M. Meier, D. Reinhardt, S. Wendzel (Hrsg.): Sicherheit 2016,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2016 229

On the Security of the ZigBee Light Link Touchlink Commissioning Procedure

Frederik Armknecht[1], Zinaida Benenson[2], Philipp Morgner[3], Christian Müller[4]

Abstract: Specifications of security mechanisms often lack explicit descriptions of the envisioned security goals and the underlying assumptions. This makes it difficult for developers and customers to understand the level of security provided by the systems. Moreover, this omission has repeatedly resulted in practical attacks that violate the implicit security assumptions of the specifications. In this work, we illustrate this effect on the example of the ZigBee Light Link (ZLL) profile, currently one of the most popular standards for smart lighting in domestic environments. We first provide a concise description of ZLL commissioning procedure for initiating and extending a network of smart bulbs, extracted directly from the specification. We then discuss how the commissioning protocol can be transformed into a formal security model, but also highlight where this is subject to interpretations because of the unclear implicit security assumptions. The proposed security model is flexible, i.e., it can be extended to capture further security requirements or attacker classes, and hence provides a solid foundation for rigorous security analyses of ZLL and other ZigBee profiles.

Keywords: Smart Home, ZigBee Light Link, Security Model.

1 Introduction

Smart buildings comprise interconnected household, office and personal devices, and systems designated to support and enhance the everyday life of inhabitants. Applications include alarm systems, energy management, building automation, or support systems for elderly and disabled people [BLM+11, HH12, MH12]. These systems can have a substantial impact on safety and privacy of the users, as they collect and process sensor data about the residents, and also automatically trigger different actuators.

Therefore, security mechanisms should ensure that only authorized entities control and manage smart objects, as well as that only authorized smart objects communicate with each other and access the gathered information [UJS13, ARA12]. Despite the enormous relevance of this topic, a comprehensive framework for a sound analysis of security features in smart building networks does not exist. Even worse, for most existing mechanisms in place, a rigorous description of the underlying security assumptions is missing.

This is not of academic interest only. On the contrary, security holes discovered in practice are often difficult to fix afterwards and mostly result in high costs, as the security disasters of GSM [BBK03], WEP [BHL06] and Bluetooth [JW01, HHPT13] have shown. Quite often in these cases, the discovered vulnerabilities were either apparent in the specifications from the very beginning, or resulted from the inaccurate or missing assumptions

[1] University of Mannheim, armknecht@uni-mannheim.de
[2] University of Erlangen-Nuremberg, zinaida.benenson@fau.de
[3] University of Erlangen-Nuremberg, philipp.morgner@fau.de
[4] University of Mannheim, christian.mueller@uni-mannheim.de

about what to protect (security goals) and against whom to protect (attacker motives and capabilities).

Security mechanisms of ZigBee Pro [Zig12b], one of the most popular standards for low-power and low-cost networks in smart buildings, similarly have turned out to be insufficient over and over again [Wri09, VHP+13, Zil15]. This creates the impression of smart buildings being insecure in general, which may unsettle potential customers. Therefore, it is necessary to provide a concise and well-founded assessment of ZigBee security.

In this work, we make a first step towards this goal and introduce a formal security model for the touchlink commissioning procedure described in the ZigBee Light Link (ZLL) [Zig12a] profile specification. We demonstrate how to express ZLL security mechanisms by means of a formal security model, at the same time discussing the shortcomings of the corresponding specification where the readers have to rely on personal interpretations. The introduced security model can be easily extended for analysis of further security goals and mechanisms in smart buildings and similar systems.

This paper is organized as follows: In Section 2, we outline related work. We provide background on ZigBee and ZLL in Section 3, and describe ZLL touchlink commissioning in Section 4. In Section 5, we present the formal security model for ZLL and discuss the security of ZLL within this model. We conclude this paper in Section 6.

2 Related Work

To the best of our knowledge, neither a formal concept of a security model nor a sound security framework for the ZLL standard or parts thereof has been published yet. However, several works analyze security of ZigBee products and outline security flaws. Thus, Wright [Wri09] introduced the KillerBee framework as a penetration testing tool for ZigBee networks. He exposed security flaws in the key provisioning protocol of the ZigBee Pro specification by showing that the network key is sent unprotected on the wireless channel. Vidgren et. al. [VHP+13] demonstrated the exploitation of the key exchange process in the Standard Security mode of ZigBee. As a consequence to this attack, they conclude that the Standard Security mode should be removed from the ZigBee specification.

Zillner and Strobl [Zil15, ZS15] analyzed security measures of the ZigBee Pro Home Automation (ZHA) and the ZLL profiles using their SecBee framework and found numerous protocol and implementation flaws. For example, they exploit the ZLL touchlink commissioning process to seize control over ZLL devices and show that unauthorized disclosure of the ZLL master key in March 2015 compromised the security of the entire ZLL ecosystem.

3 ZigBee Light Link

In this section, we introduce ZLL and provide an overview of the technical background and security mechanisms.

3.1 ZigBee Overview

ZigBee is a standard that connects embedded technologies in Wireless Personal Area Networks (WPANs). The specification is maintained by the ZigBee Alliance, a non-profit organization that comprises over 250 members. It defines the network, security, and software layers and supervises the conformance and interoperability of ZigBee-certified products.

The first ratified ZigBee specification was published in 2004, referred to as *ZigBee 2004*. Two years later, in the *ZigBee 2006* specification, cluster libraries and security features were introduced. A new security model including a so-called *trust center* was introduced in the *ZigBee 2007* specification, and in the same year, the initial *ZigBee Pro* specification containing additional software features and enhanced security mechanisms was released, which has been subject to several revisions, the most recent in 2012. The newest *ZigBee 3.0* standard is expected to be published in 2016 and is not publicly available yet. In this paper, we thus consider the current *ZigBee Pro* specification [Zig12b].

ZigBee Pro includes different *profiles*, which comprise customized sets of features and protocols for a specific application area, such as Home Automation (ZHA) [Zig13b] and Smart Energy (SE) [Zig13a] profiles. Here we focus on the ZLL profile [Zig12a] that is implemented, for example, by Philips Hue and Osram Lightify are smart lighting systems.

3.2 ZLL Technical Background and Security Mechanisms

The ZLL profile describes three different node types: ZigBee Coordinator (ZC), ZigBee Router (ZR), and ZigBee End Device (ZED). In a ZigBee network, there exists exactly one designated ZC, which acts as the router initially forming the network. In contrast, there is no fixed number of ZRs and ZEDs. ZRs act as full-function devices that route data frames to other nodes, whereas ZEDs do not have router functionality and need a ZC or ZR parent device for network participation. Most ZigBee devices form mesh networks, but also other network topologies are feasible, such as star and tree.

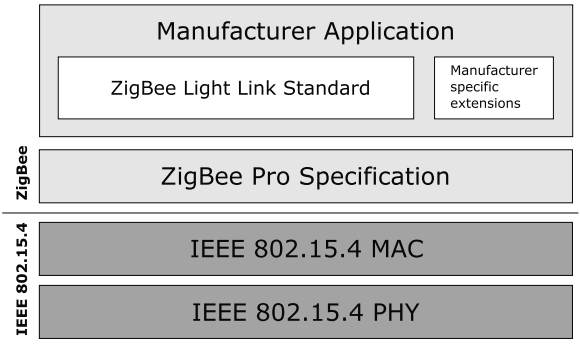

Fig. 1: Overview of the stack architecture of ZLL devices.

As illustrated in Figure 1, the ZLL stack consists of four layers: physical (PHY), medium access control (MAC), network (NWK), and application (APL). The two lower layers,

PHY and MAC, are defined in the IEEE 802.15.4-2003 specification [IEE03]. This specification is also incorporated into other WPAN standards, most prominently Thread Group [Sil15] and WirelessHART [Int10]. The PHY layer is the lowest layer and defines the physical interface. ZigBee products use radio-frequency communication in the 2.4 GHz ISM band. The MAC layer provides functionalities to transmit data frames in a reliable manner by managing access to the radio channel via CSMA/CA mechanisms, sending beacon frames, acknowledgement frames, and applying synchronization mechanisms.

The network layer is specified in the ZigBee Pro standard and manages network topology, routing, parts of the security services, and acts as a message broker. The application layer contains the manufacturer applications. This layer also includes the application profiles, which for ZLL products is the ZLL standard. Since the touchlink commissioning procedure is defined in the ZLL standard, this procedure is also defined in this layer.

Security measures are only applied to the two upper layers, NWK and APL. The two lower layers also support encryption and integrity mechanisms, however, these are not supported in ZigBee products. ZLL devices use the AES-CCM* authenticated encryption scheme with a 128-bit key. ZLL devices hold three types of cryptographic keys: link keys, the network key, and the ZLL master key. A link key is only shared between two devices in a network and used for direct unicast communication. Consequently, a ZLL device may hold multiple link keys. The network key is used for broadcast communication and shared between all devices of a network. The ZLL master key is shared between all devices that support the ZLL profile and is distributed to manufacturers by the ZigBee Alliance and protected by a non-disclosure agreement (NDA). Nonetheless, the ZLL master key was disclosed in March 2015 as stated by Zillner [Zil15].

4 ZLL Touchlink Commissioning

For the initial setup, a device needs to be associated to the customer's WPAN. The procedure of connecting a new device to a network for the first time is called commissioning. ZLL enables two commissioning modes: touchlink commissioning and classical commissioning. Since we focus on the touchlink commissioning, the classical commissioning is not further considered in this paper.

4.1 Protocol Description

We explain the commissioning protocol for joining a new device to an existing network in an abstract manner (see Figure 2), omitting the details that do not pose any additional information concerning the scope of this paper.

The touchlink commissioning protocol is executed between two entities, an initiator and an end device. The initiator is usually either a controller or a bridge, i. e., a ZLL device characterized by a physical button which is pushed to start the commissioning procedure. The end device represents the ZLL device to be added to the network. The initiator starts

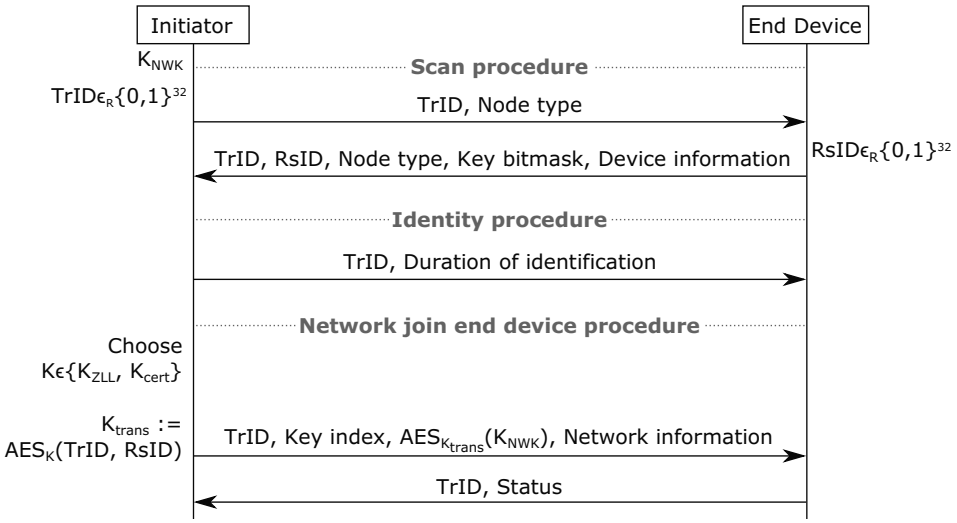

Fig. 2: Touchlink commissioning protocol in the ZLL profile using either the ZLL master key K_{ZLL} or the certification key K_{cert} to encrypt the network key K_{NWK}. `TrID` and `RsID` denote the random 32-bit transaction and response identifier, respectively. The notation $x \in_R \{0,1\}^l$ denotes a randomly chosen bit string x of length l.

the device scan procedure by sending *scan requests* on different channels as defined in the ZLL specification. These scan requests include the randomly generated transaction identifier `TrID` and the node type of the initiator (cf. Section 3.2). The end device replies with a *scan response* containing `TrID`, a random response identifier `RsID`, its own node type, the list of all supported keys referred to as `key bitmask`, and some further information.

The device scan may yield multiple potential devices from which the user can select at least one for the next steps. The user has the option to send an *identity request* to a device, upon which the target device performs a predefined identification action, e.g., a light bulb will flash a few times. An *identity request* contains the corresponding `TrID` as well as the duration of the identification action.

To join a new end device to a network, the initiator encrypts the current network key using as outlined in Section 4.2, builds a *network join end device request* containing this encrypted network key, `TrID`, the key index as well as network information, and then sends this request to the selected end device. On receiving the message, the joining end device replies with a *network join end device response* which includes a status flag set to indicate success.

4.2 Network Key Encryption

The *scan response* contains a key bitmask, which indicates the keys available to the end device. These can be the *ZLL master key*, the *Certification key*, or the *Development key*.

The initiator compares the received key bitmask to its own keying material and chooses the appropriate key. As the Development key uses a slightly different encryption algorithms and is only allowed during the development phase, here we concentrate on the algorithm that is utilized with the ZLL master key and the Certification key.

The ZLL master key, denoted K_{ZLL}, is distributed to certified manufacturers of ZLL products and bound with a non-disclosure agreement (NDA). The Certification key is used to evaluate the security mechanisms during the certification phase of a ZLL product and is defined in the ZLL specification.

To encrypt the randomly chosen network key, the initiator expands the transaction identifier and the response identifier to a 128-bit number which represents the plaintext input to the AES-ECB encryption, while either the ZLL master key or the Certification key is used as encryption key. The resulting ciphertext output is called Transport key. In the next step, the actual network key, denoted K_{NWK} is encrypted under the Transport key, again using the AES-ECB encryption mode.

We note that AES-ECB does not support any integrity protection, which makes the following Denial of Service attack theoretically possible: The attacker disturbs the part of the message containing encrypted K_{NWK} in such a way that some bits in the AES-ECB encryption are flipped, and everything else remains unaffected. In this case, the End Device will accept the message and decrypt the supposed K_{NWK} into garbage without noticing.

5 Security Model

5.1 The Contradictory ZLL Security Assumptions

As discussed in Section 1, a concise security model is indispensable for a meaningful security analysis. Such a security model has to specify at least the following three components:

1. *System model* describes what entities and possible actions are considered, such as types of devices and communication between them.
2. *Security goals* specify what should be protected, e.g., confidentiality and integrity of data.
3. *Attacker model* establishes against which attack types security is envisioned. For example, it describes the capabilities of the attackers, such as eavesdropping or physical device compromise.

Indeed, without knowing the attackers' capabilities and the security goals of a system, it is impossible to claim that a system is secure or insecure, because one does not know what should be protected (goals) and against which threats (attackers).

The ZLL system model is rigorously described in the 120 pages of the corresponding specification [Zig12a]. Also the functionality of the ZLL Touchlink Commissioning protocol

and other security protocols is precisely specified, including the format and the order of all protocols messages. Unfortunately, this document does not say anything about the attacker model and the security goals, apart form stating that *"The ZLL master key is a secret shared by all certified ZLL devices. It will be distributed only to certified manufacturers and is bound with a safekeeping contract"* [Zig12a, page 104].

The reason behind this approach is quite understandable: whereas message formats and order of protocol messages are crucial for the operation of the network (they are *functional* requirements), attacker model and security goals are *non-functional* requirements: if the attacker is not present, the ZLL network is going to work as long as all specified mechanisms are implemented correctly.

Turning to the ZigBee specification [Zig12b], we find the "Security Assumptions" section on pages 426-427. This is the only place in this 600 pages long document where the security model is considered. The description of security assumption is quite informal. Especially relevant for our work is the following statement: *"[...] due to the low-cost nature of ad hoc network devices, one cannot generally assume the availability of tamper-resistant hardware. Hence, physical access to a device may yield access to secret keying material and other privileged information, as well as access to the security software and hardware."*

Thus, when trying to extract the attacker model and the security goals from the specifications, we discover contradictory statements that make the derivation of a precise formal security modeling impossible. On the one hand, ZigBee specification envisions the attacker that is able to gain access to the internal secrets of the devices by physical tampering. On the other hand, ZLL specification builds the security of the entire ZLL ecosystem on the single master key that is the same for all ZLL devices.

In the following, we nevertheless attempt to model ZLL security more precisely and show possible pitfalls. We start with presenting a system model, then discuss (some) security goals, and finally address the attacker model.

5.2 System Model

Within the formal system model, one has to specify the involved entities and the actions they can take. Here, we exclude the attacker as this is part of the attacker model presented in Section 5.4. On a high level, the system comprises devices that can communicate with each other and also jointly execute protocols.

Nodes. The entities of the system are given by a set of all ZLL nodes $\mathcal{N}$.[5] This set is not fixed but may vary over time, i.e., nodes may be added or removed. Among the set of nodes, there is a particular class of nodes called the *coordinator nodes*.

In cryptographic models, one usually assumes that the set of all nodes is fixed. In contrast, we assume a process Create that has no input and outputs a new node. That is, we express

[5] We use the term *node* instead of ZigBee device to be consistent with other models (cf. [GRT12, VJU+12]).

by $\mathsf{n} \leftarrow \mathsf{Create}$ the event that Create has been initiated and its output is a new node n. In this case, the set of all nodes is extended by the new node n, that is, $\mathscr{N}$ is updated to $\mathscr{N} \cup \{\mathsf{n}\}$. Nodes that have been created by Create (and only those) are called *genuine* nodes and we refer by $\mathscr{N}_{\text{gen}}$ to the set of all genuine nodes.

We do not further specify Create, as its real-world meaning can depend on the concrete use case. For example, Create may refer to the process of getting (buying) nodes from a certain vendor only, or may comprise nodes that have been approved by a certain authority.

Protocols. Nodes can communicate with each other and also jointly run protocols. To express the situation that two nodes $\mathsf{n1}$ and $\mathsf{n2}$ jointly run a protocol Π, we adopt the following common notation from cryptography:

$$\Pi : [\mathsf{n1} : x_1, \mathsf{n2} : x_2] \rightarrow [\mathsf{n1} : y_1, \mathsf{n2} : y_2] \tag{1}$$

This expresses that both nodes run the protocol Π where x_1 and x_2 denote the inputs of $\mathsf{n1}$ and $\mathsf{n2}$, respectively, and y_1 and y_2 the respective outcomes of the protocol. Here, "input" and "outcome" refer to locally known values only. For example, x_1 is the value used by $\mathsf{n1}$ to run the protocol but does not necessarily mean that it is sent to $\mathsf{n2}$. We use $\perp$ to indicate an uninitialized parameter as well as an intended lack of input or output, e. g., $y_2 = \perp$ would refer to the case that $\mathsf{n2}$ has no internal output of the protocol.

In our system model, we cover two protocols: Π_{join} and Π_{cmd}. In a nutshell, the first protocol coordinates the inclusion of nodes into the network, while the second allows nodes within a network to exchange commands. Note that a full system model would require to specify the *correctness* of protocols. We omit this here due to space limitations. For the same reason, we discuss the protocols here on an abstract level only and shortly address later how these are realized in ZLL. The Π_{join} protocol is executed between a node n and a coordinator c and has the following specification:

$$\Pi_{\text{join}} : \left[\mathsf{c} : x, \mathsf{n} : x'\right] \rightarrow [\mathsf{c} : \delta, \mathsf{n} : y] \tag{2}$$

where x, x' are the inputs of c and n, y is the local outcome of n and $\delta \in \{\text{accept}, \text{reject}\}$ denotes whether the protocol run is successful or not.Similarly, the Π_{cmd} protocol is executed between two nodes $\mathsf{n1}, \mathsf{n2}$:

$$\Pi_{\text{cmd}} : \left[\mathsf{n1} : x, \mathsf{n2} : x'\right] \rightarrow [\mathsf{n1} : \perp, \mathsf{n2} : \delta] \tag{3}$$

Again, x, x' denote the inputs of the nodes and $\delta \in \{\text{accept}, \text{reject}\}$ denotes whether the protocol run is successful or not.

Networks. As already mentioned, nodes can be grouped into *networks*. Each network has exactly one coordinator node c. We use this fact to formally define a network:

Definition 1 (Network)**.** *Consider the set of all nodes $\mathscr{N}$. A subset $\mathscr{N}_{\mathsf{c}} \subseteq \mathscr{N}$ is called a* network *with respect to some coordinator node c if for any $\mathsf{n1} \in \mathscr{N}_{\mathsf{c}}$ and $\mathsf{n2} \in \mathscr{N}_{\mathsf{c}}$ the following holds: If $\mathsf{n1}$ runs the Π_{cmd} protocol with $\mathsf{n2}$, then $\mathsf{n2}$ eventually accepts.*

5.3 Security Goals

In the following, we specify the security goals of ZLL. They are not given in the specification and hence are based on our interpretation. According to the specification, each node is equipped with a master key K_{ZLL} which is used in the Π_{join} protocol to transport a network key K_{NWK} from the coordinator to the node. The knowledge of the network key K_{NWK} is mandatory to make the other node accept in the Π_{cmd} protocol. Based on these observations, our interpretation of the security goals are:

1. Only nodes that know the master key K_{ZLL} should be allowed to join a network.
2. Only nodes that know the network key K_{NWK} should be allowed to send commands to other nodes in the network.

As the knowledge of K_{ZLL} is the only condition for being able to join the network, we make the following interpretation: the set of genuine nodes $\mathcal{N}_{\text{gen}}$ is, according to the specification, nodes that know K_{ZLL}.[6] Likewise, we can say that in ZLL, the set of nodes $\mathcal{N}_{\mathsf{c}}$ that belong to the same network are characterized by sharing the same network key K_{NWK}.

Based on these considerations, we propose the following two security definitions:

Definition 2 (Security of Π_{join})**.** *Π_{join} is $\varepsilon_{\text{join}}$-secure if the probability that Π_{join} between nodes* c *and* n *is successful does not exceed $\varepsilon_{\text{join}}$ for any $\mathsf{n} \notin \mathcal{N}_{gen}$.*

Definition 3 (Security of Π_{cmd})**.** *Π_{cmd} is ε_{act}-secure if for any network $\mathcal{N}_{\mathsf{c}}$ the following holds: for any $\mathsf{n1} \notin \mathcal{N}_{\mathsf{c}}$ and $\mathsf{n2} \in \mathcal{N}_{\mathsf{c}}$ the success probability of Π_{cmd} between* $\mathsf{n1}$ *and* $\mathsf{n2}$ *does not exceed ε_{act}.*

These security goals are met in the system if both security definitions hold for small values $\varepsilon_{\text{join}}$ and ε_{act}. We note that further security requirements can be identified, such as data confidentiality, but again this has to be omitted due to lack of space.

5.4 Attacker Model

In the next step, we formalize the attacker. Different attacker types are imaginable here. To keep the model as flexible as possible, we follow the common approach in cryptography and formalize the attacker's capabilities by so-called queries. That is, formally an action of the attacker is captured by saying that the attacker makes the corresponding query (e.g., to a hypothetical party which then executes this action on behalf of the attacker). Obviously, the more types of queries an attacker can make, the more powerful she is.

The first two queries express the capability of an attacker to control the communication between nodes:
RECEIVE(n) – Receive all messages sent to node n (including broadcasts).

[6] More generally, one could say that there is an external procedure which guarantees that only genuine nodes known K_{ZLL}. But this would be out of scope of ZLL and hence is not considered here.

SEND(n, m) – Send message m to node n.
In principle, this would be sufficient to for two nodes to execute a certain protocol. However, protocol runs may involve secrets that are unknown to the adversary. On the other hand, it may be possible to remotely trigger certain protocols, e.g., by touching a button on a device. This is captured by the following query:
RUN(Π, n1, n2) – triggers protocol Π between the two nodes n1 and n2.

An adversary may also be able to introduce her own nodes to the system or remove nodes. Here, we distinguish between genuine nodes (which result from the Create process) and nodes created by the adversary:
ADDGENUINE(n) – adds a genuine node n to $\mathcal{N}$, i. e., a node created by Create.
ADDCUSTOM(n) – adds a node n to $\mathcal{N}$, which is under the full control of the adversary, meaning that she knows the complete state of the node including any secrets.
REMOVE(n) – the node n is removed completely from the system, e.g., by destroying it, removing the power source, etc.

Finally, an adversary may bring a (possibly genuine) node under her full control, meaning that she learns all internal values and can reprogram the node at her wish. We denote the corresponding query by CORRUPT(n).

5.5 Discussion

We already discussed that in ZLL, the Π_{join} protocol actually only checks if the node knows the master key K_{ZLL}, meaning that in reality, genuine nodes are characterized by knowing this value. Likewise, members of a network are characterized by sharing the same network key K_{NWK}. Observe that the Π_{join} protocol essentially established the network key between two nodes by sending its encryption using K_{ZLL} as the encryption key. Moreover, in Π_{cmd}, the commands are sent encrypted under the encryption key K_{NWK}. Thus, one can show[7] that if the encryption scheme is secure and both keys are secret, then both security definitions are fulfilled.

As the ZigBee specification admits that nodes may be physically attacked, the ZLL master key is subject to unauthorized disclosure. After this actually happened in March 2015 [Zil15], virtually any device is now able to pretend being a genuine node, violating the security definition given in Def. 2. Similarly, the network key K_{NWK} may be leaked, allowing a node to execute the Π_{cmd} protocol without having run Π_{join} before.

However, as we already stated, the choice of security definitions are also subject to interpretations. One could likewise define a network to be the set of all nodes that successfully executed the Π_{join} protocol with c. This definition would be violated if K_{NWK} is leaked. Still, we don't see this as the "correct" definition as the purpose of the Π_{join} protocol should be to achieve a given, independent security goal and not be the basis of a security definition. In any case, this again shows the necessity of giving concise security definitions. Moreover, observe that the security goals are violated only in presence of a physical

[7] In fact, one can conduct a full formal security proof if one assumes a secure encryption scheme. But again, this has to be omitted due to lack of space.

attacker, i.e., an attacker who can submit CORRUPT(n) queries. This demonstrates the importance of providing a clear attacker model.

6 Conclusion

In this paper, we provided technical background of the ZLL profile with focus on security mechanisms, especially on those used in the commissioning process, which is crucial to enable successful communication among ZLL nodes. Based on this, we developed a formal security model specifically for ZLL and reviewed the security mechanisms within this model. While this represents a necessary step towards a sound security analysis of ZLL, several fundamental security goals such as confidentiality and availability are not covered yet and are thus left for future research.

Acknowledgments This research was supported by the DFG project "Developing and Applying a Sound Security Framework for Sensor Networks".

References

[ARA12] Muhammad Raisul Alam, Mamun Bin Ibne Reaz, and Mohd. Alauddin Mohd. Ali. A Review of Smart Homes - Past, Present, and Future. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 42(6):1190–1203, 2012.

[BBK03] Elad Barkan, Eli Biham, and Nathan Keller. Instant ciphertext-only cryptanalysis of GSM encrypted communication. In *Advances in Cryptology-CRYPTO 2003*, pages 600–616. Springer, 2003.

[BHL06] Andrea Bittau, Mark Handley, and Joshua Lackey. The final nail in WEP's coffin. In *Security and Privacy, 2006 IEEE Symposium on*, pages 15–pp. IEEE, 2006.

[BLM+11] A. J. Bernheim Brush, Bongshin Lee, Ratul Mahajan, Sharad Agarwal, Stefan Saroiu, and Colin Dixon. Home automation in the wild: challenges and opportunities. In Desney S. Tan, Saleema Amershi, Bo Begole, Wendy A. Kellogg, and Manas Tungare, editors, *Proceedings of the International Conference on Human Factors in Computing Systems, CHI 2011, Vancouver, BC, Canada, May 7-12, 2011*, pages 2115–2124. ACM, 2011.

[GRT12] Laura Gheorghe, Razvan Rughinis, and Nicolae Tapus. Adaptive Security Framework for Wireless Sensor Networks. In Fatos Xhafa, Leonard Barolli, Florin Pop, Xiaofeng Chen, and Valentin Cristea, editors, *2012 Fourth International Conference on Intelligent Networking and Collaborative Systems, INCoS 2012, Bucharest, Romania, September 19-21, 2012*, pages 636–641. IEEE, 2012.

[HH12] Amy S. Hwang and Jesse Hoey. Smart Home, The Next Generation: Closing the Gap between Users and Technology. In *Artificial Intelligence for Gerontechnology, Papers from the 2012 AAAI Fall Symposium, Arlington, Virginia, USA, November 2-4, 2012*, volume FS-12-01 of *AAAI Technical Report*. AAAI, 2012.

[HHPT13] Keijo Haataja, Konstantin Hyppönen, Sanna Pasanen, and Pekka Toivanen. *Bluetooth Security Attacks: Comparative Analysis, Attacks, and Countermeasures*. Springer Science & Business Media, 2013.

[IEE03] IEEE Computer Society. IEEE Standard for Information Technology - Telecommunications and Information Exchange Between Systems - Local and Metropolitan Area Networks Specific Requirements Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs). *IEEE Std 802.15.4-2003*, pages 1–670, 2003.

[Int10] International Electrotechnical Commission. *IEC 62591 Ed. 1.0: Industrial communication networks – Wireless communication network and communication profiles – WirelessHART*, 2010.

[JW01] Markus Jakobsson and Susanne Wetzel. Security weaknesses in Bluetooth. In *Topics in Cryptology – CT-RSA 2001*, pages 176–191. Springer, 2001.

[MH12] Sarah Mennicken and Elaine M. Huang. Hacking the Natural Habitat: An In-the-Wild Study of Smart Homes, Their Development, and the People Who Live in Them. In Judy Kay, Paul Lukowicz, Hideyuki Tokuda, Patrick Olivier, and Antonio Krüger, editors, *Pervasive Computing - 10th International Conference, Pervasive 2012, Newcastle, UK, June 18-22, 2012. Proceedings*, volume 7319 of *Lecture Notes in Computer Science*, pages 143–160. Springer, 2012.

[Sil15] Silicon Labs. *Application Development Fundamentals: Thread, Rev. 0.3*, 2015.

[UJS13] Blase Ur, Jaeyeon Jung, and Stuart Schechter. The current state of access control for smart devices in homes. In *Workshop on Home Usable Privacy and Security (HUPS)*, 2013.

[VHP+13] Niko Vidgren, Keijo Haataja, Jose Luis Patino-Andres, Juan Jose Ramirez-Sanchis, and Pekka Toivanen. Security Threats in ZigBee-Enabled Systems: Vulnerability Evaluation, Practical Experiments, Countermeasures, and Lessons Learned. In *46th Hawaii International Conference on System Sciences, HICSS 2013, Wailea, HI, USA, January 7-10, 2013*, pages 5132–5138. IEEE, 2013.

[VJU+12] Marco Valero, Sang Shin Jung, A. Selcuk Uluagac, Yingshu Li, and Raheem A. Beyah. Di-Sec: A distributed security framework for heterogeneous Wireless Sensor Networks. In Albert G. Greenberg and Kazem Sohraby, editors, *Proceedings of the IEEE INFOCOM 2012, Orlando, FL, USA, March 25-30, 2012*, pages 585–593. IEEE, 2012.

[Wri09] Joshua Wright. KillerBee: Practical ZigBee Exploitation Framework. ToorCon 11, 2009.

[Zig12a] ZigBee Alliance. *ZigBee Light Link Standard Version 1.0 – Document 11-0037-10*, April 2012.

[Zig12b] ZigBee Standards Organization. *ZigBee Specification – Document 053474r20*, September 2012.

[Zig13a] ZigBee Alliance. *Smart Energy Profile 2 Application Protocol Standard – Document 13-0200-00*, April 2013.

[Zig13b] ZigBee Alliance. *ZigBee Home Automation Public Application Profile Version 1.2 – Document 05-3520-29*, June 2013.

[Zil15] Tobias Zillner. White paper: ZigBee Exploited – The good, the bad and the ugly. Technical report, Cognosec, August 2015.

[ZS15] Tobias Zillner and Sebastian Strobl. ZigBee Exploited – The good, the bad and the ugly. Black Hat USA, 2015.