FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Centralized Key Exchange for Constrained Devices in Building Automation Systems

MASTERARBEIT

zur Erlangung des akademischen Grades

Master of Science

im Rahmen des Studiums

Software Engineering und Internet Computing

eingereicht von

Robin Müller
Matrikelnummer 0926507

an der Fakultät für Informatik
der Technischen Universität Wien

Betreuung: Ao. Univ. Prof. Dr. Wolfgang Kastner
Mitwirkung: Kolleg. Dipl.-Ing. Andreas Fernbach, BSc

Wien, 24. August 2016

Robin Müller

Wolfgang Kastner

FAKULTÄT
FÜR !NFORMATIK

Faculty of Informatics

Centralized Key Exchange for Constrained Devices in Building Automation Systems

MASTER'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

Software Engineering and Internet Computing

by

Robin Müller
Registration Number 0926507

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: Ao. Univ. Prof. Dr. Wolfgang Kastner
Assistance: Kolleg. Dipl.-Ing. Andreas Fernbach, BSc

Vienna, 24th August, 2016

Robin Müller   Wolfgang Kastner

Erklärung zur Verfassung der Arbeit

Robin Müller
Eduard Rösch-Straße 8A, 2000 Stockerau

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 24. August 2016

Robin Müller

Kurzfassung

Gebäudeautomationssysteme gewinnen in modernen Bauvorhaben zunehmend an Relevanz, typischerweise um Energieeffizienz und Raumkomfort zu steigern. Eine der großen Herausforderungen bei solchen Systemen ist Netzwerksicherheit. Insbesondere drahtlose Kommunikationswege zwischen einzelnen Netzwerkknoten sollten verschlüsselt und authentifiziert sein. Dabei ist besonderes Augenmerk auf die Effizienz der Verschlüsselung zu legen, da die meisten Geräte, die typischerweise im Einsatz sind, nur über sehr geringe Rechenleistung verfügen. Aufgrund dieser Performance-Einschränkungen sind potenzielle Lösungen auf effiziente, symmetrische Kryptographieverfahren angewiesen. Die daraus resultierende Schwierigkeit liegt darin, ein passendes Schlüsselaustauschschema zu finden, welches es ermöglicht, die geheimen Schlüssel effizient zwischen einzelnen Geräten auszutauschen, ohne dabei die Skalierbarkeit und Erweiterbarkeit des Systems zu gefährden.

In dieser Arbeit wird ein zentralisiertes Schlüsselaustauschschema vorgestellt, bei dem symmetrische Schlüssel mithilfe einer vertrauenswürdigen dritten Partei erzeugt und ausgetauscht werden können. Dabei wird die Tatsache zu Nutze gemacht, dass Gebäudeautomationssysteme typischerweise bereits über eine zentrale Kontrollstelle verfügen, die als Mittelsmann bei dem Schlüsselaustausch dienen kann. Ein Prototyp wurde implementiert, um die Realisierbarkeit dieses Vorgehens zu demonstrieren und dessen Performance zu evaluieren, wobei ein besonderes Augenmerk auf Speicher- und Energieeffizienz liegt. Der Vergleich zwischen einem ungesicherten Verfahren sowie der sicheren Implementierung zeigt einen akzeptablen Anstieg der gemessenen Kennwerte. Des Weiteren sollte es möglich sein, das vorgeschlagene Schema in Zukunft mit flexiblen Verschlüsselungsprotokollen wie etwa DTLS zu integrieren.

Abstract

Building automation systems are becoming increasingly relevant in modern building projects to improve energy efficiency and increase inhabitant comfort. One of the ongoing challenges in such systems is providing security to the network. In particular, wireless message exchanges between individual network nodes need to be encrypted and authenticated efficiently, since the wireless devices typically deployed in most building automation systems are low-powered microcontrollers. Due to these performance constraints, potential solutions are limited to using efficient, symmetric cryptography. The resulting challenge lies in finding a suitable key-exchange scheme to establish the shared keys that is both scalable and extensible.

In this thesis, a centralized key exchange scheme is proposed that is capable of establishing secret keys for arbitrary node pairs by using the building controller as a trusted third party. The proposed scheme is well suited for the particular network topology typically found in building automation systems while requiring only symmetric cryptography. A prototype was implemented to demonstrate the feasibility of the proposed approach and to evaluate the performance of the implementation with regard to memory and energy efficiency. The evaluation shows an acceptable overhead when using end-to-end encryption compared to an unsecured approach. In addition, the proposed scheme could also be easily integrated with more flexible encryption protocols such as DTLS in the future.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

1.1 Motivation

The Internet of Things (IoT) as a whole has come to represent a very broad and popular field of research in recent years. With its numerous facets and application domains it covers a wide range of topics and presents new challenges in those fields. Due to this inherent variety, the Internet of Things typically doesn't allow for blanket solutions or one-size-fits-all approaches. Different research domains often have vastly different requirements and assumptions that they are built upon, and, as a result, require the development of new approaches and solutions that are specifically tailored to their particular problem spaces. Ongoing research shows progress on individual issues and challenges, but due to the relative novelty of the field as a whole, there is still a shortage of tailor-made, future proof solutions for most application domains. One such problem domain are building automation systems (BAS), which play an increasingly important role in modern building projects.

Building Automation Systems

Historically, the primary functionality of building automation systems used to be the automation of heating, ventilation and air-conditioning systems. Today, the goal usually is to improve overall system efficiency and reduce the resulting energy consumption. Over time, their application has been extended to cover other basic functionality such as lighting, as well as safety critical services such as fire and smoke detectors. These automation networks typically consist of a wire-based backbone network which fulfills most of the core functions. In addition, small, wireless microcontrollers are often used that fulfill supporting roles, such as sensor devices. The advantage of their wireless nature lies in the fact that they can operate in environments that are difficult to access physically and don't require additional wiring. Networks utilizing such wireless components are usually called wireless sensor networks, or WSNs for short. With the increasing relevance of the Internet of Things and the resulting growing integration and networking capabilities comes a necessity to properly secure the network and its individual devices in order to maintain the expected functionality.

To achieve this, much like in traditional networks, a secure BAS solution first and foremost needs to guarantee data confidentiality and integrity. Improving availability and reliability in addition to this would be desirable, but isn't always possible due to device and performance constraints. In order to find a suitable solution, it is necessary to take into account the particular limitations and characteristics of building automation systems and work around them or use them to an advantage.

1.2 Problem Statement

At this time, an ideal solution for providing security to building automation systems hasn't emerged yet. The inherently different nature of wireless sensor networks and building automation systems as opposed to traditional network models causes a range of new problems, which require the development of new technologies and solutions.
The two following factors in particular are the cause of most security issues faced by WSNs and BAS. As a result, these two factors will be the defining criteria when designing a suitable solution:

- The open and accessible nature of wireless sensor networks: Most sensor nodes run unguarded in a public environment, which allows potential attackers to physically access the hardware itself and possibly tamper with it. Traditional networks are generally based on the assumption that the devices are protected and a breach of hardware security presents an uncommon worst-case scenario. Sensor networks on the other hand must be able to withstand such attacks in a reliable manner which does not interfere with the correct operation of the network as a whole. Additionally, accessing and manipulating a hardware node can provide relatively easy access into the underlying network, which makes it necessary for the system to remain stable in the face of malfunctioning or malicious nodes.

- The constrained nature of the networked devices: Sensor nodes in WSNs usually consist of low-power microcontrollers with very limited hardware. In addition, they sometimes have to be powered solely by batteries. In order to increase their battery lifetime, it is necessary to use very energy efficient protocols and implementations. This holds especially true for cryptographic functions, as those tend to be rather resource-intensive. The fact that most devices in use today do not include a dedicated cryptography co-processor further exacerbates this problem, as software implementations of most cryptographic functions (especially those based on public-key cryptography) are not particularly efficient. This further limits the number of practically usable protocols and cryptographic systems even further.

These two factors unfortunately make tried-and-trusted security approaches that find widespread use in other application domains rather unsuitable for adoption in a building automation context. Finding a solution that provides security (primarily in the form of confidentiality and integrity) to the network while taking the above problems into consideration will be the primary challenge of this thesis.

1.3 Aim of the Work

The goal of this thesis is to develop a communications protocol which fulfills the security requirements presented by the building automation domain, while at the same time taking into consideration the specific characteristics and restrictions of building automation networks. Aside from the essential message encryption and authentication mechanisms, this protocol will also need to provide some additional functions, such as key-exchange and key-revocation capabilities. The development will be guided based on a number of assumptions regarding the nature and topology of these networks, as well as potential threats they are faced with. The scope of the solution will be confined to the building automation domain, as the assumptions that the approach will be based on might not hold true for other application domains in the Internet of Things.
In particular, the goal is to develop a protocol that is able to withstand the most prevalent attack types against building automation systems, as well as provide an energy-efficient solution which offers acceptable performance and longevity even when running on constrained sensor nodes, while not compromising important factors such as extensibility.

1.4 Methodological Approach

Initially, an extensive literature analysis is conducted to establish an overview of the problem domain, the current state of the art in building automation, as well as existing approaches for dealing with security issues in building automation networks. Alternatively, promising approaches for less specific problems can be investigated, such as wireless sensor networks or the Internet of Things in general. While some of these approaches may not be directly applicable to the building automation domain, they could provide some fundamentals and valuable insight into related problems and partial solutions.
Next, the primary threats involved in running a building automation network have to be identified. This will allow the security requirements necessary for developing a robust protocol suitable for deployment in a building automation environment to be established. Based on these requirements, existing technologies will be evaluated for their applicability to the problem domain. Since it is unlikely that a single solution that covers all the given requirements already exists, different approaches will probably have to be combined and extended to fulfill the stated design goals.
After this analysis, the design and a first version of the protocol will be presented. Suitable technologies identified in the previous step can be used as partial solutions to specific problems. Consequently, a prototype will be implemented in order to demonstrate the feasibility of the proposed approach on a constrained hardware platform. This prototype

will then be evaluated in the context of the stated design goals as well as in terms of performance.

1.5 Structure of the Work

In the next chapter, the basic concepts and functions of building automation systems are outlined, along with some of the technologies commonly used in such systems. Additionally, existing approaches are presented that deal with the security of building automation systems and the resulting challenges. Chapter 3 covers the core assumptions that the proposed solution is built upon, along with an attack analysis in the context of building automation. Subsequently, the final design requirements that guide the remaining work are defined. Chapter 4 describes the solution in detail, as well as some potential pitfalls and vulnerabilities that were identified during design, along with suggestions on how to deal with those. In Chapter 5, the implementation and hardware details are discussed and various performance evaluations are performed. Chapter 6 concludes this thesis and proposes some future work.

CHAPTER 2

State of the Art

This chapter is meant to provide an overview over the fundamentals of building automation and related security concepts in general. Initially, the basics of building automation systems will be explained as well as a number of involved technologies. This will be followed by a more detailed analysis of existing approaches typically taken in regards to security and cryptography in such systems.

2.1 Basic Concepts

2.1.1 Wireless Technologies in Building Automation Systems

Traditionally, building automation systems were built around wired networks. Individual components were usually connected to each other as well as the backbone through wired connections, such as Ethernet- or fieldbus-based networks or other proprietary technologies. Increasingly, wireless technologies have been finding their way into automation systems as part of the integration effort caused by the popularity of the IoT. These wireless devices are usually connected to the backbone network through the use of gateways and can often form individual mesh networks. In such a mesh network, all devices (or “nodes”) are able to fulfill routing functions to create a large, interconnected network of devices without the use of dedicated routing hardware.
When looking at automation systems at large, their individual components and layers can often be viewed as a pyramid. Figure 2.1.1 shows such a pyramid, with the management level at the top including the overall management functions as well as visualizations of output and operating parameters. Below that is that automation level, with various control and metering units, as well as access control mechanisms and possible data aggregation and evaluation devices. At the bottom of the pyramid lies the field level, consisting of sensors and actuators, as well as user-operated I/O devices such as light switches. Wireless technologies such as 6LoWPAN typically operate at the field level

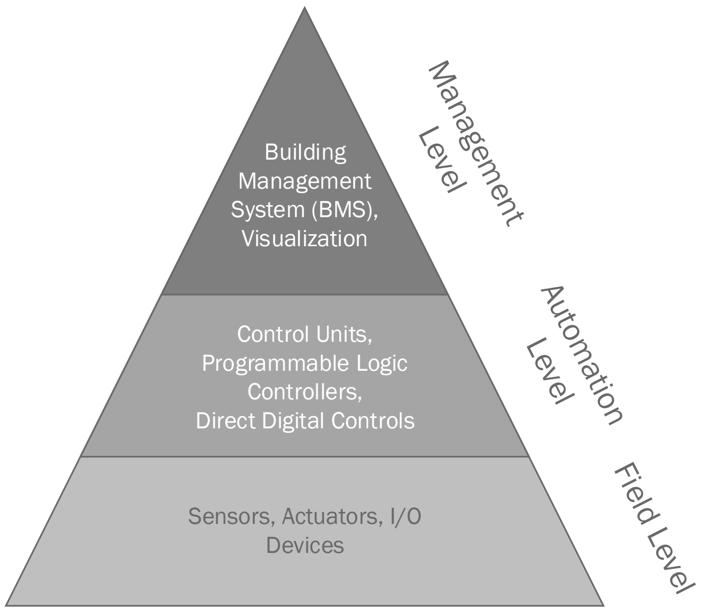
Building Management System (BMS), Visualization
Control Units, Programmable Logic Controllers, Direct Digital Controls
Sensors, Actuators, I/O Devices
Management Level
Automation Level
Field Level

Figure 2.1: The automation pyramid

too, often forming independent mesh networks and connected to the upper levels of the system through gateways and routers.

The wireless nodes in such systems are mostly small, cheap devices with limited computing capabilities, which are able to communicate with each other through wireless radios. They are spread throughout a building and fulfill various roles in the system, but they can typically be classified as either sensors or actuators. Sensors include data aggregation devices such as temperature sensors and light detectors as well as user-controlled input devices such as light switches. Actuators are mostly remote controllable devices such as lights, blinds or HVAC systems.
BAS nodes are usually based on some form of microcontrollers – constrained devices with very limited memory and low computing power. Due to the fact that sensor nodes are often located in places that are hard to access and maintain, some of them have to be powered by batteries. This is one of the most important limitations to keep in mind, as any software running on the devices must attempt to be as efficient as possible in order to reduce power consumption and maximize battery life. This is further exacerbated by the fact that radio transmitters, both when sending and receiving, tend to consume a lot of power.
For devices such as these, energy harvesting techniques can often be utilized to prolong their battery life [1]. However, especially in indoor environments the available options are rather limited, and, as a result, harvesting techniques typically employed in WSNs are mostly constrained to solar power (when available) and radio energy. Radio-frequency

(RF) based harvesting techniques usually rely on collecting ambient RF energy stemming from the constant broadcasts by countless radio transmitters in operation today (such as mobile phones, wireless routers or radio and television broadcasts) [2]. Alternatively, dedicated RF transceivers can be used to transmit energy directly, similar to wireless charging techniques.

Regardless of potentially used harvesting techniques, energy efficiency is still a primary constraint when designing any sort of WSN or BAS application. To enable an energy-efficient information exchange between nodes, low-rate wireless personal area networks (LR-WPANs) can be utilized. The IEEE 802.15.4 standard, defined in 2003, was designed to provide a low-complexity specification of the physical and media access control layers to be used by low-cost and low-power devices. The upper layers are not defined by IEEE 802.15.4, which allowed various extensions to be built on top of it, some of which enjoy successful widespread adoption in contemporary sensor networks (the most prominent among them probably being the ZigBee standard).

One such extension which will be important throughout this paper is the 6LoWPAN specification [3]. 6LoWPAN (IPv6 over Low power Wireless Personal Area Networks) was an attempt to extend the use of IPv6 to even small sensor devices. The Internet of Things (and subsequently also wireless sensor networks and building automation systems) with its large number of connected devices is particularly constrained by the lack of available address space offered by the aging IPv4 protocol [4]. To mitigate this problem, the adoption of its successor protocol IPv6, which offers a virtually unlimited address space, has been strongly pushed in IoT-circles.
And while IPv6 can alleviate some of the major problems that the IoT is faced with, supporting it within a WSN is not trivial. The IEEE 802.15.4 standard [5], which constitutes the basis on which most wireless communication in a WSN is built, is not inherently compatible with a comparatively complex protocol such as IPv6. In order to support IPv6 and allow the interconnection of WSN devices and the rest of the Internet, an adaptation layer is required. 6LoWPAN provides such an adaptation layer. It allows fully IPv6-conform packets to be translated and travel over IEEE 802.15.4 links, as well as providing 802.15.4 devices with directly addressable IPv6 addresses. To facilitate this translation one or more border routers that connect the 802.15.4 wireless network to the full IPv6 network are required. They are able to compress and decompress packets traveling in and out of the 6LoWPAN, as well as providing routing capabilities for messages inside the network. Since 802.15.4 mesh networks cannot support the routing protocols typically employed by IPv6 based networks, new protocols had to be developed [6][7][8].

The 802.15.4 standard as well as the 6LoWPAN form a common base upon which many contemporary solutions are being built. A number of high-level protocols and multi-layer frameworks exist on top of 6LoWPAN and 802.15.4. One example of such a framework is Thread [9], which is an IoT networking protocol primarily focused on home automation devices. While still relatively new, it is currently being pushed by numerous companies in an effort to become the de-facto industry standard in the field of home

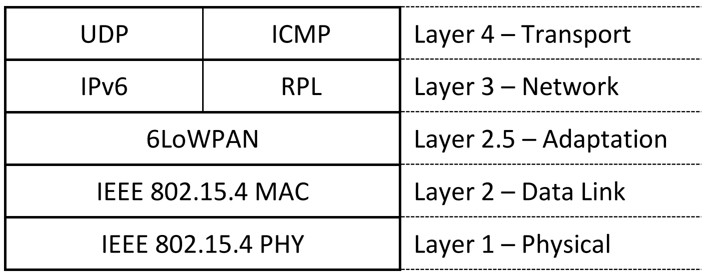

Figure 2.2: 6LoWPAN protocol stack

automation.
Another popular and well-documented example is ZigBee [10], which is more often found in industrial control settings and, while not utilizing 6LoWPAN, is also based on the 802.15.4 standard.

An important mode of message exchange in building automation networks that is typically less relevant in the Internet at large are broadcast messages. These can fulfill various functions in a WSN – global broadcasts can transmit commands to all nodes within a network, local broadcasts are typically used for routing functions or neighbor discovery, and multicast groups (such as those implemented in IPv6) can be used to address a specific subset of nodes. An example use-case for multicast groups would be a light switch that can remotely toggle a group of lamps. The lamp controllers could all be part of the same multicast group, and the switch would only have to send a single command to that group address in order to toggle all the lamps.
IEEE 802.15.4 does not inherently support message multicasting, but some efforts have been made to implement support for IPv6 multicasts over 6LoWPAN. Early implementations such as the Trickle Multicast [11], which was later incorporated into the Multicast Protocol for Low-Power and Lossy Networks [12], came with a number of significant drawbacks, namely high complexity which results in bad scalability, and low performance in general. Later attempts have tried to alleviate some of these problems by using different approaches. Stateless Multicast RPL Forwarding (SMRF) is a more lightweight mechanism and considerably outperforms the Trickle Multicast algorithm in both transmission delay as well as energy efficiency [13].

2.1.2 Physical Security

Since sensor nodes typically run unattended in an uncontrolled environment, they can be susceptible to physical compromises. Nodes located in public spaces can be accessed by potential attackers and tampered with. Once a node has become physically compromised, an attacker can be assumed to be able to take full control of the device by modifying its software or firmware. While tamper-proof hardware exists, it is typically not

widely deployed due to being expensive. Also, the efficiency of most tamper-protection mechanisms is questionable given a sufficiently motivated and equipped attacker [14]. So while heightened physical security and increased tamper-resistance can help to improve the overall security of a system, they should never be considered to be perfectly secure and as such aren't sufficient to provide a truly secure environment on their own.
A typical BAS can't be expected to provide even basic physical security, so the designed software solution should be able to continue operating to an acceptable degree even in the face of individual node compromises.

2.1.3 Information Security

The fact that many message exchanges in an IoT-enabled BAS happen over wireless connections makes them particularly susceptible to various forms of eavesdropping. As a result, securing the information exchange between nodes will be one of the primary goals of this thesis. Typically, this can be achieved by encrypting and authenticating the content of the messages (payload) that are exchanged between nodes. There are two basic families of cryptographic functions that are in widespread use when it comes to encryption – symmetric and asymmetric ones. The primary functional difference between the two lies in how the secret key is utilized. Symmetric cryptography requires the same key to be used when encrypting and decrypting information, while asymmetric cryptography uses separate keys for encryption and decryption, only one of which typically has to remain secret. This distinction is very important, as it restricts the way that the secret keys can be generated and exchanged.

Due to the long-standing necessity of securing sensible information on the Internet, efficient and functional systems have been developed in the past. Widespread and well-proven protocols such as Transport Layer Security (TLS) are able to provide confidentiality and authenticity for classical networks and devices [15]. Unfortunately, TLS itself was designed to be used with reliable, stream-based protocols such as TCP. As such, it is not well suited for use in a wireless sensor network, where message exchanges are often unreliable and based on connectionless protocols such as UDP. This led to the development of Datagram Transport Layer Security (DTLS), which will be explained in more detail below. On the cryptography side, TLS supports the majority of common algorithms and ciphers that are in use today. In practical terms, the TLS handshake most often uses asymmetric cryptography to establish a secure symmetric key, which is then used for encrypting the actual communication payload.

Unfortunately, adapting asymmetric cryptography for use in wireless sensor networks is rather impractical on current hardware. While offering many desirable properties, such as not requiring a secure channel to exchange the encryption key and providing message authentication capabilities through signed certificates, asymmetric cryptography suffers from performance issues especially on constrained devices [16]. Commonly employed asymmetric techniques such as RSA (named after its designers Rivest, Shamir and Adleman) [17] are considerably slower when compared to symmetric approaches such as AES (Advanced Encryption Standard) [18][19]. At the same time, they require a much larger key size to achieve the same level of security, which leads to increased memory

requirements and further exacerbates the performance issues on an already constrained platform. These issues currently make them mostly unfeasible for application in wireless sensor networks, as the high computational complexity causes bad performance and high energy consumption.
Symmetric cryptography on the other hand is very efficient and offers acceptable performance even on constrained devices. The drawback is that it requires the two communicating parties to first establish a shared secret before being able to securely communicate with each other. Finding a suitable key exchange process is one of the primary challenges when dealing with security on constrained devices.
In general, it can be said that it is currently not possible to design a universal protocol that is based solely on symmetric cryptography and suits all applications in the Internet of Things. It is however possible to find acceptable solutions for specific problem domains, as long as one is willing to accept some trade-offs and make certain assumptions that limit the problem space. Various approaches to handling specific security issues on constrained devices have been explored in literature, some of which will be outlined in more detail below.

2.2 Existing Approaches

2.2.1 Asymmetric Cryptography

As mentioned above, asymmetric or public-key cryptography systems (such as RSA and Elliptic Curve Cryptography, ECC) tend to be very slow and resource intensive. This is especially true when they are implemented in software and run on a microcontroller's CPU. RSA, which currently is the most widely adopted public-key cryptosystem (PKC), requires key sizes of at least 1024 bit and subsequently suffers from poor performance on memory constrained devices. Elliptic Curve Cryptography (ECC) is an attempt to alleviate some of RSA's problems by employing considerably smaller keys and thus resulting in a much better performance across the board (a 256-bit ECC key is roughly equivalent to a symmetric 128-bit AES implementation, where RSA would require a 3072-bit key to achieve the same level of security) [20]. When compared at a 128-bit security level, private key operations of ECC are reported to be 10 times faster than RSA [21]. This performance difference is even more pronounced at larger key sizes. ECC is increasingly being adopted into industry standards and it looks promising that it will eventually be able to replace most of the less efficient RSA implementations. Using ECC on constrained hardware is possible, although the performance of software implementations is still subpar when compared to symmetric approaches.

One possible solution to this would be to use dedicated cryptographic co-processors, which are able to perform commonly used operations such as those required by RSA and ECC with a much higher performance and lower energy consumption. Unfortunately, they are not yet widely adopted in the domain of building automation. The primary reasons behind this are likely financial constraints, as using more complex devices with integrated co-processors is considerably more expensive than using simpler hardware. So while approaches utilizing asymmetric cryptography might not be very feasible today, they are likely to become more appealing as devices with integrated cryptographic co-processors become cheaper and more ubiquitous.

To bridge this gap, Pecho, Nagy and Hanáček propose using smartcards (based on the ISO 7816 standard) as a replacement for dedicated cryptography hardware [22]. Modern smartcards offer the advantage of supporting not only popular encryption systems, such as RSA-2048 and occasionally even ECC, but also a secure element for key storage. The authors tested and compared the performance of RSA cryptographic primitives on such a microcontroller/smartcard hybrid as opposed to a typical software implementation and were able to achieve a speedup and reduction in power consumption by a factor of up to 30 when using RSA-1024 and a factor of 70 to 90 when using RSA-2048. These results seem very promising, and while installing a smartcard in every sensor node might not be practical or particularly cost-efficient either, it goes to show that by using dedicated hardware it is possible to considerably reduce the power consumption of PKC, increase the battery life to acceptable levels as well as achieve a high degree of security at the same time. While the authors mention ECC as an alternative, they limit themselves to RSA due to the lack of ECC support in most of today's smartcards. It stands to reason that as ECC becomes more prevalent, even higher time and power savings could be achieved, as the smaller key sizes used by ECC allow for faster implementations compared to similarly secure RSA keys.

Another approach at offloading computationally intensive operations to more powerful hardware was suggested by Huang et al. [23]. They propose to execute most of the expensive cryptographic operations, such as public key operations, on fully powered devices instead of doing so on the constrained sensor nodes themselves. This approach does not directly solve the issues that sensor networks are typically faced with, but it could be useful in certain scenarios where public key operations are less common and the infrastructure of the network allows nodes to offload specific tasks to more powerful devices.

2.2.2 Symmetric Cryptography and Key Exchange

Having established asymmetric cryptography as currently not feasible, the main focus will be on symmetric encryption systems. Even software implementations offer high performance on constrained devices, and hardware implementations of common algorithms such as AES are much more widespread than their asymmetric counterparts. The primary challenge when utilizing symmetric cryptography in a BAS context lies in securely exchanging the encryption keys. Since both communicating parties must share the same secret key and public-key cryptography cannot be relied upon to establish it, a different approach has to be taken. One advantage that can be leveraged in a BAS is the fact that every node has to be configured at some point during its deployment. During this node configuration phase, it is trivial to generate keying material and include it in the configuration. Keys deployed in such a manner are referred to as pre-loaded or pre-shared keys (PSK).

In an attempt to secure the traffic traveling through a WSN, a few basic key exchange schemes have been identified in literature as well as multiple more complex ones [24]. The basic schemes will be briefly described, along with their disadvantages and vulnerabilities.

- Network-wide shared key
  In this scheme, there exists a single secret key that is shared between all nodes. Each node is able to communicate with every other node using this key. At the same time, each node is able to decipher the traffic of any other node, since they all use the same key. While this scheme is very efficient in terms of memory consumption due to requiring only a single key, the obvious drawback is security.
  An attacker would only need to compromise any one node in order to extract the global key. He would then be able to not only decrypt all traffic exchanged on the network, but also to inject arbitrary messages into the network, potentially impersonating selected nodes.

- Fully-pairwise keys
  On the other end of the spectrum, there is the fully-pairwise key scheme, where each pair of nodes in the network has its own, unique secret key. This allows each node to communicate securely with every other node, while keeping the consequences of a node compromise to a minimum. A compromise would only reveal the pairwise keys of the compromised node, allowing the attacker to decipher any traffic traveling to and from this particular node. Presumably, this is information that the attacker is already able to obtain through having compromised the node in the first place, so there is no additional security risk involved.
  The obvious disadvantage of this scheme is the bad scalability of the network. The fact that each node has to save a pairwise key for every other node means that this scheme is not feasible in a large network consisting of devices with very limited memory. Even for smaller networks the scheme is impractical, as adding new nodes to the network would require all existing nodes to be updated using the new nodes' keys in order to be able to communicate with them.

- Random pairwise keys
  To find a middle ground between using a global key and a fully pairwise scheme, Eschenauer and Gligor [25] proposed using a probabilistic approach, where each node stores only a limited number of randomly chosen keys, taken out of a key pool. Two neighboring nodes that share a common key are able to communicate securely with each other. Given a sufficiently dense network, the likelihood of the network being fully connected is very high. For example, in a network of 10000 nodes, where each node has an expected number of 40 neighbors, every node has to store 250 keys out of a pool of 100000 in order to have a 0.99999 probability of the network being fully connected [25]. The authors propose that two nodes that wish to communicate with each other can attempt to establish a pairwise key through the secure link formed by a sequence of nodes using the randomly distributed keys.
  The main disadvantages of this scheme lie in its inherent randomness and its unsuit-

ability to sparsely connected networks, where the probability of link establishment is low.

The above three schemes are rather basic approaches for exchanging symmetric keys in WSNs. Neither of them is universally applicable to WSNs at large or even particularly useful in practice, since their drawbacks either prevent efficient use (fully pairwise and random schemes) or present a security liability (network wide key). However, there exist some advanced approaches that build upon these basic schemes but provide additional flexibility, and as such, are better suited to WSNs in general.

6LoWPAN Security

When communicating over a 6LoWPAN, security mechanisms of the lower layers can be used as a means to provide confidentiality and authenticity to the network. In particular, the MAC layer of the IEEE 802.15.4 standard specifies a number of security modes which support encryption, freshness checks and some simple access controls. Due to 802.15.4 being such a low-level protocol however, some features had to be left unspecified due to being reliant on the implementation of the upper layers. As a result, key exchange and revocation mechanisms aren't specified and must be implemented manually based on their particular operating environment.

Krentz, Rafiee and Meinel describe such a security sublayer for 6LoWPAN which provides an adaptable pairwise key-exchange scheme (APKES) [26]. The key establishment is based upon a plug-in system which supports various key exchange schemes. Depending on the specific requirements, an appropriate one can be selected, which is then extended using additional security measures to prevent flooding and replay attacks. The supported schemes include the above mentioned random pairwise keys scheme, Blom's Scheme (which attempts to derive keys from a small amount of secret data) [27], and the Localized Encryption and Authentication Protocol (LEAP) [28], which will be described in more detail below. Using the established pairwise keys it is then possible to use hop-to-hop encryption and authentication to secure the data traveling over the network. In addition to this, they also propose the easy broadcast encryption and authentication protocol (EBEAP), which can be used to secure local broadcast frames. Due to the energy and performance drawbacks of public-key cryptography the authors opted to use exclusively symmetric encryption in the form of CCM* mode for AES to provide authentication and confidentiality in their implementation.

An attempt at integrating existing solutions and providing end-to-end security to WSNs was made by Raza et al. [29]. They designed a compressed lightweight implementation of IPsec for 6LoWPAN, which is typically used by the IP protocol stack to provide authentication and encryption between communicating entities. Their implementation enables the communication between IP enabled sensor networks and the traditional Internet. This approach seems very promising due to the tried and tested nature of the underlying technology. Unfortunately, typical key exchange mechanisms employed in IP networks are based on IKE (Internet Key Exchange) and are yet to be tested for feasibility in 6LoWPAN-based networks. As a result, Raza et al. limit their current implementation to the use of pre-shared keys.

A topic that is often left out or unspecified in many security implementations is that of broadcast encryption. Securing broadcast or multicast frames is an important feature for some applications in the BAS domain. An example for such an application could be a control element that supports multiple distributed devices, such as a group of remotely controllable lights.

Research into broadcast encryption has been undertaken with various intentions and using different approaches. Fiat and Naor formally defined broadcast encryption for the first time in 1994 [30]. Boneh and Franklin devised a simple identity-based encryption scheme (IBE) [31] which is able to derive encryption keys based on a node's identity. This scheme has then been extended by Baek et al. in order to improve computational efficiency by reducing the number of required computations [32]. Other groups have conducted research into increasing the performance of these schemes to make them more suitable for wireless sensor networks and resource constrained devices in particular [33][34].

LEAP+

In order to deal with the varying security requirements posed by different message exchange schemes (for example: broadcast vs. end-to-end) Zhu et al. argue that a single keying mechanism is not sufficient [28]. They propose LEAP+, a combined approach which uses four different types of keys for every sensor node, each of which fulfills a different purpose. They also describe the key establishment and exchange as well as a re-keying mechanism to deal with compromised nodes. Their approach seems promising, since they accept the fact that no single solution is sufficient to deal with all the requirements. Instead, they attempt to combine partial solutions to solve the individual sub-problems. The design of LEAP+ is based on a number of assumptions and restrictions that are a good fit for the building automation domain, such as limiting itself to symmetric cryptography which is more resource-friendly than public-key based systems. Since the key establishment scheme of LEAP+ plays a role in the proposed prototype implementation, its structure is described in more detail below.
Networks in LEAP+ consists of a single base station and an arbitrary number of sensor nodes. To be able to handle the different communication patterns, all nodes in LEAP+ contain the following four key types:

- Individual key: Every node has its own unique key which is shared only with the base station. It is used for securing communications between the node and the base station, such as encrypting and authenticating sensor readings. Conversely, the base station can use this key to encrypt sensitive information, such as new keying material, when transmitting it to the sensor node.
  This key is generated based on the base station's master key and the node's ID, and is pre-loaded into the node before deployment. To save space, the base station only needs to store the master key and can generate the individual node keys on

demand based on the nodes' IDs. This way the base station doesn't have to save all the individual node keys permanently.

- Global key: A single, network-wide shared key which is primarily used by the base station for broadcasting messages to the entire network, such as commands or queries. Using a global key in this scenario is not a disadvantage from a security standpoint, as a compromised node which reveals the global key would reveal the broadcast traffic regardless of the encryption scheme in use. What is necessary however, is an efficient re-keying mechanism to update the global key in case a compromised node is detected and revoked in order to prevent future broadcast messages from being compromised.
  Much like the individual keys, the global key can be easily established by pre-sharing. Node revocation is realized using the µTESLA protocol for broadcast authentication [35] and key updates use a secure distribution protocol based on cluster keys (see below).

- Pairwise key: A node shares a pairwise key with each of its immediate neighbors. These keys can be used for general communications that are routed through multiple nodes and require privacy or source authentication, such as sensor readings or control commands. Message contents are always encrypted during individual hops by using the pairwise keys of the communicating nodes. Note that message contents are visible in the clear to the relaying node in between two hops. This makes LEAP alone insufficient for securing message transfers over multiple hops and requires the addition of an end-to-end security solution.
  The authors of LEAP present two schemes for establishing these pairwise keys: A basic one, which rests on the assumption that a newly deployed node will not be physically compromised within seconds of its deployment. Therein a node broadcasts a HELLO message to its neighborhood, and for every ACK reply received it generates a pairwise key for that neighbor. These keys are based on a pre-loaded initial key, which is erased from a node after the pairwise keys have been established. That way, if a node becomes compromised after the initial setup interval has expired, the attacker can only access the established pairwise keys. He cannot infer or generate any further keys since the initial key used for generating them has already been erased. The scheme also allows adding new nodes after the initial deployment, as long as the initial key is stored in a secure place and can be loaded into a newly deployed node. The author also describes an extended key establishment scheme, which is capable of dealing with more powerful attacks (such as an attacker obtaining the initial key by either compromising a node within the assumed "secure" time limit or by attacking the key authority).

- Cluster Key: This key is shared by a node and its direct neighbors and is used for local broadcast messages such as routing control information. Each node holds its own unique cluster key, as well as the cluster keys of all its neighbors. The neighbors can then use a node's key to decrypt or verify its messages, and use their

own cluster keys for encrypting messages.
Cluster key generation directly uses the previously established pairwise keys. A node generates a new cluster key for its neighborhood, encrypts it with the pairwise key of each neighbor and then transmits the encrypted keys to its neighbors. If a compromised node is revoked, its neighbors generate new cluster keys and re-establish them the same way.

While these key mechanisms cover a number of typical building automation use-cases, the system has been designed with general sensor networks in mind. As such, there are some shortcomings when trying to adapt it for use in a building automation network, in particular the lack of end-to-end encryption between two nodes which are not in a direct neighborhood relation. In order to comply with the design requirements of a BAS, some adjustments and extensions to LEAP+ would be necessary.

DTLS

Datagram Transport Layer Security (DTLS) [36] is an adaptation of TLS for use with datagram-based network protocols such as UDP. Analogous to TLS it offers various key exchange mechanisms and message encryption and authentication options. DTLS' reliance on asymmetric cryptography during key exchanges currently makes it unsuitable as a stand-alone solution in a building automation context due to the aforementioned performance concerns. It does however offer support for a key exchange scheme based on pre-shared keys (PSKs) and identities. While PSKs are impractical to use in open networks such as the Internet, they can represent an efficient way of establishing keys in an environment with somewhat limited network complexity, such as a BAS.
In such a context, it is possible to use DTLS-PSK as a useful building block for a bigger solution. This could be achieved by combining the tried and tested secure message exchange and handshake protocols offered by DTLS with an external key establishment protocol or even just pre-loaded keys. DTLS, being based on TLS, also offers the upside of continued development and support into the future. This fact could prove advantageous in terms of extensibility and maintainability. If constrained devices eventually become powerful enough, the established DTLS framework should allow for an easy transition from PSK-based solutions to more flexible approaches such as ECC.

Additionally, efforts are being undertaken towards adapting DTLS for use with multicast group messages. Nikitin et al. have proposed a DTLS modification [37] that allows secure two-way group communication built on top of the CoAP protocol. Their work was then further expanded as an IETF Internet-Draft [38], which could provide a solid starting point for implementing a workable multicast security solution.

Key exchange scheme	Cryptography	Performance	End-to-end encryption	Compromise resilience	Scalability
Global key	Symmetric	High	No	Very low	Good
Fully pairwise keys	Symmetric	High	Yes	High	Bad
Random pairwise keys	Symmetric	High	No	High	Variable[1]
LEAP+	Symmetric	High	No	High	Good
Centralized key exchange	Symmetric	High	Yes	High[2]	Good
Asymmetric schemes	Asymmetric	Low	Yes	High	Good

Table 2.1: Comparison between various key exchange schemes

[1] Scalability depends largely on network density. Sparse networks can end up not being fully connected

[2] Compromise resilience of individual nodes is good, but the central server can pose a single point of failure

CHAPTER 3

Methodology

In the Internet of Things, varying application domains are faced with different requirements and threats. Considering this fact is crucial for developing a suitable solution for a given problem, since there doesn't exist a single ideal solution covering all the use-cases and application domains.
Conducting an analysis to identify the technical and security requirements of the building automation domain will be the first step toward finding an acceptable solution.

The usage of the term building automation will be limited to systems which manage a building's electrical and mechanical equipment. Traditionally, this included HVAC (heating, ventilation and air conditioning) systems as well as lighting and power management. Today, security-critical services (such as fire detectors) are increasingly being integrated into building automation systems. Throughout this work, the term building automation system (BAS) is used to reference these kinds of devices as well as their underlying infrastructure.
As already mentioned, the primary goals of using a building automation system are to reduce the overall energy consumption of a building (for example by automatically turning off lighting or HVAC systems in currently uninhabited areas) as well as increasing the comfort of its inhabitants. Secondary uses can include managing security systems, such as fire alarms and access controls. These two groups represent the usecases that will be primarily considered going forward, and any threats will be examined based on their adverse impact on these application areas. This chapter will not take into consideration the impact on more general-purpose connected devices, such as those typically found in home automation or the Internet of Things in general.

3.1 Network Assumptions

3.1.1 Components

When considering a building automation system, a typical model is assumed that is based on a wired backbone network that houses most critical functions. This network can be supported by a wireless mesh network, which is connected to the backbone through a number of gateways or routers. Since the focus of this thesis lies on the wireless components, the wired backbone network will be largely disregarded in terms of functionality. The wireless mesh network (Figure 3.1) typically consists of the following components:

- Sensor nodes and input devices: Sensor nodes function as the data aggregation devices in a building automation system. They automatically relay their sensor readings to either the controller or to other nodes. Other input devices, such as switches, control panels or alarms can be manually interacted with by building inhabitants or personnel. They generate control signals which, similarly to sensor readings, are relayed either to the controller or to other nodes.
  In a mesh-based wireless network, they also fulfill the secondary role of providing routing capabilities when no dedicated access point is nearby. In terms of computing power, these are usually constrained devices with little resources. While most input devices will generally have a constant power supply, sensor nodes can often operate solely on batteries.

- Actuators: These react to control signals originating either from other nodes or the controller. They can cover a variety of usually physical functions, such as lighting, HVAC or door controls. Similar to sensor nodes, actuators contain only limited computational capabilities. However, due to the mechanical nature of their attached devices, they tend to be connected to a constant power supply in some way, making restrictions caused by batteries less of a concern.

- Controller: In this model, the existence of a single base station or building controller is assumed. It will usually be located on the backbone network and fulfills the role of the central controlling entity of a building, the aggregation point for all sensor data, as well as the key-management authority. In practice, these roles can be split between various devices, but for the sake of simplicity a single entity is assumed.
  Unlike sensor nodes and actuators, the controller is not computationally limited but represents a server-class device with powerful computing and storage capabilities, as well as a constant power supply.

- Routing Hardware: This class includes gateways, wireless routers, and access points. They form the bridges between the controller (which sits on the backbone network) and the wireless, 6LoWPAN-based components.
  In this model they operate as dumb devices which only fulfill routing functions and serve to connect the sensor nodes to the controller.

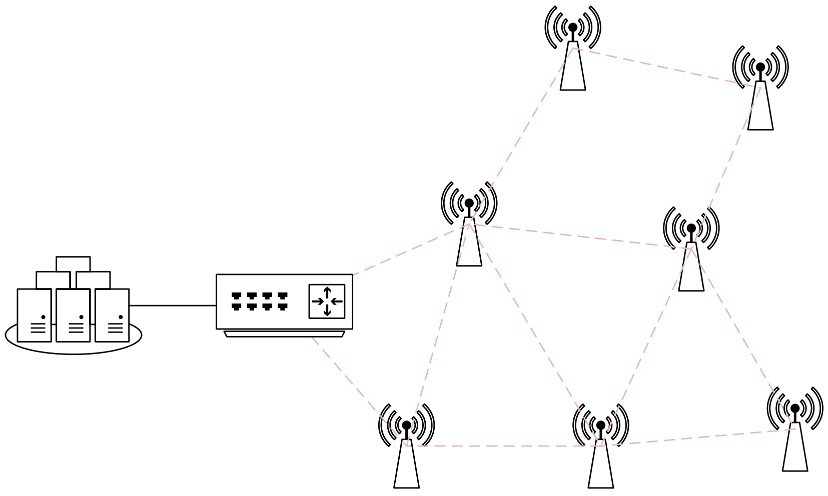

Figure 3.1: Sample image of a mesh network, consisting of a multiple sensor nodes connected to the backbone network through a gateway.

Due to the common nature of sensor nodes and actuators as constrained devices, they are considered as the same class of device. While they fulfill different practical functions, this distinction is not relevant when developing a communication protocol. For the remainder of this thesis, whenever *nodes* or *sensor nodes* are referred to, those considerations equally apply to actuator devices.
Furthermore, it is assumed that there are no mobile nodes in a building automation network. In other words, after being deployed, a node doesn't change its physical location anymore. This restriction is important since it enables the use of simpler, more efficient routing and key exchange strategies.

3.1.2 Message Exchange Schemes

In addition to the network components, it is also important to consider the types of messages being exchanged in a network. During operation, building automation networks generally have a limited number of relevant communication modes. The following modes will be primarily considered during development, as they represent the most common and important use-cases:

- Node-to-controller (and vice versa): This communication mode is in effect whenever a node sends data directly to the controller, such as sensor readings or user inputs. Conversely, the controller can also send messages to specific nodes. Examples for this could be control commands or security-related data such as new cryptographic material. This is likely to be the most common message exchange pattern in a functioning BAS, and as such requires an efficient implementation.

- Node-to-node: In some networks, nodes have the capability of communicating directly with each other. This can be useful to realize simple sensor/actuator relationships without having to route control information through the controller, such as when operating a remote light switch. Depending on the nature of the nodes, this message exchange pattern can also occur very often in a BAS and requires an efficient implementation.

- Local broadcasts: These are usually broadcasts by a node to its immediate neighborhood and are often used for relaying or gathering routing control information, as well as for neighbor discovery during node deployment.

- Group multicasts: Nodes have the ability to join multicast groups and send messages to all members of a group. In practice, this can be used to control multiple actuators that belong together (such as all the lights in a room) using only a single multicast message.

3.2 Security Assumptions

When determining possible threats and attack vectors, a number of assumptions about the components of a BAS have to be made:

I Sensor devices are publicly accessible. While this does not necessarily hold true for all of the devices, at least some number of them will generally be in openly accessible places, which allows a potential attacker direct physical access to them.

II Sensor devices are not tamper-proof. After an attacker has obtained physical access to a device, they are able to modify both the soft- and even hardware of said device.

III The medium air is publicly accessible, meaning that an attacker can freely listen in on the raw transmissions occurring over wireless links in their vicinity, as well as inject arbitrary frames into the network.

IV After deployment of a sensor node, there is a short time window where the node can be considered secure. In combination with assumption II this means that while an attacker is able to take over a node eventually, they are not able to do so immediately after deployment. The reason for this is that accessing as well as the process of compromising a node take a non-zero amount of time. This window can generally be assumed to be in the order of a few seconds to minutes.

3.3 Attack Motivation

Knowing the possible motivations of an attacker can be very helpful when trying to secure a system from intrusion. In this case, what does an attacker stand to gain from attacking and compromising a building automation system?

- Disrupting regular system operation: Based on various kinds of denial-of-service attacks, a malicious attacker can attempt to prevent the system from functioning correctly. This can take different forms, such as shutting down sensor or actuator devices or feeding malicious data into the system to prompt incorrect reactions, such as excessive heating or cooling. This in turn can manifest itself as decreased comfort of the building's inhabitants and increased operating costs.
  These attacks are typically purely destructive in nature by causing damage to the building operator, but gain the attacker little or no direct value. They are usually also the easiest to detect due to their immediate effects on observable factors.

- Breaching confidentiality: An attacker can attempt to read the data being transmitted over a BAS network. While building control data itself is generally not particularly sensitive or financially valuable, it can provide insight into the system's inner mechanisms, which in turn can help an attacker gain access to the system or building in a subsequent attack.
  Passive eavesdropping attacks can be among the hardest to detect, as they are non-intrusive and can often be conducted without the need to modify or tamper with any hardware or network traffic.

- Gaining unauthorized physical access: In some scenarios, attacks like this can be the most critical in terms of exploitability – if the BAS includes access control mechanisms such as door controls, compromising the system can gain an attacker illegitimate access to a building or to specific areas of a building. This access could then be used to further facilitate theft, espionage or sabotage.
  Preparations leading up to an attack like this can often be carried out without attracting much attention, making them hard to prevent. For this reason, special care should be taken when integrating physical access controls with a BAS.

- Gaining unauthorized access to an internal network: In a networked system where the BAS is connected to a building's local area or corporate network, vulnerabilities in the BAS can provide an attacker with an entry point to the rest of the network. This access can be further leveraged to facilitate espionage, deploy malware or conduct other malicious activities in a corporate network.
  The detection and prevention of attacks in this category is mostly reliant on having a solid network design and effective intrusion detection and prevention mechanisms for the internal network.

Cyber attacks against IT systems are usually conducted with a specific goal in mind. This goal typically involves gaining either physical access to a facility or access to networked resources in order to steal information or deploy additional malware that enables follow-up attacks. Attacks that are purely destructive in nature, such as increasing the power consumption of a system or depleting sensor batteries, without providing a financial or informational incentive for an attacker are usually less likely to occur.

3.4 Attack Analysis

Attacks against BAS can generally be separated into two groups – *network attacks* and *device attacks* [39].

Network attacks aim to access, modify or interrupt data while it is being transmitted. This is usually achieved by either accessing the network medium directly or by using an already compromised device to access its network interface. Common attacks in this category are listed below, as well as the primary measures for dealing with them:

- Sniffing: An attacker can eavesdrop on the network traffic that is being exchanged between two nodes. This occasionally happens by wiretapping wired networks, but is mostly relevant in wireless networks, where it is possible to listen in on network traffic with as little as a laptop with a wireless receiver. Any potential attacker must be assumed to be able to read and record the raw network packets being transmitted by nodes in their vicinity. To prevent eavesdropping and breach of confidentiality, an encrypted communication channel can be employed, which allows two endpoints to communicate securely with each other. Most encryption schemes require the two endpoints to first establish a shared secret key somehow. While robust key exchange schemes for classical networks already exist, finding a suitable scheme for networks of constrained devices will be one of the main challenges of this thesis.

- Replay attacks: After listening in on the network traffic, an attacker can save the raw network packets sent by a legitimate device and re-send (replay) them at a later point in time, hoping to trigger a reaction from the system. For instance, by re-sending the command to open an electronically locked door, an attacker could attempt to gain access to it. This attack even works with encrypted packets, as the attacker doesn't need to know the decrypted content of the message when replaying it.
  Preventing replay attacks is fairly straightforward – a simple way to do so is to add a sequence number to every message, and accepting only messages with increasing sequence numbers. An old, replayed message would have a lower sequence number, and hence it would be rejected by the receiver. This typically requires the sequence number to be encrypted and authenticated along with the contents of the message to prevent an attacker from modifying the number.

- Modifying messages: After first obtaining legitimate network packets, an attacker can attempt to modify and then feed them back into the network in order to force a specific reaction from the system. This is possible if the network uses a weak

message encryption or authentication scheme.
To prevent an attacker from being able to read or modify a message, a node can encrypt its outgoing messages using a robust encryption scheme and append a message authentication code (MAC) to ensure that the message can not be tampered with.

- Denial of service: The term denial of service includes all attacks that disrupt the correct functioning of a system. Typically, this includes flooding a part of the network with a high volume of messages in a short time frame in order to cause the system to start discarding messages (including legitimate ones) by taking up all the available bandwidth. Depending on the computing power of the hardware being targeted, denial of service attacks could traditionally require a lot of resources. Unfortunately, constrained devices are especially vulnerable to this kind of attack as they are not only very limited in terms of computing power, but can also offer additional restrictions such as limited battery life. An attacker could attempt to repeatedly connect to a device or cause unnecessary workload in order to quickly deplete its battery or render it inaccessible.
  Various detection and mitigation strategies have been developed over time to deal with classical denial of service attacks, but dealing with battery-draining attacks could prove more challenging. In general though, performing expensive operations on demand on battery-powered devices should be avoided as long as the requesting device is not fully trusted.

The other group of attacks are device attacks, where an attacker attempts to gain access to the functions of a device or to take over control of the device completely. These attacks are generally either enabled by remotely exploiting faults in the device software or by physically accessing the device and manipulating it directly. While the possibility of these device attacks creates certain requirements for the development of a secure communication system, preventing them altogether is an unrealistic prospect. Rather, the aim should be to develop a robust protocol which mitigates the impact of individual node compromises. The following are some attacks that can result from an attacker being able to compromise a device:

- Compromise of keying material: An attacker that has physically compromised a device will generally have access to all stored data, including encryption keys and certificates. Depending on the encryption scheme in use, this can pose a threat to the network or localized parts of it, as the confidentiality of communications might no longer be guaranteed.
  To mitigate the impact of such an attack, a good keying mechanism is necessary. In the event of a breach where keying material of individual nodes is leaked, the network traffic of unaffected nodes shouldn't be compromised. Additionally, a mechanism for revoking and issuing new keys is helpful to further reduce the consequences of a node compromise.

- Feeding malicious data: If an attacker manages to gain full control over a sensor device, they might be able to provide arbitrary data readings to the system. The consequences of such an attack largely depend on the kind of data that is being manipulated. Taking temperature data as an example, such an attack could lead to the air conditioning system behaving incorrectly, excessively heating or cooling the room, reducing comfort while increasing operating costs. Alternatively, manipulating routing control data in a network can lead to lost or delayed messages and can be considered a form of denial-of-service.
  Preventing a compromised node from sending false data is only possible after the compromise has been detected. Inconsistent data or statistical analysis can reveal a malfunctioning or misbehaving node, after which it can be blacklisted or have its keys revoked in order to prevent the system from reacting to the incorrect data.

- Providing access to the system: Compromising a sensor node gives an attacker direct access into the underlying system. Depending on the interfaces and authentication mechanisms used, a compromised node can provide access to many functions or services of the system that would be unnecessary during normal node operation. For example, an insecure interface or lacking access control mechanisms could allow an attacker to access actuators or control functions such as HVAC or even door controls from a simple sensor node.
  To prevent a compromised node from accessing arbitrary data or services, a robust access control mechanism is required.

In general, as long as a node does not misbehave, detecting a node compromise can be a difficult task. This makes it necessary to have authentication and access control mechanisms in place to reduce the impact of an undetected node compromise as much as possible. These mechanisms are typically implemented at the data and application layers and have to be customized individually for each system. As a result, describing and implementing access controls is out of scope for this thesis and won't be covered further.

3.5 Design Requirements

The definite aim of this paper is to design a secure communications protocol for use in building automation networks. This protocol should protect from network attacks and mitigate the impact of most device attacks as far as possible. Based on the above assumptions, threats, attacks and usecases, the primary design requirements for such a protocol have been identified. Existing technologies as well as the prototype implementation are evaluated according to their adherence to these requirements.

- Provide confidentiality and integrity: An attacker should not be able to read or decipher the contents of any messages exchanged on the network. Neither should they be able to alter the message without the receiver being able to detect that a modification has occurred.

- Survivability: Since the physical security of individual sensor nodes cannot be guaranteed, the network at large must be able to operate even in the case of node compromises. Individual undetected compromises should not break the confidentiality of the entire network by revealing sensitive information to an attacker.

- Energy efficiency: The limited battery life of sensor nodes requires the protocol design and implementation to be energy efficient and to avoid computationally expensive operations as much as possible.

- Extensibility: In the case of expanding building infrastructure, it should be possible to securely integrate new nodes into the network without having to re-configure existing ones.

In addition to the above properties, the implementation also has to fulfill the following two functional requirements:

- Provide an efficient key-exchange mechanism: In order to be able to provide the aforementioned confidentiality and integrity properties, a key-exchange mechanism is required. Functionally, two arbitrary nodes on the network must be able to securely establish a common cryptographic key, regardless of their physical location in the network. The exchange mechanism must be suitable for use on constrained hardware platforms while offering an acceptable performance.

- Provide a key-revocation mechanism: In the scenario of a node compromise, the system should be able to notify individual nodes about the compromise and revoke any affected key if necessary. This way the compromised node can be prevented from initiating or continuing communications with legitimate nodes.

CHAPTER 4 Solution

The primary goals of the desired security protocol are to provide confidentiality and authenticity for the communicating devices. The two most important message exchange patterns that have to be covered in a building automation system are usually direct communication between either two nodes, or a node and the control server.

In networked systems there is usually a multitude of approaches that can provide the desired security properties. Each of them comes with its own advantages and disadvantages, and they typically differ in what network layer they operate on. The most commonly found classifications are transport security and message security.

- Transport Security: Transport security mechanisms attempt to secure data at a low level, independently of the upper layers. That way the security mechanism doesn't interfere with application layer protocols and can be implemented transparently. Security options are usually limited by and dependent on the specific transport layer implementation, and can vary greatly based on the circumstances. As an example, 6LoWPAN offers a limited form of transport security by means of the 802.15.4 MAC security specified in the standard. More flexible approaches such as DTLS are becoming increasingly mature and are on track to be adopted in frameworks such as CoRE (Constrained RESTful Environments).
- Message Security: In this case, security is applied to individual messages themselves rather than the underlying connection as a whole. This way, encryption and authentication metadata is embedded in the message itself, making each message self-contained. Message security also decouples the security from the underlying transport protocol, effectively making it usable regardless of the specifics of the lower layers. Especially when implementing end-to-end security solutions, message security is often the preferred approach.

Since end-to-end encryption coupled with message authentication is one of the most straightforward ways of providing confidentiality and authenticity to a message exchange, message security was chosen as the preferred approach. Unfortunately, implementations that can be typically found in IP networks are generally not viable for sensor networks due to performance reasons. The goal is to provide a more lightweight approach that is limited to symmetric cryptography, which provides an acceptable performance even on constrained devices. Regardless of the chosen security scheme, the main challenge that needs to be solved when using a symmetric approach is the secure establishment of encryption keys.

4.1 Design Fundamentals

Due to the nature of building automation systems, the message exchange patterns remain predictable and mostly limited in scope. Specifically, not every node needs to be able to communicate with every other node. Most wireless nodes will only ever need to communicate with a control server (usually through means of a gateway) as well as a small number of other nodes that fulfill related functions. Since building automation systems typically don't require support for mobile nodes with changing locations, these communication patterns remain static and don't change throughout the lifetime of a building.

In practice, these limitations result in each node only needing to store the keys of its relevant communication partners. This is feasible even on constrained devices, since the number of required keys will generally be small, as opposed to a fully pairwise key scheme as described in Section 2.2.2. With that in mind, the primary challenge is how to exchange encryption keys between related nodes in a secure and efficient manner. There are two different cases that need to be covered: Establishing keys between a node and the controller, and establishing keys between two arbitrary nodes.

Establishing a pairwise key between node and controller

Solving this scenario is rather trivial. These key pairs (henceforth referred to as *individual keys*) are unique to each node and can simply be preloaded into a node's memory and added to the controller's keyring during the node deployment phase. While this means that the controller has to save the individual key of each node in order to be able to communicate with it, this shouldn't be an issue in practice, as symmetric keys are generally very small in size and the storage and memory of the controller are virtually unlimited. Should a node become compromised by an attacker, only its own individual key is revealed, while the individual keys of all other nodes remain secure.

Establishing a pairwise key between two arbitrary nodes

This scenario is considerably more challenging, since the system needs to be designed with extensibility in mind. This means that new nodes can be added to the network after the initial deployment phase. These new nodes should be able to exchange keys and communicate with previously deployed nodes. Loading the new keying material into existing nodes manually is not practical, which is why a different approach is suggested.

The idea is to use a trusted third party, the controller, to generate the secret keys for any node pairs (henceforth referred to as *partner nodes*, respectively *partner keys*). The already established encrypted channels between nodes and the controller can then be used to transfer the keying material securely to the two partner nodes. At a basic level, the key exchange works as follows:

A node wishing to establish a secure key with another node can simply send a request including the target's address to the controller. The controller will then generate a new key and send it, along with the two nodes' addresses, to both parties (see Figure 4.1). The nodes can now store this pairwise key and use it to communicate securely with each other. All message exchanges with the controller during this process are secured using the nodes' individual keys. The controller immediately discards any partner keys after generating them, so a potential compromise of the controller would not retroactively reveal any of the previously generated pairwise keys.

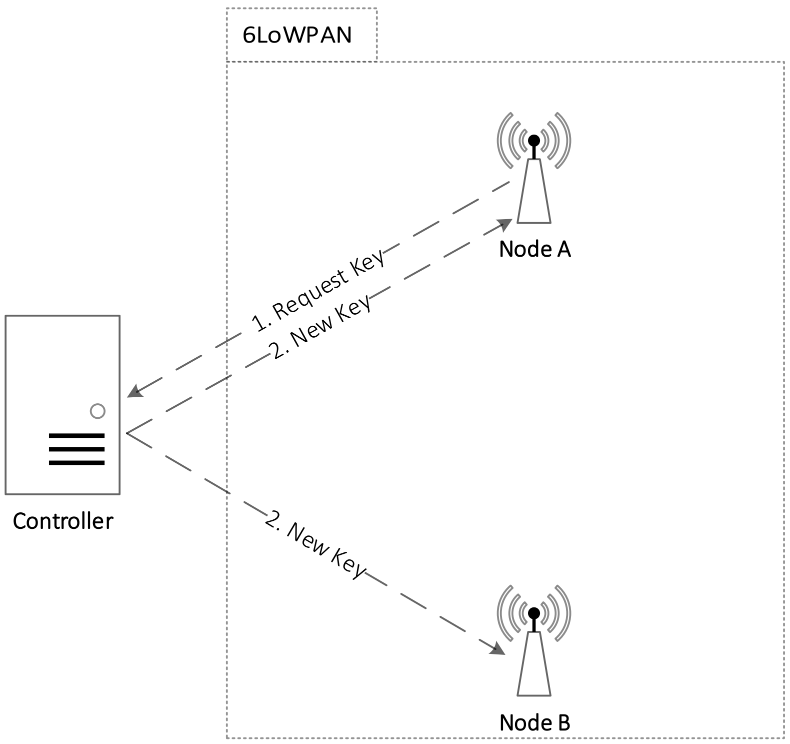

Figure 4.1: Schematic example of a basic key exchange

This key exchange design using a trusted intermediary was selected due to the nature of building automation systems. Most such systems already include some sort of central management server which supervises building activity, accumulates data, is sufficiently secure and by extension lends itself to managing keys as well. It also offers the additional advantage of being able to verify all key requests at a central entity, potentially filtering out bogus requests before they can consume the limited computing resources of the constrained nodes. Furthermore, it can be assumed that the communication patterns within building automation systems rarely, if ever, change – sensor/actuator relationships will remain mostly static throughout the lifetime of a building, with only occasional additions or repairs happening. New nodes typically need to be configured on a per-node basis anyway, making the pre-loading of keying material only a small additional effort. The combination of these properties makes a complex dynamic system, which supports mobile and frequently changing nodes, unnecessary in the building automation domain.

One disadvantage of the basic approach described above is that while keys generated in the past are not recoverable outside of compromising one of the two communicating nodes, nothing prevents the controller from issuing new keys and sending key update instructions to chosen nodes. This way, a compromised controller or leaked individual keys could allow an attacker to issue fabricated keys to arbitrary node pairs, thereby breaking the confidentiality of their exchanged messages. Alternatively, the attacker could disrupt communications between existing partner nodes by issuing both of them new, but differing keys, which would effectively render them unable to communicate with each other.
While a compromise of the control server represents a worst-case scenario in terms of a security breach, it is still possible to mitigate the effects of such an incident. In order to do so, a node will additionally attempt to verify a newly issued key with its partner before overwriting any previous keys. This way, the nodes can make sure that the key request was legitimate.
This verification process consists of a random challenge, encrypted using the new partner key. Upon receipt of the verification request, the partner node will reply with an *ACK* message including the challenge. After a final *RE-ACK* the verification process is closed and both nodes can save the new partner key for future communications (see Figure 4.2). If the partner hasn't actually requested a new key, it will not send the *ACK* message, causing the first node to discard the newly issued key after a specified timeout period.

Key Revocation

Secure key revocation can be realized using a method similar to the key exchange. The controller can send an encrypted key revocation request to any target node. This request has to include the address of the compromised node whose key is being revoked and is authenticated using the individual key of the target node. The target node can verify the authenticity of the revocation request by using its individual key, removing the compromised node and any associated keying material from its memory.

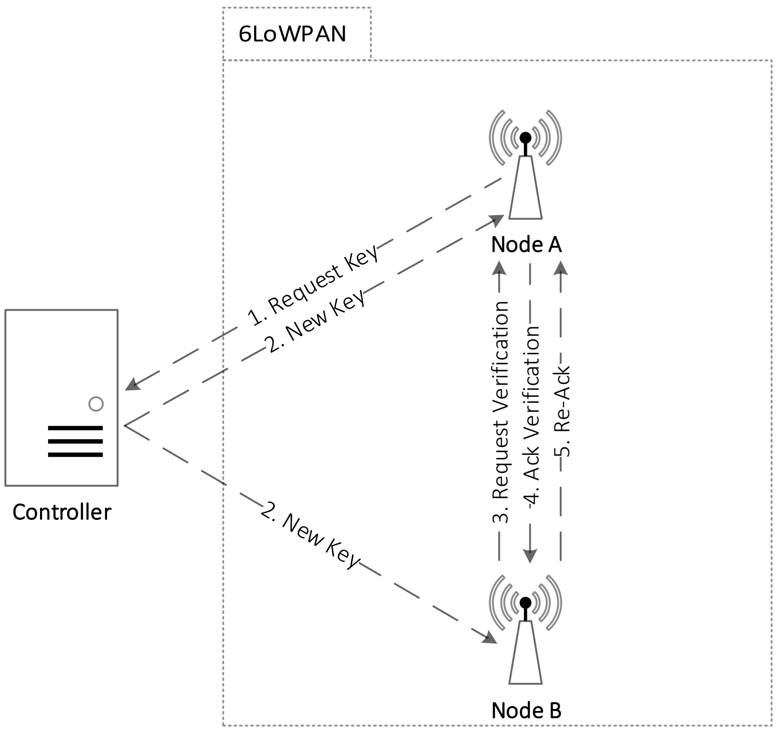

Figure 4.2: Schematic example of a key exchange with verification

4.2 Detailed Key Exchange Process

Let A and B be two nodes, where A wishes to establish a new partner key with B. A will start the key exchange procedure by sending a key request to the controller C. This key request is encrypted and authenticated using A's own, individual key ik_A, and contains B's node address, $addr_B$. Node A will also save this key request in order to be able to confirm it later.
The controller will then generate a new, random partner key $pk_{A,B}$. This key, along with the IP address of the other node, will be sent to both nodes.
These two messages are separately encrypted and authenticated using the individual keys of nodes A and B (ik_A and ik_B, respectively). This can be achieved using an authenticated encryption mechanism that provides both these features in a single pass. After sending out the key, the controller has to discard any sensitive information, in particular the key $pk_{A,B}$ itself, to prevent a potential future attacker from obtaining or deriving it.
Upon receiving the generated key, node B will wait for a short period of time and then send a verification request to node A, whose address was included in the key message. This verification request consists of a random challenge and is encrypted and authenticated

using $pk_{A,B}$.
At the same time, node A should have also received the key message and assigns the new key $pk_{A,B}$ to node B as a preliminary key. It will now be waiting for a key verification request from node B. Once it receives the verification request, it should be able to successfully decrypt it using $pk_{A,B}$ and obtain the challenge. At this point, if node A has legitimately requested a key exchange with the source of the verification request, it will respond with an ACK message including the challenge that is encrypted and authenticated using $pk_{A,B}$. If node A hasn't requested a key, or the key request has timed out in the meantime, it will discard the verification request.
Node B, upon receipt of the ACK, will decrypt it and compare the challenge in the ACK message to the original challenge. If the challenges match, the node can assume that the key request was legitimate and permanently store $pk_{A,B}$ along with node B's IP address for future communications. To finish the transfer, it will send a last *RE-ACK* message to node A, which, upon receipt, can now also save $pk_{A,B}$.
In summary, the protocol trace looks as follows:

$$\begin{aligned}
&A \rightarrow C : \{REQUEST\_KEY(addr_B)\}_{ik_A}\\
&C \rightarrow A : \{NEW\_KEY(pk_{A,B}, addr_B)\}_{ik_A}\\
&C \rightarrow B : \{NEW\_KEY(pk_{A,B}, addr_A)\}_{ik_B}\\
&B \rightarrow A : \{REQUEST\_VERIFY(challenge)\}_{pk_{A,B}}\\
&A \rightarrow B : \{ACK\_VERIFY(challenge)\}_{pk_{A,B}}\\
&B \rightarrow A : \{RE\_ACK\}_{pk_{A,B}}
\end{aligned}$$

Symbol	Meaning
A,B,...	nodes A,B,...
$\{...\}_k$	message whose payload is encrypted using the key k
$A \rightarrow B$	A sends a message to B
$addr_N$	IPv6 address of node N

Table 4.1: Notations

Using a random challenge in this exchange is necessary, otherwise a compromised controller could simply issue a bogus partner key to an arbitrary node, immediately followed by a forged ACK message to confirm it. Including the challenge will instead cause a node to discard the spoofed ACK since the response doesn't match the challenge.

4.2.1 Reliability of the Key Exchange

Owing to the fact that the entire key exchange happens over an unreliable communication medium, it is important that the protocol is robust and can handle lost or delayed packets. To achieve this, time-out periods can be utilized to make sure that node pairs don't end up in inconsistent states, where one of the nodes has saved a new key while its partner

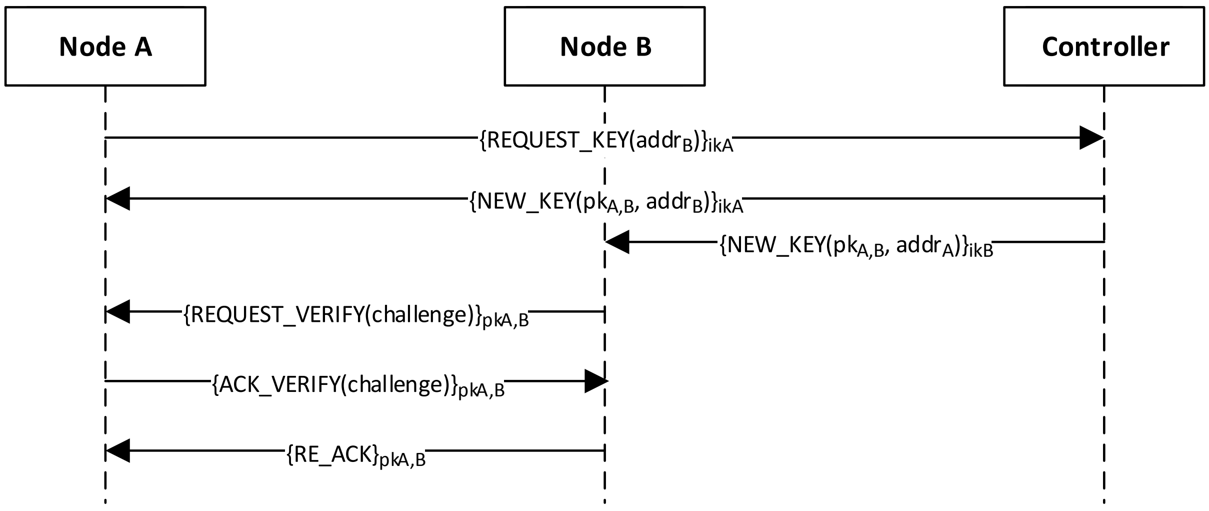
Node A
Node B
Controller
{REQUEST_KEY(addr$_B$)}$_{ikA}$
{NEW_KEY(pk$_{A,B}$, addr$_B$)}$_{ikA}$
{NEW_KEY(pk$_{A,B}$, addr$_A$)}$_{ikB}$
{REQUEST_VERIFY(challenge)}$_{pkA,B}$
{ACK_VERIFY(challenge)}$_{pkA,B}$
{RE_ACK}$_{pkA,B}$

Figure 4.3: Sequence diagram of the key exchange process

has discarded the key exchange. At the same time, the re-transmitting and re-requesting of messages should be kept to a minimum to reduce the possibility of a malicious node continually draining the power and resources of other nodes by repeatedly requesting key verification. In general, each time a node has completed a step of the key exchange, it will start a timer. Should this timer expire before the next phase of the exchange is complete, it will discard all previous progress and any received keying material. The nodes will not attempt to re-transmit any messages, with the exception of the final *RE-ACK*, the reason for which will be explained below.
This means that usually any lost messages will cause the key exchange to time out and be discarded. If that happens, the requesting node has no choice other than restarting the process and sending another key request to the controller.

The following scenarios describe in detail how packet loss is handled at particular points during the key establishment process:

- If the initial key request message gets lost, the requesting node A will simply time out and discard its key request. At this point, the target node B is still unaware of the exchange process, so it is unaffected by the lost packet.
- If the key request comes through to the controller, but one (or both) of the key notifications from the controller become lost, one of two things can happen:
  In case the message to node B or both messages are lost, B will once again be unaware of the exchange process and not react in any way. Node A will time out eventually while waiting for the verification request from B, discarding a potentially

received key.
Should the message to node A get lost, node B will receive the new key, triggering a key verification request, which will be discarded by A, as it has not received a new key yet. Node B will time out waiting for an ACK message, assume that the key request wasn't legitimate and discard the exchange.

- The above thing also happens when both nodes receive the key, but the verification request is lost. In this case node A will time out waiting for the verification request, while node B will time out waiting for an ACK message and assume that the request wasn't legitimate.

- If the verification request is successful but the ACK message is lost, node A will be waiting for the RE-ACK until its timer expires, while node B times out waiting for the ACK, at which point both nodes will discard the key exchange.

- A problematic situation arises when the ACK message is successful, but the RE-ACK is lost. What would happen then is that node B would store the key upon receiving the ACK, but node A, not having received the RE-ACK, would assume that the ACK has been lost somewhere and discard the exchange. This would result in a one-sided key establishment, effectively rendering the two nodes unable to communicate with each other.
  To reduce the probability of such an event occurring in practice, node A will attempt to re-transmit the ACK a specified number of times if it doesn't receive a RE-ACK immediately. Only after multiple failed attempts it will finally discard the exchange. While this doesn't entirely eliminate the possibility of a failed key exchange, it considerably reduces the likelihood of it happening due to a random link failure.

Optionally, a number of re-transmissions could be included on most of the message exchanges to further improve the overall reliability of the key establishment process at the expense of energy efficiency in the situation of illegitimate exchange or verification requests.

It should also be noted that in the proposed solution nodes are identified based on their IP address in the 6LoWPAN. This approach could be impractical in a real-world application, as nodes requesting new keys would need to know their partner's IP addresses. If node IP addresses in the network aren't assigned statically, figuring out specific node addresses could be non-trivial. Despite these concerns it was decided to use this IP-based scheme for the proof-of-concept due to its simplicity. A more reasonable approach for a real application would be to instead use some sort of node ID in the request. This ID would ideally have to be known during deployment time and assigned statically to all nodes. The controller could maintain a database of all node IDs and their corresponding addresses and would then be able to handle the request and forward it to the correct node. Regardless, the choice of a different addressing scheme doesn't introduce any additional security vulnerabilities, making it primarily an issue of practicality.

4.2.2 Key Exchange Vulnerabilities

Notably, the proposed scheme is still vulnerable to forged key notifications if the attacker controls a node or radio transceiver within radio range of one of the two partner nodes, in addition to having compromised the controller. In such a scenario, they can send a fabricated key notification from the controller and listen to the resulting verification request. They can then decrypt this message using the fabricated key, obtain the correct challenge value and reply with a spoofed ACK message, again encrypted using the fabricated key.
This attack can be mitigated by employing link-layer security, which ensures that all message exchanges between arbitrary nodes are always encrypted and verified during individual hops using additional neighborhood or cluster keys (as described in Section 2.2.2). That way an attacker can no longer obtain the contents of the verification request message by passively listening, since the packet is additionally encrypted during transmission using the pairwise keys of the nodes on the route between the two endpoints. To be able to access the contents of the message, the attacker would have to control a legitimate node on that route in order to intercept the request challenge between two hops and inject a spoofed ACK message into the exchange.
As a result, utilizing link layer security can considerably reduce the impact of a compromised controller, as it no longer represents a single point of failure in regards to the operation of the entire system. In order for an attacker to disrupt the secure communication between existing nodes, it would now be necessary to additionally gain control of a node capable of intercepting and modifying network traffic during the key verification process. In practice, this can be considered sufficiently secure, since a situation where an attacker is able to compromise not only the controller, but also arbitrary nodes in the network can already be considered a catastrophic security failure.

4.2.3 Key Revocation Vulnerabilities

As described above, a simple centralized key revocation scheme could be realized in a manner analogous to the key establishment process, that is by authenticating and verifying key revocation requests using nodes' individual keys. Unfortunately, implementing a key revocation scheme this way nullifies the advantages achieved by verifying new key requests between nodes. In particular, a compromised controller could simply send forged key revocation requests to arbitrary nodes, causing them to delete selected keys and being unable to securely communicate with the targeted nodes. This way, the controller would once again pose a single point of failure in the system. While this may be acceptable in some settings due to the low likelihood of a controller compromise, it would be desirable to circumvent this issue altogether, while still supporting efficient, centralized key revocation.

One possible way of achieving that goal would be to use a separate set of individual keys (unique to each node) for key revocations. Storing this key set offline and separate from the controller's other keying material would make it more resilient to a controller compromise. If a node compromise would be detected that required a revocation, the necessary revocation keys could be obtained from the offline storage and used to authenticate the revocation requests. Using an offline solution like this seems justifiable based on the fact that key revocations should represent a rare scenario that typically requires intervention from a system operator anyway. This way the additional overhead caused by having to access the offline key storage poses less of an issue in practice.

CHAPTER 5

Implementation

5.1 Architecture

To show the feasibility of the suggested approach, a sample system that demonstrates the core features was implemented. This prototype implementation is able to communicate through 6LoWPAN and its architecture consists of two primary parts:

- Demo-node: A small sensor node running a simple application in Contiki OS that sends periodic temperature readings over a 6LoWPAN-based network to the controller. It supports the designed key-exchange scheme as well as link-layer security in the form of APKES [26].

- Controller application: A Java application running on a desktop-class device which represents the building control server. It can send and receive UDP messages over an IPv6 link and supports key and node management.

In order for the IPv6 backbone to be able to communicate with the 6LoWPAN-based sensor network one additional component is required:

- Border-router: The border-router is essentially a bridge that connects the 6LoWPAN to the Ethernet-based network that houses the control server. It runs on the same hardware as other sensor nodes, but uses two network interfaces and is able to pass traffic from one network to the other. In practice, a border-router would be required at every point in the building where traffic passes from an IPv6-based backbone network to the 6LoWPAN or vice versa.
  Since devices on either side of the bridge can be addressed using their full IPv6 addresses, the border-router appears transparent to all actors, and, together with the 6LoWPAN adaptation layer, provides the impression of one interconnected IPv6 network.

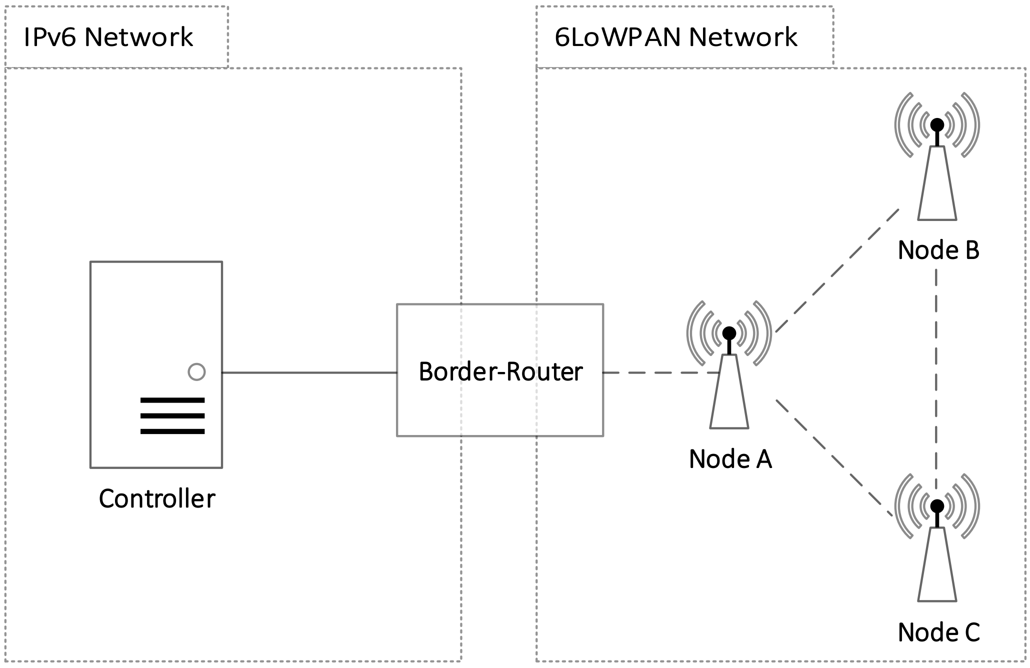

Figure 5.1: Schematic view of the network architecture and its components

Test setup

The test setup consists of a single controller and a number of demo nodes. The demo nodes can communicate with each other through a 6LoWPAN and are connected to the controller by the means of a border-router, which forwards IPv6 traffic originating at the controller to the 6LoWPAN-based node network and vice versa. The schematic view of an example test setup is visible in Figure 5.1.

5.2 Components

5.2.1 Software

The sensor node component of the prototype is implemented in Contiki OS [40]. Contiki is an open-source operating system specifically designed for low-power devices that supports most popular microcontrollers and hardware configurations. Additionally, it offers various development tools such as the Cooja network simulator, which was the primary tool used for testing and debugging during development. Cooja allows simulating 6LoWPAN-based networks consisting of arbitrary numbers of various node types (called motes) and device configurations while providing considerably higher testing throughput than running the applications on physical hardware. This is achieved by virtually eliminating any waiting periods present in physical hardware testing, such as the need for repeated flashing of new software versions onto the device EPROM each time changes are made to the program code. Naturally, after the prototype was implemented, the functionality of the solution was also tested on physical devices to show the feasibility of running it on real hardware.

5.2.2 Hardware

The node demo application was ultimately run and tested on Zolertia Z1 motes [41], which are based on the MSP430 16-bit microcontroller and specifically designed for use in sensor networks. They include an IEEE 802.15.4 compliant CC2420 radio transceiver [42] as well as an on-board temperature sensor and accelerometer.

5.2.3 Cryptography

To provide message encryption and authentication for the system, AES in CCM* mode [43] was chosen. CCM* is a slightly extended version of CCM mode (Counter with CBC-MAC) [44], which is an authenticated encryption algorithm that provides a combination of confidentiality as well as authentication. It uses 128-bit AES as its underlying cipher. Unlike pure CCM, CCM* also supports encryption-only in addition to authenticated encryption. When using authenticated mode, a MAC is generated over the plaintext and encrypted along with the message. During the message decryption process the received MAC is compared against the MAC generated from the decrypted message. Should they not match, the message is discarded.

CCM Fundamentals

CCM* is a counter-based cipher, which means that it requires not only a secret key, but also an initialization vector (IV, sometimes called nonce). To preserve the security properties of the cipher, it is important to never use an IV with any given key more than once. Notably, the IV does not have to be secret though, which means it can be prepended to the message as plaintext. A common way of preventing reuse of an IV is to use a sequential counter value, which is incremented by one after each message. Unless the counter reaches its maximum value and overflows, this ensures that an IV will never be reused.
CCM* uses a 13 byte IV, 8 of which represent the last 8 bytes of the link-local IP address of the source node. The last 5 bytes of the IV are the counter value, incremented by one after every message. A 5 byte counter allows for a maximum unsigned value of approximately 10^{12}, which represents the maximum number of messages that can be exchanged before the counter overflows. In case this value is ever reached, the counter has to be reset and a new key must be assigned to prevent the reuse of counter values with the previous key. One simple way of doing so is to apply a one-way function to the

old key to generate a new one. However, due to the high message cap and the resulting unlikelihood of an overflow in day-to-day operations (sending 1000 messages per second, which is a highly unrealistic assumption for a BAS, it would take over 31 years to reach the maximum value), a counter overflow handling wasn't implemented in the prototype solution.
The MACs generated by CCM* support variable length, but are typically set to 8 bytes. Combined with the 13 byte IV, this results in a total overhead of 21 bytes per encrypted message transfer.

The choice fell on CCM* primarily because there already exist efficient implementations for Contiki OS and because the combined nature of the authenticated encryption makes an additional message authentication mechanism unnecessary. That being said, any sufficiently secure symmetric cipher coupled with some form of message authentication should be suitable for use in this design.

5.2.4 Link-Layer Security

In addition to providing end-to-end encryption between nodes and/or the controller, it was also decided to incorporate link-layer security to increase the robustness of the solution to random node compromises. Contiki OS does not support link-layer security out of the box, but there exists a working implementation developed by Konrad-Felix Krentz [26]. While link-layer security alone doesn't provide all the desired properties, such as end-to-end encryption between individual nodes, it helps alleviate some of the issues identified in the system. Primarily, it offers hop-to-hop encryption based on an adaptable pairwise key establishment scheme (APKES). One of the supported schemes is based on LEAP, which has been described in detail in Section 2.2.2. It allows new nodes to establish secure keys with their immediate neighbors, as long as there is a short time frame after deployment during which the node can be considered secure.
An additional advantage of combining APKES with the proposed solution is that APKES prevents injection attacks by rejecting packets from unknown or unverified neighbors. This property greatly reduces the viability of some attacks, such as the key-invalidation attack described in Section 4.2.2, where an attacker could use a compromised controller or leaked keystore to disrupt already established key pairs.
Furthermore, the security layer also provides authentication and encryption of broadcast frames, based on the keys established through APKES. This is mostly useful for securing local broadcasts, such as those used by 6LoWPAN routing and neighborhood discovery protocols. Notably, the proposed end-to-end encryption scheme runs at the application layer, so it doesn't conflict with the security layer in any way. As a result, using APKES is optional, and in fact it could even be replaced with any other link-layer security implementation that provides similar properties. Further details regarding the exact functionality of the security layer implementation will be omitted at this point and instead, the interested reader is pointed to the associated papers by Krentz et al. [26].

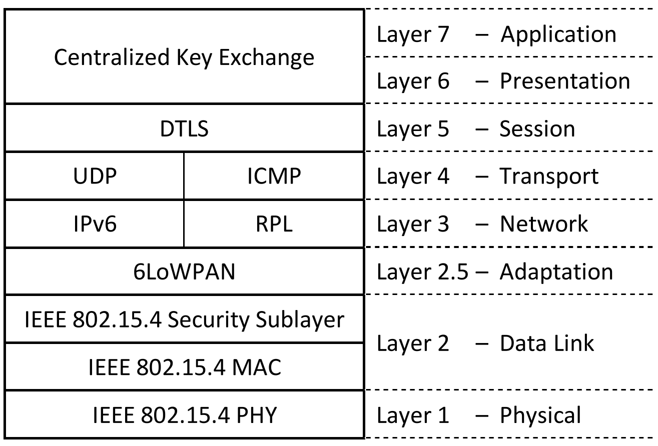

Figure 5.2: 6LoWPAN protocol stack with possible security features

5.2.5 CoAP / DTLS Integration

Many building automation usecases rely on RESTful web services. In order to be able to support such services even on constrained devices, the Constrained Application Protocol (CoAP) was developed. It is currently widely in use in the IoT and in WSN environments. In an attempt to integrate this solution with CoAP, it was looked into existing ways of securing CoAP traffic. One promising approach was to use CoAP in combination with DTLS. DTLS, as described in Section 2.2.2, is an adaptation of Transport Layer Security (TLS) for use with UDP. While its PKC-based key exchange mechanisms aren't particularly well suited for use in a constrained environment, it offers the ability to use pre-shared keys (PSKs) for secure communication. Thereby a pre-shared key is used during the DTLS handshake to establish a secure connection with the peer. This PSK-based key scheme could theoretically be adapted to work in combination with the proposed centralized key exchange. In particular, the keys established through this scheme could be further utilized as PSKs in DTLS-secured communication channels. This approach would have the advantage of making the solution independent of any higher-level protocols or layers, such as CoAP. Once the key exchange is complete and the necessary keys are saved in the DTLS keystore, CoAP (or any other high-level protocol) can take over and utilize the secure communication channel provided by DTLS without having to interact directly with any secure keys or cryptographic mechanisms.
Some research into using CoAP over DTLS in Contiki was conducted and found that the 6lbr project [45] offers a functioning DTLS-CoAP integration for Contiki. Unfortunately tinyDTLS, the underlying DTLS implementation, didn't seem to support using multiple identities in conjunction with pre-shared keys at the time of writing, preventing a straightforward integration of the centralized key exchange. Nevertheless, the general concept of using the key exchange with DTLS-PSK seems sound and creating a working implementation, while out of scope for this thesis, should only be a matter of extending

tinyDTLS to support the necessary features.

5.3 Evaluation

One of the primary goals when designing the system was to ensure acceptable performance even on constrained hardware. The two most important metrics evaluated were power consumption and memory usage. These properties were compared across various security configurations in order to show the performance implications of the suggested approach and to demonstrate its viability. The following four configurations were tested:

- No end-to-end encryption, no link-layer security
- End-to-end encryption, no link-layer security
- No end-to-end encryption, link-layer security
- End-to-end encryption, link-layer security

When comparing the various configurations, option 1 (no encryption, no link-layer security) served as a baseline value that the other configurations were matched against.

In addition to power and memory requirements, a number of measurements regarding the time required for completing various security related tasks were also conducted, both concerning the duration of the key exchange as well as DTLS/CoAP-based functions. These measurements were performed in a simple testbed which is not fully representative of a complex live system. Nevertheless, they were able to provide a general impression of the temporal overhead caused by including the various security features.

5.3.1 Power Consumption

Ensuring a reasonable power consumption was one of the most important aspects of the proposed solution. It is made necessary due to the presence of devices with a limited power supply, such as a battery. In order to achieve this goal, computationally intensive operations, such as asymmetric cryptography, had to be avoided and replaced by more efficient albeit less flexible approaches.

To actually evaluate the power consumption of the implementation, the Powertrace tool offered by Contiki-OS was used. Powertrace estimates the energy consumption during operation by continuously monitoring the state of the hardware components, specifically the CPU and radio transceiver. It records the time spent in each state measured in CPU ticks, which can then be converted to the absolute time based on the CPU clock frequency.
The total time a component spends in a specific state can then be used to obtain its net power consumption by multiplying the time with the estimated power draw of the given component in that state. The average power consumptions of the individual components

Component	Current Consumption	Notes
MSP430f2617	0.5ma	Active Mode @1MHz (LPM)
	< 10ma	Active Mode @16MHz
CC2420	<1µA	OFF Mode
	18.8mA	Receive Mode
	17.4mA	Transmit Mode @0dBm

Table 5.1: Approximate power consumption of Z1 components [41]

on the testing devices were obtained from the Zolertia Z1 datasheet [41]. Table 5.1 provides a listing of the relevant components and their electrical characteristics. For the actual evaluation of the power consumption, it was chosen to compare the power required to complete a single message sending or receiving operation. Since individual nodes typically either send (sensors, control elements) or receive (actuators) the majority of the time, but not both, those two scenarios are evaluated separately.

Regardless of the chosen mode, the CPU spends most of its time between individual message exchanges in low power mode (LPM). The remaining time between exchanges is taken up by tasks regularly scheduled by the OS. As a result, the total power consumption between two send or receive cycles is largely dependent on the length of the idle period between them. Longer waiting periods cause a higher total consumption, reducing the relative difference when comparing various security configurations. Since the main focus lies on the power required by sending/receiving and the associated cryptographic operations, it was attempted to minimize the influence of the idle period. To achieve this, all time spent in LPM mode was disregarded and a period of only one second between individual message exchanges was chosen, reducing unrelated CPU usage and idle radio time to a minimum. This way, the largest part of power usage can be attributed to actual cryptographic and network operations, allowing the comparison of the overhead caused by the different security configurations. Additionally, the time that the radio transceiver spends in transmit and receive modes is factored in, since longer messages require increased radio activity.

Additional workload generated by the partner key establishment process was not considered, as it typically only happens a limited number of times throughout the lifetime of a device, unlike day-to-day operations which will represent the majority of the energy consumption. Furthermore, the key establishment process follows a normal message exchange cycle and doesn't involve any additional high-complexity computations, putting its power consumption in line with normal node operations.

The power consumption of the node startup phase, during which a node sets up its basic functions and networking capabilities was not included in the final evaluation either. This process typically takes around a minute, but only needs to to be performed once during the lifetime of a node and would unnecessarily increase the total power consumption and thus reduce the relative differences between the individual test configurations. An

additional factor that needs to be taken into consideration is RPL convergence. Especially during the early message exchanges, the RPL routing algorithm is still attempting to establish the network topology, causing considerably higher than usual radio traffic, significantly increasing power consumption during some measurement cycles. Much like the node startup phase, this process cannot be avoided entirely and would skew the evaluation results. To avoid this, all measurement cycles that had significantly increased CPU and radio usage due to RPL exchanges were filtered out. For the actual measurement, the nodes completed a total of 110 message exchange cycles each. Of these 110, typically around 5 had to be dropped due to RPL. The power consumption during the remaining message exchanges was then averaged and compared between the various security configurations. Figures 5.3 and 5.4 give a breakdown of the power required by the CPU and radio in the four different configurations.
The evaluation indicated an increase in power consumption of approximately 50% during both sending and receiving. While this number seems rather high, it is limited to the actual encryption, sending and receiving operations due to the measurement methodology. In a real-world application a node consumes a considerable amount of power even when idle, mostly due to the CPU still drawing power even in LPM and the radio periodically turning on to listen for traffic. Measuring the total power consumption during normal operations would therefore result in a much lower relative overhead. With that being said, the measured overhead during send/receive operations appears to be within an acceptable range.

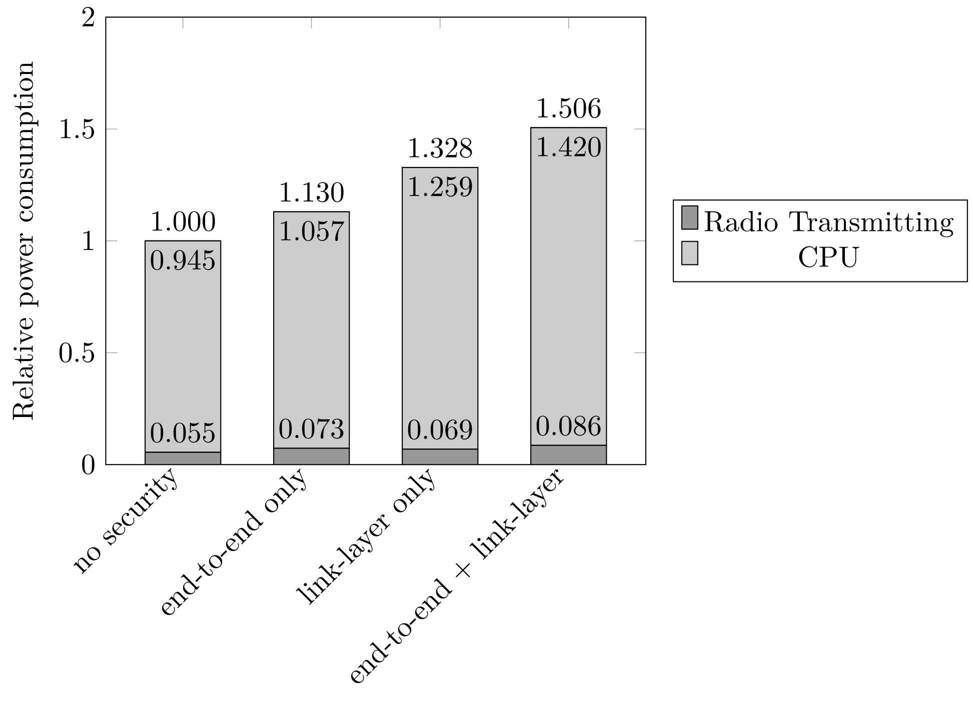

Figure 5.3: Relative power consumption during send cycles

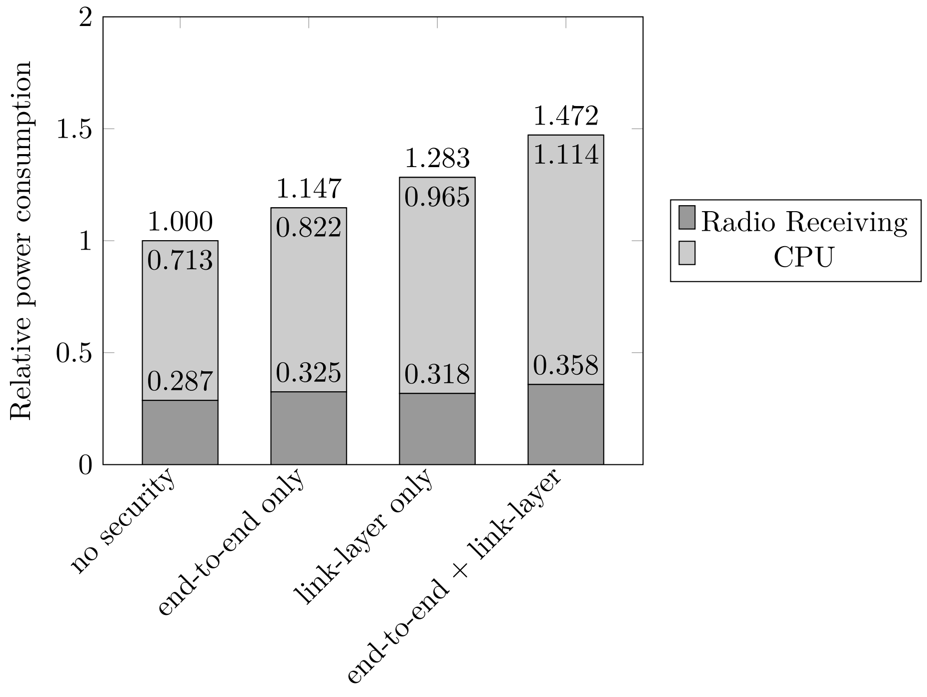

Figure 5.4: Relative power consumption during receive cycles

5.3.2 Memory Usage

The second important aspect to keep in mind while designing a solution for constrained devices is memory consumption. The RAM and ROM available on most devices is typically fairly limited. The Z1 mote that was used for evaluating the implementation offers 92kB of Flash Memory and 8kB of RAM, the majority of which is already occupied by the operating system. The implemented solution has to fit within the remaining memory and still offer at least some upward room for implementing additional features in the future.

As it turns out, enabling end-to-end encryption requires approximately an additional 3400 bytes of program memory and 550 bytes of RAM, which represents an overhead of 8% and 11%, respectively.

Adding link-layer security further increases the memory requirements, making the implementation move rather close to the 8kB RAM limit of the device. While some further micro-optimization may be possible, substantial improvements are unlikely. The RAM requirements of the end-to-end encryption are largely dependent on the number of known nodes, and since the size of the table storing these nodes is chosen statically, a large table size can considerably increase the RAM footprint. Each entry, consisting of node address, state, permanent key, preliminary key, verification challenge, frame counters and an associated timer, takes up approximately 88 bytes of RAM. The above measurements were taken with a table size of 5, allowing each node to store 4 partner nodes in addition to the controller, which always takes up the first table entry. In practice, the table size should be chosen according to system requirements and available memory.

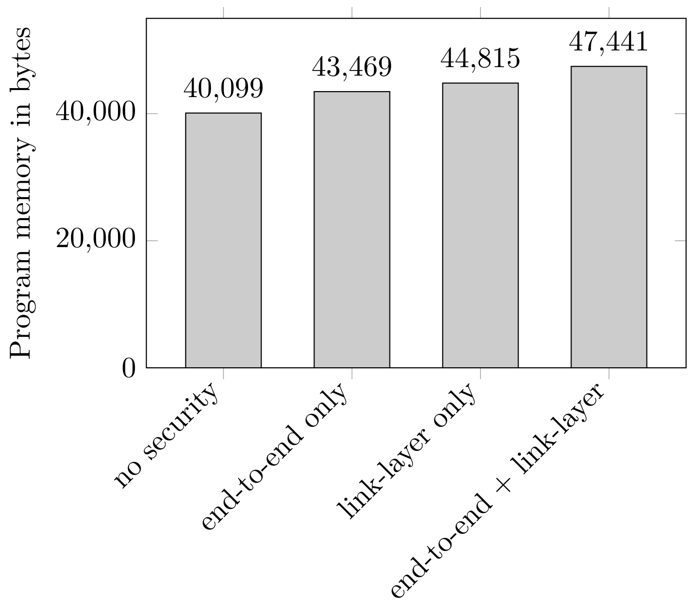

Figure 5.5: Program memory consumption of prototype implementation

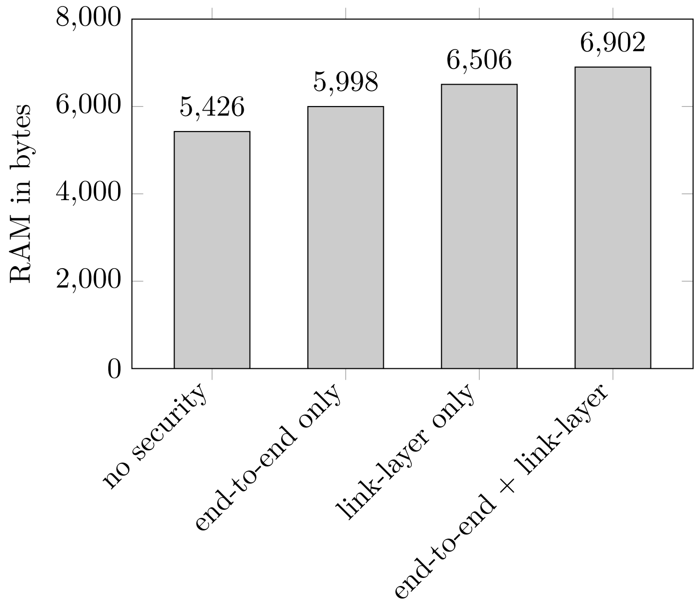

Figure 5.6: RAM consumption of prototype implementation

5.3.3 Timing Measurements

In order to provide a general impression of the temporal overhead of both the proposed solution as well as a possible DTLS-based approach, some additional timings were compared. In particular, the following measurements were performed (Table 5.2):

- Duration of the centralized key exchange: First, the time elapsed between the sending of the initial key request message from the node to the controller and the receipt of the final *RE-ACK* message from the partner node was measured.
  On average this process took 72 milliseconds, with individual measurements varying between 64 and 79 milliseconds.

- Duration of a DTLS handshake: The time required for completing a DTLS-PSK handshake between a node and the control server was measured. This handshake has to be performed before any DTLS-CoAP requests can be handled.
  Completing the DTLS handshake on average took 604 milliseconds, with values varying between 500 and 776 milliseconds. While this is rather long, it should be acceptable due to the fact that the handshake, as well as the centralized key exchange, only have to be completed a single time. Repeating them at a later point is not needed as long as keys or DTLS cookies don't have to be renewed.

Additionally, the performance of the DTLS-PSK-based approach described in Section 5.2.5 in comparison to a plain CoAP approach was evaluated. To that end, the two following metrics were measured (Table 5.3):

- Duration of a plain CoAP request/response cycle: The response time of a simple CoAP GET request by the controller, sent to a node.
  Here, the measured duration was an average of 39 milliseconds, with values between 25 and 65 milliseconds.

- Duration of a DTLS-CoAP request/response cycle: The response time of a DTLS-secured CoAP GET request by the controller, sent to a node. This excludes the handshake duration.
  In contrast to the plain CoAP request, DTLS-CoAP took slightly longer with an average of 63 milliseconds, with values between 54 and 84 milliseconds.

Considering the obtained values, using DTLS shouldn't cause a substantial slowdown of system response times, assuming that any required key exchanges and DTLS handshakes have been completed in advance. Since most functions in a BAS are not time-critical, the additional delays caused by using DTLS instead of plain CoAP are insignificant. Even for tasks where an immediate response would be desirable due to user input (such as light or door switches), the increased response times would be well within the acceptable margin of 100 milliseconds suggested by interface usability guidelines [46].

	Key Establishment	DTLS Handshake
	77 ms	543 ms
	66 ms	566 ms
	78 ms	572 ms
	71 ms	500 ms
	79 ms	549 ms
	71 ms	569 ms
	64 ms	584 ms
	71 ms	729 ms
	72 ms	776 ms
	69 ms	650 ms
Average	**72 ms**	**604 ms**

Table 5.2: Individual duration measurements for the centralized key establishment and the DTLS handshake

	Plain CoAP	DTLS-CoAP
	52 ms	57 ms
	43 ms	81 ms
	40 ms	54 ms
	31 ms	55 ms
	35 ms	57 ms
	45 ms	58 ms
	56 ms	57 ms
	39 ms	82 ms
	43 ms	84 ms
	32 ms	77 ms
	38 ms	61 ms
	33 ms	63 ms
	38 ms	55 ms
	38 ms	62 ms
	27 ms	62 ms
	27 ms	61 ms
	27 ms	57 ms
	65 ms	57 ms
	37 ms	58 ms
	25 ms	68 ms
Average	**39 ms**	**63 ms**

Table 5.3: Individual duration measurements for plain CoAP requests in comparison to DTLS-CoAP requests

CHAPTER 6

Conclusion and Future Work

As building automation systems become more ubiquitous, it gets increasingly important to properly secure them. While a number of approaches and partial solutions for securing and authenticating the network traffic of wireless sensor networks exist, none of them offers an adequate, overarching solution, yet. Even technologies such as DTLS, while well-tested and proven in practice due to the related TLS, cannot solve all of the problems faced by WSNs. In particular, avoiding asymmetric cryptography due to performance reasons and finding a suitable key-exchange based solely on symmetric methods has proven challenging.

In this thesis, such a key-establishment scheme is proposed. It is primarily focused on deployment in building automation systems due to the centralized network structure and architecture typically found in those systems. It allows arbitrary nodes in a network to exchange secure keys by using a trusted intermediary. Some problems and possible attacks related to this scheme are described as well as means by which they can be prevented. To demonstrate the feasibility of this approach, a prototype implementation is provided that allows secure key establishment, end-to-end encrypted communication between two nodes and additionally supports link-layer security over a 6LoWPAN. It was implemented in Contiki-OS, an operating system focused on low-power devices.

The prototype was then evaluated in terms of energy consumption and memory usage, the two primary considerations when developing solutions for constrained devices. The evaluation showed that even with the added end-to-end encryption, both the energy consumption as well as memory usage remained at an acceptable level. These results indicate that this approach to implementing authenticated end-to-end security in the context of building automation is feasible as long as certain requirements, such as a trusted entity, can be guaranteed.

Nevertheless, it is accepted that a centralized approach is not necessarily usable in or suited to all WSN scenarios, and as such, additional research into more flexible approaches is recommended. However, due to the diverse nature of WSN and IoT applications, a single one-size-fits-all solution is unlikely to exist. That being considered, reducing the problem space to particular problem domains can yield solutions that, while not necessarily flexible, are much more effective and better suited to their respective environments. Furthermore, it can be expected that as technology continues evolving, less compromises regarding hardware and performance will be necessary. This should make more flexible approaches, such as those using asymmetric cryptography, increasingly feasible in sensor networks.

In addition to the provided prototype implementation, further research was conducted into integrating the solution with existing, more versatile approaches, such as the aforementioned DTLS. Initial testing shows promising results with DTLS-PSK, which is based on pre-shared keys and offers most features required by a BAS while in theory being possible to combine with the proposed centralized key-exchange approach. Furthermore, DTLS-CoAP performance was tested and compared to an unsecured, plain CoAP approach, illustrating an acceptable overhead. For the future, further research into designing a flexible DTLS framework based around pre-shared keys and identities is recommended. Ideally, this would allow custom key exchange schemes to be easily plugged into the PSK system. This way, existing applications that currently rely on widespread protocols such as CoAP could easily incorporate a secure DTLS layer that offers centralized (or other) key exchange capabilities.

Another area with numerous open questions is that of broadcast encryption. A number of theoretical approaches to realizing broadcast encryption has been described in Section 2.2, but none of them so far offer a really practical solution. One promising approach was proposed recently by Tiloca et al. [38]. It is based on DTLS, which again offers the potential advantage of allowing easy integration of efficient key exchange schemes in the future.

In general, flexible integration approaches such as DTLS should be preferred over ones that are more limited in scope. The primary contribution of this thesis is a centralized key exchange scheme that could be easily integrated with such frameworks. Moreover, the conducted performance evaluation shows acceptable performance and memory overheads when using symmetric cryptography compared to an unsecured solution, both in a direct AES-CCM* implementation as well as when integrating with DTLS-CoAP. The proposed approach, potentially in combination with DTLS, offers a solution that should be well suited to providing security in a building automation context.

Bibliography

[1] S. Akbari, "Energy harvesting for wireless sensor networks review", in *2014 Federated Conference on Computer Science and Information Systems*, 2014-09, pp. 987–992. DOI: 10.15439/2014F85.

[2] H. Jabbar, Y. S. Song, and T. T. Jeong, "RF energy harvesting system and circuits for charging of mobile devices", *IEEE Transactions on Consumer Electronics*, vol. 56, no. 1, pp. 247–253, 2010-02, ISSN: 0098-3063. DOI: 10.1109/TCE.2010.5439152.

[3] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler, "Transmission of IPv6 packets over IEEE 802.15.4 networks", RFC Editor, RFC 4944, 2007-09, http://www.rfc-editor.org/rfc/rfc4944.txt. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4944.txt.

[4] The Number Resource Organization. (2011). Free pool of IPv4 address space depleted, [Online]. Available: https://www.nro.net/news/ipv4-free-pool-depleted (visited on 2016-04-05).

[5] 2016. [Online]. Available: http://www.ieee802.org/15/pub/TG4.html (visited on 2016-10-10).

[6] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, and R. Alexander, "RPL: IPv6 routing protocol for low-power and lossy networks", RFC Editor, RFC 6550, 2012-03, http://www.rfc-editor.org/rfc/rfc6550.txt. [Online]. Available: http://www.rfc-editor.org/rfc/rfc6550.txt.

[7] S. D. Park, "Hierarchical routing over 6LoWPAN (HiLow)", IETF Secretariat, Internet-Draft draft-daniel-6lowpan-hilow-hierarchical-routing-01, 2007-06, http://www.ietf.org/internet-drafts/draft-daniel-6lowpan-hilow-hierarchical-routing-01.txt. [Online]. Available: http://www.ietf.org/internet-drafts/draft-daniel-6lowpan-hilow-hierarchical-routing-01.txt.

[8] S. D̃. Park, "6LoWPAN Ad Hoc On-Demand Distance Vector Routing (LOAD)", IETF Secretariat, Internet-Draft draft-daniel-6lowpan-load-adhoc-routing-03, 2007-06, http://www.ietf.org/internet-drafts/draft-daniel-6lowpan-load-adhoc-routing-03.txt. [Online]. Available: http://www.ietf.org/internet-drafts/draft-daniel-6lowpan-load-adhoc-routing-03.txt.

[9] [Online]. Available: `http://threadgroup.org/` (visited on 2016-10-18).

[10] [Online]. Available: `http://www.zigbee.org/` (visited on 2016-10-18).

[11] P. Levis, T. Clausen, J. Hui, O. Gnawali, and J. Ko, “The trickle algorithm”, RFC Editor, RFC 6206, 2011-03, `http://www.rfc-editor.org/rfc/rfc6206.txt`. [Online]. Available: `http://www.rfc-editor.org/rfc/rfc6206.txt`.

[12] J. Hui and R. Kelsey, “Multicast protocol for low-power and lossy networks (MPL)”, RFC Editor, RFC 7731, 2016-02, `https://www.rfc-editor.org/rfc/rfc7731.txt`. [Online]. Available: `https://www.rfc-editor.org/rfc/rfc7731.txt`.

[13] G. Oikonomou and I. Phillips, “Stateless multicast forwarding with RPL in 6LowPAN sensor networks”, in *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2012 IEEE International Conference on*, 2012-03, pp. 272–277.

[14] R. Anderson and M. Kuhn, “Tamper resistance: a cautionary note”, in *Proceedings of the 2Nd Conference on Proceedings of the Second USENIX Workshop on Electronic Commerce - Volume 2*, ser. WOEC’96, Oakland, California: USENIX Association, 1996, pp. 1–1. [Online]. Available: `http://dl.acm.org/citation.cfm?id=1267167.1267168`.

[15] T. Dierks and E. Rescorla, “The transport layer security (TLS) protocol version 1.2”, RFC Editor, RFC 5246, 2008-08, `http://www.rfc-editor.org/rfc/rfc5246.txt`. [Online]. Available: `http://www.rfc-editor.org/rfc/rfc5246.txt`.

[16] F. Amin, A. H. Jahangir, and H. Rasifard, “Analysis of public-key cryptography for wireless sensor networks security”, *Proceedings of World Academy of Science: Engineering and Technology*, vol. 43, p. 530, 2008-07.

[17] J. Jonsson and B. Kaliski, “Public-key cryptography standards (PKCS) 1: RSA cryptography specifications version 2.1”, RFC Editor, RFC 3447, 2003-02, `http://www.rfc-editor.org/rfc/rfc3447.txt`. [Online]. Available: `http://www.rfc-editor.org/rfc/rfc3447.txt`.

[18] *Federal information processing standards publication (FIPS 197). Advanced Encryption Standard (AES)*, 2001.

[19] G. Singh and Supriya, “Article: a study of encryption algorithms (RSA, DES, 3DES and AES) for information security”, *International Journal of Computer Applications*, vol. 67, no. 19, pp. 33–38, 2013-04, Full text available.

[20] K. Maletsky, “RSA vs ECC comparison for embedded systems (white paper)”, Tech. Rep., 2015-07.

[21] N. Gura, A. Patel, A. Wander, H. Eberle, and S. C. Shantz, "Cryptographic hardware and embedded systems - CHES 2004: 6th international workshop cambridge, ma, usa, august 11-13, 2004. proceedings", in, M. Joye and J.-J. Quisquater, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, ch. Comparing Elliptic Curve Cryptography and RSA on 8-bit CPUs, pp. 119–132, ISBN: 978-3-540-28632-5. [Online]. Available: `http://dx.doi.org/10.1007/978-3-540-28632-5_9`.

[22] P. Pecho, J. Nagy, and P. Hanacek, "Power consumption of hardware cryptography platform for wireless sensor", *Parallel and Distributed Computing Applications and Technologies, International Conference on*, pp. 318–323, 2009.

[23] Q. Huang, J. Cukier, H. Kobayashi, B. Liu, and J. Zhang, "Fast authenticated key establishment protocols for self-organizing sensor networks", in *Proceedings of the 2Nd ACM International Conference on Wireless Sensor Networks and Applications*, ser. WSNA '03, San Diego, CA, USA: ACM, 2003, pp. 141–150, ISBN: 1-58113-764-8. [Online]. Available: `http://doi.acm.org/10.1145/941350.941371`.

[24] C.-Y. Chen and H.-C. Chao, "A survey of key distribution in wireless sensor networks", *Security and Communication Networks*, vol. 7, no. 12, pp. 2495–2508, 2014, ISSN: 1939-0122. [Online]. Available: `http://dx.doi.org/10.1002/sec.354`.

[25] L. Eschenauer and V. D. Gligor, "A key-management scheme for distributed sensor networks", in *Proceedings of the 9th ACM Conference on Computer and Communications Security*, ser. CCS '02, Washington, DC, USA: ACM, 2002, pp. 41–47, ISBN: 1-58113-612-9. [Online]. Available: `http://doi.acm.org/10.1145/586110.586117`.

[26] K.-F. Krentz, H. Rafiee, and C. Meinel, "6LoWPAN security: adding compromise resilience to the 802.15.4 security sublayer", in *Proceedings of the International Workshop on Adaptive Security*, ser. ASPI '13, Zurich, Switzerland: ACM, 2013, 1:1–1:10, ISBN: 978-1-4503-2543-1. [Online]. Available: `http://doi.acm.org/10.1145/2523501.2523502`.

[27] R. Blom, "An optimal class of symmetric key generation systems", in *Proc. Of the EUROCRYPT 84 Workshop on Advances in Cryptology: Theory and Application of Cryptographic Techniques*, Paris, France: Springer-Verlag New York, Inc., 1985, pp. 335–338, ISBN: 0-387-16076-0. [Online]. Available: `http://dl.acm.org/citation.cfm?id=20177.20199`.

[28] S. Zhu, S. Setia, and S. Jajodia, "LEAP+: efficient security mechanisms for large-scale distributed sensor networks", *ACM Trans. Sen. Netw.*, vol. 2, no. 4, pp. 500–528, 2006-11, ISSN: 1550-4859. [Online]. Available: `http://doi.acm.org/10.1145/1218556.1218559`.

[29] S. Raza, S. Duquennoy, T. Chung, D. Yazar, T. Voigt, and U. Roedig, "Securing communication in 6LoWPAN with compressed IPsec", in *Distributed Computing in Sensor Systems and Workshops (DCOSS), 2011 International Conference on*, 2011-06, pp. 1–8.

[30] A. Fiat and M. Naor, "Broadcast encryption", in *Proceedings of the 13th Annual International Cryptology Conference on Advances in Cryptology*, ser. CRYPTO '93, Santa Barbara, California, USA: Springer-Verlag New York, Inc., 1994, pp. 480–491, ISBN: 0-387-57766-1. [Online]. Available: `http://dl.acm.org/citation.cfm?id=188105.188198`.

[31] D. Boneh and M. Franklin, "Advances in cryptology — CRYPTO 2001: 21st annual international cryptology conference, santa barbara, california, usa, august 19–23, 2001 proceedings", in, J. Kilian, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, ch. Identity-Based Encryption from the Weil Pairing, pp. 213–229, ISBN: 978-3-540-44647-7. [Online]. Available: `http://dx.doi.org/10.1007/3-540-44647-8_13`.

[32] J. Baek, R. Safavi-Naini, and W. Susilo, "Public key cryptography - PKC 2005: 8th international workshop on theory and practice in public key cryptography, les diablerets, switzerland, january 23-26, 2005. proceedings", in, S. Vaudenay, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, ch. Efficient Multi-receiver Identity-Based Encryption and Its Application to Broadcast Encryption, pp. 380–397, ISBN: 978-3-540-30580-4. [Online]. Available: `http://dx.doi.org/10.1007/978-3-540-30580-4_26`.

[33] M. J. Bohio and A. Miri, "An authenticated broadcasting scheme for wireless ad hoc network", in *CNSR*, 2004.

[34] Y. Chen, G. Yang, and Y. Chen, "An efficient broadcast encryption scheme for wireless sensor network", in *Wireless Communications, Networking and Mobile Computing, 2009. WiCom '09. 5th International Conference on*, 2009-09, pp. 1–4.

[35] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. E. Culler, "SPINS: security protocols for sensor networks", *Wirel. Netw.*, vol. 8, no. 5, pp. 521–534, 2002-09, ISSN: 1022-0038. [Online]. Available: `http://dx.doi.org/10.1023/A:1016598314198`.

[36] E. Rescorla and N. Modadugu, "Datagram transport layer security version 1.2", RFC Editor, RFC 6347, 2012-01, `http://www.rfc-editor.org/rfc/rfc6347.txt`. [Online]. Available: `http://www.rfc-editor.org/rfc/rfc6347.txt`.

[37] K. Nikitin, "DTLS adaptation for efficient secure group communication", Master's thesis, KTH, School of Information and Communication Technology (ICT), 2015.

[38] M. Tiloca, S. Raza, K. Nikitin, and S. Kumar, "Secure Two-Way DTLS-Based Group Communication in the IoT", Internet Engineering Task Force, Internet-Draft draft-tiloca-dice-secure-groupcomm-00, 2016-04-16, Work in Progress, 35 pp. [Online]. Available: `https://tools.ietf.org/html/draft-tiloca-dice-secure-groupcomm-00`.

[39] W. Granzer, F. Praus, and W. Kastner, "Security in building automation systems", *IEEE Transactions on Industrial Electronics*, vol. 57, no. 11, pp. 3622–3630, 2010-11.

[40] [Online]. Available: `http://www.contiki-os.org/` (visited on 2016-10-10).

[41] Zolertia. (2010). Zolertia Z1 datasheet, [Online]. Available: `http://zolertia.sourceforge.net/wiki/images/e/e8/Z1_RevC_Datasheet.pdf` (visited on 2016-04-06).

[42] Texas Instruments. (2013). CC2420 RF transceiver, [Online]. Available: `http://www.ti.com/product/CC2420/description` (visited on 2016-07-19).

[43] R. Struik, "Formal specification of the CCM* mode of operation", IEEE, Tech. Rep., 2004-09, Doc no. IEEE 15-04-0537-00-004b.

[44] M. J. Dworkin, "Sp 800-38c. recommendation for block cipher modes of operation: the CCM mode for authentication and confidentiality", Gaithersburg, MD, United States, Tech. Rep., 2004.

[45] Cetic. (). 6lbr project, [Online]. Available: `http://cetic.github.io/6lbr/` (visited on 2016-07-19).

[46] J. Nielsen, *Usability Engineering.* San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993, ISBN: 0125184050.