but unauthenticated device.

3. *Security can be based on the reuse of keys by each layer.* This reduces the storage costs.

4. *End-to-end security is enabled, based on a shared key between only two devices.* This actually limits the trust requirement to only two devices communicating.

5. *The security level used by all devices in a given network, and by all layers of a device shall be the same.* If an application requires more security, then it shall form its own separate network with a higher security level.

Other decisions such as key update/expire, counter overflow, loss of synchronization, error conditions arising from securing frames, etc. should be included in the application profiles and must be addressed correctly in the real implementations.

4.3 Security Keys and Trust Center

ZigBee devices use 128-bit symmetric encryption keys to provide security amongst a network. A *Link Key* (**LK**) is shared by two ZigBee devices and it is used to secure unicast communication between APL peer entities. LK is used as the basis of the security services (i.e. key transport, authentication) in *High Security* mode (**HS**). A *Network Key* (**NK**) is shared amongst all devices in a network and it is used to secure broadcast communications in a network. NK is used as the basis of the security services (i.e. authentication, frame security) in *Standard Security* mode (**SS**). A *Master Key* (**MK**) is used to establish a key and it is shared pairwise between two ZigBee devices. The intended recipient is always aware of the key type that is used in frame protection.

The security keys can be acquired in different ways depending on their types as shown in Table 1. *Key Transport* is the case that the Trust Center of the network sends the key to the device. *Key Establishment* is the method that is used to establish a pairwise key (*i.e. LK*) between two devices. Note that for this method, a pre-shared key (*i.e. MK*) is required between two devices. *Pre-installation* is the case that the device acquires the key before joining the network.

In ZigBee-2007, NKs have two different types: standard (**SNK**), and high-security (**HSNK**). It is stated that NK distribution and initialization of the Frame Counters (*See Subsection 4.4*) depend on the type of the NK, but in both cases the messages are secured in the same way. The availability of the security keys to the different layers and different security modes are given in Table 2. Some key types are optional for some security modes and they are indicated with "(O)" in the table. In addition, all ZigBee layers must share the active NK and associated incoming/outgoing frame counters.

In order to avoid reuse of keys across different security services, it is possible to derive different keys from the LK. Uncorrelated keys can be derived using a one-way function so that

Table 3: Derived Key Types (0x: Hexadecimal)

Key Type	**Computation**	**Explanation**
Key-Transport Key	HMAC(0x00)$_{LK}$	Protects transported NKs
Key-Load Key	HMAC(0x02)$_{LK}$	Protects transported MKs and LKs
Data Key	LK	Equal to the LK

Table 4: Key Distribution

Purpose	**Device receives from TC**	**Via**
Trust Management	Initial MK or Active NK	Unsecured Key Transport
Network Management	Initial active NK and updated NKs	Secured Key Transport
Configuration	MK or LKs	Secured Key Transport

execution of different security protocols can be logically separated. Three types of secret keys can be derived from a LK as listed in Table 3. The derivation of these keys, except Data Key, requires computation of a *Keyed-Hashing for Message Authentication Code* (**HMAC**) [4]. All the derived keys must share the associated frame counters.

Trust Center In each secure ZigBee network, there exists a unique Trust Center (**TC**) application which is trusted by all the devices in the network. TC distributes keys as part of network and end-to-end application configuration management. In high-security (commercial) applications devices are using MK, whereas in low-security (residential) applications devices are using NK to initiate secure communication with TC. In parallel with the key acquirement schemes in Table 1, MK and NK can be obtained by either pre-installation or a kind of key transport which is called *in-band unsecured key transport*. Certainly, the latter option is not tolerable in vulnerable situations. The interaction between a ZigBee device and the TC for different purposes is given in Table 4.

As we mentioned before, there are three logical device types in a ZigBee network: *coordinator*, *router*, and *end device*. TC is not a device type, but an application. In a ZigBee network, the ZigBee Coordinator configures the security level (which can also be *unsecured*). The ZigBee Coordinator also configures the TC of the network, which is by default the ZigBee Coordinator itself.

Security Modes TC can be configured to operate in either Standard Security mode (SS) or High Security mode (HS). In SS mode, which is designed for the residential applications, the TC is required to maintain the SNK and control the policies of network admittance. In HS mode, which is designed for the commercial applications, the TC is required to maintain a list of all the devices in the network (this is actually optional in SS mode), all the relevant keys (MKs, LKs, HSNKs) and control the policies of network admittance. As a result, the required memory of the TC grows with the number of the devices in the network in HS mode, but not in SS mode. In addition, the implementation of the Symmetric-Key Key Exchange (*See Subsection 4.7*) and the Mutual Entity Authentication (*See Subsection 4.8*) protocols are mandatory in HS mode. Right now, among the application profiles mentioned in Section 2, only the Commercial Building Automation (**CBA**) profile uses HS mode.

4.4 Network Layer Security

The Network (NWK) layer provides functionality to ensure correct operation of the MAC layer and also provides suitable service interface to the APL layer. When a NWK layer frame needs to be secured, the NWK layer secures it by using AES [3] encryption/authentication in the *Enhanced Counter with CBC-MAC* (**CCM***) mode of operation (*see Subsection 4.9*). The NWK layer processes outgoing/incoming frames in order to securely transmit/receive them. The upper layers control the security processing operations by setting up the security keys,

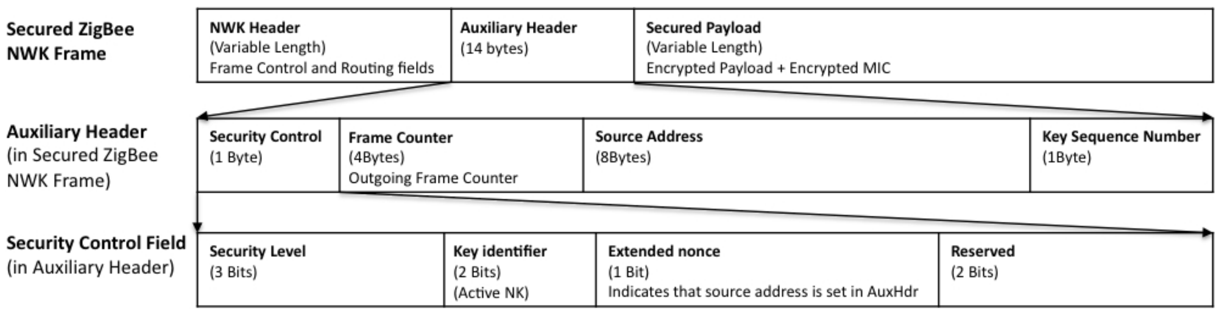
Secured ZigBee NWK Frame
NWK Header (Variable Length) Frame Control and Routing fields
Auxiliary Header (14 bytes)
Secured Payload (Variable Length) Encrypted Payload + Encrypted MIC
Auxiliary Header (in Secured ZigBee NWK Frame)
Security Control (1 Byte)
Frame Counter (4Bytes) Outgoing Frame Counter
Source Address (8Bytes)
Key Sequence Number (1Byte)
Security Control Field (in Auxiliary Header)
Security Level (3 Bits)
Key identifier (2 Bits) (Active NK)
Extended nonce (1 Bit) Indicates that source address is set in AuxHdr
Reserved (2 Bits)

Figure 2: Secured ZigBee NWK Frame

Table 5: NWK Frame Security (||: Concatenation)

Processing Outgoing Frames	**Processing Incoming Frames**
1. Retrieve active NK, outgoing frame counter, key sequence number and security level from the NWK Layer Information Base (**NIB**). 2. Set AuxHdr using the parameters in Step 1. 3. Execute CCM* mode encryption and authentication with the parameters: length of MIC (obtained from security level), NK, and CCM* Nonce (CCM* Nonce uses the values from AuxHdr and constructed as "Source Address || Frame Counter || Security Control". 4. Construct the outgoing frame, depending on the encryption requirement, as in Fig. 2. 5. Increase outgoing frame counter in NIB. 6. Set the *Security Level* subfield of AuxHdr as "000".	1. Determine security level from NIB and overwrite it to the *Security Level* subfield. 2. Determine the key sequence number, sender address, and the *Received Frame Count* from AuxHdr. 3. Obtain the corresponding (matching the key sequence number) security material from NIB. If *Received Frame Count* is less than *Frame Count* then FAIL. 4. Execute CCM* mode decryption and authentication with the same parameters as in outgoing frames. 5. Set Frame Count to "Received Frame Count + 1". Store Frame counter and the Sender Address in the NIB.

frame counters and the security level.

The structure of a secured NWK frame is given in Fig. 2. A secured NWK frame is a NWK frame that has an Auxiliary Header (**AuxHdr**) and this is indicated in the *Frame Control* field of the NWK Header. AuxHdr includes the *Frame Counter*, which has the purpose of providing frame freshness and preventing processing of duplicate frames. An important point here is that the word "secured" does not necessarily mean that encryption is applied (*i.e. in the case of integrity-only protection*). Therefore, a Secured Payload does not need to have an encrypted payload. The *Security Level* field in the Security Control part of the AuxHdr indicates which security level is applied. The level can be None (ne security at all), MIC (integrity protection only, with three different MIC lengths: MIC-32, MIC-64, MIC-128), ENC (encryption only), and ENC-MIC (both encryption and integrity protection with three different MIC lengths: ENC-MIC-32, ENC-MIC-64, ENC-MIC-128). The Message Integrity Code (**MIC**) is computed using the NWK header, the AuxHdr and the (encrypted) payload.

The processing of the incoming and outgoing frames are given in Table 5. An interesting security precaution here is hiding the security level in the last step of Outgoing Frames Processing. Although the rationale behind this action is not defined in the specification, it is clear that it is not a significant protection since there are only eight choices (as listed above).

4.5 Application Layer Security

As shown in Fig. 1, the Application (APL) Layer is composed of the APS sublayer, the ZDO and manufacturer defined application objects. A maximum of 240 distinct application objects can be defined. ZDO is responsible for initializing APS, NWK, Security Service Provider, and assembling the information from applications. ZDOs are applications that employ NWK and APS primitives to implement ZigBee End Devices, ZigBee Routers and ZigBee Coordinators. When an APL layer frame needs to be secured, the APS sublayer handles security. Therefore, APL security actually covers APS sublayer security which is explained in Subsection 4.6.

4.6 Application Support (APS) Sublayer Security

The Application Support (APS) sublayer provides an interface between the NWK and the APL layers through a general set of services (for use of ZDO and Application Objects) provided by APS data and management entities. The APS sublayer processes outgoing/incoming frames in order to securely transmit/receive the frames and establish/manage the cryptographic keys. The upper layers issue primitives to APS sublayer to use its services. APS Layer Security includes the following services: *Establish Key*, *Transport Key*, *Update Device*, *Remove Device*, *Request Key*, *Switch Key*, *Entity Authentication*, and *Permissions Configuration Table*. Below we explain the current usage of these services, keeping in mind that they can be extended in the future.

The **Establish Key** service is the mechanism for establishing a LK between two ZigBee devices. In ZigBee-2007, Symmetric-Key Key Exchange (**SKKE**) is the method for key establishment and MK is the trust information used in key establishment (*See Subsection 4.7*). Recently, Public-Key Key Establishment (**PKKE**) is included in a ZigBee application profile (*See Subsection 6.1*).

The **Transport Key** service provides secure or unsecure means to transport a NK, LK, or MK.

The **Update Device** service provides secure means for a ZigBee Router to inform the TC that a device changed its status (*i.e. joined or left the network*).

The **Remove Device** service provides secure means for the TC to inform a ZigBee Router that one of his children should be removed.

The **Request Key** service provides secure means for a device to request the *active NK* or the *end-to-end application MK* from another device (*i.e. TC*).

The **Switch Key** service provides secure means for TC to inform a device to switch to the alternate NK.

The **Entity Authentication** service, *which was not present in ZigBee-2006*, provides authenticity between two devices based on a shared secret (*i.e. NK*).

The **Permission Configuration Table** (**PCT**) stores the information of which devices have authorization to perform which commands. In addition, PCT determines whether security based on LK is required or not. Maintaining a PCT is optional.

The services of the APS sublayer are issued in *APS Command Frame*s. The structure of the APS Command Frames is given in Fig. 3. The first two fields of all the frames, the Frame Counter and the APS Counter, form the *APS Header*. The remaining fields form the *APS Payload*, whose first field *APS Command Identifier* indicates the type of the command frame (SKKE, Transport-Key, Update-Device, Remove-Device, Request-Key, Switch-Key, Entity Authentication, etc.), as shown in Fig. 3.

There are some important points regarding the APS command frames in Fig. 3. All *Key Establishment* (*e.g. SKKE*) command frames are sent unsecured. The *Status* field of the *Update-Device* command indicates the security mode (SS/HS), the security of the frame (Secured/Unsecured), and the type of the join (Join/Rejoin); unless the device is leaving. *Partner*

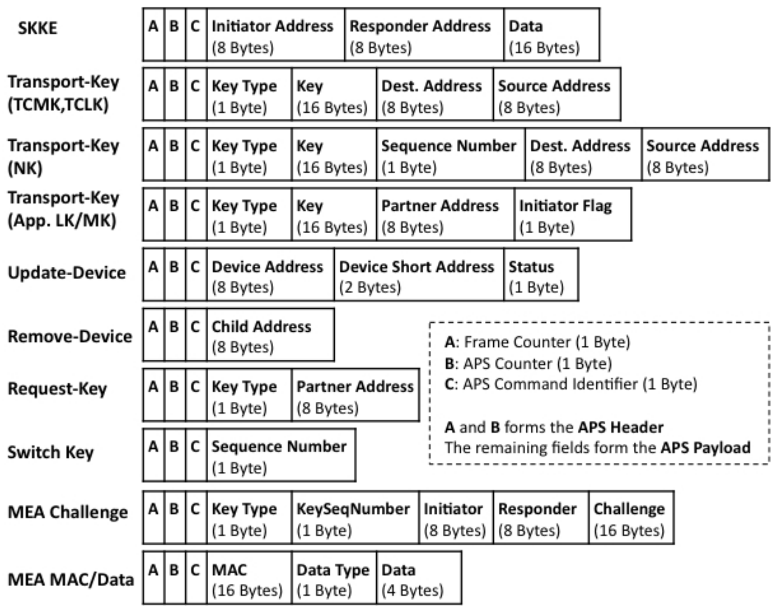
SKKE
A | B | C | Initiator Address (8 Bytes) | Responder Address (8 Bytes) | Data (16 Bytes)
Transport-Key (TCMK,TCLK)
A | B | C | Key Type (1 Byte) | Key (16 Bytes) | Dest. Address (8 Bytes) | Source Address (8 Bytes)
Transport-Key (NK)
A | B | C | Key Type (1 Byte) | Key (16 Bytes) | Sequence Number (1 Byte) | Dest. Address (8 Bytes) | Source Address (8 Bytes)
Transport-Key (App. LK/MK)
A | B | C | Key Type (1 Byte) | Key (16 Bytes) | Partner Address (8 Bytes) | Initiator Flag (1 Byte)
Update-Device
A | B | C | Device Address (8 Bytes) | Device Short Address (2 Bytes) | Status (1 Byte)
Remove-Device
A | B | C | Child Address (8 Bytes)
Request-Key
A | B | C | Key Type (1 Byte) | Partner Address (8 Bytes)
Switch Key
A | B | C | Sequence Number (1 Byte)
MEA Challenge
A | B | C | Key Type (1 Byte) | KeySeqNumber (1 Byte) | Initiator (8 Bytes) | Responder (8 Bytes) | Challenge (16 Bytes)
MEA MAC/Data
A | B | C | MAC (16 Bytes) | Data Type (1 Byte) | Data (4 Bytes)
A: Frame Counter (1 Byte)
B: APS Counter (1 Byte)
C: APS Command Identifier (1 Byte)
A and B forms the APS Header
The remaining fields form the APS Payload

Figure 3: APS Command Frames

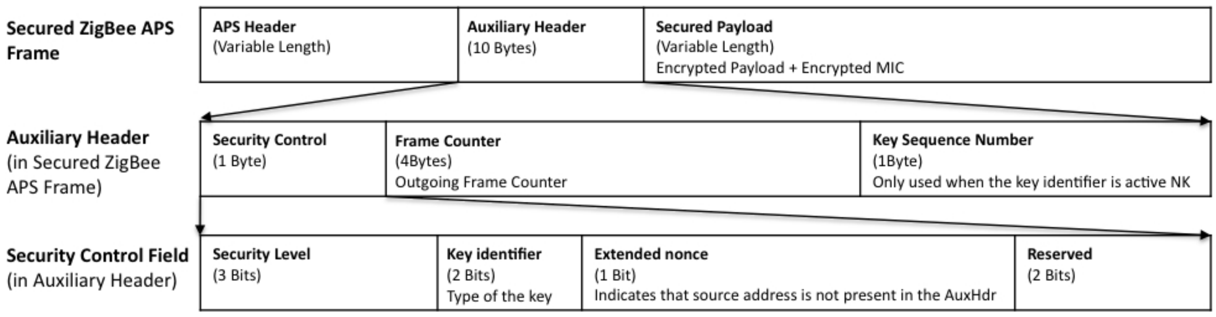
Secured ZigBee APS Frame
APS Header (Variable Length)
Auxiliary Header (10 Bytes)
Secured Payload (Variable Length) Encrypted Payload + Encrypted MIC
Auxiliary Header (in Secured ZigBee APS Frame)
Security Control (1 Byte)
Frame Counter (4Bytes) Outgoing Frame Counter
Key Sequence Number (1Byte) Only used when the key identifier is active NK
Security Control Field (in Auxiliary Header)
Security Level (3 Bits)
Key identifier (2 Bits) Type of the key
Extended nonce (1 Bit) Indicates that source address is not present in the AuxHdr
Reserved (2 Bits)

Figure 4: Secured ZigBee APS Frame

Address field of the *Request-Key* command is not present when the key type is NK or TCLK (note that TCLK means LK of the TC).

We have mentioned in Table 5 that the NKs are stored in the NWK Layer Information Base (**NIB**). Likewise, the MK/LK pairs and the relevant information is stored in the APS Layer Information Base (**AIB**).

The processing of the incoming and outgoing frames in APS layer is given in Table 6. Note that the security level is hidden the same way as in the NWK layer frame security.

4.7 Symmetric-Key Key Establishment Protocol

In the Symmetric-Key Key Establishment (SKKE) protocol, an initiator device U establishes a LK (or LK_{UV}) with a responder device V using a shared secret MK (or MK_{UV}). As we mentioned in Table 1, MK may either be pre-installed or transported from the TC. We present the SKKE protocol with computational details in Table 7. In the first two messages, the devices exchange their 16-byte challenges. In the last two messages, the devices exchange the data they have computed using the challenges and the device identities. Note that a *device identity* is the unique 64-bit device address, and ***kdf*** (defined in [2]) is the key derivation function that takes two parameters: the shared secret bit string, and the length of the keying data to be generated. After verifying that they received the correct values, they use another value as the LK that both of them can compute.

Table 6: APS Frame Security (||: Concatenation)

Processing Outgoing Frames	**Processing Incoming Frames**
1. Obtain the security material (from NIB or AIB), and the key identifier. If the key identifier is *active NK*, then either APS or NWK layer will apply security (not both of them). 2. Extract the outgoing frame counter (and the key sequence number if the key identifier is active NK) from the security material. 3. Obtain the security level from NIB. Set the AuxHdr (using all these parameters) as in Fig. 4. 4. Execute CCM* mode encryption and authentication, construct the CCM* nonce as "Source Address || Frame Counter || Security Control" (using the values from AuxHdr). 5. Construct the outgoing frame, depending on the encryption requirement, as in Fig. 4. 6. Increment and store the outgoing frame counter. 7. Set the *Security Level* subfield of AuxHdr as "000".	1. Determine the sequence number, key identifier and received frame counter value from the AuxHdr. 2. Obtain the appropriate security material from NIB or AIB depending on the key identifier. If *Received Frame Count* is less than *Frame Count* then FAIL. 3. Determine the security level from NIB and overwrite it to the *Security Level* subfield. 4. Execute CCM* mode decryption and authentication with the same parameters as in outgoing frames. 5. Unsecured APS frame will be constructed using the output of CCM*. 6. Set and store the Frame Count as "Received Frame Count + 1".

All the SKKE messages use the frame format as shown in Fig. 3. The *Data* field in a SKKE frame stores the value of either a challenge or a MAC (not the MAC layer, but the Message Authentication Code) tag.

4.8 Mutual Entity Authentication Protocol

In the Mutual Entity Authentication (**MEA**) protocol, an initiator device U and a responder device V mutually authenticate each other based on a secret key (NK). The devices authenticate each other by using random challenges with responses based on a NK. We present the MEA protocol with computational details in Table 8. In the first two messages, the devices exchange their challenges. Note that, **OFC** stands for the Outgoing Frame Counter of the device. In the last two messages, the devices exchange the data they have computed using their information and their frame counter values. They use the frame counter values they received and their previous knowledge to verify that they received the correct values, and thereby authenticate each other.

The MEA protocol uses two different frame formats for the challenge and the data, as shown in Fig. 3. The *MAC*, *Data Type*, and *Data* fields have the values field is the *MacTag*, *Frame Counter*, and the field is the *Frame Counter Value*, respectively.

4.9 CCM* Mode of Operation

CCM* is a generic encryption and authentication block cipher mode which is defined for use with only block ciphers having 128-bit block size. The AES-CCM* mode of operation is an extension of the AES-CCM mode that is used in IEEE 802.14.5-2003 [1] and provides capabilities for authentication, encryption, or both. Securing a NWK or an APS frame is actually based on AES-CCM* mode of operation in a particular security level.

Table 7: SKKE Protocol (H: Hash Function, MAC: HMAC Function, ||: Concatenation, 0x: Hexadecimal)

SKKE-1	U→V	QEU
SKKE-2	V→U	QEV

Computation in U		Z = MAC{U‖V‖QEU‖QEV}$_{MK}$
		KKeyData = kdf(Z, 256)
		MacKey = Leftmost 128bits of KKeyData
		= H(Z‖0x00000001)
		KeyData = Rightmost 128bits of KKeyData
		= H(Z‖0x00000002)
		MacData2 = 0x03‖U‖V‖QEU‖QEV
		MacTag2 = MAC(MacData2)$_{MacKey}$

SKKE-3	U→V	MacTag2
Verification in V		MacData2 = 0x03‖U‖V‖QEU‖QEV Verify MacTag2 using MacData2

Computation in V		Z = MAC{U‖V‖QEU‖QEV}$_{MK}$
		KKeyData = kdf(Z, 256)
		MacKey = Leftmost 128bits of KKeyData
		= H(Z‖0x00000001)
		KeyData = Rightmost 128bits of KKeyData
		= H(Z‖0x00000002)
		MacData1 = 0x02‖V‖U‖QEV‖QEU
		MacTag1 = MAC(MacData1)$_{MacKey}$

SKKE-4	V→U	MacTag1
Verification in U		MacData1 = 0x02‖V‖U‖QEV‖QEU Verify MacTag2 using MacData2
U and V use KeyData as the new LK LK = H(MAC{U‖V‖QEU‖QEV}$_{MK}$‖0x00000002)		

Table 8: MEA Protocol (MAC: HMAC Function, ||: Concatenation, 0x: Hexadecimal)

MEA-1	U→V	QEU
MEA-2	V→U	QEV

Computation in U		MacData2 = 0x03‖U‖V‖QEU‖QEV‖OFC$_U$
		MacTag2 = MAC{MacData2}$_{NK}$

MEA-3	U→V	MacTag2, OFC$_U$
Verification in V		MacData2 = 0x03‖U‖V‖QEU‖QEV‖OFC$_U$ Verify MacTag2 using MacData2

Computation in V		MacData1 = 0x02‖V‖U‖QEV‖QEU‖OFC$_V$
		MacTag1 = MAC(MacData1)$_{NK}$

MEA-4	V→U	MacTag1, OFC$_V$
Verification in U		MacData1 = 0x02‖V‖U‖QEV‖QEU‖OFC$_V$ Verify MacTag1 using MacData1

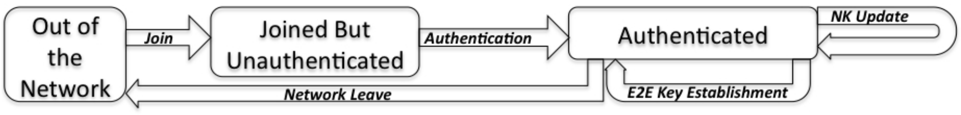

Figure 5: Security Procedures

5 The Security Procedures

In this section, we present a procedural description of how the security services in Section 4 are used. The security procedures consist of *joining a secured network*, *authentication*, *NK update*, *end-to-end application key establishment*, and *network leave*. We tried to visualize the security procedures in a state machine format as in Fig. 5. The arrows in the figure are the procedures, and the boxes are the states of a ZigBee device. Note that the procedure *End-to-End Key Establishment* is only valid in HS mode. Also note that it is also possible to change states without using procedures, as in the case of time outs, missed key update, etc.

5.1 Joining A Secured Network

This procedure is applied when a joiner device communicates with a router to join a secure network or when a device in a network has missed a key update and wants to receive the latest NK. The joiner device may begin the procedure by transmitting an unsecured beacon request frame. The joiner device receives beacons from routers and decides which network to join. After that the joiner device sends an association/rejoin request to the router. The router, knowing the joiner device's address and security capabilities (and in the case of *rejoin* whether the NK was used to secure the rejoin request command), will send an association/rejoin response command to the joiner device. If the joiner device receives a positive association/rejoin response command, the joiner is declared as *joined but unauthenticated* and the next phase shall be the Authentication routine. The protocol narration for the *Join* procedure is given in Appendix Subsection A.2.

5.2 Authentication

A ZigBee device that successfully finished the *joining a secured network* procedure is declared as *joined but unauthenticated* and shall start the *authentication* procedure. If the device is not a router, then after successful completion of the authentication it will be declared as *joined and authenticated.* If the device is a router, then after successful completion of the authentication followed by the initiation of routing operations it will be declared as *joined and authenticated.*

A new feature of the ZigBee-2007 is that, not only the newly joined device but also the neighbouring routers must be authenticated by using the MEA protocol.

Authentication depends on many different parameters such as the security mode (SS, HS), the presence of a router between TC and the device, the security level (None, ENC, MIC, ENC-MIC), the preconfiguration of the device (Not preconfigured, Preconfigured with NK/MK/LK), profiles, etc.

In SS mode, there are three main scenarios:

NPK: Device is not preconfigured with a valid key: TC sends active NK to the device in plaintext.

PK-NK: Device is preconfigured with active NK: TC sends fake NK (all zeros) to the device in plaintext.

PK-TCLK: Device is preconfigured with TCLK: TC sends NK to the device, encrypted by TCLK (LK of the TC).

In HS mode, there are two main scenarios:

NPK: Device is not preconfigured with a valid key: According to its configuration, TC sends either TCLK or TCMK in plaintext (*We separated these cases as NPK-TCLK and NPK-TCMK, see Appendix*). If TC sends TCMK, then it also makes key establishment to derive a TCLK.

PK-TCMK: Device is preconfigured with TCMK: TC establishes TCLK with the device, using key establishment service.

In case of HS, the joiner establishes entity authentication with the router to complete the authentication procedure. In the case of a separate router existing between the TC and the device, the communication between the router and the TC is secured in APS layer using active NK in SS, TCLK in HS. The protocol narrations for the *Authentication* procedure is given in Appendix Subsection A.3.

5.3 NK Update

We have mentioned that, maintaining a list of all the devices in the network is optional in SS but mandatory in HS for a TC. In SS mode, TC broadcasts new SNK to all the devices. In HS mode, TC sends new HSNK to each device using unicast communication.

If the device is capable of storing an alternate NK (*i.e. FFD, see Section 2*), then it will replace its alternate NK with the new key it received. If the device is not capable of storing an alternate NK (*i.e. RFD, see Section 2*), then it will replace its current NK and ignore any Switch-Key command. In any case, all incoming frame counters and the outgoing frame counter of the appropriate NK shall be set to 0. The sequence number of the new key will be the sequence number of the previous key incremented by 1 in mod 256. The protocol narration for *NK Update* procedure is given in Appendix Subsection A.4.

5.4 End-to-End Application Key Establishment

When End-to-End Application Security between two devices is required, the devices will run this procedure which also requires the collabration of the TC. Depending on the configuration of the TC, the devices can either receive an application LK (**appLK**) or an application MK (**appMK**) from TC. In the case of receiving an appMK, it is necessary to establish an appLK afterwards, using the SKKE protocol. The protocol narration for the *End-to-End Application Key Establishment* procedure is given in Appendix Subsection A.5.

5.5 Network Leave

The Network Leave procedure actually works in two different ways: Remove-Device and Device-Leave. In *Remove-Device*, the TC wants a ZigBee device to be removed from the network. After TC informs the router about the situation, the router sends a leave command to the relevant device. In *Device-Leave*, a device decides to leave the network and sends a leave command to the router. Then, the router informs the TC about the situation. In either case, TC shall delete the device from its list (which is optional in SS mode). If TC and the router share a LK, then the messages between the two will be secured with LK, otherwise with NK. The messages between router and the leaving device will be secured by NK. The protocol narration for the *Network Leave* procedure is given in Appendix Subsection A.6.

6 Related Work

6.1 Public Key Cryptography and Key Establishment

The ZigBee Specification uses symmetric cryptography, and therefore lacks (*1*) session key distribution without the intervention of a key distribution center (trust center), (*2*) digital signature and the non-repudiation property. Public-Key (**PK**) cryptography could alleviate these shortcomings but one needs to consider the trade-off between the performance and the security. The *computational complexity* of symmetric cryptography is much less, whereas *key management* (the process of selecting, distributing and storing keys) is complex and relatively insecure compared to the PK-cryptography.

[11] is one of the oldest papers with a proposal for using PK-cryptography in ZigBee. It proposes dynamic establishment of the security keys between communicating nodes by using Elliptic Curve Cryptography (**ECC**) [10]. Although there are many PK systems such as RSA, DSA, and Diffie-Hellman, ECC is preferred because it requires shorter keys for the same level of protection.

[22] presents a computation time comparison of Elliptic Curve Diffie-Hellman (ECDH), Elliptic Curve Menezes-Qu-Vanstone (ECMQV) and Symmetric Key Agreement (SKA) for IEEE 802.15.4 networks. In addition, considering frame transmission time and encryption processing they claim that ECMQV is not much slower than SKA. In other words, providing better key management and security with a little more device overhead. Note that the SKA [23] is actually the same as SKKE, and ECMQV is included in the brand new ZigBee profile: Smart Energy (*see below*).

[13] proposes scalable key establishment protocols and secure routing protocols with scalable authentication schemes. The idea here is to use different protocols in different settings, i.e. pure symmetric cryptography for low security, hybrid (symmetric and PK) for medium, and pure PK-cryptography for high security. This is achieved by installing a group key and an ECC based public/private key pair into each device before they join the network, thus enabling both symmetric and PK-cryptography.

An application of the *identity-based cryptography* [20] which reduces the number of the required security keys in ZigBee networks is proposed in [19]. Identity-based cryptography has advantages over PK-cryptography because it eliminates the (need for the) public key directory. The key point here is using the identity of each entity as a public key, which results in a simplified key distribution process. However, this scheme has some disadvantages, i.e. when the private key of an entity is somehow compromised, changing the key pair is very problematic since the public key is the identity of the entity. [19] proposes a scenario in which the identity to be used is a concatenation of the device information such as function, location, timestamp etc. It also requires a private key generator which is actually the TC.

Reducing the high costs of the PK-cryptography in especially the RFD (*see Section 2*) end nodes is another important issue. [24] proposes a hybrid authenticated key establishment scheme where an implementation of ECC [25] and symmetric cryptography are combined. This combination reduces the required resources for computation in the RFD node by using symmetric key based operations. Besides, the usage of ECC avoids typical key management problems in pure symmetric key based protocols. [24] also shows that it is possible to have even more reduction by using Modular Square Root (MSR) [26] techniques instead of ECC.

ZigBee Smart Energy Profile Specification In the end of May 2008, ZigBee published the ZigBee Smart Energy Profile Specification [8], which seems to be their response to the PK-cryptography proposals. The Key Establishment Cluster defined in [8] states that two basic types of key establishment can be implemented:

1) *Symmetric Key Key Establishment (SKKE)*: Establishing a key (LK) based on a shared secret (MK). If the shared secret (MK) is compromised then established key (LK) also may be compromised.

2) *Public Key Key Establishment (PKKE)*: Establishing a key (LK) based on shared static and ephemeral public keys. The case that uses certificates signed by a Certificate Authority to transport static public keys is called Certificate-Based Key Establishment (CBKE). The Key Establishment Cluster uses CBKE with ECMQV key agreement and ECQV certificate generation [9].

Table 9: CBKE Protocol (MAC: HMAC Function, ||: Concatenation, 0x: Hexadecimal)

CBKE-1	U→V	S_U, E_U
CBKE-2	V→U	S_V, E_V

Computation in U		Z = KeyBitGen(S_U, E_U, S_V, E_V)
		KKeyData = kdf(Z, 256)
		MacKey = Leftmost 128bits of KKeyData
		= H(Z||0x00000001)
		KeyData = Rightmost 128bits of KKeyData
		= H(Z||0x00000002)
		Mac_U=MAC(0x02,S_U,S_V,E_U,E_V)$_{MacKey}$

Computation in U		Z = KeyBitGen(S_U, E_U, S_V, E_V)
		KKeyData = kdf(Z, 256)
		MacKey = Leftmost 128bits of KKeyData
		= H(Z||0x00000001)
		KeyData = Rightmost 128bits of KKeyData
		= H(Z||0x00000002)
		Mac_V=MAC(0x03,S_V,S_U,E_V,E_U)$_{MacKey}$

CBKE-3	U→V	MAC_U
CBKE-4	V→U	MAC_V
Verification in U		Verify Mac_V
Verification in V		Verify Mac_U
Use KeyData as the new LK		

The CBKE protocol is given in Table 9. *S* represents the static data, which is a combination of device address and device static public key, whereas *E* represents the ephemeral data. In the first two messages, the devices exchange static and ephemeral data. In the last two messages, the devices exchange the data they have computed using the static and ephemeral data. Note that ***KeyBitGen*** is the function for generating the ECMQV key bit stream, and *kdf* is the same key derivation function as in SKKE. After verifying that they received the correct values, they use another value as the LK that both of them can compute.

6.2 Vulnerabilities and Advices

The IEEE 802.14.5 standard and its potential security vulnerabilities are important since it is the structure that ZigBee is built on.

[21] pinpoints the problems in IEEE 802.14.5 security and classifies them as: (*1*) Initialization Vector (**IV**) Management Problems, (*2*) Key Management Problems, and (*3*) Integrity Protection Problems. Since the vulnerabilities may be avoided in different levels, the paper [21] also classifies the advice for these levels. The advice for application designers include avoiding usage of any security suite that does not have integrity protection (e.g. ENC), and for implementing their own acknowledgement system. The advice for hardware designers include improving Access Control List (ACL) and nonce usage, and eliminating the implementations of the security suites without integrity. Finally, the advice for specification/standard writers include necessary support, requirements and explanations for the vulnerable points which are pinpointed as problems.

[29] presents an attack classification based on layers. Jamming, capturing, tampering, exhaustion, collision, and unfairness are the attacks that are possible in the PHY and MAC layers. Routing disruption and resource consumption are the attack types that are possible in the NWK layer. In [29] some of these attacks are modelled in the network simulation system NS-2 [30] and the results are presented. In addition, relying heavily on the TC is an important criticism

in the paper. Distributed or hierarchical key management schemes are recommended especially for large scale networks.

We believe that *key update* is an important issue in ZigBee networks. When a device is removed or leaving the network, it still knows the security key (i.e. NK). However, updating the NK after each device leave will be costly, whereas not updating the key will be insecure. Therefore, there is a trade-off between security and performance.

6.3 Surveys and Comparisons

[28] compares the security of two important Personal Area Network (PAN) standards, ZigBee and Bluetooth. It further claims that the secure key distribution and security key storage are the issues that may have problems in the future.

Z-Wave is a rival technology to ZigBee, which is a proprietary one developed by a Danish company. [12] presents a comparison of ZigBee and Z-Wave, and discusses security issues.

[14] compares ZigBee security with other security architectures in Wireless Sensor Networks (**WSN**) such as SPINS [16], LEAP [18], TINYSEC [17] and SM [15]. In this work, authentication characteristics are focused and it is concluded that the trend has moved from pre-deployed keying mechanisms to the symmetric key agreements, then to ECC based algorithms to perform authentication in WSNs.

[37] reviews five key management schemes: Eschenauer [38], Du [39], LEAP, SHELL [40] and PANJA [41]. This study evaluates this schemes considering the new trends in IEEE 802.15.4 and ZigBee. They conclude that future developments could incorporate the flexibility of LEAP with the adjustable robustness of Du or Eschenauer. In addition, for highest robustness Shell, and for highly scalability Panja can be preferred.

Claiming that being the first study that analyses SKKE, [27] presents a performance analysis of the key exchange mechanism. Their results indicate that even for small network size frequent key exchanges impose a serious performance burden on the data traffic.

As we mentioned before, TINYSEC is a WSN protocol that can be compared to ZigBee. [35] claims that, TINYSEC achieves low energy consumption by low security, ZigBee suffers from high energy consumption by high security, whereas their new proposal MiniSec obtains low energy consumption and high security. The idea behind this assertion is employing Offset Codebook Mode (OCB) [36] in encryption, sending only a few bits of the IV, and having different energy-optimized communication modes for unicast and broadcast communication.

Although the issue of contention resolution is beyond this survey, [34] is an interesting work since it presents the first application of *probabilistic model checking* to the IEEE 802.15.4 standard.

[33] is a recent extensive survey on WSNs; however the ZigBee section covers ZigBee-2006 only, like all other references. In addition, [42] is also a useful survey for WSNs, although it is not directly related to ZigBee.

6.4 Burglary Protection

Burglary protection is an interesting topic in pervasive computing. The attacks on security protocols are not the only threats in ZigBee networks, *theft* is also a serious security threat.

[31] proposes a security policy which is an extension of a more general model called "Resurrecting Duckling" [32]. The main idea in [31] is to chain the devices in a network or in friendly networks in such a way that a device will only function when it can see all of its friends. This idea is realized with protocols defined for device association and presence verification, and some real life scenarios are presented in the paper.

7 Conclusion

ZigBee is an emerging wireless network standard with low resource requirements. The latest version of the ZigBee Specification, ZigBee-2007, enhances the security of ZigBee. In this paper we presented a high level self-contained overview of the ZigBee-2007 security. We explained the key points of the specification such as *security in different layers*, presented the computations behind key establishment and authentication schemes such as *SKKE*, *CBKE*, *MEA*, and went deeper into the essential protocol narrations such as *Authentication*, *NK Update*, *etc.* We expect that ZigBee's wide applicability in many areas will necessitate much more work in ZigBee security. We plan to extend our work with the analysis and verification of the ZigBee security protocols.

References

[1] *IEEE Std. 802.15.4-2003, Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (WPANs)*, IEEE, 2003.

[2] *ANSI X9.63-2001, Public Key Cryptography for the Financial Services Industry - Key Agreement and Key Transport Using Elliptic Curve Cryptography*, American Bankers Association, November 2001.

[3] *FIPS Pub 197, Advanced Encryption Standard (AES), Federal Information Processing Standards Publication 197*, US Dpt. of Commerce/N.I.S.T, Springfield, Virginia, November 2001.

[4] *FIPS Pub 198, The Keyed-Hash Message Authentication Code (HMAC), Federal Information Processing Standards Publication 198*, US Dpt of Commerce/N.I.S.T., Springfield, Virginia, March 2002.

[5] *ZigBee Specification*, ZigBee Alliance, r17, January 2008.

[6] *ZigBee Specification*, ZigBee Alliance, r13, October 2006.

[7] *ZigBee Specification*, ZigBee Alliance, r06, December 2004.

[8] *ZigBee Smart Energy Profile Specification*, ZigBee Alliance, r14, May 2008.

[9] *Standards for Efficient Cryptography: SEC 1 (working draft) ver 1.7: Elliptic Curve Cryptography*, Certicom Research, November 2006.

[10] N. Koblitz, "Elliptic Curve Cryptosystems", *Mathematics of Computation*, 48, pp. 203-209, 1987.

[11] M. Blaser, "Industrial-strength Security for Zigbee: The Case for Public-key Cryptography", *Embedded Computing*, 2005.

[12] M. Knight, "How safe is Z-wave?", *Engineering and Technology*, 2006.

[13] Q. Huang, H. Kobayashi, B. Liu, D. Gu, and J. Zhang, "Energy/Security Scalable Mobile Cryptosystem", *Proc. IEEE Int'l Symp. on Personal, Indoor and Mobile Radio Comm.*, Vol. 3, pp. 2755-2759, September 2003.

[14] D. Boyle, and T, Newe, "Security Protocols for Use with Wireless Sensor Networks: A survey of security architectures", *Proc. Int'l Conf. on Wireless and Mobile Comm.*, 2007.

[15] J. Heo, C.S. Hong, “Efficient and Authenticated Key Agreement Mechanism in Low-Rate WPAN Environment”, *Proc. IEEE Int’l Symp. on Wireless Pervasive Computing*, January 2006.

[16] A. Perrig, R. Szewczyk, J.D. Tygar, V. Wen, and D. Culler, “SPINS: Security Protocols for Sensor Networks”, *Wireless Networks*, 8(5), pp. 521-534, 2002.

[17] C. Karlof, N. Sastry, and D. Wagner, “TinySec: A Link Layer Security Architecture for Wireless Sensor Networks”, *Proc. 2nd ACM Int’l Conf. on Embedded Networked Sensor Systems*, pp. 162-175, November2004.

[18] S. Zhu, S. Setia, and S. Jajodia, “LEAP: Efficient Security Mechanisms for Large-Scale Distributed Sensor Networks”, *Proc. ACM Conf. on Computer and Comm. Security*, pp. 62-72, October 2003.

[19] S.T. Nguyen, and C. Rong, “ZigBee Security Using Identity-Based Cryptography”, *Lecture Notes in Computer Science*, 4610, pp. 3-12, 2007.

[20] A. Shamir, “Identity-based Cryptosystems and Signature Schemes”, In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47-53. Springer, Heidelberg, 1985.

[21] N. Sastry, and D. Wagner, “Security Considerations for IEEE 802.15.4 Networks”, *Proc. 3rd ACM Workshop on Wireless security*, October 2004.

[22] J. Heo, and C.S. Hong, “Efficient and Authenticated Key Agreement Mechanism in Low-Rate WPAN Environment”, *Proc. 1st IEEE Int’l Symp. Wireless Pervasive Computing*, pp. 30-34, January 2006.

[23] *ISO/IEC 9798-2, Information Technology - Security Techniques - Entity Authentication Mechanism - Part 2, “Mechanisms Using Symmetric Encipherment Algorithm”*, International Standardization Organization, 1994.

[24] Q. Huang, J.I. Cukier, H. Kobayashi, B. Liu, and J. Zhang, “Fast Authenticated Key Establishment Protocols for Self-Organizing Sensor Networks”, *Proc. Int’l Conf. on Wireless Sensor Networks and Applications*, pp. 141-150, September 2003.

[25] *Standard for efficient cryptography, SEC 1: Elliptic Curve Cryptography. Version 1.0*, September 20, 2000. Certicom Corporation.

[26] D.R. Stinson, “The Rabin Cryptosystem, Cryptography: Theory and Practice”, Section 4.7, CRC Press (1995).

[27] M. Khan, F. Amini, J. Misic, and V. B. Misic, “The Cost of Security: Performance of ZigBee Key Exchange Mechanism in an 802.15.4 Beacon Enabled Cluster”, *Proc. 3rd IEEE Int’l Conf. on Mobile Ad Hoc and Sensor Systems*, pp. 876-881, October 2006.

[28] O. Hyncica, P. Kacz, P. Fiedler, Z. Bradac, P. Kucera, and R. Vrba, “On Security of PAN Wireless Systems”, *Lecture Notes in Computer Science*, pp. 178-185, 2006.

[29] J. Zheng, M.J. Lee, and M. Anshel, “Toward Secure Low Rate Wireless Personal Area Networks”, *IEEE Trans. on Mobile Computing*, Vol. 5, No. 10, pp. 1361-1373, October 2006.

[30] Network Simulator - NS2, USC Information Sciences Institute, 2005.

[31] M.Ø. Pedersen, J.I. Pagter, “The All-Or-Nothing Anti-Theft Policy: Theft Protection for Pervasive Computing”, *Proc. IEEE Computer Society AINA Workshops*, pp. 626-631, 2007.

[32] F. Stajano and R. Anderson. "The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks", *Proc. 7th Int'l Workshop Security Protocols*, 1999.

[33] P. Baronti, P. Pillai, V.W.C. Chook, S. Chessa, A. Gotta, and Y.F. Hu, "Wireless sensor networks: A survey on the state of the art and the 802.15.4 and ZigBee standards", *Computer Communications*, v.30 n.7, pp.1655-1695, May 2007.

[34] M. Fruth, "Probabilistic Model Checking of Contention Resolution in the IEEE 802.15.4 Low-Rate Wireless Personal Area Network Protocol", *Proc. 2nd Int'l Symp. on Leveraging Applications of Formal Methods, Verification and Validation*, pp. 290-297, November 2006.

[35] M. Luk, G. Mezzour, A. Perrig, and V. Gligor, "MiniSec: A Secure Sensor Network Comm. Architecture", *Proc. IEEE Int'l Conf. on Inf. Processing in Sensor Networks*, April 2007.

[36] P. Rogaway, M. Bellare, and J. Black, "OCB: A block-cipher mode of operation for efficient authenticated encryption", *ACM Trans. on Information and System Security*, vol. 6, no. 3, pp. 365-403, August 2003.

[37] J.C. Lee, K.H. Wong, J. Cao, H.C.B. Chan and V.C.M. Leung, "Key management issues in wireless sensor networks: current proposals and future developments", *IEEE Wireless Comm.*, April 2007.

[38] L. Eschenauer and V.D. Gligor, "A Key-Management Scheme for Distributed Sensor Networks", *Proc. 9th ACM Conf. Comp. and Comm. Sec.*, pp. 41-47, 2002.

[39] W. Du, J. Deng, Y.S. Han, P.K. Varshney, J. Katz, and A. Khalili, "A Pairwise Key Predistribution Scheme for Wireless Sensor Networks", *Proc. 10th ACM Conf. Comp. Comm. Sec.*, pp. 42-51, 2003.

[40] M. F. Younis, K. Ghumman, and M. Eltoweissy, "Location-Aware Combinatorial Key Management Scheme for Clustered Sensor Networks", *IEEE Trans. Parallel and Distrib. Sys.*, vol. 17, pp. 865-82, 2006.

[41] B. Panja, S.K. Madria, and B. Bhargava, "Energy and Communication Efficient Group Key Management Protocol for Hierarchical Sensor Networks", *Proc. IEEE Conf. Sensor Networks, Ubiquitous, and Trustworthy Comp.*, pp. 384-93, 2006.

[42] J.P. Walters, Z. Liang, W. Shi, and V. Chaudhary, "Wireless Sensor Network Security: A Survey", *Security in Distributed, Grid, Mobile, and Pervasive Computing*, Ed. Y. Xiao. Auerbach Publications, CRC Press, 2007.

[43] J. Zheng, and M.J.Lee, "Will IEEE 802.15.4 Make Ubiquitous Networking a Reality?- A Discussion on a Potential Low Power, Low Bit Rate Standard", *IEEE Comm. Magazine*, June 2004.

A Protocol Narrations

A.1 Guide For Reading Narrations

Protocol Narration Convention Extending the classical *Alice-Bob* protocol narration style, we used message (primitive) titles in boldface fonts, and message fields inside angle brackets. The protocol narration convention is given below:

Format:	A. B→C: D-E.F <M>
A:	Message Number (*1,2,3,...*)
B and C:	Sender and Receiver (Trust Center: *TC*, Router: *R*, All Routers: *Rn*, Device: *D*, All Devices: *Dn*, Initiator: *I*, Responder: *Res*)
D:	Message Layer and Entity (APS Management Entity: *APSME*, NWK Management Entity: *NLME*)
E:	Command Type (*SWITCH-KEY, ESTABLISH-KEY, etc.*)
F:	Primitive Type (*request, response*)
M:	Message structure, including field names (*Accept, SrcAddress, etc.*) and possible values (*True, Raddr, etc.*)

Example:
3. D→TC: APSME-ESTABLISH-KEY.response
<InitiatorAddress(=TCaddr), Accept(=TRUE)>

This is the third message of the protocol, sent by a device to the TC, the message is an APS layer management entity message, the command type is *establish key*, the primitive type is *response*. The message has two fields, the first field *InitiatorAddress* has the value *TCaddr*, and the second field *Accept* has the value *TRUE*.

Cryptographic Notation We used three different cryptographic operations: *Encryption*, *HMAC*, and *Hash*. The notation is given below:

{Message}:Key : *Message* is encrypted with *Key*
MAC{Message}:MacKey : HMAC of *Message*, computed using *MacKey*
H(Message) : Hash of *Message*

A.2 Join

The Join procedure depends on the MAC layer command frames that are defined in the IEEE 802.15.4 standard, and being the only exception in the appendix, we omitted the message details.

1. D → Rn: Beacon request command (unsecured)
2. Rn → D: Beacon (unsecured)
3. D → R: Association request command
4. R → D: Association response command

A.3 Authentication

Protocol Naming Convention In order to have a shorter and a systematic way of naming protocols, we created the protocol naming convention below for the Authentication protocols:

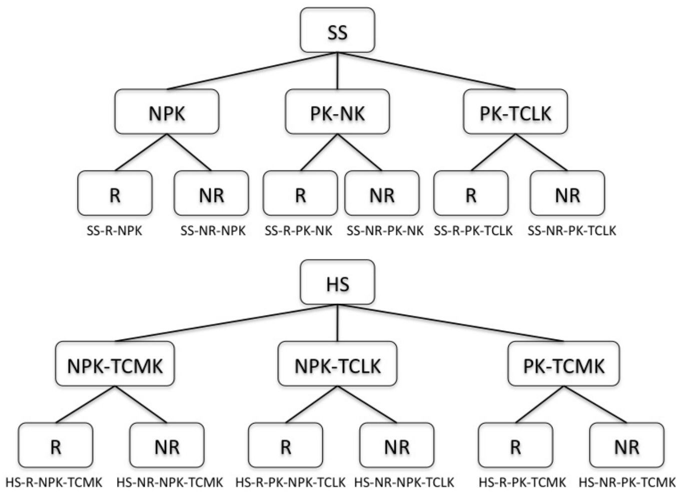

Figure 6: Authentication Protocols

	Format: G-H-I[-J]
G:	Security Mode (Standard Security: *SS*, High Security: *HS*)
H:	Presence of a separate router (A separate router exists between TC and D: *R*, TC serves as the router: *NR*)
I:	Key knowledge of the device (Preconfigured with a key: *PK*, Not preconfigured: *NPK*)
J:	Preconfigured or "to be configured" key type (None, Network Key: *NK*, Trust Center Link Key: *TCLK*, Trust Center Master Key: *TCMK*)

Example: SS-R-PK-TCLK means:
Standard Security Mode,
TC is not the router,
Device is preconfigured with TCLK

The protocols in the Authentication part can be summarized as in Fig. 6. In order to save space, we present the narrations in the case that TC is the router.

SS-NR-NPK

1. TC→D: APSME-TRANSPORT-KEY.request (send active SNK)
<DestAddress(=Daddr), Keytype(=SNKtype), KeySeqNumber(=activeKeySeqNo), NetworkKey(=activeSNK), UseParent(=FALSE), ParentAddress(=No need to set)>

SS-NR-PK-NK

1. TC→D: APSME-TRANSPORT-KEY.request (send active SNK)
<DestAddress(=Daddr), Keytype(=SNKtype), KeySeqNumber(=Zero), NetworkKey(=Zero), UseParent(=FALSE), ParentAddress(=No need to set)>

SS-NR-PK-TCLK

1. TC→D: APSME-TRANSPORT-KEY.request (send active SNK)
<{DestAddress(=Daddr), Keytype(=SNKtype), KeySeqNumber(=activeKeySeqNo), NetworkKey(=activeSNK), UseParent(=FALSE), ParentAddress(=No need to set)}:TCLK>

HS-NR-NPK-TCMK

1. TC→D: APSME-TRANSPORT-KEY.request (send TCMK)
<DestAddress(=Daddr), Keytype(=TCMKtype), ParentAddress(=TCaddr), Key(=TCMK)>
2. TC→D: APSME-ESTABLISH-KEY.request (establish TCLK)
<ResponderAddress(=Daddr), UseParent(=FALSE), ResponderParentAddress(=No need to set), KeyEstablishmentMethod(=SKKE)>
3. D→TC: APSME-ESTABLISH-KEY.response (establish TCLK)
<InitiatorAddress(=TCaddr), Accept(=TRUE)>
4. TC↔D: SKKE (*See Subsection 4.7 for details*)
5. TC→D: APSME-TRANSPORT-KEY.request (send active HNK)

<DestAddress(=Daddr), Keytype(=HNKtype), KeySeqNumber(=activeKeySeqNo), NetworkKey(=activeHNK), UseParent(=FALSE), ParentAddress(=No need to set)>
6. D→TC: APSME-AUTHENTICATION.request (entity auth.)
<PartnerAddress(=TCaddr), Action(=INITIATE), RandomChallenge(=Drandom)>
7. TC→D: APSME-AUTHENTICATION.request (entity auth.)
<PartnerAddress(=Daddr), Action(=RESPOND_ACCEPT), RandomChallenge(=TCrandom)>
8. D↔TC: MEA (*See Subsection 4.8 for details*)

HS-NR-NPK-TCLK

1. TC→D: APSME-TRANSPORT-KEY.request (send TCLK)
<DestAddress(=Daddr), Keytype(=TCLKtype), ParentAddress(=TCaddr), Key(=TCLK)>
2. TC→D: APSME-TRANSPORT-KEY.request (send active HNK)
<DestAddress(=Daddr), Keytype(=HNKtype), KeySeqNumber(=activeKeySeqNo), NetworkKey(=activeHNK), UseParent(=FALSE), ParentAddress(=No need to set)>
3. D→TC: APSME-AUTHENTICATION.request (entity auth.)
<PartnerAddress(=TCaddr), Action(=INITIATE), RandomChallenge(=Drandom)>
4. TC→D: APSME-AUTHENTICATION.request (entity auth.)
<PartnerAddress(=Daddr), Action(=RESPOND_ACCEPT), RandomChallenge(=TCrandom)>
5. D↔TC: MEA (*See Subsection 4.8 for details*)

HS-NR-PK-TCMK

1. TC→D: APSME-ESTABLISH-KEY.request (establish TCLK)
<ResponderAddress(=Daddr), UseParent(=FALSE), ResponderParentAddress(=No need to set), KeyEstablishmentMethod(=SKKE)>
2. D→TC: APSME-ESTABLISH-KEY.response (establish TCLK)
<InitiatorAddress(=TCaddr), Accept(=TRUE)>
3. TC↔D: SKKE (*See Subsection 4.7 for details*)
4. TC→D: APSME-TRANSPORT-KEY.request (send active HNK)
<DestAddress(=Daddr), Keytype(=HNKtype), KeySeqNumber(=activeKeySeqNo), NetworkKey(=activeHNK), UseParent(=FALSE), ParentAddress(=No need to set)>
5. D→TC: APSME-AUTHENTICATION.request (entity auth.)
<PartnerAddress(=TCaddr), Action(=INITIATE), RandomChallenge(=Drandom)>
6. TC→D: APSME-AUTHENTICATION.request (entity auth.)
<PartnerAddress(=Daddr), Action(=RESPOND_ACCEPT), RandomChallenge(=TCrandom)>
7. D↔TC: MEA (*See Subsection 4.8 for details*)

A.4 NK Update

We have two different NK Update protocols for SS and HS modes.

SS

1. TC→Dn: APSME-TRANSPORT-KEY.request
<DestAddress(=Broadcastaddr), KeyType(=SNKtype), KeySeqNumber(=newKeySeqNumber), NetworkKey(=newSNK), UseParent(=FALSE), ParentAddress(=No need to set)>
2. TC→Dn: APSME-SWITCH-KEY.request
<DestAddress(=Broadcastaddr), KeySeqNumber(=newKeySeqNumber)>

HS

1. TC→D: APSME-TRANSPORT-KEY.request
<DestAddress(=Daddr), KeyType(=HSNKtype), KeySeqNumber(=newKeySeqNumber), NetworkKey(=newHSNK), UseParent(=FALSE), ParentAddress(=No need to set)>
2. TC→D: APSME-SWITCH-KEY.request
<DestAddress(=Daddr), KeySeqNumber(=newKeySeqNumber)>

A.5 End-to-End Application Key Establishment

This procedure depends on the configuration of the TC. Therefore, we have two different protocols.

TC is configured to send AppLK

1. I→TC: APSME-REQUEST-KEY.request
<DestAddress(=Resaddr), KeyType(=AppKeytype), PartnerAddress(=Resaddr)>
2. TC→I: APSME-TRANSPORT-KEY.request
<DestAddress(=Iaddr), KeyType(=AppLKtype), PartnerAddress(=Resaddr), Initiator(=TRUE), Key(=newAppLK)>
3. TC→Res: APSME-TRANSPORT-KEY.request
<DestAddress(=Resaddr), KeyType(=AppLKtype), PartnerAddress(=Iaddr), Initiator(=FALSE), Key(=newAppLK)>

TC is configured to send AppMK

1. I→TC: APSME-REQUEST-KEY.request
<DestAddress(=Resaddr), KeyType(=AppKeytype), PartnerAddress(=Resaddr)>
2. TC→I: APSME-TRANSPORT-KEY.request
<DestAddress(=Iaddr), KeyType(=AppMKtype), PartnerAddress(=Resaddr), Initiator(=TRUE), Key(=newAppMK)>
3. TC→Res: APSME-TRANSPORT-KEY.request
<DestAddress(=Resaddr), KeyType(=AppMKtype), PartnerAddress(=Iaddr), Initiator(=FALSE), Key(=newAppMK)>

4. I→Res: APSME-ESTABLISH-KEY.request (establish TCLK)
<ResponderAddress(=Resaddr), UseParent(=FALSE), ResponderParentAddress(=No need to set), KeyEstablishmentMethod(=SKKE)>
5. Res→I: APSME-ESTABLISH-KEY.response (establish TCLK)
<InitiatorAddress(=Iaddr), Accept(=TRUE)>
6. I↔Res: SKKE

A.6 Network Leave

This procedure has different protocols, depending on the initiator of the procedure.

Remove-Device

1. TC→R: APSME-REMOVE-DEVICE.request
<ParentAddress(=Raddr), ChildAddress(=Daddr)>
2. R→D: NLME-LEAVE.request
<DeviceAddress(=Daddr), RemoveChildren(=TRUE/FALSE), Rejoin(=FALSE)>

Device-Leave

1. D→R: NLME-LEAVE.request
<DeviceAddress(=Daddr), RemoveChildren(=TRUE/FALSE), Rejoin(=FALSE)>
2. R→TC: APSME-UPDATE-DEVICE.request
<SrcAddress(=Raddr), DeviceAddress(=Daddr), Status(=DeviceLeft), DeviceShortAddress(=Dshortaddr)>