Secure key distribution in heterogeneous interoperable industrial Internet of Things

Fergal MARTIN-TRICOT, Cédric EICHLER, Pascal BERTHOMÉ
LIFO - INSA Centre-Val de Loire
88 Boulevard Lahitolle
18000 BOURGES, France
E-Mail: firstname.lastname@insa-cvl.fr

***Abstract*—Industry 4.0 proposes the inclusion of *IoT* components in industrial contexts to have a better control on production and logistic processes.**

Unfortunately, growing connectivity in a formerly isolated world induces major security risks which are especially critical in an industrial context. Furthermore, the highly fragmented nature of the industrial *IoT* market imposes interoperability management. Interoperability and heterogeneity generally poses security challenges, exacerbating these risks.

oneM2M is a promising standard that can enable interoperability and ensure communication with various tier protocols. The interface between oneM2M and tiers protocols has however to be trusted, which is undesirable. In this paper, we focus on the first phase of an approach to enable end-to-end encryption between oneM2M and Zigbee devices: secure key exchange. We fully define an exchange protocol taking place during the enrolment of a Zigbee device within oneM2M. We provide and discuss an implementation of this protocol, demonstrating its technical feasibility.

I. Introduction

During the last years, the industry adopted *Machine to Machine* (***M2M***), a paradigm of communication that allows direct machine to machine exchanges [1]. This approach provides tools to have a better vision and therefore control on procedures. The next step of this transformation is now on track: Industry 4.0 [2]. The idea is to use the *Internet of Things* (***IoT***) to interconnect every component of the industrial process and thus have a real-time vision of it.

The industrial context implies some paramount security requirements induced by the criticality of processes in terms of safety and industrial secrecy. For this reason, protocols used for industrial *IoT* must be blameless on the security mechanism employed.

Furthermore, there is a lot of protocols that address different needs. The most interesting ones for industrial concerns can be divided in two main categories: *Wireless Local Area Network* (***WLAN***) and *Low Power Wide Area Network* (***LPWAN***) [3].

WLAN are suitable to communicate at room or building scale. This is the perfect choice to have, for example, a sensor network to monitor a facility production. This category comprises classical technologies such as WiFi or Bluetooth but also Mesh networks [4] that offers some interesting properties like better range and energy efficiency [5]. The most used and developed mesh *IoT* technologies are Zigbee [6], Thread [7] and Z-Wave [8].

On the other hand, *LPWAN* usually rely on a network widely deployed and proposed by a service provider. The main difference with 3G or 4G networks is that the protocol is more optimized to reduce power consumption. These protocols can thus work in a wide area and allow a device to be connected anywhere [9]. In an industrial approach, it can be used in shipment and supply chain.

Due to the multiplicity of manufacturers, protocols, and their complementarity, we consider that a realistic use case must involve various different protocols. It raises a known problem in computer science: protocol interoperability. When two different technologies are involved, data structures are different and communication becomes more complicated. In this context, security management is still an issue [10] as different approaches must coexist, raising the question of security at their interfaces.

A. Scenario and objective

Our scenario takes place in a factory which needs to interconnect *IoT* devices. A central network, dedicated to the control and supervision of all devices, communicates with different protocols. Since interoperability is centred around this central network, our main objective is to **ensure data security through end to end encryption between an end device part of a tier protocol and a device within the central network.** In addition, since they may be closed and based on proprietary technologies, we consider **impossible to alter the operation of an end-device in a tier protocol**.

An existing specification, oneM2M, seems adequate to operate the central network and guarantee protocol interoperability. In [11], we proposed a high-level view of an approach to enable communication between a oneM2M node and a Zigbee device with end-to-end ciphering of the data exchanged. We chose to work on Zigbee because of its openness specification, the ease to find test devices - like XBee -, and the fact that Zigbee inspired other *WPAN* protocols. Indeed, other *WPAN* we studied (in particular Z-Wave and Thread) have similar enrolment procedure, facilitating the adaptation of our work to other protocols.

978-1-6654-2854-5/21/$31.00 ©2021 IEEE

B. Contributions and article overview

In this paper, we focus on the first phase of this approach for secure communication: enrolment and secure key provisioning. Contributions of this paper are twofold:

- we fully specify how keys may be securely provisioned during the enrolment of a ZigBee device within oneM2M.
- we show how such enrolment can be implemented and present a demonstrator validating the feasibility of the approach.

This paper is organized as follows. Section II provides an overview of the related works focused on security and interoperability of IoT protocols. Sections III and IV detail the main characteristics and relevant security mechanisms of the chosen interoperability and third-party protocols, respectively. The following sections present our contributions. Section V details the proposed enrolment procedure. Section VI describes the experimental context of our implementation. Section VII and VIII detail our proposal and the required modifications on ZigBee's and oneM2M's side, respectively. Finally, section VII concludes and discusses future work.

II. Related Work

The security of *IoT* protocols has been well studied. This is particularly true for the protocols we are interested in: *Mesh Local Area Network*.

Zigbee is probably the most mature and represented of such protocols. Modern security was introduced in version 3.0 of Zigbee and has since been largely analysed, for example by Zillner [12] and Fan et al. [13]. More recent protocols inspired by Zigbee such as Thread and Z-Wave have also been the target of in-depth security analysis. One may cite for example Dinu and Kizhvatov [14] on Thread and Rouch et al. [15] regarding Z-Wave. The consensus on these protocols is that security is acceptable in most cases [16] because most security breaches are coming from implementation issues.

In spite of the variety of proposals addressing *IoT* interoperability, solutions are largely converging. They usually imply a Gateway (e.g. [17], [18]) which has to translate data and instructions between different protocols. Regardless of the implementation and varying details, to the best of our knowledge, all of these kinds of approaches rely on a central translator interfacing the different protocols. They thus impose the same trust model: the interoperability-enabling component, generally unique and central, is necessarily trusted.

The security of a trusted interoperability gateway is generally critiqued, for example by Tarouco et al. in [19] or by Martino et al. in [20]. To improve data security, it is thus necessary to limit the number of trusted components and their trust level. End-to-end encryption is a perfect candidate to achieve this aim. This is usually difficult in heterogeneous environment, but has already been studied. In 2017, for instance, Mukherjee et al. proposed a solution to implement end-to-end encryption for Cloud-Fog communication [21] through a pre-shared key approach.

To the best of our knowledge, no existing work guarantee secure key distribution in a heterogeneous deployment to ensure end-to-end encryption during communication.

III. An interoperability solution: *oneM2M*

oneM2M is a standard initially proposed by the *ETSI* (European Telecommunication Standard Institute) and nowadays supported by a dozen of national and international standardization institutes and professional forums. It aims to provide a common framework to operate in *Machine to Machine* (*M2M*) and *IoT* paradigm [22]. In particular, oneM2M proposes a mature approach for interoperability management and numerous internal security measures [23].

A. Interoperability management in oneM2M

The oneM2M's interoperability management relies entirely on a specific entity: the *Interworking Proxy Entity* (***IPE***). The idea is to create a node supporting a non-oneM2M communication interface and a oneM2M's one.

The *IPE* is thus connected to a Third-Party protocol and has the following main functions:

- It gets data from the network it is connected to.
- It translates data between the tier network's and oneM2M's formats/ontologies.
- It ensures the integrity of the data it transmits to the oneM2M's network - using oneM2M's security measures.

A specific protocol can be made compatible with oneM2M by creating a dedicated *IPE*. Since this approach is generic, it can work with most of the existing protocols.

However, security can become a problem because every data transit unencrypted through the *IPE*. A vulnerability of the *IPE* would give read and write access to every data of the third-party network. Our ultimate goal is thus to provide end-to-end security from the tier network up to oneM2M to negate this intrinsic flaw.

B. Introduction to oneM2M's internal data security

To securely communicate within the oneM2M network, a oneM2M device can follow a procedure defined in the *Remote Security Provisioning Framework* (***RSPF***). Its aim is to provide a node with the keys it needs to authenticate and communicate securely within the network. The specification defines a specific entity to ensure key provisioning during enrolment: the *M2M Enrolment Function* (***MEF***).

In particular, two oneM2M nodes may communicate securely with end-to-end encryption if they have exchanged an *ESData* key. It can be provisioned to the two nodes which need to communicate through *RSPF*.

With this approach, data transmitted can be totally secured between two oneM2M nodes. Adapting this mechanism between a oneM2M node and a third-party device would thus provide end-to-end encryption. To do so, we must be able to **securely provide an *ESData* key to a third party device** and we must assume that we can modify the *IPE* and the oneM2M's end device which wants to communicate with the tier network. Obviously, after this step, one should still

ensure that communication is possible between the two nodes. In [11], we show how data translation could be deported in the oneM2M node after key provisioning, making communication with end-to-end encryption possible.

IV. A third-Party protocol as an example: Zigbee

To design and to assess the feasibility of our approach, we rely on a real protocol to find a working technical solution.

A. Introduction to Zigbee

Zigbee is a specification used to create *Wireless Personal Area Network* (***WPAN***) using radio communication. Introduced for the first time in 1998 and standardized in 2003, the current version, *Zigbee 3.0* [24], proposes an open and secure *WPAN* technology using a mesh network architecture.

B. Data ciphering

For securing communications, Zigbee uses *AES* ciphering algorithm with 128bits long keys in *CCM** operating mode. There are two main keys that are used through all the lifespan of a node.

The first one is the *install code (**IC**)* which is included inside the device during manufacturing. It cannot be changed, should be unique and is used for the authentication during the enrolment of the node.

The second one is the Network Key. It is shared by every node of a specific network and used to cipher every communication inside it.

C. Enrolment

The enrolment, like in every wireless protocol, is a critical phase during which the key used to cipher all communications must be transmitted to a new device. Two different security models for the enrolment exist in Zigbee 3.0: centralized and distributed. In this paper, we will focus on the centralized model, as it is the most secured.

In a centralized network, the authentication and enrolment of a new nodes is done by an entity called *Trust Centre (**TC**)*.

The key point is the usage of the *IC* to authenticate the new device to the *TC*. Once this key is shared with the *TC* (QR code scanning for example), the enrolment procedure can be done securely. Fig. 1 depicts the enrolment of a new node within a Zigbee network. This procedure is divided in three steps: first, a scan to find reachable networks, then the standard association procedure and finally the emission of the network key to the new node within an *APS - Transport Key* message.

After the enrolment phase, a node is able to communicate securely using the *Network Key* with every device of the network.

V. Secure key distribution in heterogeneous network

A. The working principle

The *IPE* and the Zigbee gateway have total access on data that transit through them. To protect the data, we must cipher it from the sender to the receiver. We propose a way to divert

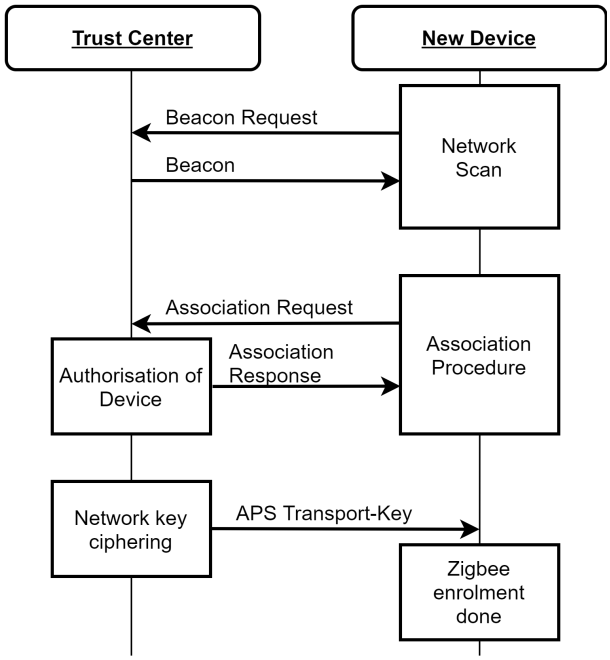

Fig. 1. Zigbee standard enrolment procedure

the enrolment principle of Zigbee to deploy a ciphering key into a Zigbee node, from the oneM2M network.

Our proposal relies on the creation of an *Enrolment Gateway (**EG**)* within the oneM2M's *MEF* which would receive and deal with Zigbee's enrolment procedure. This way, it becomes possible to deliver a specific ciphering key for a Zigbee node. After the provisioning of this key, the Zigbee node will continue to work normally but the message sent will be ciphered using a key only known by the recipient within the oneM2M's network.

In addition to the *MEF*, our complete solution requires the separation of the *IPE* in two entities: the *IPE-OCI* and the *IPE-NI* [11]. The former is involved in the communication and the deport of the translation capabilities within the oneM2M node. In this paper, as we focus on the enrolment part and therefore drop the *IPE-OCI*. The *IPE-NI (IPE - Network Interface)* is the remaining part of the *IPE*. It only transmits communications without doing anything else. In what follows, we ignore the *IPE-NI* for simplicity's sake, as it does not intervene in the enrolment apart for forwarding messages.

B. Enrolment in detail

Fig. 2 shows how the enrolment process of a new Zigbee node is done along our solution. With this approach, the only node that has to know the *network key* is our *EG* (which is part of a MEF). The Trust Centre is consequently just an intermediary for the key distribution and never has access to the delivered keys.

Here are the details about the three involved actors:

- Our ***EG*** is designed to manage Zigbee enrolment from the oneM2M network (within the MEF). It is a whole new entity and must implement all of Zigbee's security

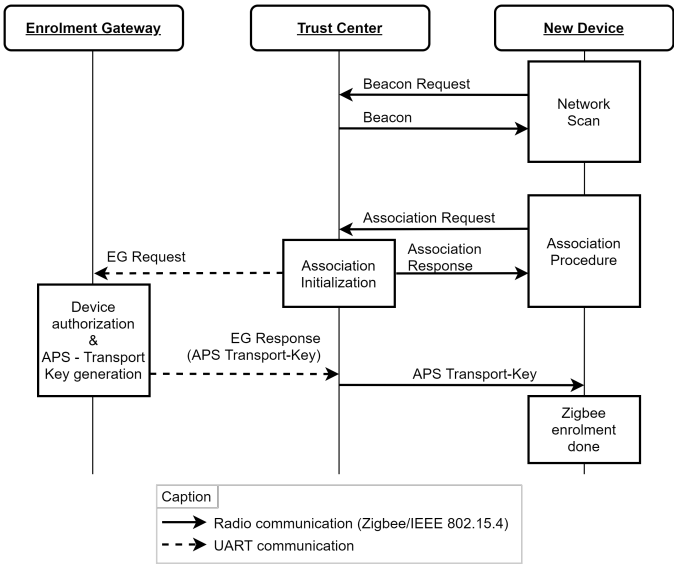

Fig. 2. Sequence diagram of the enrolment procedure

primitives to create the secure *APS - Transport Key* message.

- The **Zigbee Trust Centre** (classically within the gateway) is an existing entity which must be deeply transformed. During enrolment, its role is limited to the management of the IEEE 802.15.4 association procedure before transmitting the Zigbee enrolment request to the *EG*.
- Lastly, the **new device**, is a normal Zigbee device going through enrolment. Like every Zigbee device, it has an *install code* which was provisioned during factory and must be inquired within the *EG* before the enrolment (out-of-scope operation). It is designated as *enrolee* in the following.

As shown in Fig. 2, the enrolment operation begins with an *IEEE 802.15.4* scan which is initiated by the enrolee to find a *PAN* that accepts new devices. Like every *PAN* coordinator, our Zigbee gateway answers with a beacon containing all the information about its network.

After receiving the beacon, the enrolee sends an *Association Request* to the *PAN* coordinator, our gateway, which answers with an *Association Response* and transmits the request to our *EG*.

The *EG* can now send the *network key* within an *APS - Transport Key* message. The key distribution is secured from man-in-the-middle attack thanks to standard Zigbee security which relies on a pre-shared key: the *install code*.

After receiving its key, the enrolee is ready to start securely communicating with end-to-end encryption. The only remaining step is to distribute the key to the oneM2M device, using standard oneM2M security mechanisms (*RSPF*).

VI. Implementation overview

The main purpose of our implementation is to illustrate and demonstrate the feasibility of our solution for end-to-end data security with its most innovative aspect: the enrolment management.

Therefore, this first iteration covers the enrolment management, within a oneM2M deployment, of a Zigbee device. As discussed, it implies the creation of the *EG*, within a *MEF* and some modifications on the Zigbee Gateway.

In order to implement our solution and thus validate its technical feasibility, we worked with different hardware and software. For the hardware part, we selected a Nordic Semiconductor development card: the nRF 52840. It is designed and shipped with firmware to develop *IoT* solutions with Zigbee, Thread or Bluetooth.

The details of the chosen architecture, as depicted in Fig. 3, are as follows:

- The **Zigbee gateway** uses an nRF 52840 card. For this entity, we wanted to work with a Zigbee implementation and modify it to answer our needs. Unfortunately, we didn't find any open-source implementation of the latest Zigbee version. Thus, we are using an IEEE 802.15.4 SDK which is the basis of the Zigbee protocol.
- The **sniffer** uses the same development card with a specific firmware to use it as an IEEE 802.15.4 sniffer. Its main purpose is to verify the correct emission of the different messages composing our enrolment procedure.
- For the **end-device** we choose to work with another nRF 52840 card using the Zigbee CLI firmware from the Nordic's SDK.
- Last, the ***EG*** is implemented with Golang language within a computer. This entity is communicating directly to the *Zigbee gateway* thanks to an *UART* connexion (*USB*).

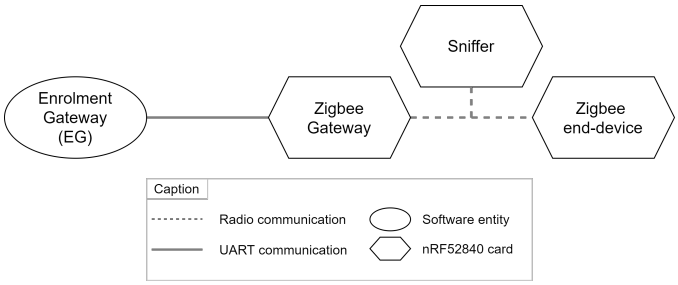

Fig. 3. Architecture of our implementation

VII. Implementation of a Zigbee gateway

A. Working principle

Due to the use of a SDK, our implementation shall start with a configuration step. To do so, we use the *MAC Sublayer Management Entity (**MLME**)* which offers call for the stack initialization. In our example, we are using it to configure the following variables on the gateway:

- The extended address (64 bits)
- The short address (16 bits)
- The beacon used with its payload and its length
- The flag to allow association

- And, last but not least, we use the *mlme_start_config* to initialize the network and configure its characteristics.

After the radio configuration, we initialize the *UART* communication with the *EG*. Devices are then ready to start the enrolment process depicted in Fig. 2 and the Zigbee gateway waits for the reception of a beacon request from a Zigbee node.

B. Beacon generation

The beacon is a specific message type that is sent periodically or on demand (depending on the network configuration) by a Zigbee router. In both case, it is composed of two parts: an IEEE 802.15.4 and a Zigbee one. The former is managed by the stack we are using and generated according to the previously defined configuration. Therefore, only the latter part has to be managed.

Fig. 4 shows the data structure of a Zigbee beacon, which is encapsulated within an IEEE 802.15.4 header.

```
ZigBee Beacon, ZigBee PRO, EPID: NordicSe_8f:ce:29:a4:10
   Protocol ID: 0
   Beacon: Stack Profile: ZigBee PRO, Router Capacity, End Device Capacity
      .... .... .... 0010 = Stack Profile: ZigBee PRO (0x2)
      .... .... 0010 .... = Protocol Version: 2
      .... .1.. .... .... = Router Capacity: True
      .000 0... .... .... = Device Depth: 0
      1... .... .... .... = End Device Capacity: True
   Extended PAN ID: NordicSe_8f:ce:29:a4:10 (f4:ce:36:8f:ce:29:a4:10)
   Tx Offset: 16777215
   Update ID: 0
```

Fig. 4. Zigbee's Beacon payload

It is composed of information about the network capabilities (protocol and stack version, transmission offset...) as well as some other concerning the sender of this beacon (router, end-device capacity). This beacon is thus dynamically generated by our implementation depending on the configuration.

C. Communication with the EG

We created a simple communication protocol with a data structure allowing the use of any communication medium. In our demonstrator, it is done though an *UART* connexion, but our protocol could be encapsulated within, for example, a classical IP communication.

In this first version, we didn't add data ciphering as we consider that the data transmitted are either non-critical or already protected by Zigbee ciphering (for the *APS - Transport Key*). We defined two main messages type which are build as follows:

- An **EG Request** message includes extended and short addresses of the gateway and the new device. This is all the information needed by the *EG* to generate the *APS - Transport Key* message.
- An **EG Response** is a message that just include, as a payload, the full *APS - Transport Key* packet that is transmitted by the *EG* and relayed to the device by the Zigbee gateway.

VIII. Implementation of an Enrolment Gateway

The *EG* is one of the most important part of our implementation as it is a trusted entity which has to replace the Zigbee's *Trust Centre*.

A. Working principle

We choose to use the Golang language for its ease of use and modularity. Moreover, it is possible to import and use some C functions. This *EG* has to meet three main objectives:

- Communicate with the Zigbee gateway (with our custom protocol)
- Forge the different headers composing the *APS - Transport Key* message
- Being able to cipher with Zigbee's primitives

The operation of this entity can be separated in four major steps:

1) An *EG Request* is received from the *UART* connexion
2) The *EG* uses the information of this message to forge the headers
3) The *IC* is transformed to cipher the payload containing the network key
4) The payload and header are assembled and sent back to the Zigbee gateway within an *EG Response* message

For every cryptographic function, we based our work on an open-source project in C[1].

B. APS - Transport Key generation

The main part of the *EG* work is to forge the whole packet which transports the network key to the new device. We forge every header according to the information within the message received from the Zigbee gateway. We will here present briefly the different headers involved.

1) The first one is the *IEEE 802.15.4* data header. It contains the necessary information for the packet to be delivered to the right recipient: destination and source address, *PAN* identifier, a sequence number as well as a frame control field to set some flags.
2) The second is the *Zigbee Network Layer Data*. This is the first layer of the Zigbee stack and is made to extend the functionalities of the previous header. It always includes extended addresses as well as a radius and some flags to define the packet type (in our case, a Data frame).
3) The third is the *Zigbee Application Support Layer*. This layer ensures the Zigbee acknowledgment if needed and give information about the security of the next layer (enabled in our case).
4) The last one is the *Zigbee Security Header* which informs about the key used to cipher its payload and contains an integrity code.

Our *EG* has to forge only the headers above the IEEE 802.15.4 layer (2 to 4) as the first is generated by our Zigbee gateway's stack.

[1] *SecBee* project by Cagnosec (https://github.com/Cognosec/SecBee)

The payload of this packet is a *Zigbee Transport Key Header* which, as it names suggests, transport the network key. Fig. 5 presents the structure of this header.

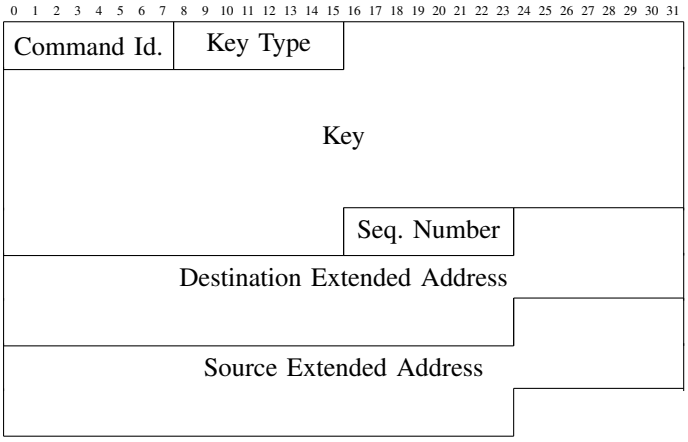

Fig. 5. *Zigbee Transport Key Header* for an *APS Transport Key* message

In our case, the sequence number must be the same as the one of the previous header and the Key Type is set to 0x01 (Standard network key). After the generation of this header, it has to be ciphered with the *install code* of the enrolee.

The first step is to transform the *install code* according to the Zigbee specification which uses Matyas-Meyer-Oseas hash function to transform the key. After doing this, the encryption of the payload can be done with an *AES CCM** algorithm.

At the end, the payload is assembled with the previously generated headers and sent back to the Zigbee gateway within our custom *EG Response* message.

IX. Conclusion

Integration of *IoT* within the industry, as the foundation of the 4th industrial revolution, is crucial. Yet, it poses new security threats, especially when considering a heterogeneous, interoperable environment.

In this paper, to achieve end-to-end data security within a heterogeneous industrial *IoT* architecture, we propose a solution for secure key distribution between a promising standard handling for interoperability management, oneM2M, and a tiers protocol. Using Zigbee as a use-case, we interface the internal security mechanisms of both protocols to handle Zigbee node enrolment within oneM2M and thus centralize key management. In addition to its wide use, the security model of Zigbee is close to others, such as Z-Wave's and Thread's, making our solution adaptable to these kinds of protocols.

Our implementation validates the feasibility of the enrolment management of a Zigbee network from within a oneM2M network. Hence, it becomes possible to have a total control on key distribution and authorized nodes.

The keys being exchanged, we plan on working on the second part of our initial proposal: communication with end-to-end encryption. To this end, we must implement IPE's translation capabilities within a oneM2M node, modify the Zigbee gateway to handle communication, and rely on an implementation of oneM2M containing the relevant security mechanisms defined within the standard.

References

[1] J. Latvakoski, A. Iivari, P. Vitic, B. Jubeh, M. B. Alaya, T. Monteil, Y. Lopez, G. Talavera, J. Gonzalez, N. Granqvist, M. Kellil, H. Ganem, and T. Väisänen, "A survey on M2M service networks," *Computers*, vol. 3, no. 4, pp. 130–173, 2014.

[2] H. Lasi, P. Fettke, H.-G. Kemper, T. Feld, and M. Hoffmann, "Industry 4.0," *Business & Information Systems Engineering*, vol. 6, no. 4, pp. 239–242, Aug 2014.

[3] K. Mekki, E. Bajic, F. Chaxel, and F. Meyer, "A comparative study of LPWAN technologies for large-scale IoT deployment," *ICT Express*, vol. 5, no. 1, pp. 1–7, mar 2019.

[4] Y. Liu, K.-F. Tong, X. Qiu, Y. Liu, and X. Ding, "Wireless mesh networks in IoT networks," in *2017 International Workshop on Electromagnetics: Applications and Student Innovation Competition*. IEEE, 2017, pp. 183–185.

[5] C. Gomez and J. Paradells, "Wireless home automation networks: A survey of architectures and technologies," *IEEE Communications Magazine*, vol. 48, no. 6, 2010.

[6] ZigBee, "ZigBee: Securing the IoT," ZigBee Alliance, Tech. Rep., 2017.

[7] Thread, "Thread technical overview," Thread, Tech. Rep., Oct. 2015.

[8] ABR, "Introduction to the Z-Wave security ecosystem," Sigma Design, Tech. Rep., 2016.

[9] K. E. Nolan, W. Guibene, and M. Y. Kelly, "An evaluation of low power wide area network technologies for the internet of things," in *2016 International Wireless Communications and Mobile Computing Conference (IWCMC)*. IEEE, sep 2016.

[10] M. Elkhodr, S. Shahrestani, and H. Cheung, "The internet of things : New interoperability, management and security challenges," *International Journal of Network Security & Its Applications*, vol. 8, no. 2, pp. 85–102, mar 2016.

[11] F. Martin-Tricot, C. Eichler, and P. Berthomé, "An enrolment gateway for data security in heterogeneous industrial internet of things," in *2020 International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*. IEEE, sep 2020.

[12] T. Zillner, *ZigBee Exploited - The Good, The Bad and The Ugly*, Cagnosec, August 2015.

[13] X. Fan, F. Susan, W. Long, and S. Li, "Security analysis of Zigbee," 2017.

[14] D. Dinu and I. Kizhvatov, "EM analysis in the IoT context: Lessons learned from an attack on Thread," pp. –, 2018.

[15] L. Rouch, J. François, F. Beck, and A. Lahmadi, "A Universal Controller to Take Over a Z-Wave Network," in *Black Hat Europe 2017*, London, United Kingdom, Dec. 2017, pp. 1–9.

[16] D. Celebucki, M. A. Lin, and S. Graham, "A security evaluation of popular internet of things protocols for manufacturers," in *Consumer Electronics (ICCE), 2018 IEEE International Conference on*. IEEE, 2018, pp. 1–6.

[17] H. Derhamy, J. Eliasson, and J. Delsing, "IoT interoperability—on-demand and low latency transparent multiprotocol translator," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1754–1763, oct 2017.

[18] M. Blackstock and R. Lea, "IoT interoperability: A hub-based approach," in *2014 International Conference on the Internet of Things (IOT)*. IEEE, oct 2014.

[19] L. M. R. Tarouco, L. M. Bertholdo, L. Z. Granville, L. M. R. Arbiza, F. Carbone, M. Marotta, and J. J. C. de Santanna, "Internet of things in healthcare: Interoperatibility and security issues," in *2012 IEEE International Conference on Communications (ICC)*. IEEE, jun 2012.

[20] B. D. Martino, M. Rak, M. Ficco, A. Esposito, S. Maisto, and S. Nacchia, "Internet of things reference architectures, security and interoperability: A survey," *Internet of Things*, vol. 1-2, pp. 99–112, sep 2018.

[21] B. Mukherjee, R. L. Neupane, and P. Calyam, "End-to-end IoT security middleware for cloud-fog communication," in *2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud)*. IEEE, jun 2017.

[22] *Functional Architecture - TS-0001 - v3.22.1*, oneM2M, Feb. 2021.

[23] *Security Solutions - TS-0003 - v3.14.0*, oneM2M, Feb. 2021.

[24] ZigBee Alliance, *ZigBee Specifications*, Aug. 2015.