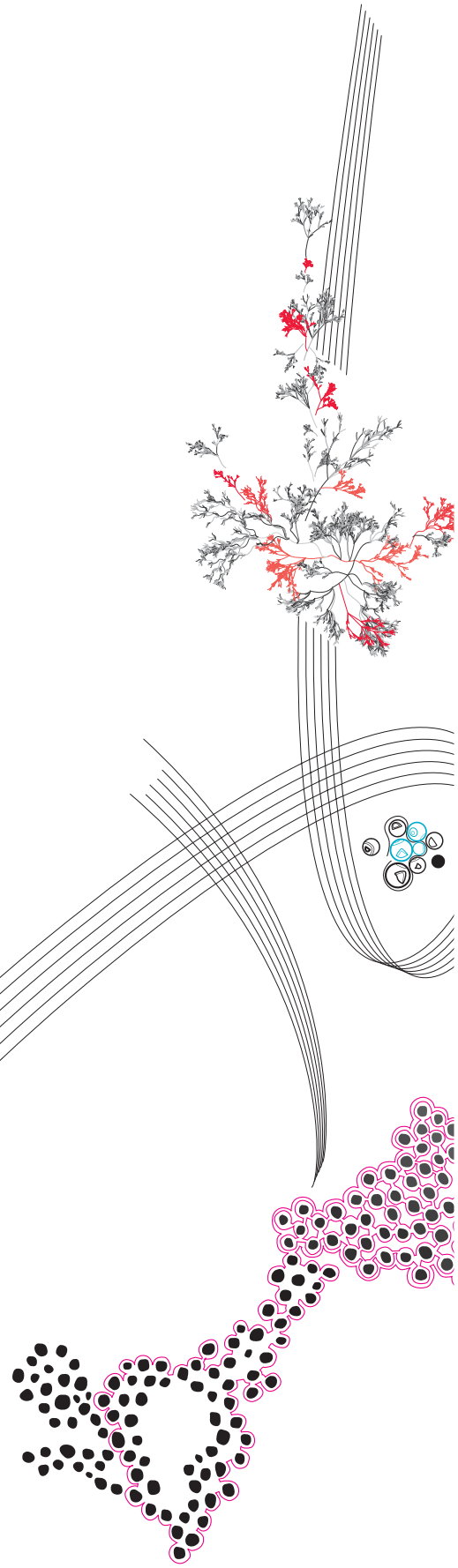


MSc Final Thesis Computer Science

Lightweight Public Key Infrastructure for IoT

Pim Frans Beune

Supervisor: M. Elhajj
Committee member: F. Bukhsh

July 10, 2023

Department of Semantics, Cybersecurity & Services
Faculty of Electrical Engineering,
Mathematics and Computer Science

UNIVERSITY OF TWENTE.

Abstract

The term Internet of Things (IoT) refers to real-world physical objects that have sensors, computing power, and software that may communicate to other systems and devices over the Internet to exchange data. Such devices can greatly improve the quality of life of its users. For example, voice-controlled devices can allow visually and/or mobility impaired people to control home appliances. Nevertheless, the field of IoT has numerous security challenges that need to be addressed. Researchers have reported various attacks against the confidentiality, integrity, and availability of IoT devices. Additionally, because these IoT devices come in such large numbers, they are a very attractive target for adversaries. Most of these security challenges can be solved by making use of cryptographic tools, one of which is Public Key Infrastructure (PKI). Public Key Infrastructure is a suite of soft- and hardware that enables computer systems to control public-key encryption. The goal of a PKI is to make it easier to transfer information securely over the internet and to verify the identities of the parties involved in the transmission. Unfortunately, IoT vendors have been sluggish to embrace PKI for technical and financial reasons. For example, because of the nature of traditional PKI, the architecture introduces a single point of failure. This is an issue especially important in IoT, as the large number of devices introduce a large attack surface. Furthermore, because of the resource-constrained nature of IoT devices, the traditional cryptographic tools will have severe performance limitations when executed on regular IoT hardware.

This thesis aims to address the aforementioned issues by researching the academic landscape regarding PKI for IoT, in order to identify issues and opportunities in designing such a tailored PKI. Furthermore, we introduce a decentralized lightweight PKI system that makes use of lightweight cryptography and certificates, and is thus suited for computationally limited IoT devices. Moreover, because of the decentralized architecture of this novel PKI system, the architecture scales well and is thus especially fitting for IoT devices. We not only find that it is feasible to implement a PKI for IoT, but that this PKI, in some aspects, performs better than existing PKIs in the literature.

Acknowledgements

With this thesis, I am completing my five-year study here at the University of Twente. I would like to express my sincere gratitude to everyone who supported me throughout the journey of finishing this master's thesis.

First and foremost, I am immensely grateful to my supervisor, Dr. Mohammed Elhajj, for his guidance, expertise, our weekly meetings, and the quick responses to my questions. His feedback has been very helpful in shaping this research work.

I would also like to extend my appreciation to the members of my thesis committee, Dr. Faiza Bukhsh, for her time, expertise, and comments. Her input has greatly enhanced the quality of the final parts this thesis.

Moreover, I want to thank Christopher, Joppe, and Tijmen for the many evenings of hanging out at *'t Volbert*. Thanks to them, I was able to get my mind off, so that I could restart work with full focus the next day. I would like to give special thanks to Joppe, whose expertise was invaluable in conducting the energy analysis for my research.

Furthermore, I would like to offer my gratitude to the friends I made during my study here. Armin, Darell, Feije, Joost, Jorrit, Karel, Mart, Max, Neil, Rifqi, and Stan, your company has made my time here at the university a great experience. I eagerly look forward to reuniting with each of you.

Finally, I want to express my gratitude to my parents and sister. Steven, Simone, and Marleen, thank you for supporting me through the journey of this thesis and my entire study here at the University of Twente.

Contents

List of Figures

List of Tables

List of Acronyms

AES Advanced Encryption Standard. 12, 30, 41, 46, 65

AVISPA Automated Validation of Internet Security Protocols and Applications. 63, 66

CA Certificate Authority. 13, 14, 29, 32, 35, 44, 45, 51–54, 68–70

CBOR Concise Binary Object Representation. 7, 21, 23, 32, 45, 50, 55–62, 70

CH Cluster Heads. 27, 28, 40

CL-Atse Constraint-Logic-based Attack Searcher. 66

CN Common Name. 51

CoAP Constrained Application Protocol. 7, 21, 23, 29, 38, 42

CPS Cyber-Physical System. 17, 18, 21, 31

CPU Central Processing Unit. 14, 28, 30, 38, 40, 46, 48, 55

CT Certificate Transparency. 32

DDoS Distributed Denial-of-Service. 12

DN Distinguished Name. 51

DNA Deoxyribonucleic acid. 30, 31, 33–35, 39, 41, 69

DoS Denial-of-Service. 14, 28, 30–32, 35, 37–40, 54

ECC Elliptic Curve Cryptography. 23, 24, 26–30, 32, 33, 35, 38–40, 44, 46, 50, 52, 60, 61, 64, 65, 69, 70

ECDH Elliptic-Curve Diffie-Hellman. 27, 28, 45, 61, 64, 65

ECDLP Elliptic-Curve Discrete Logarithm Problem. 44, 45, 64

ECDSA Elliptic-Curve Digital Signature Algorithm. 28, 33, 61, 64, 65

EGC Elliptic Galois Cryptography. 29

HLPSL High-Level Protocol Specification Language. 66

HTTP Hypertext Transfer Protocol. 38

IBE Identity-Based Encryption. 32

IF Intermediary Format. 66

IoT Internet of Things. 2, 7, 9, 12, 14–19, 21–26, 28–32, 35, 37, 39, 43, 44, 50, 53, 54, 60, 61, 65, 68–71

JSON JavaScript Object Notation. 45, 50, 56, 57, 59, 62

MAC Message Authentication Code. 28

MITM Man-in-the-middle attack. 7, 27, 40, 47, 65, 66

NVS Name Value Storage. 29

OCSP Online Certificate Status Protocol. 29, 30, 37

OS Operating System. 55

PK Public Key. 13

PKI Public Key Infrastructure. 2, 7, 9, 13–19, 21–26, 28–38, 43–45, 52–54, 64–66, 68–71

PSK Pre-Shared Key. 14

PUF Physical Unclonable Function. 29, 31, 34, 35, 37, 69, 70

RA Registration Authority. 32

RAM Random Access Memory. 28, 30, 32, 38, 40, 42, 46, 48, 55, 61

ROM Read Only Memory. 31, 32, 38–40, 42

RQ Research Question. 15, 16

RSA Rivest Shamir Adleman. 33–35, 44, 50, 52, 70

SLR Systematic Literature Review. 7, 15, 16, 19–21, 25, 26, 69

SSL Secure Sockets Layer. 38

TLS Transport Layer Security. 32

TOFU Trust on First Use. 46, 71

TTP Trusted Third Party. 28

UUID Universally Unique Identifier. 7, 45–47, 50

VA Validation Authority. 28

WSN Wireless Sensor Network. 17, 18, 21, 24, 27–29, 31, 32

Chapter 1

Introduction

The term Internet of Things (IoT) refers to real-world physical objects that have sensors, computing power, and software that may communicate to other systems and devices over the Internet to exchange data [3]. By utilizing wearables and personal mobile devices like smartphones, the IoT is transforming people's immediate surroundings into a cyberphysical system with which they can engage. For example, numerous home-automation devices allow consumers to automatically turn off all their lights as soon as they leave home. People-centric IoT solutions greatly influence the daily activities of the elderly and the disabled, improving their autonomy and self-assurance [4]. For example, voice-controlled devices can allow visually and/or mobility impaired people to control home appliances [5]. IoT devices are characterized by being inherently heterogeneous [6] and having stringent resource constraints [7]. Additionally, they are relatively small of size, however, there are billions of these devices currently deployed worldwide [8]. Finally, these objects can communicate wirelessly using protocols such as Wi-Fi, Bluetooth and Zigbee.

Nevertheless, the field of IoT has numerous security challenges that need to be addressed [9]. Researchers have reported various attacks against the confidentiality [10–12], integrity [13–15], and availability [16, 17] of IoT devices. For example, Cui et al. [14] show how firmware upgrades can be used to an attacker's advantage to introduce malicious firmware alterations into embedded devices. They demonstrate a proof-of-concept printer malware that is capable of network spying, data exfiltration, and propagation to regular PCs and other embedded devices. Additionally, because these IoT devices come in such large numbers [8], they are a very attractive target for adversaries [18], who abuse these devices in order to create a botnet. A notable example of this is the Mirai botnet [19], where attackers managed to gain access to 600k IoT devices, creating a botnet of devices that will do anything the attackers tell the devices to do. Consequently, this botnet targeted service provider Dyn with a Distributed Denial-of-Service (DDoS) attack in October 2016, and as a result, numerous high-profile websites like Reddit, Spotify and Twitter were down for multiple hours [20].

Most of these challenges can be solved by making use of cryptographic primitives[1]. We shall now describe some of these (well-known) primitives.

- *Symmetric-key* cryptography employs the same secret key for both the encryption and decryption of data. Given that the secret key is strong enough, it will be infeasible for an adversary given just a ciphertext to reveal and/or modify the underlying plaintext. A notable example of symmetric-key encryption is the Advanced Encryption Standard (AES) block cipher [21].
- *Public-key cryptography* employs pairs of related keys. A public key and its accompanying private key make up each key pair [22]. The private key must be kept hidden in order for

[1] Although most internet-connected devices already have security measures in place, such as password authentication that make use of these cryptographic primitives, we want to point out that it is vital to properly implement these security measures and cryptographic primitives. For example, as shown in the Cui et al. [14] paper, a large share of the discovered printers had default passwords in place, rendering the authentication virtually useless.

public-key cryptography to be secure. Anyone with access to a public key (which can be distributed) can encrypt a message, creating a ciphertext. However, only those who have access to the associated private key can decode the ciphertext to reveal the original message [23].

- A *cryptographic hash function* accepts an input message of any length and outputs a fixed-length "message digest" of the input. For a hash function to be cryptographically secure, it must be computationally impossible to construct any message given a certain target message digest, or to produce two messages with the same message digest [24]. Therefore, cryptographic hash functions serve to verify the integrity of transmitted data.
- Finally, *Public Key Infrastructure (PKI)* is a suite of soft- and hardware that enables computer systems to control public-key encryption. The goal of a PKI is to make it easier to transfer information securely over the internet where more rigorous authentication is needed (instead of, e.g., passwords) to verify the identities of the parties involved in the transmission and the integrity of the data being transferred. PKI makes use of digital certificates, which it is able to create, distribute, edit and revoke. A digital certificate links public keys to the identities (also known as "subjects") of various entities (like domain names) [25]. A Certificate Authority (CA) issues and registers certificates as part of a process that establishes the linking.

In Figure 1.1, a succinct overview of the PKI structure is given. It consists of five steps:

Step 1: An entity, e.g., a bank (*bank.com*) requests a certificate from a CA, by sending it the server's Public Key (PK) and identity.

Step 2: After successfully verifying the identity of the bank, the CA sends back a certificate for the identity and PK of *bank.com*, that is signed with the private key of the CA.

Step 3: A user wants to securely connect to *bank.com*. In order to do this, it will verify the PK the bank offers, by requesting its certificate.

Step 4: Upon request, the bank hands out the certificate to the user.

Step 5: The user will check that the signature on the certificate is indeed from the CA, and that the certificate contains the PK and identity *bank.com*. If these conditions are met, the user will trust the bank and start a secure connection using the bank's public key.

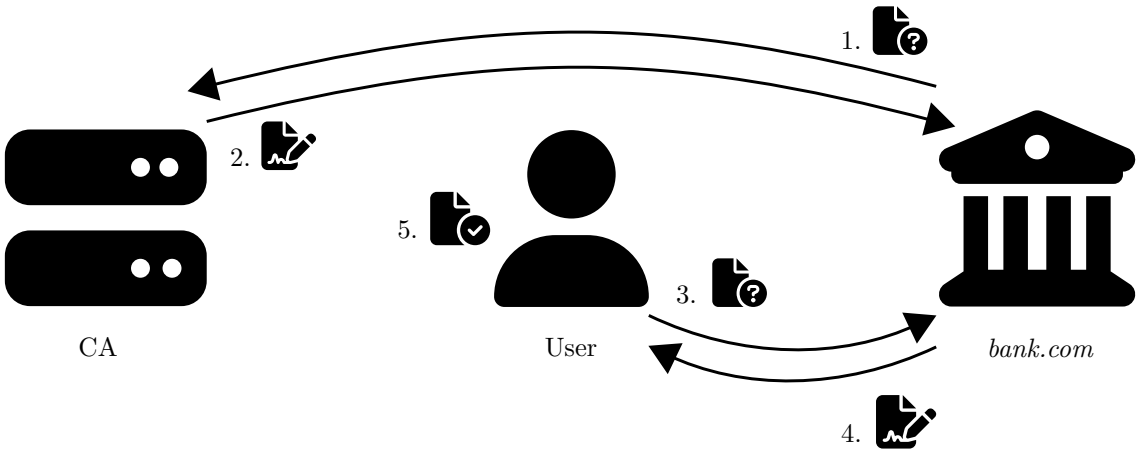


Figure 1.1: Concise overview of the PKI structure.

1.1 Problem statement

IoT vendors have been sluggish to embrace PKI for technical and financial reasons. Here, we list a number of important factors why traditional PKI is not adequate for the current IoT landscape.

1. Because of the nature of traditional PKI architectures, the CA introduces a single point of failure. Therefore, it is possible for attackers to issue certificates for anybody they want if they manage to hijack the CA. This is an issue especially important in IoT environments, as the enormous number of IoT devices creates a large potential attack surface.
2. Because of the resource-constrained nature of IoT devices, the traditional cryptographic primitives mentioned earlier will have severe performance limitations when executed on regular IoT hardware. To illustrate, we present the results from Blanc et al. [26], who ported 12 encryption algorithms to multiple IoT hardware platforms (x86_64, MSP430, AVR and ARM), in order to compare their performance. As can be seen in Table 1.1, there is a fifty-two-fold difference in execution time of identical encryption algorithms when ran on x86_64 (traditional PCs) and MSP430 (one of many IoT CPU architectures).
3. In the current system, it is impractical to untrust certificates in the case of a CA breach. The only option to guarantee certificate validity in the event that it is discovered that a CA has issued false certificates is to ban all certificates issued by the compromised CA. However, it is possible that this would also lead to the rejection of non-malicious certificates, which would negatively impact legitimate users.
4. Pre-Shared Keys (PSK), an alternative to PKI for IoT devices, is easier (and thus cheaper) to implement. However, they are problematic when connected to the internet; since the PSKs need to be enrolled prior to deployment, this is usually done by the manufacturers. Consequently, manufacturers have knowledge of all PSKs, allowing them to decrypt any traffic of these IoT devices.

This thesis aims to address the aforementioned issues by introducing a decentralized PKI system that makes use of lightweight cryptography and certificates, and is thus suited for computationally limited IoT devices. The aforementioned points stress the importance of this research, along with the numerous attacks recorded against IoT devices [8, 10–18].

1.2 Research objective

There is a vast amount of literature that proposes novel PKI architectures and cryptographic protocols tailored to IoT devices. However, some of these proposed works have minor to major flaws:

- Numerous articles propose lightweight PKI solutions that have a centralized architecture, therefore imposing a single point of failure [27–29].
- Some works are vulnerable to Denial-of-Service (DoS) attacks [7, 30].
- Proposed works [31] use a timestamp as the seed for random key generation, allowing adversaries to significantly reduce the potential keyspace. Consequently, this renders brute-forcing keys highly effective.
- One-time pads are reused [32]. This imposes a significant vulnerability because it allows attackers to get the XOR result of two plaintext messages if they can get hold of two ciphers

Table 1.1: Factor difference in average execution time (cycles) of identical encryption algorithms between x86_64 and other architectures (specified in the first row).

	MSP430	AVR	ARM
Average difference (factor)	52.32	32.41	6.67

that were XORed with the same one-time pad (key). Then, they can use statistical analyses to get both plaintexts separately.

Because of the numerous earlier mentioned obstacles with the current IoT landscape, and the deficiency of proposed PKI architectures in the present literature, the aim of this thesis is to develop a PKI that is tailored to IoT devices, i.e., that addresses the current problems with PKI for IoT. To this end, we have developed a number of Research Questions (RQs):

RQ1: What are the current issues regarding PKI for IoT?

RQ2: What are the lightweight PKI solutions already present in the literature?

RQ3: How do we implement a lightweight PKI solution for the IoT?

RQ4: How will this new lightweight PKI solution perform compared with traditional and literature PKI solutions?

1.3 Methodology

The methodology of this thesis is based on the engineering cycle of Wieringa [33]. It consists of four steps: in the first step, *problem investigation*, we will be investigating what problems must be improved, and why. The first step will be completed by answering the first two research questions, which in turn will be done by designing and performing a Systematic Literature Review (SLR), which is described in the next chapters. With these chapters, we aim to identify the current academic landscape regarding PKI for IoT, including issues and opportunities. In the second and third step, *treatment design and implementation*, we will be designing and implementing artifacts that could treat the problem(s). These steps correspond to the third research question, and will be completed by creating a new PKI architecture for the IoT, with the aim to address the issues and opportunities found in the first two research questions. In the final step, *implementation evaluation*, we will be determining the success of the artifacts in terms of treating the problem. This step will be completed by evaluating the proposed implementation in terms of performance and security. Once the evaluation has completed, we are able to answer the final research question by comparing the resulting metrics to those present in the literature.

1.4 Thesis outline

The structure of this thesis is as follows: the first chapter, Chapter 2, will design the Systematic Literature Review (SLR) we will be performing to discover the current academic landscape regarding PKI for IoT. In Chapter 3, we will in fact conduct all steps of this SLR, and report their results. Next, Chapter 4 will summarize all articles that are the result of the SLR in the previous chapter. Furthermore, it will describe a number of limitations found during the article analysis, and it will conclude with a concise overview of all papers and their characteristics. Then, Chapter 5 will propose a novel PKI architecture tailored to IoT devices, that addresses the limitations found in the literature analysis in Chapter 4. Then, in Chapter 6 and 7 we will be evaluating the proposed design and implementation in terms of performance and security respectively. Finally, we will be concluding this thesis in Chapter 8.

Chapter 2

Designing the Systematic Literature Review

A Systematic Literature Review (SLR) is an examination of the available research on a clearly defined subject that employs systematic methods to find, choose, and evaluate pertinent articles as well as to gather and analyze data from those studies. For an SLR, reproducible and transparent approaches must be employed [34]. There are numerous types of literature reviews, such as exploratory reviews, which aim to identify theories, empirical data, and research methodologies that have been published in academic literature [35]. While exploratory reviews only look at the research work on a surface level, we want to identify knowledge gaps in the state of the art, thus we decided that for this thesis, an SLR would be the most appropriate option.

This chapter presents the framework used to conduct the systematic literature review. Firstly, we will compose a specific search string and use this string to search the determined online databases. After that, we will filter out irrelevant articles and analyze characteristics of the remaining corpus. Finally, we will present the knowledge gaps and possible new research directions. A clear overview of the framework is displayed in Figure 2.1.

2.1 Research questions

For the literature aggregation and analysis, we have developed the following research questions:

RQ1: What are the current issues regarding PKI for IoT? This question aims to discover current and relevant issues that could occur when designing and/or implementing a Public Key Infrastructure for an IoT architecture.

RQ2: What are the lightweight PKI solutions already present in the literature? This question focuses on highlighting recent studies regarding PKI solutions for the IoT, along with their characteristics. These solutions could already (to some extent) address the relevant issues described in RQ1.

2.2 Methodology

The SLR consists of four phases: (i) searching the online databases, (ii) filtering results, (iii) analyzing and categorizing results, and (iv) presenting knowledge gaps and new research directions.

2.2.1 Searching the online databases

In the first phase, we compose a keyword-based search string and search online databases such as IEEE and Google Scholar using this search string. Furthermore, we will be looking for existing

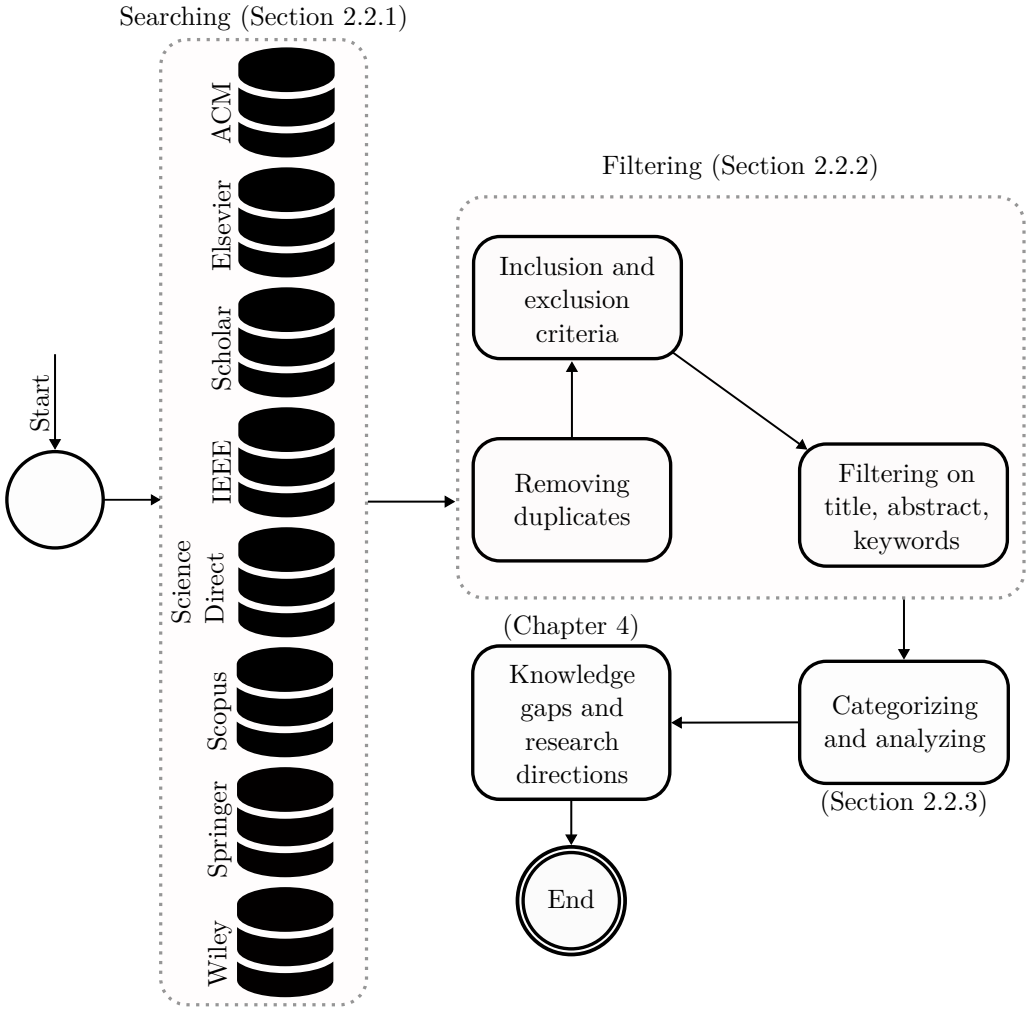


Figure 2.1: Overview of the systematic literature framework.

master theses regarding (lightweight) PKI for IoT. Because these theses are often not findable in databases mentioned before, we will use search engines such as Google to find these.

After identifying the research questions, we composed two search terms to examine the literature. These terms are then used to form a search query, where the query has to contain both search terms. Table 2.1 contains the search terms. A star (*) represents a wildcard. This allows for results that include "Internet of Thing" or "Internet of Things", for example.

For the term "IoT", we included "Internet of Things" and "IoT" since it is one of the subjects of our study. We also used the terms "Internet of Everything", and "Web of Things" because they are closely related to IoT. In addition, we have incorporated "Cyber-Physical System" (CPS), which is a term that is frequently used in place of IoT [36]. Wireless Sensor Network (WSN) is also included since some research papers consider them to be IoT applications [37]. Finally, we included the word "Lightweight" as that is a required characteristic of proposed solutions, i.e., IoT devices are inherently limited in computation and therefore require a lightweight PKI solution. Nevertheless, this keyword was added only after realizing the number of resulting articles was on the low side. More on this in Section 3.2.

For the term "PKI", we included "Public Key Infrastructure" and "PKI", as it is another subject of our study. We also included the term "Cryptography" and "Cryptology" as cryptography plays a large role in PKI.

Table 2.1: Search terms used to compose the search query. A star (*) represents a wildcard.

IoT	("Internet of Thing*" **OR** "Web of Thing*" **OR**"Internet of Everything" **OR** "IoT" **OR** "Cyber Physical System*" **OR** "Cyber-Physical System*" **OR** "CPS" **OR** "Wireless Sensor Network*" **OR** "WSN" **OR** "Lightweight")
PKI	**AND** ("Public Key Infrastructure*" **OR** "PKI" **OR** "Cryptography" **OR** "Cryptology")

Finally, we will search the online databases listed in Table 2.2 with the earlier specified search terms. We want to note that for searching the master theses, the search terms from Table 2.1 have not been used, as search engines often do not properly work with boolean operators. Instead, the search terms *"Master Thesis PKI IoT"* were used.

2.2.2 Filtering results

In this phase, we filter out unwanted results by looking at the title, abstract and keywords of the papers. Furthermore, we will construct inclusion and exclusion criteria and filter the set of papers with these criteria. Moreover, we will filter out duplicate papers found among multiple online databases.

A set of inclusion and exclusion criteria has been established as part of the review to ensure the selection of appropriate research papers. They can be seen in Table 2.3. We want to elaborate on a number of criteria. *IC.1* states "*Articles that are primary studies.*" The goal is to only include research articles that offer original insights rather than reviews of previous literature. *IC.3* states "*Articles that are published in journals or conferences.*" We aim to include papers that are of significant scientific quality. This criterion can be assured by only including works from journals or conferences, where works go through a relatively rigorous reviewing process.

2.2.3 Analyzing and categorizing metadata of results

In the third phase, we analyze the metadata of the results by categorizing the papers by number of citations and publication year. This can be achieved by creating bar and/or pie charts. Moreover, we will analyze the most frequently used title words of articles by creating a bar chart. Furthermore, we will analyze co-authorship and/or co-citations of the papers, which can be done by creating the respective graph with tools such as *VOSviewer* [1]. Finally, we will categorize the papers on whether they propose a novel cryptographic protocol or PKI framework. We will also categorize the papers on what techniques/mechanics are used (e.g., Elliptic Curve Cryptography, decentralized technologies such as blockchain).

Also, we will count the instances of keywords mentioned in publications by authors, evaluate the frequency of keyword co-occurrence to determine trends in IoT and PKI articles. Finally, we will analyze the most cited papers in the filtered corpus.

Table 2.2: Online digital databases used for the research.

Database	URL
ACM	https://dl.acm.org
Elsevier	https://www.elsevier.com/
Google Scholar	https://scholar.google.com
IEEE	https://ieeexplore.ieee.org
ScienceDirect	https://sciencedirect.com
Scopus	https://www.scopus.com
Springer	https://springer.com
Wiley	https://onlinelibrary.wiley.com

Table 2.3: Inclusion and exclusion criteria for research papers.

Criterion	Details
Inclusion	• Articles that are primary studies. • Articles that present challenges, implementations, or results regarding PKI or cryptography for IoT. • Articles that are published in journals or conferences.
Exclusion	• Articles with fewer than 10 citations. • Articles not written in English. • Articles without any references.

2.2.4 Presenting knowledge gaps and new research directions

In the final phase, we will have read and analyzed all remaining papers. With this information, we hope to identify knowledge gaps in the literature and possible new research directions. This will be done in Chapter 4. In the next chapter, we will be performing the SLR, and classify and analyze the resulting papers.

2.3 Conclusion

In this chapter, we have designed the framework for the SLR we will be performing in the next chapter. We have composed a number of research questions and formed a methodology for searching, filtering, and analyzing articles. The next chapter will perform all the steps listed in this chapter, and present the results of each step.

Chapter 3

Performing the Systematic Literature Review

This chapter explains how the SLR is performed. We focus on providing information about the activities involved in the identification, selection, and reporting of the research works. We also talk about the classifications that were employed.

3.1 Searching the online databases

We gathered 324 research works that may be related to the study topic from the use of the search query across the digital libraries.

In some databases, employing the search string yielded tens of thousands of results. Therefore, we chose to employ the search string on the article title only, that is, the title must contain the entire search string. Applying this constraint yielded far fewer results, which made it feasible to work with the found papers. Moreover, the webpage of Elsevier seemed to forward to ScienceDirect upon querying the database, therefore, we consider these two databases to be equal.

Table 3.1: Number of search results per online database

Database	Number of results
ACM	2
Elsevier	5
Google Scholar	148
IEEE	49
Master Theses	5
ScienceDirect	5
Scopus	113
Springer	0
Wiley	2

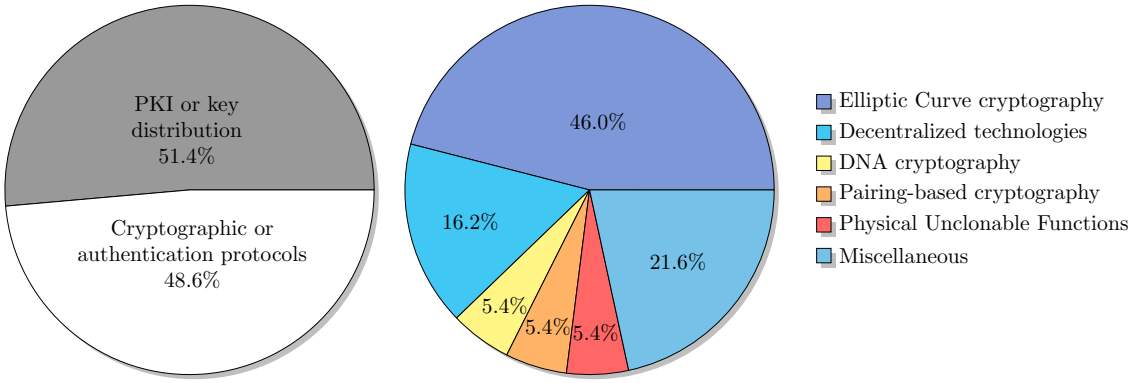
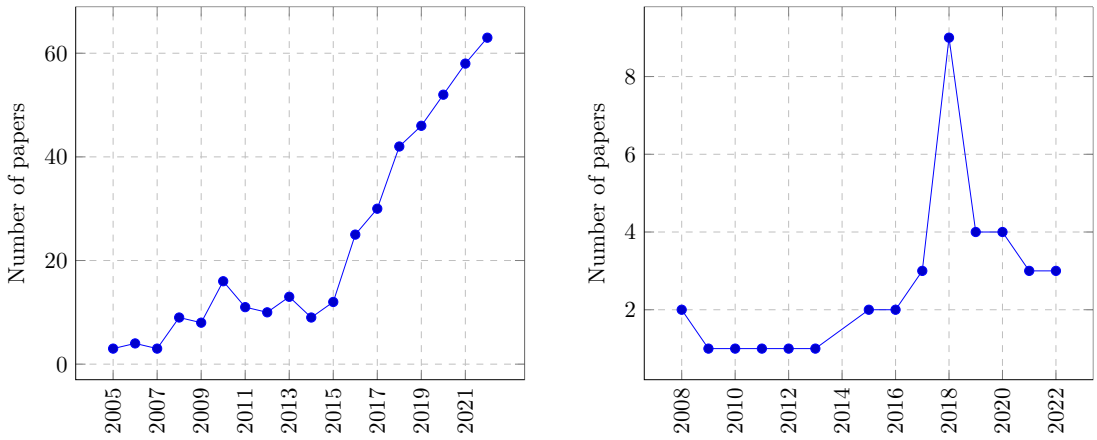


Figure 3.1: The left chart shows the percentage of the papers that address cryptographic or authentication protocols, and papers that address PKI or key distribution. The right chart shows the main mechanics/technologies used in papers of remaining corpus. Miscellaneous technologies include identity-based cryptography, Constrained Application Protocol (CoAP), Concise Binary Object Representation (CBOR) and X.509 optimizations.

It is important to note that before aggregating results of the online databases, we already applied *EC.1* (*"Articles with less than 10 citations"*). This is because the online interface made it easy to identify articles with this constraint, and this would have been much harder to do later in the SLR process. However, this constraint is not applied to the Master Theses, as these articles are not present in the literature databases. Moreover, we want to point out that ScienceDirect does not support wildcards, and does not support over 8 boolean operators. Therefore, for ScienceDirect only, we adapted our search term to

("Internet of Things" **OR** "IoT" **OR** "Cyber Physical System" **OR** "CPS" **OR** "Wireless Sensor Network" **OR** "WSN") **AND** ("Public Key Infrastructure" **OR** "PKI" **OR** "Cryptography")

because we deem these keywords the most relevant.



(a) Before filtering (Section 3.1).

(b) After filtering (Section 3.2).

Figure 3.2: Number of articles in corpus per year.

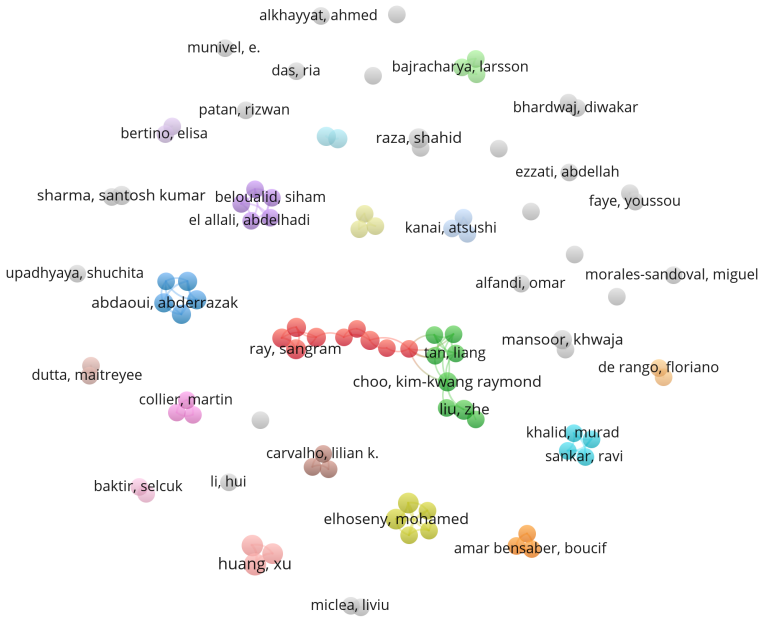


Figure 3.3: Co-authorship graph on PKI, Cryptography and IoT before filtering (Section 3.1). Created with VOSviewer [1].

Table 3.1 lists the number of results per online database. It is interesting to note that some databases yielded very few results (after filtering on *EC.1*), and one database (Springer) even yielded zero results.

3.2 Filtering results

The initial database queries yielded a total of 324 articles. After removing duplicates, we were left with 117 distinct articles, a reduction of around 64%.

Next, we applied the inclusion and exclusion criteria to the corpus of papers. This left us with 50 articles. A couple dozen papers have been removed as they are not primary studies, i.e., these papers exclusively reviewed previous literature. Moreover, a significant number of articles had fewer than 10 citations, so these were removed from the corpus as well. Additionally, one article in the corpus had no references and was therefore removed.

After this, we filtered the papers on the title, abstract and keywords. As a result, ten papers were removed as they were deemed irrelevant. For example, one paper proposed a storage scheme tailored to IoT devices. This resulted in a corpus of 29 articles. Because we deemed this number of articles to be on the low end, we decided to redo the searching and filtering procedure including the "Lightweight" search term (Table 2.1). We decided to add this search term because IoT devices are inherently limited in terms of computation, and require therefore a lightweight PKI/cryptography solution. As a result, we managed to find eight more relevant papers, bringing the total number of articles to 37.

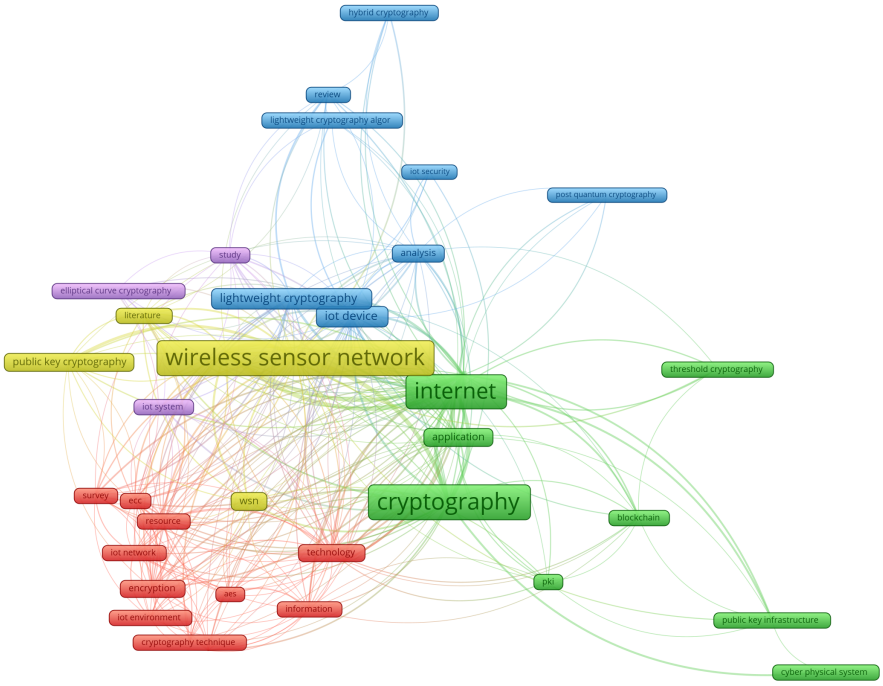


Figure 3.4: Co-occurrence of keywords (limited to 33 keywords) (before filtering, Section 3.1). Created with VOSviewer [1].

3.3 Analyzing and categorizing metadata of results

The majority (51.4%, Figure 3.1, left) of the remaining papers introduce novel PKI solutions for the IoT, while a smaller portion of the papers presents novel cryptographic or authentication protocols. Also, there seems to be a trend among techniques used to implement a PKI or cryptography solution for the IoT. As can be seen in Figure 3.1 (right), 46% of the papers in the remaining corpus make use of Elliptic Curve Cryptography (ECC). Other techniques used are decentralizing technologies, such as blockchain. A couple of papers use pairing-based cryptography, and finally, a very small number of papers employ techniques such as X.509 optimizations, CBOR and CoAP.

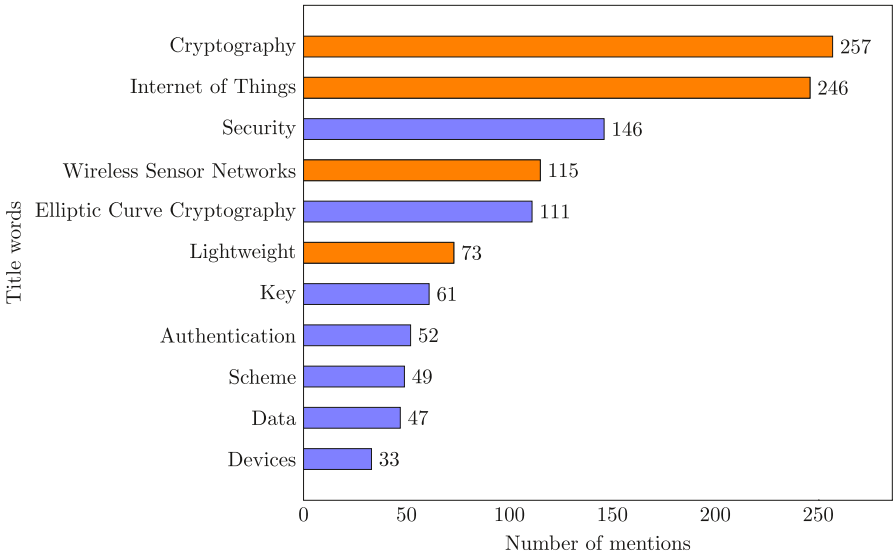


Figure 3.5: The most frequently mentioned words used in article titles, before filtering (Section 3.1). Search terms (Table 2.1) are highlighted in orange.

Table 3.2: Most-cited articles on PKI, cryptography and IoT after filtering (Section 3.2).

Number	Article name	Number of citations
1	Liu and Ning [39] 'TinyECC: A configurable library for elliptic curve cryptography in wireless sensor networks'	688
2	Shi and Gong [40] 'A new user authentication protocol for wireless sensor networks using elliptic curves cryptography'	141
3	Elhoseny et al. [41] 'A secure data routing schema for WSN using Elliptic Curve Cryptography and homomorphic encryption'	107
4	Szczechowiak et al. [42] 'On the application of Pairing Based Cryptography to Wireless Sensor Networks'	73
5	Khari et al. [43] 'Securing Data in Internet of Things (IoT) Using Cryptography and Steganography Techniques'	71

Furthermore, Figure 3.2 plots the number of articles per year before filtering the papers (Section 3.1) and after filtering (Section 3.2). Before filtering, the number of articles published climbed from 9 in 2014 to 63 in 2022, a seven-fold increase. This shows a clear increasing trend in the number of articles of the past 8 years. After filtering, the trend is still visible, but definitely not as clear as before filtering. This could be due to the fact that the papers have not been filtered on a date criterion.

Next, a co-authorship graph was made using VOSviewer (a data visualization program) [1] and includes co-authors who have written at least two papers (see Figure 3.3). The output is the largest graph with the co-authorship relationships of 95 writers. We want to note that this graph was generated using the full corpus (324 articles), as the filtered corpus of only 37 articles did not result in any notable relations. This graph shows numerous author clusters, however, there appears to be only a single connection between two clusters. This is achieved by Joel Rodrigues (Instituto de Telecomunicações, Portugal), who connects the red cluster (consisting of numerous Indian scholars) to the green cluster (mostly consisting of scholars from Asian universities), by writing a paper about a blockchain-based secret sharing scheme for Industrial IoT devices [38]. These connected clusters consist of 17 distinct authors. Nevertheless, there seems to be no co-authorships between other articles, although there is a vast number of other unconnected clusters.

We also built a graph of the co-occurrence of keywords, that contains the 33 most frequently used keywords, and is connected by co-occurrences of other keywords (Figure 3.4). For the same reason as the co-authorship graph, we opted to use the entire unfiltered corpus (Section 3.1). While the colored clusters are not very distinguishable, we do identify the blue (8 keywords), green (8 keywords), and red (10 keywords) clusters, that contain keywords related to cryptography and security techniques for the IoT.

Additionally, we counted the number of unique words included in the article titles in order to examine trends in IoT and PKI articles (see Figure 3.5). We opted not to assess keyword frequency of the articles, because this resulted in too few results. It is important to note that some words such as "IoT" and "Internet of Things" have been merged for a more comprehensive overview. Furthermore, we used the entire unfiltered corpus for this analysis (Section 3.1). Figure 3.5 shows that aside from the search terms, which are highlighted in orange (Table 2.1), the most frequent article title words are "Security" (146 mentions), "Elliptic Curve Cryptography" (111 mentions), and "Key" (61 mentions).

Furthermore, Table 3.2 displays the five most-cited publications of the filtered corpus. In total, they were cited 1080 times. All papers propose work that can help secure data transfer between IoT devices and other IoT/regular devices. Four out of the five papers use ECC as the main technology [39–41, 43], one uses pairing-based cryptography [42].

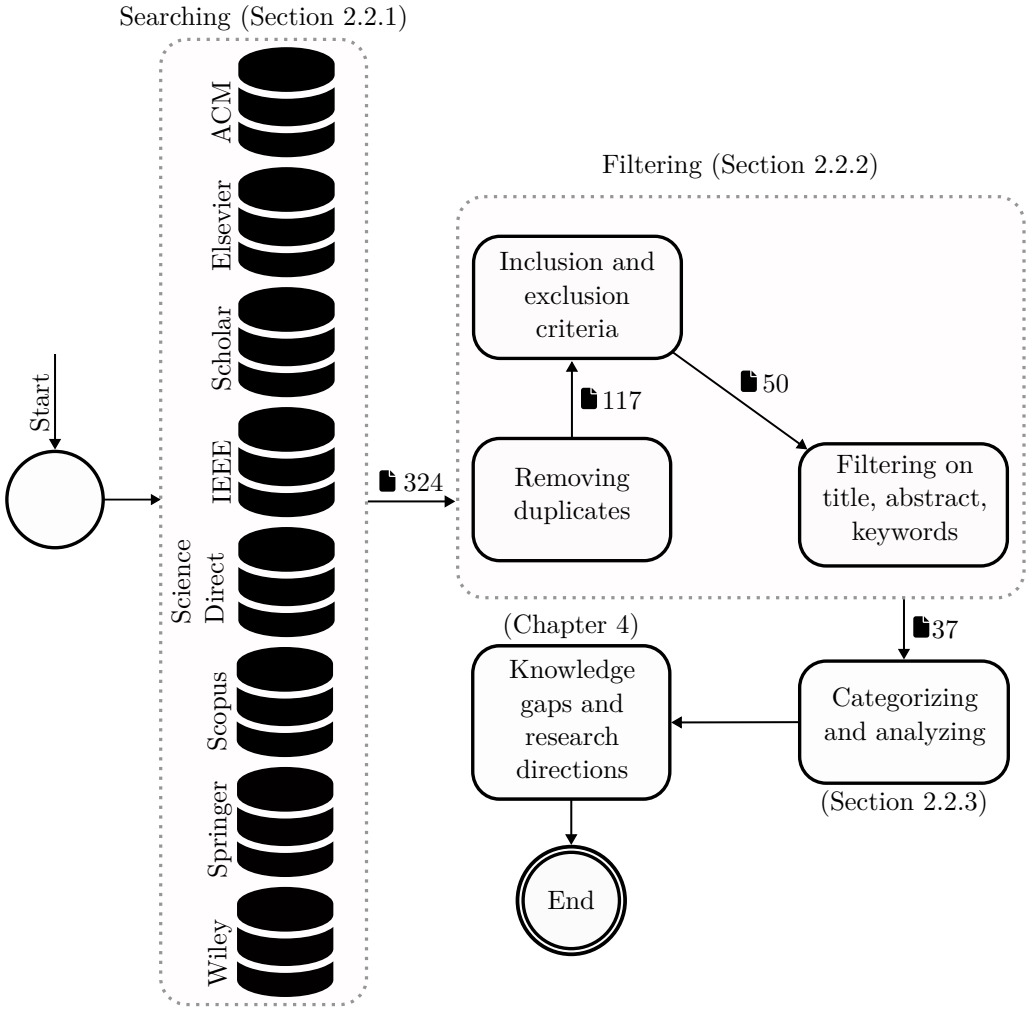


Figure 3.6: Overview of the systematic literature framework. Also, the number of articles remaining after each step are shown.

A final overview of the SLR procedure can be seen in Figure 3.6. It is similar to Figure 2.1, however, it also contains the number of articles remaining in each step of the SLR.

3.4 Conclusion

In this chapter, we have performed the SLR using the steps listed from the previous chapter. Our initial corpus of articles consisted of 324 articles. Nevertheless, after removing duplicates and filtering out irrelevant articles we ended up with a final body of 37 articles. Additionally, we have classified the articles based on whether they introduce lightweight cryptography, or lightweight PKI for the IoT. Furthermore, we have further classified the latter category on what technologies are used in the articles. The next chapter will summarize and discuss the 37 articles, by discussing all proposed technologies and creating a comprehensive overview of all articles and their properties.

Chapter 4

Literature analysis

This chapter summarizes the 37 articles that are the result of the SLR in Chapter 3. We classify the articles by the main technology used, such as ECC and pairing-based cryptography. Furthermore, we will discuss all technologies by listing their advantages and disadvantages when implemented in a lightweight PKI. Finally, we create a comprehensive overview of the papers by listing their characteristics, and presenting the results of their security and performance analyses. These overviews can be found in Table 4.2 and 4.3. Figure 4.1 shows an overview of the technologies found in the articles in this SLR. A single section is dedicated for every technology, which also briefly introduces said technology.

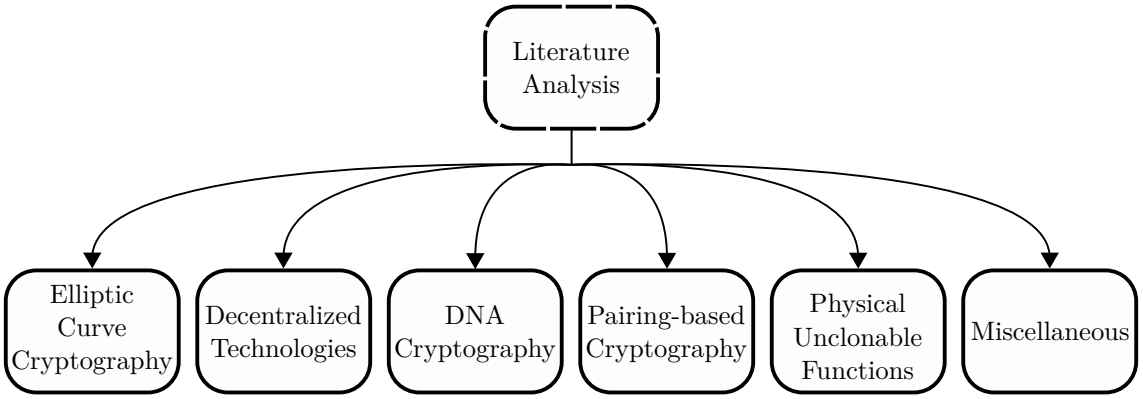


Figure 4.1: Overview of the discovered technologies used in the articles in this SLR.

4.1 Elliptic Curve Cryptography

Elliptic Curve Cryptography (ECC) is a method of public-key encryption that is built on the algebraic structure of elliptic curves. Compared to non-EC encryption, ECC enables smaller keys while offering comparable security [44–46].

Tewari and Gupta [47] developed a simple ECC-based authentication protocol for IoT devices. In this protocol, the client and server exchange a number of randomly chosen points on an elliptic curve, in order to verify each other's private keys. They also provided an evaluation of the protocol's performance and security. Mutual authentication, confidentiality, anonymity, forward secrecy, and other attacks are all safeguarded by the protocol. Comparing the use of ECC to other asymmetric approaches, the overhead of computation and communication is reduced. The suggested protocol is quick to implement, has little overhead (1152 bits in communication).

Shi and Gong [48] present a novel authentication protocol for WSNs based on ECC, which overcomes the weaknesses in [49]. The protocol consists of four phases; user registration, login, authentication, and the final phase allows the user to update their password. In terms of security, computation and communication cost, the suggested protocol outperforms its predecessor by 30% according to the authors. This protocol can prevent general security problems and offer reciprocal authentication to safeguard both inner and outside security, unlike the work of [49]. The suggested approach is therefore better fitted to WSN deployments.

Shah and Shah [32] developed ECIOT, a protocol to establish a secret session key with a server, through the use of Elliptic-Curve Diffie-Hellman (ECDH). The established key will be used as a one-time pad (i.e., XOR cipher) to encrypt communication between two entities. The ECIOT protocol has been implemented on the MIRACL cryptography library. Nevertheless, the article fails to provide a security and a performance analysis. Additionally, the fact that the proposed protocol uses the same one-time pad per session, i.e., it reuses one-time pads, introduces a major vulnerability; if adversaries are able to obtain two ciphers XORed with the same key, they are able to retrieve the XOR result of the two plaintext messages. With this, they can perform statistical analyses in order to obtain both plaintexts individually.

In Rajendiran et al. [50] a unique method for effective and secure key-predistribution in a WSN is proposed. Each sensor node has a unique point on an elliptic curve allocated to it. For each sensor, a private key is created using the point doubling procedure. If two sensor nodes share at least one private key, they can securely connect. Two sensor nodes will employ an intermediary sensor for which both parties hold the keys if they themselves do not share keys. Their simulation findings demonstrate that, when compared to previous schemes [51], the suggested model has superior resilience to network attacks (such as brute force and Sybil attacks) and needs less memory to provide better connectivity and resilience.

Qazi et al. [45] propose an effective and safe technique for communication security across a wireless sensor network consisting of 10 (weak or strong) nodes. Strong nodes store the authentication information for all nodes, while weak nodes only save the information for strong nodes. Communication between weak nodes is therefore transmitted via a strong node. The methodology has been implemented with ECC, and it has been assessed in terms of two kinds of communication: that which occurs between nodes of the same kind and that which occurs between nodes of different types. According to the authors, the scheme shows to be resilient against various attacks, and introduces only a slight overhead in weak node to weak node communication. Finally, key generation between two nodes only takes 50 ms.

Louw et al. [52] created an ECC-based key distribution protocol. In this protocol, if a client wants its key distributed, it sends its address and public key to the server. Then, the server encrypts this data using a start-up key and distributes the cipher across the network. According to the authors, the protocol complies with the minimal standards necessary for a key distribution plan to be deemed efficient and secure. The scheme's setup time and associated system overhead are the only points of concern. Unfortunately, the writers did not offer a thorough examination of security and performance. Additionally, one part of the algorithm uses a hard coded key to encrypt and decrypt, making it vulnerable to statistical analysis attacks.

Ju [53] created a lightweight key establishment protocol that uses ECC and integrates symmetric cryptography, hash chains, and ECDH. To make computation and communication faster, the same initial key is deployed to all devices. According to the author, despite having lower computational complexity than other key establishment protocols, the ECC operations affect the total efficiency the most. In terms of security, the scheme has perfect forward secrecy and is not vulnerable to Man-in-the-middle attack (MITM) attacks. Finally, the author determined that the key storage has a linear relation to the number of nodes, and that establishing a network of 10 nodes takes around 10 seconds.

Through the use of a dynamic clustering strategy, Elhoseny et al. [41] present a unique encryption method based on homomorphic encryption and ECC to secure transmission of data in a WSN. A genetic algorithm is used to find the best sensor nodes to serve as Cluster Heads (CH), which transport messages to base stations, with the aim of maximizing the network's overall longevity.

Public and private keys for sensors are created using ECC. By using homomorphic encryption, the CH can create the resulting data that will be delivered to the base station without having to decrypt any encrypted data in order to avoid the CH from being compromised. The author's performance analysis shows that their method requires $1.04 - 3.47\times$ less RAM and is $0.99 - 9.16\times$ faster than related works. Finally, they show that their protocol is secure against eavesdropping (passive attacks), brute force attacks, sinkhole attacks, and DoS attacks.

An open source ECC, including an ECDH and an Elliptic-Curve Digital Signature Algorithm (ECDSA), optimized for the IoT has been implemented and tested by Pinol et al. [54] To optimize the algorithms, they avoid division operations as these are computationally expensive. Instead, they use modular arithmetic. Furthermore, they implemented an optimized greatest common divisor and Euclid's algorithm. When compared to homogeneous and affine coordinate systems, their test demonstrates that the use of the Jacobian coordinate system provides higher performance and has smaller memory footprints. For a key size of 256 bits, key generation takes around 5000 ms and requires 76 mJ of energy. Furthermore, they show that signature verification and generation takes around 11 and 5 seconds, 154 and 76 mJ respectively. Finally, the protocol requires around 1800 bytes of RAM.

μPKI, a Public Key Infrastructure for wireless sensor networks, is presented by Kadri et al. [28] μPKI uses public key cryptography as a method to ensure the validity of the base station in an effort to address the security issue in WSN. μPKI consists of two parts. The first is the handshake between the base station and a particular sensor node, during which the base station and sensor node establish a session key to secure an end-to-end link. The second stage involves utilizing this session key to encrypt data in order to protect data privacy and maintain data integrity using MACs attached to each packet. Their performance analysis shows that encrypting the session key requires around 23 mJ of energy. Moreover, sending and receiving the session key takes around 4 and 2 mJ respectively. Finally, they show the sensor to sensor handshake requires around 4 mJ.

An ECC-based security framework for smart cards with IoT capabilities is proposed by Bai et al. [46] It is able to use any applications at any time, anyplace, and supports encryption and decryption, digital signature generation and verification. By implementing this security framework, users will be able to use a single IoT-enabled smart card in a smart environment to gain secure access to any applications. Moreover, safe automated device-to-device communication is provided by this IoT-enabled smart card. The authentication, integrity and confidentiality of the smart card and data are all guaranteed through message encryption and authentication, according to the authors.

With AVR and ARM CPUs, Liu and Seo [55] provide, assess, and optimize the Ted37919, NUMS256 and NUMS384 ECC curves. They do this by implementing more efficient squaring for multiplication, and multi-precision multiplication algorithms. They show the implementation breaks speed records for both ARM and AVR CPUs. Next, they implemented the NUMS curves NUMS256, Ted37919, and NUMS384, which incorporate the distinct computational benefits of the Montgomery and twisted Edward curves. Then, they show that for scalar multiplication across 256, 379, and 384-bit security prime fields, they achieved record-breaking execution times. On an ARM processor, the NUMS256 curve, for instance, only needs 1.357 M cycles, which is more than 1.6 times quicker than the often used Curve25519.

Lara-Nino et al. [56] described the design and implementation of a hardware module for ECC curves. By allowing various polynomials, field lengths, and other curve properties, this architecture demonstrates generality. This work's design operates in constant time, providing some defense against side-channel attacks. The hardware utilization of the suggested design has been assessed by the authors, and it has been contrasted with related academic works. This comparison has demonstrated that the suggested design is appealing for applications with limited resources, such those found in the IoT environment. The architecture that has been presented demonstrates hardware costs less than 1500 slices and runtimes less than 10 ms.

A Lightweight Public Key Infrastructure called LPKI is presented in Toorani and Beheshti [29] and is tailored to platforms with limited resources. It utilizes signcryption and ECC, and gives each subscriber one set of private-public keys, delegating all validations to a Trusted Third Party (TTP) known as the Validation Authority (VA). The architecture employs optimized certificates,

to increase protocol efficiency. Nevertheless, because the certificates have the same structure as the widely used X.509 certificates, they are easily compatible with the ubiquitous PKIX infrastructure. Unfortunately, neither security nor a performance analysis is present in this article.

Elliptic Galois Cryptography (EGC) is an encryption scheme that is introduced and described in Khari et al. [43], which is used to encrypt private information obtained from various medical sources. In the proposed work, various IoT devices send information using the proposed protocol to the controller. The data is encrypted using the EGC scheme by the controller, and the encrypted message is then steganographically buried within layers of a picture. According to the authors, better security was offered by the proposed EGC protocol using the new ECC over Galois field. Furthermore, they show that the protocol's mean squared error is at least 52 times lower than that of existing approaches, and encryption and decryption each take about 0.4 seconds.

In Albalas et al. [44], a secure and resource-efficient CoAP protocol is designed for the application layer of IoT networks, which ensures authentication and authorization, confidentiality, and integrity between devices on IoT networks. The encryption protocol used is ECC, because it employs smaller key sizes, which should result in less power usage in these networks. According to their test findings, compared to the normal CoAP protocol, authentication used 75% less energy, data integrity 56% less energy, and confidentiality 47% less energy. However, no security analysis is present.

Liu and Ning [39] propose TinyECC, a customizable library for ECC operations in WSNs. They also discussed its design, implementation, and assessment. TinyECC's configurability is an interesting characteristic, which allows developers to enable or disable particular optimizations as needed. The developers have a lot of choice when it comes to incorporating TinyECC into sensor network applications because different configurations have varying execution times and resource consumption. Their testing results revealed the TinyECC configurations that were the most efficient in terms of computation and storage. Nevertheless, the article does not provide a security analysis of TinyECC.

4.2 Decentralized technologies

Technologies with decentralized components are those whose parts are spread across various computers and which coordinate and communicate with each other. Decentralized technologies are distinguished by the fact that not all data and computation is kept in a single location. An example of a decentralized technology is the renowned Bitcoin protocol [57].

Won et al. [6] present a PKI called IoT-PKI that is decentralized and built on a blockchain network. IoT-PKI uses distributed nodes to handle scalability. User-controlled certificates are supported by IoT-PKI, which avoids single points of failure. It works with a name-value storage in the blockchain, where the name stands in for the identity and the value for the hash of a certificate. Through the use of a prototype, they have demonstrated the viability of IoT-PKI. According to their experimental findings, IoT-PKI can verify a certificate's revocation status between three and twenty-six times faster than traditional PKI can do so using Online Certificate Status Protocol (OCSP).

Hoogland [58] made *DECKIN*, a PKI solution on top of an existing blockchain protocol. It provides a practical key-management solution, a verification procedure that is used anytime a node wants to enroll an identity, and it is made for low-resource IoT devices. It uses cryptographic accumulators so that resource-constrained devices do not have to traverse the entire blockchain to find a certificate. Additionally, it employs Physical Unclonable Functions (PUFs) to address the system's key management issues. Finally, their experimental results show that certificate verification takes only 0.025 ms, and the average protocol runtime is 35 ms.

Three distinct blockchain-based alternatives to conventional CA-PKI for certificate administration are proposed and analyzed by Singla and Bertino [59]. The first proposal makes use of the Emercoin blockchain that provides a Name Value Storage (NVS). The second approach employs Ethereum smart contracts, and the final proposal uses Ethereum Light Sync mode, which does not require a remote blockchain node, unlike the first two proposals. In addition to offering a more reliable and scalable PKI, they demonstrate their viability in the context of resource-constrained IoT devices in

terms of computing power, memory, and storage capabilities. According to the authors' analysis, certificate verification requires 0382 MB of storage space and takes 128250 milliseconds, depending on the alternative. They show that the implementations outperform certificate verification in conventional PKI with OCSP check by 1.73 times, whereas they underperform conventional PKI without OCSP check by 2.24.2 times.

Magnusson [60] evaluated the performance of an existing PKI [61] that uses smart contracts on the Ethereum blockchain, by deploying it to a Raspberry Pi 2. The author found that deploying the PKI to this IoT-like device required over 20 GB of storage to store the blockchain. Furthermore, synchronizing the blockchain resulted in significant CPU and RAM utilization. The author deduces from the test findings that these issues hinder such a solution from being a practical replacement for existing solutions.

An alternate PKI methodology for the IoT is put forth by Pintaldi [62]. In order to provide an implementation that reaps the advantages of the original approach and address some of its security issues, it draws on a previous study by Won et al. [6] The Trusted Platform Module (TPM) technology, which offers a reliable device identification method, has been used to implement and improve the original solution. The proposed solution has also been the subject of various tests to highlight potential advantages and disadvantages in comparison to the conventional centralized PKI. According to their experimental results, a handshake in this blockchain-based system takes anywhere between 5 and 13.5 times longer to complete. However, providing and managing keys and certificates for IoT devices is significantly easier. Additionally, this solution's distributed design addresses the traditional PKIs single points of failure.

In the context of the Internet of Things, the thesis of Champagne [30] proposed a workable substitute for PKI based on Certificate Authorities. They tested their design by implementing a proof of concept on the Ethereum blockchain. This proof of concept uses a smart contract as its core, which represents the trusted authority. Through their findings, they discovered that the solution addressed the drawbacks of the conventional PKI while being suitable for the IoT. They demonstrate that the complete framework uses about 20 Kb of RAM, and certificate signing takes about a minute. Additionally, the required maximum upload and download speeds are roughly 22 Kbps and 15 Kbps, respectively. Nevertheless, under this framework some entities' identities are not verified, which could lead to a DoS attack.

4.3 DNA cryptography

In DNA cryptography, DNA is employed as an information carrier, which explores DNAs excellent energy efficiency, and high information density inherent in DNA molecules for cryptographic uses like signatures, encryption, and authentication [63].

A unique DNA mapping method for ECC is given in Tiwari and Kim [64]. This mapping converts plaintext sequences to a pseudo-random sequence before encryption, and makes the overall scheme thus more secure. Both encryption and decryption are performed using the mapped characters. The proposed scheme's security research reveals that existing ECC systems are more resistant to timing attacks when used with the suggested DNA mapping. The suggested system was successfully implemented on an IoT device. They demonstrate that the suggested encryption and decryption take a linear amount of time in relation to the size of the input data. As a result, the suggested DNA (re)mapping method strengthens existing ECC solutions without using a lot of resources or time.

Al-Husainy et al. [65] present an adaptable, lightweight encryption scheme for IoT devices. The scheme only uses XOR and rotate operations which are faster than e.g., mathematical operations. Furthermore, the supplied information is represented as 2D matrices, or blocks, of bits using a straightforward structure. As a result, the IoT device's processor will be better able to execute the multitasking idea and manage these blocks. Additionally, it makes use of a configurable block size to increase the system's adaptability to different IoT systems with diverse memory capacities. In comparison to AES, the suggested encryption scheme has demonstrated superior performance in terms of encryption time, according to the authors. The suggested encryption method makes

use of DNA sequences to generate keys with a high level of randomness that makes them difficult for attackers to decrypt. Additionally, the authors show that the suggested encryption scheme had high confusion and diffusion effects. Moreover, the proposed encryption scheme passed the avalanche test with a score of above 50%, indicating that it is resistant to attacks utilizing statistical analysis.

4.4 Pairing-based cryptography

A pairing is a mathematical function that takes two elliptic curve points as inputs and produces an element in a finite field [66]. Pairing-based cryptography makes use of such a pairing to create cryptographic primitives.

In Yu and Li [67], a novel anonymous authentication key agreement method for home-based IoT using pairing-based cryptography is presented. This scheme enables secure communication in an architecture of servers and resource-constrained IoT devices in a home area network, and users outside the network. The scheme consists of three steps; the initialization step, which is performed by the home server, the anonymous registration step, which is performed by the sensor devices, and finally, the anonymous authentication key agreement step, in which all devices generate session keys. The new protocol conceals the devices' and the user's identities. Finally, they use BAN logic [68] to prove the security of the scheme, which demonstrates that communication between sensors, users and servers is secure. Additionally, their performance analysis demonstrates that the entire scheme takes around 3.1 seconds to run.

Szczechowiak et al. [42] implemented three pairing-based cryptography systems for wireless sensor networks. The schemes allow new nodes to be added at any time, scale well, and there is no extra communication overhead caused by the cryptographic operations. Their implementations have been tested on a variety of WSN processors, and all the implementations are the quickest at the time of writing, according to the authors. Additionally, they demonstrate that the pairing schemes require between 34 and 47 KB of ROM. Furthermore, they offer a novel key exchange protocol that may be advantageous in more computationally limiting scenarios.

4.5 Physical Unclonable Functions

A piece of hardware known as a Physical Unclonable Function (PUF) provides a physically unique response (among PUFs) for a given input, because it has intrinsic randomness created during production [69].

Chanda et al. [27] designed a lightweight PKI system for usage in CPS devices. The suggested solution introduces the idea of session keys to deal with certificate verification and revocation and employs Physical Unclonable Function (PUF) to make the key generation process simpler. The suggested approach also takes into account PKI scheme security requirements such node registration and secure random number generation. Their experimental findings show that the average duration of key generation is 4.68 μs and requires 1.31 μJ of energy. Furthermore, they demonstrate that the proposed solution requires 680 KB of storage and 2280.67 KB of communication on average. Finally, they proved the scheme to be secure under the Real-Or-Random (RoR) model and is resilient against DoS, and malicious public key changes.

Siddiqui et al. [70] introduce an IoT network and device authentication protocol based on PUF-based digital certificates. The protocol introduces a secure key mechanism to defend PUF IoT devices from differential analysis attacks. To give a verifiable security analysis through authentication and verification procedures, they validate the proposed approach using the Tamarin prover. The protocol also protects devices against message manipulation, impersonation and replay attacks, session forgery and hijacking, according to the authors.

4.6 Miscellaneous articles

Schukat and Cortijo [71] present a number of PKI for IoT criteria and demonstrate how they need to be modified and tweaked before PKI can be effectively implemented in the IoT ecosystem. The requirements were initially introduced for the internet and traditional client/server communication. Reducing certificate size, utilizing only a root CA and no intermediate CAs, generating entropy using CAs (rather than the devices themselves), and using multiple certificates per device are a few of these modifications.

In order to address the issue of the CAs availability, Aljadani and Gazdar [72] propose a novel distributed PKI for WSN-based applications. In this architecture, numerous nodes are elected to be CA of their assigned cluster. Once a node is selected to be CA, it will hand out short-term certificates to other nodes in its cluster. Additionally, a RA in each cluster will aid in removing erroneous sensor nodes. According to the authors, the proposed architecture can withstand numerous attacks like sinkholes and DoS. Nevertheless, no performance analysis is available.

Forsby et al. [73] have supplied compression and encoding algorithms for X.509 certificates and have specified a lightweight variant of the X.509 certificate for the IoT. The lightweight certificate is compliant with the X.509 standard, which enables use in all current PKI implementations. Their experimental results show that the lightweight certificate variant is 4.5× smaller than a regular certificate, and even 14.8× smaller with the proposed compression.

In Höglund et al. [7], the authors outline several obstacles to enabling PKI for the Internet of Things, as well as two solutions: certificate overhead reduction and secure enrollment. They do this by developing a new kind of X.509 certificate and shrinking its size by employing CBOR encoding. They have demonstrated their ability to securely complete initial enrollment and re-enrollment, and minimize X.509 overhead for the intended IoT applications. The authors demonstrate that the protocol uses around 4 KB of ROM and 1 KB of RAM. Furthermore, the compressed ECC certificates are only 150 bytes in size. Nevertheless, the proposed system does not protect itself against DoS attacks.

PKIoT is an architecture that Marino et al. [74] propose with the goal of making certificate-based authentication practical for IoT devices with limited resources. The PKIoT architecture enables IoT nodes to delegate difficult security-related activities to a remote server. According to their present condition and level of trust in the server, nodes can freely choose which tasks to delegate. The PKIoT architecture provides an expandable, compatible, and flexible solution as a result. They also created a novel sort of compact certificate, which when used in place of standard X.509 certificates allows for even more reductions in transmission overheads but necessitates PKIoT support on both ends of the communication. Finally, their experimental results show that PKIoT is around 12× faster in key generation, 10× and 12× faster in signature generation and verification respectively, in comparison with not employing the PKIoT architecture.

A protocol based on Identity-Based Encryption (IBE) is proposed by Anggorojati and Prasad [75] to secure inter-domain communication on the Internet of Things. The major contributions include an IBE-based key-escrow-free authentication mechanism, a system for looking up IBE parameters in other domains, and a method for creating shared secret keys for secure device communication. According to the authors, the method offers mutual authentication, however there is no performance analysis present.

In order to highlight the potential issues that need to be addressed, Diaz-Sanchez et al. [76] give a thorough analysis of numerous security facets of PKI and Transport Layer Security (TLS), with a special emphasis on existing certificate pinning methods. They examined and contrasted the various certificate pinning proposals in the IoT context. They discovered that IoT adoption of Certificate Transparency (CT) is necessary in a scenario including a widespread attack. Nonetheless, global attack detection may not be possible given the heterogeneity of systems and devices.

Henriques and Vernekar [31] suggest a method for securing communication within an IoT system that combines symmetric and asymmetric cryptography. Compared to utilizing only asymmetric cryptography, asymmetric and symmetric cryptography together speed up encryption. Nevertheless, the article does not provide a performance and security analysis, and is vulnerable to timing

attacks as the random keys are generated using a timestamp as the seed.

4.7 Discussion

This section will discuss the results of the SLR by listing the advantages and disadvantages of the proposed technologies, that have emerged from the literature review (Figure 4.1). A concise overview is given in Table 4.1.

Elliptic Curve Cryptography (ECC) is a public key encryption technique that can be used for creating a Public Key Infrastructure (PKI). Here are some of the advantages and disadvantages of using ECC for creating PKI:

- Advantages:
    1. Security: ECC is considered to be more secure than traditional public key encryption algorithms like RSA, as it requires smaller key sizes to provide the same level of security [44, 45]. This makes ECC a good choice for systems where resources are limited.
    2. Key management: ECC allows for easier key management, as it requires shorter key lengths than other encryption techniques, making key distribution and storage more manageable.
- Disadvantages:
    1. Slower signature verification: while ECDSA signature generation is faster than RSA, it is much slower in verification, compared with RSA [77]. This is due to the fact that ECDSA signature verification is computationally more complex than generation.
    2. Patents and licensing: Certain ECC algorithms and the ways in which they are implemented may be covered by patents. This may restrict their use and adoption in particular applications, especially in open-source initiatives [78].

Decentralized technologies can be used for creating a PKI, where trust and authentication are established through a decentralized network instead of a central authority. Here are some of the advantages and disadvantages of using decentralized technologies for creating a PKI:

- Advantages:
    1. Security: Decentralized PKI can be more secure than traditional PKI, as it is less susceptible to attacks on central authorities or single points of failure. Instead, trust and authentication are established through a distributed network of nodes, making it harder for attackers to compromise the system.
    2. Transparency: Decentralized PKI can offer greater transparency and accountability, for example when all transactions and operations are recorded on a public blockchain or distributed ledger. This can increase trust in the system and reduce the potential for fraudulent activities.
- Disadvantages:
    1. Governance: Decentralized PKI requires a governance model to ensure that the network is secure, reliable, and operates in the best interest of its users. This can be difficult to achieve, and disagreements over governance can lead to network fragmentation and reduced trust in the system [79].
    2. Trust model complexity: decentralized PKI systems might necessitate that users build trust connections with numerous authorities or entities, which could make the trust model more complex. In contrast to a centralized PKI with a single trusted authority, managing and confirming trust relationships across many domains might be more difficult.

DNA cryptography is an emerging field that explores the potential of using DNA molecules for encryption purposes. While it is an innovative technology, it is still in the early stages of

development, and there are several advantages and disadvantages to consider when using DNA cryptography for creating a PKI:

- Advantages:
  1. High Information Density: The information contained in DNA molecules is extraordinarily dense. Large amounts of data can be stored on each DNA strand, which could make it possible to encrypt and store a lot of data in a small space [80].
  2. Energy efficiency: utilizing the parallelism of DNA strands, cryptographic processes based on DNA can be carried out simultaneously. The simultaneous processing of many DNA sequences increases the effectiveness of cryptographic operations [63].
- Disadvantages:
  1. Complexity: DNA cryptography is a complex field that requires specialized knowledge and expertise to implement. This makes it challenging for organizations to adopt and implement the technology.
  2. Key reuse: Traditional cryptography, is resistant if the attacker lacks sufficient processing power. Increasing the key size is a simple way to increase security. Since a DNA key is difficult to modify, it is essentially a one-time pad. Reuse is not possible [81].

Pairing-based cryptography is a type of public key cryptography that uses a pairing operation to encrypt and decrypt data. Here are some of the advantages and disadvantages of using pairing-based cryptography for creating a PKI:

- Advantages:
  1. Advanced techniques: Pairing-based cryptography makes it possible to perform advanced cryptographic operations that are impractical or time-consuming to carry out using conventional cryptographic methods. It offers features such as efficient zero-knowledge proofs [82], identity-based encryption [83], and attribute-based encryption [84].
  2. Better security: Pairing-based cryptography offers higher levels of security as it uses complex mathematical operations to encrypt and decrypt data. It requires therefore a smaller keysize than e.g., RSA for comparable security [85].
- Disadvantages:
  1. Complexity: Pairing-based cryptography is a complex system that requires specialized knowledge and expertise to implement. This makes it challenging for organizations to adopt and implement the technology [85].
  2. Key escrow: With the traditional PKI system, the problem of key escrow in pairing-based cryptography does not exist [85]. Key escrow is a system in which the decryption keys required to open encrypted data are kept secret by a third party. This can have severe consequences for confidentiality and integrity when this third party is not the actual owner of the key.

Physical Unclonable Function (PUF) is a technology that uses physical characteristics of a hardware device to generate unique keys for encryption and decryption. Here are some of the advantages and disadvantages of using PUF technology for creating a PKI:

- Advantages:
  1. Post-Quantum Security: PUFs may provide defense against attacks from quantum computers. PUFs offer post-quantum security because they are less vulnerable to attacks based on quantum algorithms because they depend on inherent physical properties rather than on mathematical calculations [86].
  2. Uniqueness: Based on the physical characteristics of the device, PUFs provide distinct and unpredictable values. PUFs are excellent for a variety of security applications,

including device authentication and secure key creation, because of unique physical variations that occur during the manufacturing process [87].

- Disadvantages:
  1. Variability and reliability: Environmental changes might affect the physical characteristics used by PUFs. The security and effectiveness of PUF responses may be impacted by this variation, which might make it difficult to guarantee their accuracy and consistency.
  2. Aging and degradation: A PUFs response can alter as a result of a device's physical attributes deteriorating or changing over time. PUFs dependability and longevity may be impacted by aging effects including temperature changes or extended usage [88].

Table 4.1: Advantages and disadvantages of the proposed technologies found in the literature study.

Technology	Advantages	Disadvantages
Elliptic Curve Cryptography	Security	Slower signature verification
	Key management	Patents and licensing
Decentralized technologies	Security	Governance
	Transparency	Trust model complexity
DNA Cryptography	High information density	Complexity
	Energy efficiency	Key reuse
Pairing-based cryptography	Advanced techniques	Complexity
	Better security	Key escrow
Physical Unclonable Functions	Post-Quantum security	Variability and reliability
	Uniqueness	Aging and degradation

4.8 Conclusion

As seen in the previous section, choosing a technology for implementing a lightweight PKI for the IoT is difficult as each has advantages and disadvantages. Furthermore, there is a vast amount of material that proposes innovative PKI designs and cryptographic protocols designed specifically for IoT devices. However, several of these ideas have little to significant limitations:

- Some works make use of cryptographic algorithms (e.g., RSA) that will have severe performance limitations when deployed on resource-constrained IoT architectures [31]. Other cryptosystems, such as ECC provide much higher levels of security for much smaller keysizes, thus reducing computational and communication overhead [89].
- Various works propose centralized PKI solutions, which impose a single point of failure because of their centralized nature [27–29]. As a result, if an attacker manages to take over the CA, they will be able to issue certificates for whomever they choose. Furthermore, it is impractical to untrust certificates in the event of a CA breach in a centralized system. If a CA is found to have issued fake certificates, the only way to ensure certificate authenticity is to ban all certificates issued by the compromised CA. However, it's probable that this might also result in the denial of certificates that were not malicious, which would have a bad effect on legitimate users.
- Some works are prone to Denial-of-Service (DoS) attacks [7, 30]. DoS attacks are designed to flood a targeted system with malicious traffic or requests in order to overrun or deplete its resources, such as a server or network. As a result, the targeted system's regular operation is significantly hindered or completely interrupted, making it inaccessible or useless to authorized users. Businesses, organizations, or individuals who depend on the target system's accessibility and functionality may suffer serious repercussions as a result of this interruption.
- Adversaries can drastically restrict the possible keyspace in proposed works [31] because the seeds used to generate random keys are timestamps. In turn, this makes brute-forcing keys very efficient.
- Some protocols reuse one-time pads [32]. This introduces a major vulnerability; if adversaries are able to obtain two ciphers XORed with the same one-time pad (key), they are able to

retrieve the XOR result of the two plaintext messages. With this, they can perform statistical analyses in order to obtain both plaintexts individually.

Finally, in order to create a comprehensive overview of the reviewed papers, we extracted features from the papers and aggregated them into a table. Examples of these features are authors, performance and security analysis. The result can be seen in Table 4.2 and 4.3. We decided to make a clear distinction between specific PKI papers (Table 4.2), and more general cryptography articles (Table 4.3), such as novel authentication and encryption protocols. This is because the former category of articles will be used to discover knowledge gaps in the existing literature, while the latter category can be used as building blocks for our PKI implementation in Chapter 5. In both tables, the comparison column lists the comparison the article itself conducted against related works, which can be in terms of performance or security. If this column does not tell what the work is compared with, e.g., "3× *faster*", it must be assumed that the performance is compared with peer articles. Additionally, we made the decision to not include the "Architecture" column in Table 4.3 because, for the majority of the proposed solutions, it was irrelevant whether the architecture was centralized or decentralized because the majority of these solutions were simply encryption or authentication protocols, for example.

The next chapter will propose a novel PKI architecture with the knowledge gained in this chapter. It will introduce a decentralized PKI architecture, and propose a new type of "lightweight" certificate, in order to reduce communication overhead.

Table 4.2: Overview of the reviewed papers that present novel works regarding PKI for IoT.

Name/ authors	Architecture	Technology used	Security analysis	Performance analysis	Comparison
IoT-PKI [6]	Decentralized	Blockchain	Mitigations / solutions for: • Private key compromise • Stolen IoT device • Weak random number generator on client • No available blockchain nodes	Certificate verification time: • At blockchain node: 10 ms • At IoT device: 128 ms	3 to 26 times faster than traditional PKI
Schukat and Cortijo [71]	Centralized	X.509 optimizations	Analysis not present	Analysis not present	Analysis not present
DECKIN [58]	Decentralized	Blockchain, PUF	Secure against: • Spoofing key updates • Spoofing key revocations	• Certificate verification time: 0.025 ms • Average protocol runtime: 35 ms	Analysis not present
Chanda et al. [27]	Centralized	PUF, ECC	• Proven secure under RoR model Resilient against: • Denial of Service • Malicious public key changes	Key generation: • Duration: 4.68 μs • Energy: 1.31 μJ Memory usage: • 680 KB on device • 2280.67 KB in communication	• $14-16\times$ faster • Requires $18-25\times$ less power
Aljadani and Gazdar [72]	Distributed	Clustering technique	Resilient to: • Sinkhole attacks • DoS	Analysis not present	Analysis not present
Siddiqui et al. [70]	Decentralized	PUF	Secure against: • Chosen plaintext attack • Session hijacking • Session forgery • Impersonation attacks • Message tampering • Replay attacks	Analysis not present	Provides more security features than other related works, such as device security and privacy
Singla and Bertino [59]	Decentralized	Blockchain	Analysis not present	• Certificate verification: 128-250 ms • Storage required: 0-382 MB • Cost: \$0.07-0.18 per certificate • Time to issue certificate: 1-10 minutes	• $1.6-3\times$ faster than traditional PKI (with OCSP check) • $2.2-4.2\times$ slower than traditional PKI (without OCSP check)

Continued on next page

Table 4.2 – *Continued from previous page*

Name/ authors	Architecture	Technology used	Security analysis	Performance analysis	Comparison
μPKI [28]	Centralized	ECC	Ensures: • Confidentiality • Authentication • Integrity	• Encrypting session key: 22.82 mJ • Sending session key: 3.78 mJ • Receiving session key: 1.83 mJ • Sensor handshake: 3.70 mJ	• $1.23 - 1.48\times$ less energy than simplified Kerberos • $2.92\times$ less energy than simplified SSL
Magnusson [60]	Decentralized	Blockchain, smart contracts	Analysis not present	"Significant" CPU and RAM utilization	Analysis not present
Pintaldi [62]	Decentralized	Blockchain	Analysis not present	Analysis not present	Handshake is $5 - 13.5\times$ slower than in conventional PKI
LPKI [29]	Centralized	ECC	Analysis not present	Analysis not present	Analysis not present
Champagne [30]	Decentralized	Blockchain	Some identities are not verified, which could result in a DoS attack.	• Certificate signing: ~ 1.2 min • Memory usage: 20 Kb • Network usage: Up: 22 Kb/s, down: 15 Kb/s	Analysis not present
Höglund et al. [7]	-	X.509 optimizations	Analysis not present	• Average header size: 10 bytes • Compressed ECC certificate: ~ 150 bytes in size • ROM use: 3.7 KB • RAM use: 1.1 KB	• Average CoAP header size is $16\times$ smaller than HTTP
PKIoT [74]	Centralized	X.509 optimizations	Potentially vulnerable to: • Denial of Service • Eavesdropping • Tampering • Masquerading	• Key generation: 348 ms • Signature generation: 434 ms • Signature verification: 429 ms • Certificate verification: 469 ms, 16.57 mJ	• Key generation: $\sim 11.7\times$ faster • Signature generation: $\sim 9.5\times$ faster • Signature verification: $\sim 12.2\times$ faster
Diaz-Sanchez et al. [76]	-	Certificate pinning, certificate transparency	Analysis not present	Analysis not present	Analysis not present

Table 4.3: Overview of the reviewed papers that present novel works regarding cryptography for IoT.

Name/ authors	Technology used	Security analysis	Performance analysis	Comparison
Yu and Li [67]	Pairing-based encryption	Secure against: • User impersonation attack • Replay attack • Insider attack Other properties: • Key forward secrecy • Strong key establishment • Proven secure through BAN logic	• Whole scheme cost: 3.1 s	$1.48 - 8.38\times$ slower than peer works
Tiwari and Kim [64]	DNA	• Through the proposed DNA mapping, existing ECC is more resilient to timing and SPA attacks. The scheme: • Converts repetitive data to pseudo-random data	• Encryption and decryption take a linear amount of time in relation to the input size	Analysis not present
Tewari and Gupta [47]	ECC	Properties: • Mutual authentication • Anonymity • Confidentiality • Availability • Resistant to DoS	• Storage cost: 576 bits • Communication cost: 1152 bits	• $0.69 - 1.64\times$ less storage required • $0.88 - 1.63\times$ more communication required
Szczechowiak et al. [42]	Pairing-based cryptography	Analysis not present	• The first pairing scheme requires on average 33.54 KB of ROM on the tested platforms • The second pairing scheme requires on average 46.73 KB of ROM on the tested platforms • The fastest implementations at the time of writing	Analysis not present

Continued on next page

Table 4.3 – *Continued from previous page*

Name/ authors	Technology used	Security analysis	Performance analysis	Comparison
Shi and Gong [48]	ECC	Provides: • Mutual authentication • Perfect forward secrecy Resistant to: • Replay attack • User impersonation attack • Sensor impersonation attack • Gateway impersonation attack • Man-in-the-middle attack • Insider attack	Approx. 30% faster than its predecessor [49]	Approx. 30% faster than its predecessor [49]
ECIOT [32]	ECC, DH	Analysis not present	Analysis not present	Analysis not present
Rajendiran et al. [50]	ECC	Highly resilient to: • Sybil attack • Random attack • Brute force attack	Analysis not present	Superior resilience to attacks like brute force and Sybil
Qazi et al. [45]	ECC	Provides: • User anonymity • User untraceability • Resistance to various attacks • Session key agreement • Mutual authentication	• Key generation between two nodes: 50 ms • Authentication between two nodes only requires six packets	Analysis not present
Louw et al. [52]	ECC	Analysis not present	Analysis not present	Analysis not present
Ju [53]	ECC	• Resilient against MITM • Perfect forward secrecy	• Storage linear to number of nodes • 10 seconds to establish a 10-node network	Analysis not present
Elhoseny et al. [41]	ECC	Secure against: • Passive attack • Brute force attack • Compromised CH • Sinkhole attack • DoS attack • HELLO flood attack	• CPU cycles: 66201 • Time: 8.619 ms • RAM 281 bytes • ROM 3845 bytes	• $0.94 - 8.9$ fewer CPU cycles • $0.99 - 9.16\times$ faster • $1.04 - 3.47\times$ less RAM • $1.37 - 1.88\times$ less ROM

Continued on next page

Table 4.3 – *Continued from previous page*

Name/ authors	Technology used	Security analysis	Performance analysis	Comparison
Pinol et al. [54]	ECC	Analysis not present	For a 256-bit key size: • Key generation: ∼5000 ms, 75.93 mJ • Signature verification: ∼4-11 s, 153.84 mJ • Signature generation: ∼5 s, 76.23 mJ	Analysis not present
Bai et al. [46]	ECC	Provides: • Authentication • Integrity • Confidentiality	Analysis not present	Analysis not present
Liu and Seo [55]	ECC	Protected against: • Timing attacks • Simple side-channel attacks	Clock cycles: • NUMS256: 543/429 • Ted37919: 1126/884 • NUMS384: 1139/898	NUMS256 is 1.6× faster than Curve25519
Al-Husainy et al. [65]	DNA	• Peak signal-to-noise ratio comparable to AES	• Key size: 24 bit	• $1.5 - 5.4\times$ faster than AES • Peak signal-to-noise ratio comparable to AES
Lara-Nino et al. [56]	ECC	Analysis not present	Runtime: 2.69-6.72 ms	• Requires fewer slices • $0.01 - 8.6\times$ faster
Forsby et al. [73]	X.509 optimizations	Analysis not present	• Optimized X.509 certificate: 324 bytes • Compressed: 146 bytes	• 4.5× smaller than a regular certificate • 14.8× smaller than a regular certificate (with compression)
Anggorojati and Prasad [75]	Identity-based cryptography	• Mutual authentication	Analysis not present	Analysis not present
Khari et al. [43]	ECC	• Mean squared error: 0.02 • Peak signal-to-noise ratio: 70 dB	• Time for encryption and decryption: 0.4 sec	• $1.5 - 2\times$ faster • $52.5 - 75\times$ lower mean squared error • $1.5 - 2.3\times$ higher peak signal-to-noise ratio

Continued on next page

Table 4.3 – *Continued from previous page*

Name/ authors	Technology used	Security analysis	Performance analysis	Comparison
Albalas et al. [44]	ECC, CoAP	Ensures: • Confidentiality • Authorization • Integrity • Authentication	Power required: ~ 0.7 W	Requires $1.14 - 1.57\times$ less power than regular CoAP
TinyECC [39]	ECC	Analysis not present	• Initialization: 5.64-5202 ms, 1.4-83.84 mJ • 11.40-20.77 KB of ROM • 1.5-2.1 KB of RAM	Analysis not present
Henriques and Vernekar [31]	(A)symmetric cryptography	Analysis not present	Analysis not present	Analysis not present

Chapter 5

Proposed solution

The demand for a scalable and decentralized public key infrastructure (PKI) is on the rise as the Internet of Things (IoT) landscape expands at a rapid rate, connecting everything from sensors to appliances. A lightweight and decentralized PKI offers an appropriate solution to the particular security issues that IoT ecosystems raise.

Firstly, a PKI architecture for the IoT must be decentralized due to the nature of IoT networks. Decentralized PKIs reduce single points of failure and boost attack resistance simply by doing away with the need for a centralized authority to administer certificates. Devices may independently manage their cryptographic keys, create secure connections, and authenticate one another via a decentralized PKI, which promotes assurance within the IoT ecosystem.

Secondly, a lightweight PKI system is a PKI system created especially for IoT devices with limited resources. Such devices frequently feature little amounts of memory, computing power, and energy. The performance and efficiency of IoT devices can be hampered by traditional PKI structures, which were initially created for more powerful computing environments. IoT devices can use cryptographic processes that are tailored to their capabilities by implementing a lightweight PKI, which reduces processing overhead and energy usage while assuring reliable operation.

This chapter will describe the proposed solution for a novel PKI architecture tailored to IoT devices. The architecture is mainly based on findings during the literature analysis (Chapter 4) and hopes to address the limitations presented in Section 4.8. The contributions of this proposed solution are two-fold:

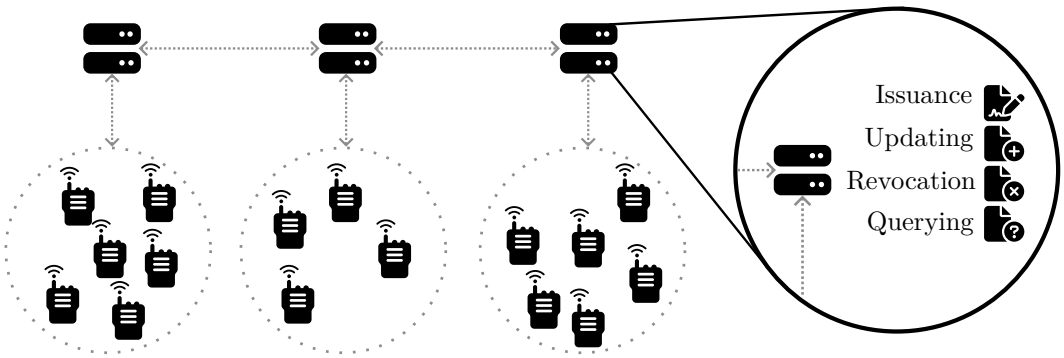
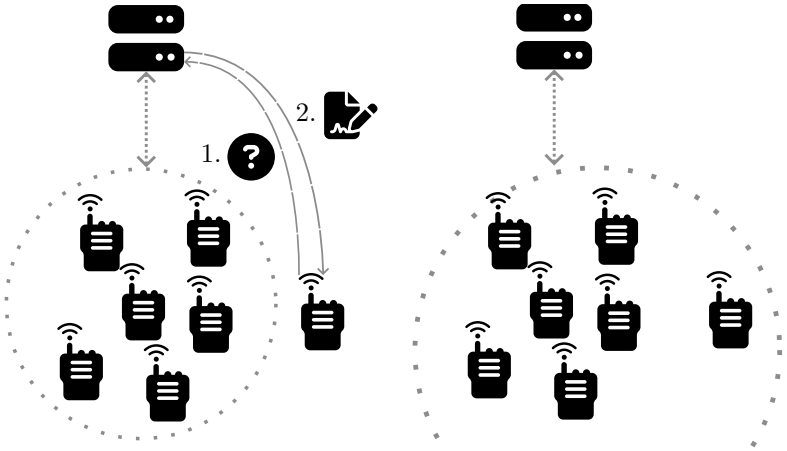


Figure 5.1: Overview of the proposed design. The gray circles represent zones, in which IoT devices are situated. The IoT devices are individually connected to their zone master located above them. Moreover, the zone masters are able to communicate among their neighbors.



(a) A node wants to be enrolled into the zone, by sending an enrollment request to the zone master. The zone master sends back a signed certificate.

(b) Next, the node is part of the zone.

Figure 5.2: Node enrollment procedure visualized.

1. In order to address the single point of failure present in traditional PKI architectures, we propose a decentralized PKI system. Previous literature has shown that this is feasible to achieve [6, 30, 58–60, 62, 70, 72]. We propose to introduce an architecture that employs "zones", where each zone has a master (See Figure 5.1). This master issues, updates, revokes, and looks up certificates for all other IoT nodes in its zone. A node can become part of a zone by sending an enrollment request to a zone master, to which the master responds with a signed certificate (Figure 5.2). Thus, to a certain extent it acts as a CA when compared with the traditional PKI architecture. Furthermore, it is optimal for the master node to have significant computational power. The general idea is that zone masters keep track of the certificates of all nodes in its zone. If a zone gets too many nodes to handle, it will convert a node in the zone to a new master node, thereby creating a new zone, which is connected to the old zone. This creates a parent-child like architecture, where each zone (apart from the root zone) has a link to its parent and child zone. This zone system allows for high scalability, where each zone can represent a single entity. For example, one zone can contain all IoT nodes in a certain smart city, while another zone contains all nodes in a certain smart building.

2. The second contribution addresses the need for a lightweight architecture, which is achieved in two following ways:

   (a) The use of lightweight cryptography: Elliptic Curve Cryptography (ECC) is public key cryptography based on the algebraic structure of elliptic curves over finite fields. Because it offers the same level of security with smaller key sizes, it has an advantage over conventional cryptographic methods like RSA [89]. ECC starts with an elliptic curve that is described mathematically by the equation:

   $$y^2 \equiv x^3 + ax + b \pmod{p} \tag{5.1}$$

   where x and y are coordinates on the curve, a and b are constants that define the shape of the curve, and p is a prime number representing the finite field. One cryptosystem that makes use of ECC is the Elliptic-Curve Discrete Logarithm Problem (ECDLP); given the curve in equation 5.1 and a point P on the curve, the ECDLP involves finding an integer n such that nP equals a specified point Q on the curve. In mathematical notation, the ECDLP can be expressed as:

   $$Q = nP$$

Table 5.1: Symbols and notations used in this chapter.

Symbol	Description
H	Hardware score of a node
U	Cryptographically secure Universally Unique Identifier (UUID)
C	Node certificate
S	Signature of C
PK	Public Key
SK	Secret Key
R	Revocation status (0 = not revoked, 1 = revoked)
$\{M\}_k$	Message M is encrypted with key k

The security of elliptic curve cryptography relies on the difficulty of solving the Elliptic-Curve Discrete Logarithm Problem (ECDLP). That is, given a curve, a base point P, and a resulting point Q, it is computationally hard to determine the integer n such that $Q = nP$ [90].

(b) The use of lightweight certificates: the proposed architecture will dramatically reduce the number and size of fields present in the conventional X.509 certificates in order to speed up certificate creation and verification. To further reduce communication overhead, we will use Concise Binary Object Representation (CBOR) to encode the certificates. CBOR is a binary data serialization format that resembles JSON in some ways. It enables the faster transport of data objects with name-value pairs than JSON. This sacrifices human comprehension in favor of more rapid processing and transfer rates.

We shall now describe a number of important procedures in this architecture. In these descriptions, we sometimes make use of symbols. Definitions of the symbols can be found in Table 5.1. Finally, lifecycle diagrams of both nodes and zones can be found in Figure 5.3.

5.1 Zone setup

In order to set up a zone, a single zone master node must be deployed. Firstly, this zone master generates a cryptographically secure Public Key PK, Secret Key SK and UUID U for itself. The zone master will essentially act as a CA, compared with traditional PKI architectures. In order for the zone to be used by other nodes, its IP address must be published. When nodes obtain the IP address of a zone, they can enroll into the zone, which is described in the next section.

The zone that is set up first will be the zone whose IP address will be published for other nodes to join. This means that if a node does not know the IP address of a zone to join, it will try to join the first zone. After that, the zone master will decide to either welcome the node into its zone, or to redirect its enrollment request to a child zone.

It is important to note that this operation is only applicable for zone masters who want to create a brand new PKI chain. When a node wants to assist this PKI by becoming a zone master, it will first have to join the PKI as a regular node.

5.2 Node enrollment

If a node wants to become part of a zone, it will first execute an Elliptic-Curve Diffie-Hellman (ECDH) key exchange so that subsequent communication can preserve confidentiality and integrity. The Diffie-Hellman key exchange technique enables two parties with no prior knowledge of one another to establish a common secret key across an insecure channel [91]. A symmetric-key cipher can then be employed with this key to encrypt subsequently sent messages. After the node and zone master have conducted a Diffie-Hellman key exchange to obtain a secure symmetric key σ, the node

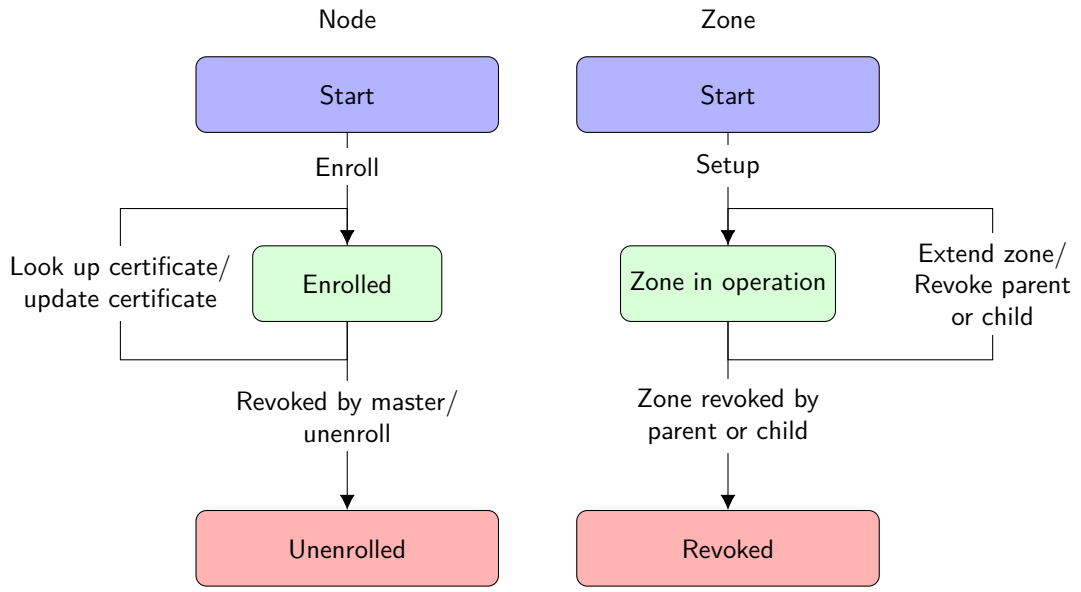


Figure 5.3: Lifecycle diagrams of a node and zone. The transitions between states contain hyperlinks to their respective sections.

will use it to securely send an enrollment request to the zone master facilitated by the Advanced Encryption Standard (AES) block cipher. This enrollment request contains the hardware score H of the node, which is calculated as follows:

$$H = (\text{Amount of RAM in MB}) + (\text{number of CPU cores}) \times (\text{CPU clock speed in MHz}). \quad (5.2)$$

Optionally, depending on the level of trust in the zone master, the node will also generate a keypair itself and send the public key along as well. Thus, the node sends the following tuple to the zone master, where square brackets indicate optional fields:

$$\lambda = \{([PK], H)\}_{\sigma}.$$

Next, the zone master generates a cryptographically secure UUID U, PK and SK (if the node did not send along a public key), and certificate C. The certificate contains the UUID U and public key PK, among others (Section 5.10). The certificate is signed with the SK of the zone master, and stored in signature S. Moreover, it sends the tuple

$$\mu = \{([PK, SK], C, S)\}_{\sigma}$$

back to the node. Finally, the zone master stores the tuple

$$\nu = (U, H, C, R = 0)$$

in an internal table, where R is the revocation status of the certificate. Because the certificate contains (among others) the public key of the node, the public key can be queried by other nodes if they want to securely communicate. We want to note that the zone enrollment relies on the principle of Trust on First Use (TOFU) [92]. That is, the zone master assumes the node to be benign and to not spoof their identity (UUID U) upon key generation. Moreover, we want to point out that after enrollment, all communication between zones and nodes is done through public key cryptography (ECC) to ensure secure and private communication. An overview of the enrollment procedure is given in Figure 5.4.

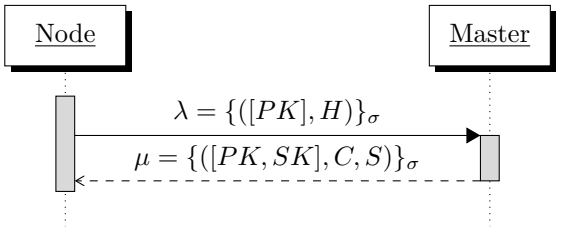


Figure 5.4: Node enrollment. Optional data is denoted by square brackets.

5.2.1 Subsequent communication

After enrollment, the node can communicate with other nodes and its zone master using public key cryptography. Nevertheless, an adversary can perform a Man-in-the-middle attack (MITM) attack, as illustrated in Figure 5.5. In this figure, a benign node requests the certificate of node x to a benign zone master. The MITM however, blocks the request from reaching the master and responds with a malicious response, encrypted with the public key of the node. Although the MITM still has to guess the type of request in order to successfully send back a malicious response, this is a significant problem.

To fix this problem, we propose that for every request, the requesting party sends along a nonce inside the encrypted message. Next, the responding party sends this same nonce inside the encrypted response. If the requesting party successfully verifies that the request nonce is identical to the response nonce, it can confirm that no MITM is present.

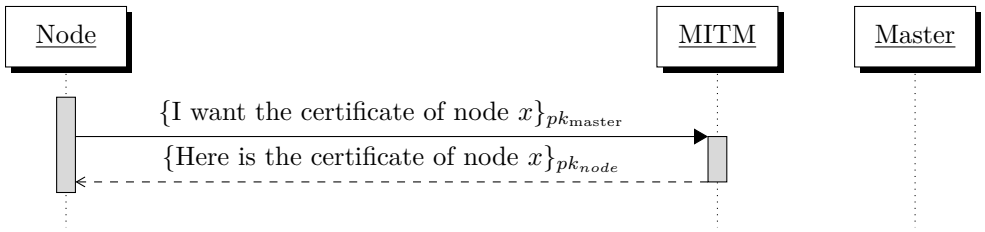


Figure 5.5: Demonstration of MITM attack.

The communication in the rest of this chapter will make use of this nonce principle.

5.3 Zone extension

If a zone master does not want to accept any more node enrollments (e.g., if the zone master has too many nodes resulting in computational limits), it will instruct the strongest node, i.e., the node with the highest H score in its zone, to create a new zone. We explain the next steps with the help of Figure 5.6. In the real architecture, nodes are identified by their UUID, but for readability purposes, we refer to them by a letter. Suppose zone master B wants to extend its zone, it will instruct the strongest node (C) to create a new zone (step 1). After this node (C) has created a new zone, the master of the old zone (B) will redirect the enrollment of new nodes to this new zone (C). Finally, the child (C) sends a recursive message to its parents (B and A) saying that it is the new child of B (step 2). Because zone master (A) knows master (B) is its child from the previous extension, it will respond to zone master (C) that it is C's grandparent (step 3). The grandparent/grandchild identification is performed in order to conduct zone master revocation, which is described in Section 5.8.

Due to accidental or purposeful misconfiguration, it can be possible that two zones redirect to each other upon enrollment of a node. When a node wants to enroll into a zone in such a case, it can get stuck in an infinite redirection loop. To solve this problem, we only allow zone masters to redirect enrollments to their child zones. Furthermore, when a zone master has vacant spots after

a number of nodes have left, it can stop redirection upon enrollment of new nodes and welcome them into its zone.

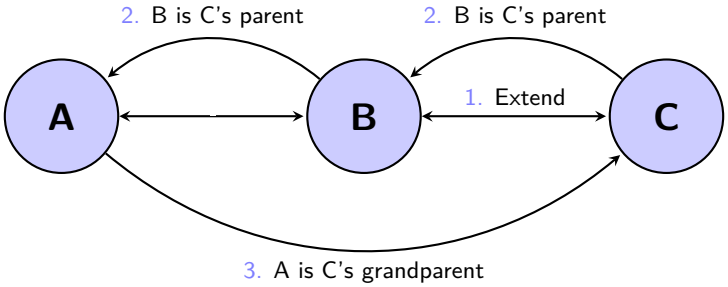


Figure 5.6: Zone extension procedure illustrated.

To conclude this section, we describe how a zone master determines whether it has reached its node limit, after which it will extend its zone. For performance purposes, we describe the following rule: if all nodes in a zone would be consecutively sending a request to the zone master, the zone master must be able to complete all requests within one second. Next, we aim to determine this node threshold for a Raspberry Pi 4 Model B with 1 GB of RAM memory, and a Quad core 64-bit CPU @ 1.8GHz, because this is the testbed for our performance analysis (Chapter 6). From this experiment it is concluded that such a Raspberry Pi is able to handle 20 consecutive enrollment requests within one second. We decided to use the enrollment operation for this benchmark, as it requires a number of cryptographic operations, which would reflect a realistic situation in terms of computing power. Given that this Raspberry Pi has a hardware score of 80031734, we deduced a formula to determine the maximum number of nodes per zone:

$$\text{Maximum number of nodes} = \text{Hardware score } (H) \times 2.5 \times 10^{-7}$$

5.4 Certificate updating

It may be necessary for a node to update their public-private keypair, in case of private key loss or compromise. Should a node want to update its keypair, it can simply send a new certificate signed with the old key to its zone master. Then, the zone master will verify the signature, and if the signature is valid, it will replace the tuple μ in its internal table. In this tuple, U and H are copied from the previous entry, and C is a new certificate of the node's new PK, signed with the SK of the zone master.

5.5 Certificate revocation

In case a node U in a zone is acting maliciously, the zone master must be able to revoke the node's certificate. If this is the case, it can simply set the revocation status R to 1 in its internal table. If other nodes want to look up the certificate of node U, the zone master will inform the nodes that U's certificate has been revoked.

5.6 Certificate lookup

When two nodes want to communicate securely, they must first get each other's certificates. When a node wants to look up the certificate C of an identity U, it will ask its zone master for its certificate. The master is faced with two possible scenarios:

1. If U is located in the master's zone, it can simply fetch the certificate from its internal table, and send it back to the node.
2. If U is not located in the master's zone, the procedure is as follows, accompanied by Figure 5.7: In this example, zone master 4 has the certificate for node 6. Node 5 asks its master (3)

for the certificate of node 6. Because node 6 is not located in the zone of master 3, it will send a recursive query to its child and parent zone masters, to ask if they have the certificate of 6. Zone master 4 responds with the certificate of node 6, and master 3 sends the certificate back to node 5.

Finally, when the node has obtained the requested certificate, it will verify all signatures of the zone masters until it reaches the root zone.

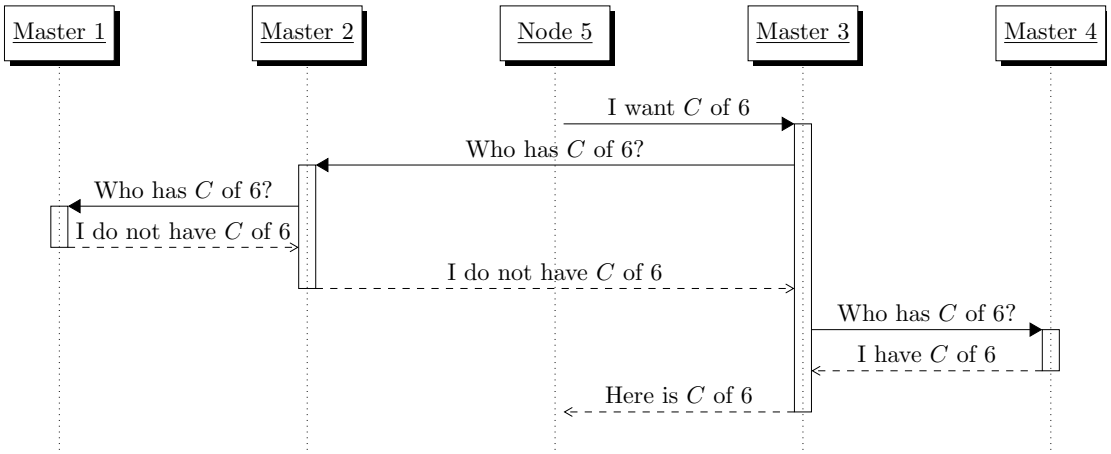


Figure 5.7: Certificate lookup procedure illustrated.

5.7 Certificate verification

Before two nodes can communicate securely, the nodes must have verified each other's certificate. To this end, both nodes must first query the public key of each other's zone master. With this, they can verify each other's certificate. Next, both nodes must traverse the entire certificate chain up to the root zone, by requesting the certificates and signatures of all zone masters that are parent of the to be verified node.

5.8 Zone master revocation

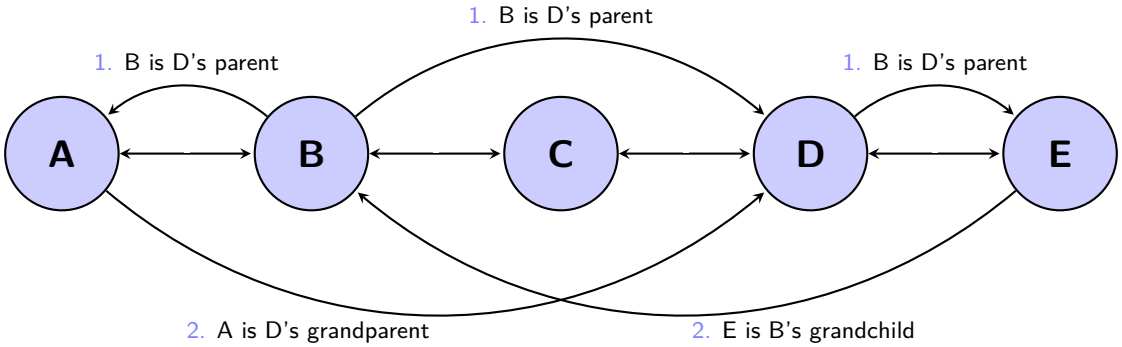
In case a zone master starts handing out malicious certificates, other zone masters must be able to get rid of the zone, i.e., revoke the zone. A zone master can revoke either its master parent or child. If a zone master decides to revoke its child zone master, its new child will become its grandchild zone master. If a zone master wants to revoke its parent zone master, the procedure is equivalent. Because the grandchildren and grandparents identify themselves in the zone extension procedure (Section 5.3), the master will know the identifiers U of its grandchild and grandparent. Finally, the zone master will send a signed message to all nodes in the revoked zone that they have to unenroll and re-enroll, so that they will be part of a valid zone again. An example is illustrated in Figure 5.8, where a zone master revokes its child zone.

Nevertheless, it is trivial that a malicious zone is not likely to update its nodes on the fact that it has been revoked. Therefore, the nodes that are in this revoked zone must be able to detect that its zone master has been revoked. To this end, we propose the following: periodically, all nodes in the zone will request proof from the parent and child zone master that they are still present. This will be done in the form of a digital signature:

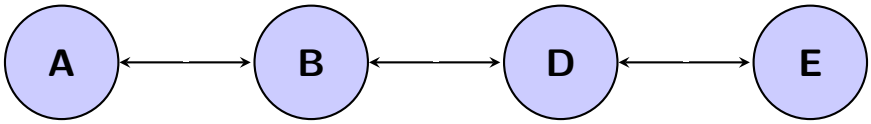
$$\text{sign}_{SK_P}(\alpha), \text{sign}_{SK_C}(\beta) \tag{5.3}$$

Where SK_P and SK_C are the secret keys of the parent and child zones, and α and β are random strings created by the verifying nodes in the zone. The only way for a malicious zone master to forge such signatures, is to know the secret key of both zone masters. Should the parent and child

zone masters return an incorrect or no signature, the node will assume that its zone has been revoked. After this, it will unenroll from the current zone and enroll into a new zone.



(a) Zone B revokes its child zone C by telling all nodes (except C) that B is the new parent of D. Because other nodes know that B is their parent or child, they will update fellow nodes by telling them they are new grandparents/children.



(b) While zone C can still operate, it is essentially removed from the zone chain.

Figure 5.8: Zone revocation illustrated: all circles represent zone masters. In the real architecture, nodes are identified by their UUID, but for readability purposes, we refer to them by a letter. In this case, zone B revokes its child zone C (Figure a), after which it is removed from the zone chain (Figure b).

5.9 Node unenrollment

When a node does not feel the necessity to communicate (securely) anymore, it can unenroll out of the zone. When a node wants to unenroll, it can simply send a signed request to the zone master, who will then revoke the node's certificate in its internal table, after successfully verifying the signature.

5.10 Certificates and keys

This section will describe a new certificate format that is based on X.509 certificates. In order to speed up certificate generation and verification, we will drastically reduce the number and size of fields found in the standard X.509 certificates. Moreover, we will encode the certificates using Concise Binary Object Representation (CBOR), to reduce the communication overhead even further. CBOR is a binary data serialization format that is somewhat related to JavaScript Object Notation (JSON). It permits the transfer of data objects containing name-value pairs (like JSON), but in a shorter way. This compromises human comprehension in favor of faster processing and transfer rates.

Finally, the key system to be used along with these certificates is ECC, as this is shown to be more efficient than other cryptosystems such as Rivest Shamir Adleman (RSA) [44–46] and has been widely implemented and tested on IoT devices [32, 39, 54, 55]. Next, we shall describe the fields found in this new certificate architecture.

5.10.1 Version number

X.509 specification This field is an integer describing the version of the certificate [2]. The versatility of X.509 version 3 offers support for other topologies like meshes and bridges [93]. Version

```
TBSCertificate  ::=  SEQUENCE  {        Certificate  ::=  SEQUENCE  {
        version                                 tbsCertificate
        serialNumber                            signatureAlgorithm
        signature                               signatureValue
        issuer                              }
        validity
        subject
        subjectPublicKeyInfo
        issuerUniqueID
        subjectUniqueID
        extensions
    }
```

Figure 5.9: Overview of a traditional X.509 certificate [2]. All fields are explained in the sections below.

3 also offers support for certificate extensions, which allow a CA to exclusively issue certificates for predetermined uses.

Optimized format By restricting this field to only version 3, it can be left out entirely.

5.10.2 Serial number

X.509 specification The serial number, which the CA assigns to each certificate, has to be a positive integer. It must be distinct for each certificate that a specific CA issues. In other words, the combination of CA and serial number uniquely identify a certificate [2].

Optimized format In order to reduce computational overhead as much as possible, the serial number will be a monotonically increasing integer, starting from 0.

5.10.3 Signature

X.509 specification The identifier for the algorithm that the CA employed to sign the certificate. The value of this field must be identical to the signature algorithm field (Section 5.10.11).

Optimized format By restricting the signature algorithm to a single algorithm, the field can be omitted entirely. The signature algorithm chosen is ECDSA NIST P-256, which is elaborated on in Section 5.10.7.

5.10.4 Issuer

X.509 specification The CA that issued the certificate is named in this field. The issuer field needs to have a Distinguished Name (DN) in it that is not empty [2].

Optimized format While the Issuer field in regular X.509 certificates contain various attributes such as country, organization, and Common Name (CN), in order to preserve space, this field shall only contain the CN.

5.10.5 Validity

X.509 specification The duration for which the CA guarantees that it will keep track of the certificate's state. The date the validity period starts (`notBefore`) and the date the validity period ends (`notAfter`) are expressed in the field as a sequence of two dates [2].

Optimized format The X.509 format allows for two date types:

1. UTCTime, which is of the format `YYMMDDHHMMSSZ`
2. GeneralizedTime, which is of the format `YYYYMMDDHHMMSSZ`

While the first date format is smaller, it does entail a drawback: the year is indicated using two characters. These two characters are interpreted as follows [2]:

- "*Where YY is greater than or equal to 50, the year SHALL be interpreted as 19YY; and*"
- "*Where YY is less than 50, the year SHALL be interpreted as 20YY.*"

This means that after the year 2049, date types of this format are not applicable anymore. Nevertheless, the optimized format will still make use of this date type. In this optimized format, the `YY` characters are to be interpreted as `20YY`.

5.10.6 Subject

X.509 specification The entity connected to the public key is identified by this field [2]. If the subject is a CA, the subject field must contain a "non-empty distinguished name matching the contents of the issuer field" [2].

Optimized format In order to provide unique identities for all nodes in the PKI, the subject will be a `uuid4`, with a length of 36 bytes.

5.10.7 Subject Public Key Info

X.509 specification This field contains the public key and specifies the algorithm that the key is used with.

Optimized format By allowing only one public key algorithm, the specified algorithm in this field can be omitted, thus decreasing computational overhead. The public key algorithm chosen is ECC, on the NIST P-256 curve. The reason for choosing Elliptic Curve Cryptography over a different asymmetric cryptosystem, e.g., RSA, is because it provides much higher levels of security for much smaller keysizes, thus reducing computational and communication overhead [89]. The P-256 curve is chosen because it provides 128 bit level security, which is considered 'acceptable' by the National Institute of Standards and Technology [89].

5.10.8 Unique identifiers

X.509 specification To address the potential for subject and/or issuer name reuse. However, it is advised that CAs adopt distinctive names and must thereafter refrain from assigning these fields [2].

Optimized format Because it is advised that subjects and/or issuer names are not reused, this field is not necessary and therefore omitted entirely.

5.10.9 Extensions

X.509 specification This field lists one or more certificate extensions. It must allow at least the following extensions [2]:

- Key Usage: this extension specifies the function of the key present in the certificate (such as certificate signing or encryption).
- Certificate policies: a collection of one or more policy information terms that describe the policy by which the certificate was issued and the permitted uses of the certificate.

- Subject alternative name: with this extension, identities can be linked to the certificate's subject. These identities can be used in place of, or in addition to, the identity listed in the certificate's subject field.
- Basic constraints: specifies "*the maximum depth of valid certification paths that include this certificate*" [2], as well as whether the subject of the certificate is a CA.
- Name constraints: All subject names in following certifications in a certification route must be contained inside the name space indicated by this extension, which can only be used in a CA certificate. The constraint for URIs only applies to the host portion of the name. The constraint may specify a host or domain but must be supplied as a fully qualified domain name. `host.example.com` and `.example.com` are two examples.
- Policy constraints: Certificates deployed to CAs may use the policy constraints extension, which restricts path validation.
- Extended key usage: This extension lists one or more additional uses for the certified public key that can be substituted for or added to those mentioned in the key usage extension. This extension will often only be shown in certificates for end entities.
- Inhibit anyPolicy: Certificates issued to CAs may utilize the inhibit anyPolicy extension. The inhibit anyPolicy extension states that, unless it occurs in an intermediate self-issued CA certificate, the special anyPolicy OID is not considered an explicit match for other certificate policies.

Optimized format In the optimized format, entities are not required to receive and parse these extensions, i.e., all extensions are listed as optional. This is done in order to reduce communication and computational overhead.

5.10.10 TBS Certificate

X.509 specification This TBS (*"To Be Signed"*) field includes the subject and issuer names, a public key related to the subject, a validity time, and other relevant data [2].

Optimized format Because this information is already listed in fields mentioned before (Section 5.10.4, 5.10.6), this field will be omitted altogether.

5.10.11 Signature Algorithm

X.509 specification The name of the cryptographic algorithm that the CA used to sign this certificate. The value of this field must be the same as in the signature field (Section 5.10.3).

Optimized format By restricting the signature algorithm to a single algorithm, the field can be omitted entirely. The signature algorithm chosen is ECDSA NIST P-256.

5.10.12 Signature Value

X.509 specification This field contains the tbsCertificate's ASN.1 DER-encoded digital signature.

Optimized format Because this field is essential for the proper functioning of the PKI, it is included in the optimized format, without any modifications.

5.10.13 Summary

An overview of the optimized X.509 certificate for IoT devices is shown in Figure 5.10.

```
TBSCertificate  ::=  SEQUENCE  {
        serialNumber          int
        issuer                string
        validity              UTCTime
        subject               int
        subjectPublicKeyInfo  string
        extensions            list
    }

Certificate  ::=  SEQUENCE  {
        signatureValue        string
    }
```

Figure 5.10: Overview of the optimized X.509 certificate for IoT.

5.11 Security considerations

This section presents a number of security considerations for the proposed solution, along with security solutions.

5.11.1 Denial-of-Service

Malicious nodes and zones are able to overwhelm the intended victim zone masters/nodes by sending them protocol messages (e.g., certificate lookup, certificate update, and enrollment requests) at a very high frequency. This can result in the victim zone master/node being overwhelmed and shutting down, essentially performing a DoS attack.

To mitigate this issue, all nodes and zone masters will implement rate limiting, which is a mechanism employed to regulate the frequency of requests exchanged by a network interface. If the traffic rate at a node or zone master exceeds a certain threshold, the zone master or node will simply drop remaining traffic. Although this can have a negative effect on benign nodes, it will prevent DoS attacks.

5.11.2 Blocking of messages

A malicious zone master is able to block any requests sent between zones. Examples of such requests are zone extension, node enrollment, certificate lookup and zone master revocation. If a zone master suspects a node to block requests, it will send three distinct dummy requests to the suspicious zone master. When none of the requests receive any response, it will revoke the zone master. If the suspicious zone master is no direct parent or child of the zone master, it will ask neighboring zones to revoke. Moreover, if a node suspects a zone master to be blocking requests, it will tell its zone master who will then perform this (three requests) test on their behalf.

5.12 Conclusion

In this chapter, we have designed a decentralized PKI architecture that employs zones, in which the zone masters act as CAs for all nodes in the master's zone. Furthermore, we have designed a new type of X.509 certificate, that will be more lightweight than its traditional variant. It has been designed by removing or reducing the size of the fields present in a classical X.509 certificate. In the next two chapters, we will validate our design by conducting a performance (Chapter 6) and security (Chapter 7) analysis.

Chapter 6

Performance analysis

This chapter will design and execute the framework for evaluating the performance of the proposed solution. The proposed solution has been implemented in Python 3.10, and uses the `pycryptodome`[1] and `cbor2`[2] libraries for cryptographic operations and CBOR encoding respectively. The implementation is hosted on a Raspberry Pi 4 Model B with 1 GB of LPDDR4-3200 SDRAM memory, and a Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit CPU @ 1.8GHz. The Raspberry Pi employs Alpine Linux (v3.17) as its host OS, because it is a lightweight distribution.

Firstly, we will describe the procedure to measure the performance of the implementation. Next, we will present the results of the previously explained methodology.

6.1 Experimental setup

In order to measure the performance of the proposed solution and implementation, we will be performing a time, memory, and power-based analysis.

- In the time analysis, we will first log the current timestamp before executing the operation. After the operation has finished, we will log the timestamp again and compute the difference to obtain the runtime duration.
- In the memory analysis, we will be using `filprofiler`[3], which is a Python tool that analyzes memory usage of programs.
- In the energy analysis, we will be conducting a baseline voltage and ampere measurement to obtain the baseline usage. When executing the operation, the peak voltage and amperage will be observed, and the differences between the baseline will be noted. By combining the voltage and amperage with the duration of the operation, the energy (J) can be calculated:

  $$\text{Energy (J)} = \text{Voltage (V)} \times \text{Amperage (A)} \times \text{Time (t)}.$$

  For the measurement, we have used dedicated Raspberry Pi energy measurement hardware. The experimental setup can be seen in Figure 6.1.

In this analysis, we will benchmark the following operations, as we deem these to be most relevant:

- Certificate size
- Certificate generation
- Node enrollment (Section 5.2)
- Zone extension (Section 5.3)

[1] https://pypi.org/project/pycryptodome/
[2] https://pypi.org/project/cbor2/
[3] https://pypi.org/project/filprofiler/

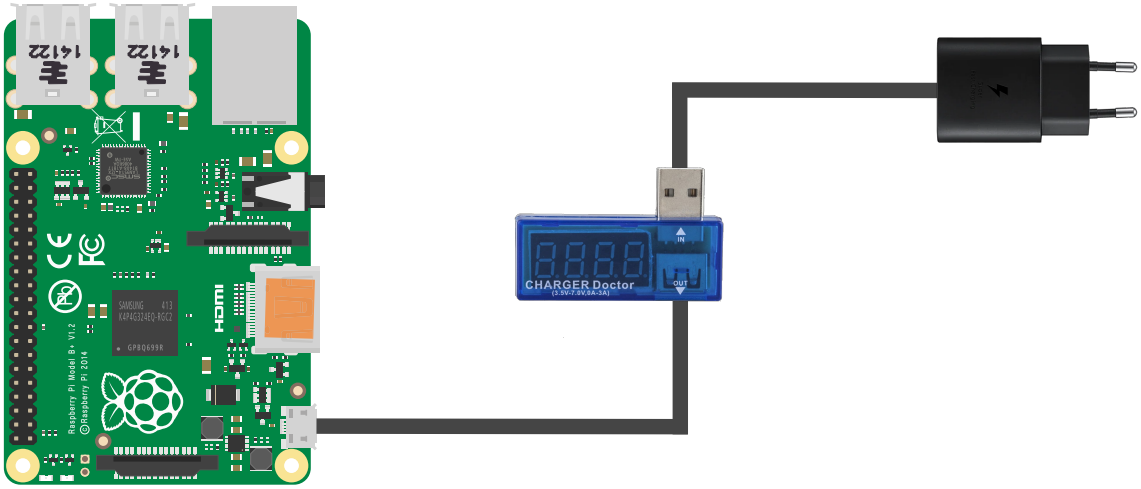


Figure 6.1: Experimental setup for the energy analysis. The dedicated energy measurement hardware sits on the connection between the Raspberry Pi and the power outlet.

- Certificate lookup (Section 5.6)
- Certificate verification (Section 5.7)

We will benchmark every one of these operations three times. The first run, the benchmark will be performed with traditional X.509 certificates. That is, an X.509 certificate with all fields included. Next, we will run the benchmark with our optimized X.509 certificates with omitted or shortened fields. Finally, we will benchmark our solution with the optimized certificates from the previous run, but we encode them efficiently using CBOR. We want to note that in the first two benchmarks, all certificates are encoded in JSON format for ease of parsing during communication. Furthermore, we considered benchmarking the number of messages sent across the network. However, we deemed this to be trivial as this can be deduced from the proposed design in Chapter 5.

We will now describe the four benchmark procedures in more detail.

6.1.1 Certificate size

The first benchmark will create three certificates, one for each category (traditional, optimized, CBOR encoded). Then, the size of the certificates will be calculated in bytes.

6.1.2 Certificate generation

In the next benchmark, the time, memory, and energy required to generate a certificate is measured, which includes the signature generation. To this end, we will generate 100 certificates and calculate the average time and memory consumption. As certificates are only CBOR encoded when transmitted across a network, it is not relevant to benchmark this category. Nevertheless, we will test the performance of CBOR encoding to find out whether generation of such certificates requires fewer resources than regular JSON encoding.

6.1.3 Node enrollment

This benchmark will create a master and client, and make the client connect to the master to enroll. The latter procedure is benchmarked in terms of time, memory, and energy consumption for both the client and master. To this end, the benchmark will be executed 100 times and the average values will be computed.

6.1.4 Zone extension

This benchmark creates a master and client as well. Next, the client enrolls into the master's zone. Finally, the master instructs the client to extend to a new zone, which is measured in terms of time, memory, and energy consumption.

6.1.5 Certificate lookup

This benchmark is parametrized by the number of zone masters. That is, this benchmark will first create x zone masters, and will enroll the first client to the first zone in the chain, and the second client to the final zone in the chain. Finally, the first client will look up the certificate of the second client, whose request is then passed along the entire zone chain. The purpose of this benchmark is mainly to find out whether CBOR encoded certificates are transferred faster than regular and optimized JSON encoded certificates.

6.1.6 Certificate verification

The final benchmark will be measuring the memory usage, runtime, and energy usage of certificate verification with five zone masters. Here, a node has to verify the certificate of a node in the final zone, and verify the five certificates of the zone masters that make up the certificate chain.

6.2 Experimental results

This section will present the results of the experimental methodology as described in the previous section. Each subsection will present the results for its respective operation.

6.2.1 Certificate size

This benchmark generated three certificates, one for each category, and measured their size in bytes. We want to point out that numerous certificates have been generated for every category. Nevertheless, the certificate size remained constant.

As can be seen in Figure 6.2, the optimization and encoding of the certificates drastically reduces their size. Optimized certificates are 46% smaller than traditional X.509 certificates, and CBOR encoded certificates are 52% smaller than traditional certificates.

6.2.2 Certificate generation

In this benchmark, we generated 100 certificates and calculated the average execution time, memory usage, and energy consumption for traditional, optimized and CBOR encoded certificates. Although it can be seen in Figure 6.3 that optimized certificates take less time and memory to be generated, it is not by a significant margin. Furthermore, it can be noted that CBOR encoding requires less energy to generate than regular JSON encoding. The greatest improvements can be

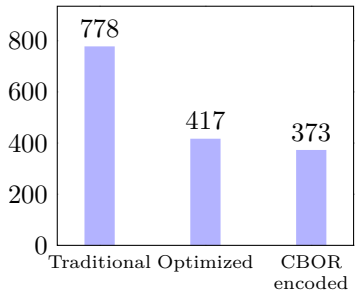
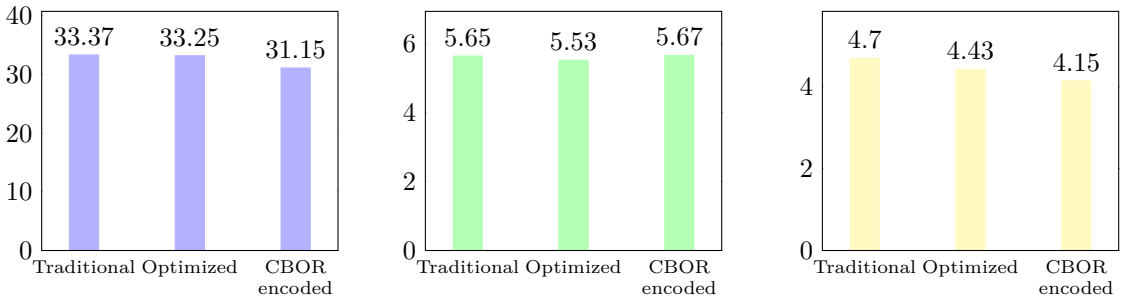


Figure 6.2: Certificate size in bytes.

seen in terms of memory, where the CBOR encoding requires > 2000 fewer bytes of memory than when generating a traditional certificate.

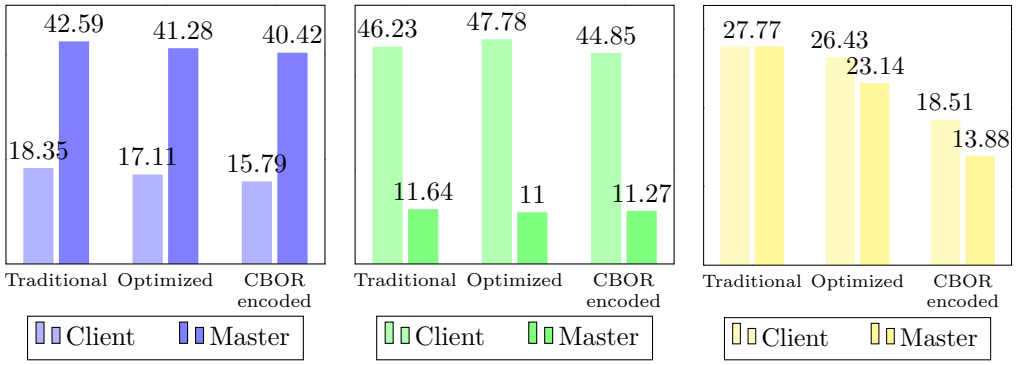


(a) Memory (kilobytes) (b) Time (ms) (c) Energy (mJ)

Figure 6.3: Benchmark of certificate generation.

6.2.3 Node enrollment

In this benchmark, the node enrollment procedure was executed 100 times, and the average duration, energy, and memory consumption of enrollment was calculated. This benchmark was conducted from both the client and master perspective, in order to give a more comprehensive overview.



(a) Memory (kilobytes) (b) Time (ms) (c) Energy (mJ)

Figure 6.4: Benchmark of node enrollment.

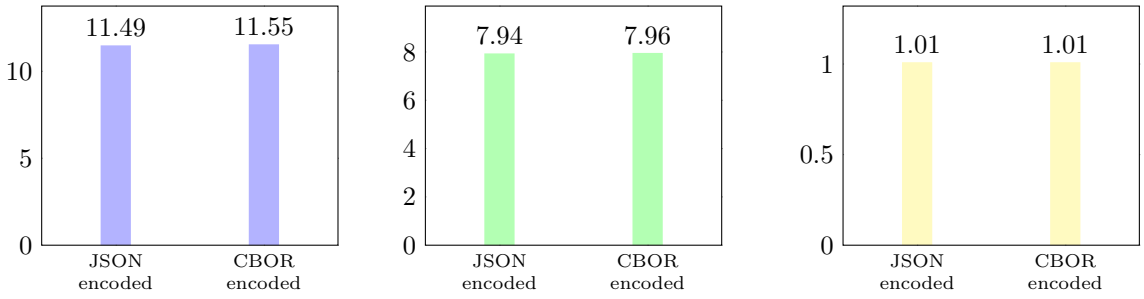
In Figure 6.4, it can be seen that the optimized certificate slightly outperforms the traditional certificate, in terms of memory and energy usage. Although the CBOR encoded certificate seems to perform better than both the traditional and optimized certificate on the client side in terms of time, this is not the case for the master side. Furthermore, it is interesting to note that the master side needs significantly more memory, which is mainly due to the cryptographic operations the master has to perform. Nevertheless, the client side requires more time to complete than the master side, which could be due to the fact that the client has to find a zone master with vacant spots, which could require the client to send numerous requests across the network. Finally, we note that on both the client side CBOR encoding and optimized certificates do have a positive effect on energy usage, where energy usage is reduced by 33% and 50% on the client and master side respectively.

6.2.4 Zone extension

In this benchmark, the zone extension procedure was executed 100 times, and the average time, energy, and memory consumption during extension was calculated.

It is important to note that the zone extension procedure does not make use of certificates that are sent across the network. Consequently, the efficiency of CBOR encoding is not leveraged as much as with for example, the node enrollment procedure. Therefore, in this benchmark, we will only compare regular JSON encoding with CBOR encoding.

Figure 6.5 demonstrates that this benchmark does not show a significant improvement in using CBOR encoding over JSON encoding, due to the aforementioned reason.



(a) Memory (kilobytes) (b) Time (ms) (c) Energy (mJ)

Figure 6.5: Benchmark of zone extension.

6.2.5 Certificate lookup

In this benchmark, there is one node in the first zone master in the chain, and another node in the final zone master. The first node will look up the certificate of the second node, so that the zone masters will have to propagate this request along all zone masters in the chain. For this parametrized benchmark, we executed the benchmark for x zone masters, where x is an integer in $[2, 150]$. We chose the upper bound of zone masters to be 150, as the Raspberry Pi was not able to handle more.

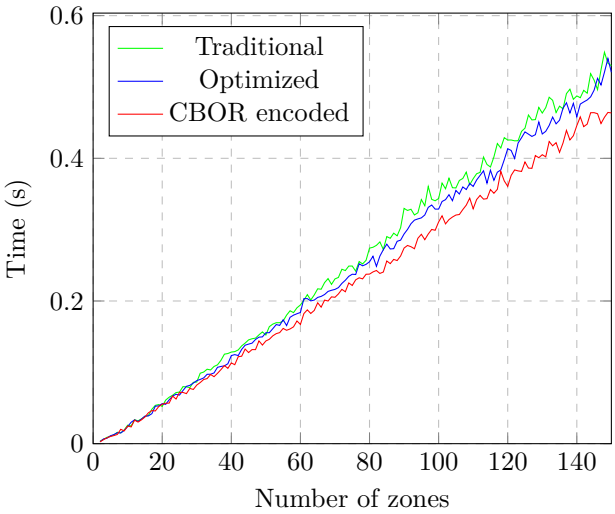
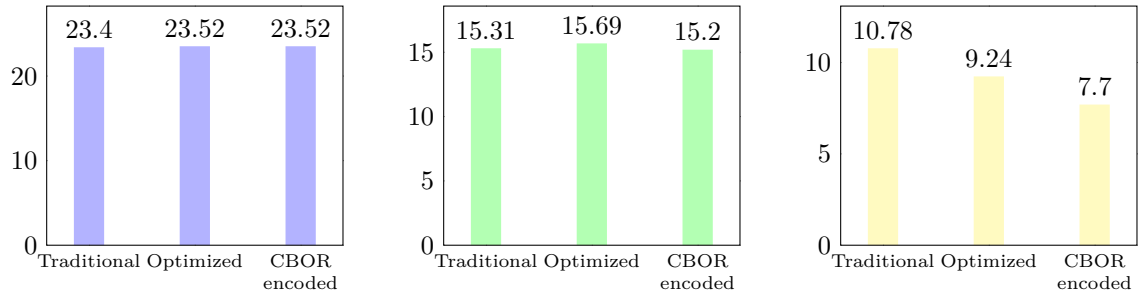


Figure 6.6: Benchmark of certificate lookup.

As depicted in Figure 6.6, a noticeable enhancement in execution time can be observed when comparing traditional certificates with optimized and encoded certificates, particularly for zone counts exceeding approximately 20. This is attributed to the fact that an increased number of messages are required to facilitate certificate lookups as the number of zones grows, thus leading to a greater utilization of the CBOR encoding technique. For example, for $x = 150$ zones, the CBOR encoded certificate performs 12.5% faster than traditional certificates, and 10.9% faster than optimized certificates.

6.2.6 Certificate verification

In this benchmark, we will be measuring the memory usage, runtime, and energy usage of certificate verification with five zone masters. Here, a node has to verify the certificate of a node in the final zone, and verify the five certificates of the zone masters that make up the certificate chain.



(a) Memory (kilobytes) (b) Time (ms) (c) Energy (mJ)

Figure 6.7: Benchmark of certificate verification.

As can be seen in Figure 6.7, while in terms of memory the CBOR encoded certificate does not yield any improvements over the traditional certificate, there is minor improvements in runtime. Furthermore, we can observe that there are significant energy usage improvements for optimized and CBOR encoded certificates.

6.3 Discussion

This section will discuss the obtained performance results by comparing them to results found in the literature (Table 4.2 and 4.3).

From the tables that have been created during the literature review, two types of metrics have been found that allow for comparison between works: certificate size and certificate verification, in terms of runtime and energy usage.

6.3.1 Certificate size

For certificate size we have the work by Forsby et al. [73], where the authors suggest a lightweight X.509 certificate format made especially for Internet of Things devices. This novel format seeks to provide safe authentication and data integrity while reducing certificate size and computational overhead. The lightweight certificate shows compliance with the X.509 standard, which allows use in all current PKI implementations. Not only is our optimized certificate larger than that of [73] (417 vs 324 bytes), this is also the case for the compressed certificate (373 vs 146 bytes). This can be due to the fact that [73] has applied more efficient techniques for optimizing certificate size. For example, for the "Subject Public Key Info" field, [73] have taken extra steps to compress the ECC keys, even before CBOR compression. Furthermore, another reason for smaller certificate sizes could be attributed to the fact that the CBOR libraries used in both works are implemented differently. Namely, the library in our work (`cbor2`) is noted to be compliant with the original CBOR specification (RFC 7049, [94]). Nevertheless, the CBOR library used in [73] has been implemented before the existence of RFC 7049 [94].

6.3.2 Certificate verification

For certificate verification time and energy consumption, we have the works by Won et al. [6], Pinol et al. [54], Singla and Bertino [59], and Marino et al. [74]. Firstly, it can be seen that our work outperforms [6] and [59] by 8.4-16.4 times. Nevertheless, due to the fact that the architectures in these works are blockchain-based, and thus completely different from our work, we will not go into much detail as we do not think that would be a fair comparison.

Next, we have Pinol et al. [54], who created an open source ECC that is optimized for the IoT and includes an Elliptic-Curve Diffie-Hellman (ECDH) and an Elliptic-Curve Digital Signature Algorithm (ECDSA). They skip division operations since they are computationally expensive in order to optimize the algorithms. They instead use modular arithmetic. The comparison shows that our implementation is 263-724 times faster than that of [54], depending on the keysize. This is mainly due to the fact that the implementation of [54] has been tested on a STM32W108CC wireless chip, which has much slower hardware than a Raspberry Pi 4B [95]. Furthermore, another reason could be that the authors have implemented their own ECC, instead of relying on libraries. This could introduce performance bottlenecks, as custom ECC implementations could be non-compliant to specifications.

The final work up for comparison is that by Marino et al. [74], who developed PKIoT, which allows IoT nodes to assign computationally intensive security-related tasks to a remote server. Nodes are free to decide which tasks to assign based on their current state and level of trust in the server. As a result, the PKIoT architecture offers a scalable, adaptable, and flexible solution. They also developed a novel type of compact certificate, which requires PKIoT compatibility on both ends of the communication but allows for even further transmission overhead reductions when used in place of normal X.509 certificates. The comparison results show that our implementation is almost 31 times faster, and consumes 2 times less energy than the work in [74]. This can be attributed to a number of factors. Firstly, the usage of different crypto libraries: while our implementation uses `pycryptodome`, [74] uses micro-ecc [96]. Due to implementation and programming language differences, these libraries can achieve different results in terms of performance. Nevertheless, the improvement in performance cannot be attributed to our hardware; the PKIoT server, used to perform cryptographic operations when nodes choose so, is a PC with an Intel Core i5-6500 @ 3.20GHz and 15.6 GB of RAM. However, it is possible that the worse performance of can be due to certificate size. In [74], on the client side certificates are stored as a simple link to the full certificate, which is located on a PKIoT server. This certificate is unfortunately a full-sized certificate, which can explain the longer verification times and higher energy consumption.

We conclude this section by giving a comprehensive overview of the discussed articles in Table 6.1.

Table 6.1: Comparison of proposed solution with those present in the literature.

Type of metric	Name	Authors' results	Our results	Improvement
Certificate size	Forsby et al. [73]	• Optimized: 324 bytes • Compressed: 146 bytes	• Optimized: 417 bytes • Compressed: 373 bytes	✗
Certificate verification	IoT-PKI [6]	• 128 ms	• 15.2 ms • 7.7 mJ	✓
	Pinol et al. [54]	• 4-11 s, 153.84 mJ		✓
	Singla et al. [59]	• 128-250 ms		✓
	PKIoT [74]	• 469 ms, 16.57 mJ		✓

6.4 Conclusions

In benchmarks 6.2.2-6.2.4, it is shown that there is no significant difference in execution time between the various types of certificates. Nevertheless, the benchmarks do mostly show that the optimized CBOR encoded certificates perform better in terms of memory usage, certificate size, and energy consumption. This can be attributed to a number of factors:

1. More effective number encoding is supported by CBOR, which employs a variable-length integer encoding system that consumes fewer bytes than JSON to represent small values. In contrast to JSON's text-based representation, CBOR permits the encoding of floating-point integers using a more compact binary format [94].

2. CBOR uses a length-prefix encoding strategy for Unicode strings, which enables it to repre-

sent strings using fewer bytes than JSON's Unicode escape sequences. This enables CBOR to support more compact encoding of Unicode strings. For instance, the Unicode characters in CBOR are never escaped, in contrast to formats like JSON. Therefore, a newline character `(U+000A)` is never represented in a string as bytes `0x5c6e` (the characters "\" and "n") but rather as bytes `0x0a`.

3. Tagging is one of the characteristics of CBOR, which may enable it to represent some categories of data more effectively than JSON. For instance, CBOR comes with built-in support for expressing dates and times, which can be done with less data than equivalent JSON expressions.

What is surprising is that the CBOR encoding seems to have no significant effect on the performance in terms of time. In fact, although the differences are barely noticeable, most benchmarks perform slower than their optimized JSON counterparts. We argue that CBOR requires more time to encode/decode than JSON, as CBOR needs to compress/decompress the data as well, and because of the aforementioned reasons.

Furthermore, we argue that optimized and CBOR encoded certificates have a positive effect on energy usage, when compared with the traditional certificates. This can be attributed to the fact that CBOR encoding requires less resources than regular JSON encoding. It is important to note that the energy measurement results could be skewed due to hardware inaccuracies.

Also, certificate lookup is visibly improved using CBOR as long as the number of zones are high enough. We suppose CBOR encoding only has an effect on operations that require CBOR data to be sent over the network, such as certificate lookup, therefore leveraging the power of efficient encoding the most.

We conclude that CBOR encoding has no significant or an adverse effect on all operations in terms of time, because the compression of CBOR encoding requires more time than regular JSON. Nevertheless, when CBOR encoded certificates are sent across numerous hosts (Section 6.2.5). In such cases, the power of CBOR encoding is properly leveraged. Furthermore, the benchmarks mostly show that the CBOR encoding shows a significant decrease in memory consumption and certificate size. By comparing our obtained results with those from the literature (Table 4.2 and 4.3), we have concluded that our certificate verification is faster and consumes less energy, but it must be taken into account that different architectures and hardware platforms could have had an effect on these outcomes. Nonetheless, we also concluded that there is still room for improvement in certificate size, as other works have shown that a smaller certificate size can be obtained.

The next chapter will validate our proposed solution in terms of security, by conducting a theoretical and formal security analysis.

Chapter 7

Security analysis

This chapter presents details of the security analysis that will be performed on the implementation. The security analysis is three-fold. Firstly, we will be conducting a theoretical security analysis using theoretical proofs. Secondly, we will be outlining a number of security guarantees, such as resilience against replay attacks. Finally, we will be performing a formal analysis using a formal verification tool called Automated Validation of Internet Security Protocols and Applications (AVISPA). However, first we will describe the adversarial model and assumptions:

7.1 Adversarial model

An adversary will possess the following capabilities:

- The adversary will have unrestricted control over the communication channel, enabling them to observe, manipulate, or replay the messages conveyed over the channel.
- The adversary has the capability to employ the following attack model and execute it within polynomial time:
  - `lookup(`N`,` M`)` This operation allows the adversary to legitimately request the public key of node N through zone master M, as if it were a regular user.
  - `enroll(`M`)` This operation allows the adversary to legitimately enroll into a zone with master M, as if it were a regular user.
  - `update(`C`,` M`)` This operation allows the adversary to update its certificate to C using zone master M.
  - `revoke(`Z`)` This operation allows a malicious zone master to revoke its child or parent zone Z.
  - `corruptNode()` This attack model represents a scenario in which a node becomes compromised or corrupted.
  - `corruptZoneMaster()` This attack model represents a scenario in which a zone master becomes compromised or corrupted.

7.2 Assumptions

- The adversary:
  - can start any number of parallel protocol sessions,
  - knows the functioning of the entire protocol,
  - can build and send messages,

  - can read, retain and block any sent message,
  - can decrypt any message for which it has the key.
- It is computationally infeasible for the adversary to break the Elliptic-Curve Digital Signature Algorithm (ECDSA), Elliptic-Curve Diffie-Hellman (ECDH), and Elliptic-Curve Discrete Logarithm Problem (ECDLP).

7.3 Theoretical analysis

The architecture employs Elliptic Curve Cryptography (ECC) to generate its keys and signatures, as outlined in Section 5.10. Additionally, the SHA-256 hashing algorithm is utilized as the secure hashing algorithm. This section will provide a theoretical security analysis of various operations present in the design of the PKI.

7.3.1 Certificate updating

In the case that a node wants to update its public-private keypair, it can issue a certificate update request to the zone master. It must be ensured that only the node who is in possession of the secret key is able to update its keypair.

Theorem 1. *A node who does not have the secret key for an identity is unable to send a legitimate update transaction to the zone master.*

Proof. If a node wants to update its keypair, it can simply send a new certificate C_{new} signed with the old key SK_{old}. In order to create this signature S, the node needs to be in possession of SK_{old}. The submission of an update transaction by an adversary, for an identity that is not under their possession, can only be achieved through the successful reconstruction of SK_{old} and the creation of a valid signature for the new certificate. If the adversary is able to do this, it would imply that the adversary is able to break the ECDSA, which is computationally infeasible for a computationally limited adversary. □

7.3.2 Node unenrollment

When a node wants to unenroll out of the zone, it can request the zone master to do so, who will then revoke the certificate of the node in its internal table. The node does this by sending a signed message requesting to unenroll, similar to the certificate update procedure.

Theorem 2. *It is impossible for an adversary to submit a legitimate revoke transaction to the zone master for an identity that they do not hold the secret key for.*

Proof. If a node wants to unenroll from the zone, it will send a request signed with SK to the zone master. If an adversary wants unenroll a node for which it does not own the private key SK, it will need to get a hold of SK. Therefore, the adversary must be able to break the ECDSA, which is computationally infeasible for a computationally bound adversary. □

7.3.3 Certificate lookup

When two nodes want to communicate securely, they will have to look up their respective certificates. This can be done by querying the zone master for the certificates. It is therefore essential that the certificates are not modified in transit. We propose the following theorem:

Theorem 3. *It is impossible for an adversary to modify a certificate in transit during a lookup procedure.*

Every certificate handed out by a zone master is signed by the zone master itself. Furthermore, every zone master public key certificate is signed by its parent. Therefore, if a node wants to check the validity of a certificate, it can check the signature on the certificate. Next, it will traverse the certificate chain until it reaches the root certificate, for which it must assume that it is to be

trusted. Therefore, if an adversary aims to perform a MITM with a malicious certificate, it must be able to forge the signature of the zone master in which a node resides. Thus, the adversary must be able to obtain the private key of the zone master, or must be able to break the ECDSA, which is computationally infeasible given a computationally limited adversary.

7.4 Security guarantees

This section will outline a number of security guarantees for the proposed PKI solution.

7.4.1 Secure against replay attacks

Our PKI is resilient against replay attacks, because the authenticity of messages sent over public key cryptography can be verified by checking the certificate, along with the signatures of all zone masters. Furthermore, messages sent over public key cryptography include encrypted nonces, which ensure that an adversary is not able to retransmit such messages (Section 5.2.1). The only potential possibility for replay attacks is that for node enrollment, because an adversary can simply retransmit an enrollment request. Nevertheless, this does not have a major impact as the zone master can simply check that a certain node has already been enrolled. If an adversary stores an enrollment request for a node that has unenrolled out of the zone, replay attacks are prevented by requiring a fresh Diffie-Hellman key exchange (that is, with new keys) for every enrollment request, regardless of whether this node has enrolled before.

7.4.2 Mutual authentication

Our system offers mutual authentication, in which all entities (such as IoT nodes and zone masters) mutually verify one other's identities. More specifically, because it is computationally infeasible to break the ECDH, adversaries are unable to spoof authentication on enrollment requests $\lambda = \{([PK], H)\}_\sigma$, as they are encrypted with symmetric key σ. Furthermore, adversaries cannot spoof messages exchanged over public key cryptography as they are protected by an encrypted nonce (Section 5.2.1).

7.4.3 Secure against impersonation attacks

An attacker is unable to create legitimate messages to send over the network unless it has access to the zone master's or an IoT node's private key. Therefore, it is impossible for an adversary to pretend to be a zone master or an IoT node.

7.4.4 Secure against Man-in-the-Middle attacks

Man-in-the-middle attack (MITM) attacks occur when an active attacker successfully poses as both the user to the server and the server to the user by intercepting the communication connection between a legitimate user and the server. After then, both the user and the targeted server will think they are speaking to one other. From the previously mentioned guarantees, we infer that our protocol can offer mutual authentication, which enables us to prevent MITM attacks.

7.4.5 Secure against brute-force attacks

The total number of unique keys used in the encryption system is what determines the size of the key space. In order to prevent brute-force attacks, an encryption algorithm's key-space needs to be sufficiently vast. Our proposed PKI employs the NIST P-256 curve, which has a keysize of 256 bits. Therefore, the entire keyspace of this encryption scheme is 1.16×10^{77}. Following, we conclude that a brute force attack is computationally infeasible for such a keyspace.

7.4.6 Secure against passive attacks

To prevent adversaries from listening in, the transmitted data is encrypted in the proposed solution using the ECC and AES encryption scheme. As a result, without the decryption key, the passive

adversaries are unable to decrypt the intercepted message. Thus, we conclude that our proposed solution is resilient against passive attacks.

7.5 Formal analysis

For the formal analysis, we have employed the Automated Validation of Internet Security Protocols and Applications (AVISPA) tool, which is frequently used to examine security protocols and cryptographic properties. Researchers and developers can use it as a platform for automatic assessment and testing of security aspects of protocols [97]. AVISPA examines the provided model under the presumptions of perfect cryptography [98] and protocol message exchange via a network controlled by a Dolev-Yao adversary [99]. Specifically, the intruder can intercept messages and analyze them if it has the decryption keys, and it can act upon this knowledge.

In this analysis, two models have been written in High-Level Protocol Specification Language (HLPSL), which AVISPA internally translates to an Intermediary Format (IF), which it feeds to a backend, Constraint-Logic-based Attack Searcher (CL-Atse). We opted to write two HLPSL models, as we deemed these models to cover all operations inside the proposed design best:

1. Messages over symmetric-key cryptography: in this HLPSL specification, the node enrollment procedure (Section 5.2) is modeled. That is, the node and zone master agree on a symmetric key using a Diffie-Hellman key exchange. Next, the node sends a symmetrically encrypted request to the zone master to be enrolled into the zone. Finally, the zone master responds to the enrollment request with a symmetrically encrypted message as well.
2. Messages over public-key cryptography (Section 5.2.1): in this specification, all other operations in the design are modeled. Firstly, The requesting party generates a nonce and sends this nonce, along with the request payload in an encrypted form (with the public key of the receiving party) to the receiver. The receiving party responds with the same nonce, concatenated with its desired response payload. The entire response is encrypted with the public key of the sending party. The reason for including a nonce in the request and response is to prevent MITM attacks, more details in Section 5.2.1.

The full HLPSL models can be found in Appendix A.

7.5.1 Results

Both proposed models have been tested using the CL-Atse backend. The simulation results showed that both models protect the confidentiality of the message payloads, and that the second HLPSL model ensures correct authentication of both communicating parties. We want to note that checking for this property in the first model is not relevant, as it is assumed that all entities with possession of symmetric keys are authenticated parties, while in the second model any unauthenticated party can send messages encrypted with the public key of the recipient.

According to the best of our knowledge, both HLPSL specifications cover the entire protocol design in a complete way. Therefore, we consider according to the formal analysis, the entire design to be secure.

We would like to note that we have decided not to conduct a comparison between our work and that of the literature, as we deemed the metrics for comparison in Table 4.2 and 4.3 too difficult to give an accurate and comprehensive comparison.

7.6 Conclusions

In this chapter, we have performed a theoretical and formal analysis of the proposed PKI design. Furthermore, we have provided a number of security guarantees, such as resilience against impersonation attacks. In the theoretical analysis, we have written a number of theorems related to various operations inside this new architecture, along with their proofs. Next, in the formal analysis, we have written to HLPSL specifications that model two types of communication in the

architecture: (i) those using symmetric-key cryptography, and (ii) those using public-key cryptography. Assuming that the two specifications fully cover the entire architecture, we conclude that along with the theoretical analysis and the security guarantees, the proposed solution is secure.

The next chapter is the final chapter of this thesis, which will conclude it entirely.

Chapter 8

Conclusions

In this thesis we have researched the academic landscape regarding Public Key Infrastructure (PKI) for the Internet of Things (IoT), in order to identify issues and opportunities in designing such a PKI. Furthermore, we have designed, implemented, and evaluated a lightweight PKI tailored to IoT devices. This chapter aims to conclude this thesis, by noting the research objectives, answering the research questions, stating the contributions, limitations, and potential future works.

8.1 Research objectives

The term Internet of Things (IoT) describes actual physical items with sensors, processing power, and software that may connect to other systems and devices via the Internet to exchange data [3]. The IoT is converting people's immediate surroundings into a cyberphysical system that they can interact with by using wearables and personal mobile devices like smartphones. However, there are many security issues in the IoT space that need to be resolved [9]. Researchers have documented numerous attempts to compromise the availability [16, 17], confidentiality [10–12], and integrity [13–15] of IoT devices. Public Key Infrastructure (PKI) is one of many solutions to address the aforementioned issues and is already used daily in traditional computer systems. PKI facilitates the generation, dissemination, modification, and revocation of digital certificates. A digital certificate serves as a connection between public keys and the identities, of numerous entities [25]. As part of the procedure to create the connection, a Certificate Authority (CA) produces and registers these digital certificates.

Unfortunately, IoT vendors have been slow to deploy PKI for a number of reasons:

1. The CA introduces a single point of failure due to the design of conventional PKI infrastructures. Therefore, if an attacker manages to take over the CA, they will be able to issue certificates for whomever they choose. This problem is particularly crucial in IoT contexts because the vast array of connected devices increases the possible attack surface.
2. Traditional cryptographic algorithms will have severe performance constraints when implemented on regular IoT hardware due to the resource-constrained nature of IoT devices. To illustrate, Blanc et al. [26] have ported 12 encryption algorithms to multiple IoT hardware platforms, to compare their performance. Their experimental results show that the algorithms run 6-50 times slower than on traditional PCs, depending on the architecture.

By creating a decentralized PKI system that uses lightweight cryptography and is suitable for IoT devices with minimal processing power, this thesis seeks to overcome the aforementioned problems. The aforementioned issues emphasize the importance of this research, as do the numerous attacks on IoT devices that have been documented [8, 10–18].

8.2 Contributions

In this thesis, we have provided two contributions. Firstly, a SLR on the academic landscape regarding PKI for the Internet of Things (IoT). This SLR has resulted in more than 30 articles, which have been analyzed based on metadata and content. From this analysis, we have concluded that in the current academic landscape PKI solutions make use of mostly five technologies: Elliptic Curve Cryptography (ECC), decentralized technologies, DNA cryptography, pairing-based cryptography, and Physical Unclonable Functions (PUFs). We have summarized all 37 articles, and provided a concise overview of all advantages and disadvantages of the five technologies found in the SLR. We end the SLR by giving a comprehensive overview of all articles, by listing their features, such as technology used, and results of their performance and security analyses. To the best of our knowledge, no such SLR has been conducted before, thus stressing the importance of this contribution.

The second contribution of this thesis is the design, implementation and evaluation of a Public Key Infrastructure (PKI), tailored to Internet of Things (IoT) devices. This implementation aims to address the research objectives stated earlier, by creating a PKI that is not only lightweight in terms of computational requirements, but that also scales well in terms of IoT devices. These two aspects are especially important in the IoT context, since such devices are characterized by having stringent resource constraints, and are expected to come by the billions in the upcoming years [8]. We have not only found that it is feasible to implement a PKI for IoT, but that this PKI, in some aspects, performs better than existing PKIs in the literature (Section 6.3). This stresses the importance of this contribution, as secure communication is very important among IoT devices, and should not introduce too significant computational and communication overhead.

8.3 Research questions

In this section, we aim to answer the research questions proposed at the start of this thesis (Chapter 1).

8.3.1 RQ1: What are the current issues regarding PKI for IoT?

As the Internet of Things (IoT) landscape expands quickly, connecting everything from sensors to appliances, there is a growing need for a Public Key Infrastructure (PKI) tailored for IoT devices. Nevertheless, the current traditional PKI architecture is not suitable for billions of computationally limited IoT devices, due to two main reasons:

1. Traditional PKI, which depends on a single Certificate Authority (CA), may run into administration and scalability issues in massive IoT deployments. IoT device numbers can reach billions [8], making managing unique certificates for each device difficult and time-consuming. Furthermore, the vast number of IoT devices increases the attack surface of the PKI, which makes it easier to become compromised, thus jeopardizing all IoT devices due to the traditional PKI's centralized nature.
2. IoT devices often have limited energy, memory, and computing resources. For IoT devices with limited resources, traditional cryptographic methods can be unusable because they frequently need a lot of computing power and memory to execute encryption and decryption operations. It is possible that these devices lack the processing power required to handle the computational overhead of conventional cryptography. It is therefore important to develop a PKI that takes this into account, by for example using lightweight cryptography such as ECC.

8.3.2 RQ2: What are the lightweight PKI solutions already present in the literature?

To answer this research question, we have conducted a Systematic Literature Review (SLR), to identify the current academic landscape regarding PKI for IoT. As a result, we have found 19

articles that propose novel PKI solutions, tailored to IoT devices (Table 4.2). The majority of these articles (32%) propose PKI systems based on decentralized technologies, such as blockchain. Moreover, other technologies used in proposed solutions include X.509 optimizations, Physical Unclonable Functions PUF, and Elliptic Curve Cryptography (ECC). All three individually contribute to 16% of the total number of articles. This reflects the need for a scalable and lightweight PKI solution, which addresses the two main problems found in the traditional PKI systems regarding IoT, namely the device's vast numbers and inherently limited computational capabilities. This idea is further cemented when studying the articles that propose cryptographic algorithms for the IoT (Table 4.3), where over 68% of those articles use ECC as their main contributing factor.

8.3.3 RQ3: How do we implement a lightweight PKI solution for the IoT?

As the IoT environment expands quickly, connecting everything from sensors to appliances, there is a growing need for a scalable and decentralized PKI. The specific security concerns that IoT ecosystems raise can be addressed using a decentralized, lightweight PKI. First, given the nature of IoT networks, a PKI architecture for the IoT must be decentralized. Decentralized PKIs increase attack resilience by doing away with the requirement for a centralized authority to manage certificates, which reduces single points of failure. Second, a lightweight PKI system is a PKI system designed specifically for IoT devices with constrained resources. These devices usually have little memory, computation, and energy capacities. Traditional PKI architectures, which were initially developed for more powerful computing environments, can hinder the performance and effectiveness of IoT devices.

We have addressed these two points by implementing a lightweight PKI. The contributions of this lightweight PKI are two-fold:

1. We have proposed a decentralized PKI system as a solution for the single point of failure that traditional PKI architectures have. We have suggested introducing an architecture that makes use of "zones", each of which has a master. All other IoT nodes in its zone have a master that issues, updates, revokes, and searches for certificates for them. Consequently, compared to the conventional PKI design, it functions much like a Certificate Authority (CA). Zone masters are responsible for keeping track of all nodes' certificates within their zone. If a zone has more nodes than it can manage, one of the nodes in the zone will be changed to a new master node, forming a new zone that is connected to the old zone. A parent-child architecture is created as a result, with links between each zone (aside from the root zone).

2. The second contribution has addressed the need for a lightweight architecture, which is achieved in two following ways: (i) Elliptic Curve Cryptography (ECC), a public key cryptography based on the algebraic structure of elliptic curves over finite fields, is a technique for using lightweight encryption. In comparison to traditional cryptographic techniques like RSA, it has an advantage because it provides the same level of security with smaller key sizes [89]. (ii) The usage of lightweight certificates: In order to speed up certificate creation and verification, the suggested design will drastically reduce the amount and size of fields included in the current X.509 certificates. We have encoded the certificates using Concise Binary Object Representation (CBOR) to further reduce communication overhead.

8.3.4 RQ4: How will this new lightweight PKI solution perform compared with traditional and literature PKI solutions?

We have gathered five articles from Table 4.2 and 4.3 that are eligible for a performance comparison. These five articles have performance results across two types of metrics: certificate size, and certificate verification (in terms of energy consumption and time).

Only one article has discussed the first type of metric, from the comparison with the article and our work, we have found that there is still room for improvement: our optimized and CBOR encoded

certificates are 28.7% and 155.47% larger respectively than those in the work. We have also argued why this could be the case.

The other four articles have discussed the other metric: certificate verification. From this perspective, our implementation performs much faster and more efficient than the works in the literature. Namely, our implementation is 8.4-724 times faster and 2.1-20 times more energy efficient, when looking at all works. Furthermore, we have argued why our implementation is more efficient. Nevertheless, it must be noted that differences in architectures and hardware performance can have a large influence on such performance results.

8.4 Limitations and future works

This section describes some limitations of the proposed PKI solution, along with potential future works to address the limitations.

8.4.1 Certificate queries

Nodes are not updated on the new certificates of other nodes due to the architecture's design. In order to share data securely, nodes must periodically ask the zone master for certificates. The zone master may be subject to a heavy workload as a result. This is a problem that must be solved, with for example client-side certificate caching. Nodes can cache certificates so that they do not have to query the zone master every time they want to communicate.

8.4.2 TOFU on enrollment

The node enrollment strategy used by the current system is based on the idea of Trust on First Use (TOFU) [92]. Essentially, the zone master assumes that the node is benign during this process and will not falsify its identity, as indicated by its UUID, when generating its certificate. For a future work, this issue can be solved, potentially by a cryptographic verification mechanism that introduces unique fingerprints for each IoT node, so that zone masters are able to distinguish between nodes.

8.4.3 Implementation bottleneck

The proposed solution has been implemented in Python, with the reason that it is just a proof-of-concept. While the implementation can be deployed on hardware such as a Raspberry Pi, this is not the case for smaller hardware such as an ESP32. For a potential future work, the proposed solution can be implemented in a lightweight variant of Python such as MicroPython[1], or an even lower level language such as C.

8.4.4 Certificate verification

If a node wants to verify a certificate of a node, it has to traverse the entire certificate chain from the node's zone master, all the way to the root zone. To this end, it has to request the certificates and signatures of all mentioned zone masters. This can impose a significant burden on the first number of zone masters, potentially bottlenecking the architecture. A potential solution could be to verify only a number of parent zone masters, while still ensuring secure certificate verification.

[1] https://micropython.org/

Bibliography

[1] Nees Jan Van Eck and Ludo Waltman. Text mining and visualization using VOSviewer. *arXiv preprint arXiv:1109.2058*, 2011.

[2] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280, RFC Editor, May 2008. URL http://www.rfc-editor.org/rfc/rfc5280.txt. http://www.rfc-editor.org/rfc/rfc5280.txt.

[3] Muhammad Shafiq, Zhaoquan Gu, Omar Cheikhrouhou, Wajdi Alhakami, and Habib Hamam. The Rise of "Internet of Things": Review and Open Research Issues Related to Detection and Prevention of IoT-Based Security Attacks. *Wireless Communications and Mobile Computing*, 2022:1–12, aug 2022. doi: 10.1155/2022/8669348. URL https://doi.org/10.1155%2F2022%2F8669348.

[4] Mari Carmen Domingo. An overview of the Internet of Things for people with disabilities. *Journal of Network and Computer Applications*, 35(2):584–596, mar 2012. doi: 10.1016/j.jnca.2011.10.015. URL https://doi.org/10.1016%2Fj.jnca.2011.10.015.

[5] Ming-Che Hsieh, Wei-Sheng Hung, Shu-Wen Lin, and Chin-Hsing Luo. Designing an assistive dialog agent for a case of spinal cord injury. In *2009 Ninth International Conference on Hybrid Intelligent Systems*, volume 1, pages 67–72. IEEE, 2009.

[6] Jongho Won, Ankush Singla, Elisa Bertino, and Greg Bollella. Decentralized public key infrastructure for internet-of-things. In *MILCOM 2018-2018 IEEE Military Communications Conference (MILCOM)*, pages 907–913. IEEE, 2018.

[7] Joel Höglund, Samuel Lindemer, Martin Furuhed, and Shahid Raza. PKI4IoT: Towards public key infrastructure for the Internet of Things. *Computers & Security*, 89:101658, feb 2020. doi: 10.1016/j.cose.2019.101658. URL https://doi.org/10.1016%2Fj.cose.2019.101658.

[8] Amy Nordrum. Popular Internet of Things Forecast of 50 Billion Devices by 2020 Is Outdated - IEEE Spectrum, 2016. URL https://spectrum.ieee.org/popular-internet-of-things-forecast-of-50-billion-devices-by-2020-is-outdated. Accessed: 2022-11-29.

[9] Nataliia Neshenko, Elias Bou-Harb, Jorge Crichigno, Georges Kaddoum, and Nasir Ghani. Demystifying IoT Security: An Exhaustive Survey on IoT Vulnerabilities and a First Empirical Look on Internet-Scale IoT Exploitations. *IEEE Communications Surveys & Tutorials*, 21 (3):2702–2733, 2019. doi: 10.1109/comst.2019.2910750. URL https://doi.org/10.1109%2Fcomst.2019.2910750.

[10] Alex Biryukov, Daniel Dinu, and Yann Le Corre. Side-channel attacks meet secure network protocols. In *International conference on applied cryptography and network security*, pages 435–454. Springer, 2017.

[11] Colin OFlynn and Zhizhang Chen. Power analysis attacks against IEEE 802.15. 4 nodes. In *International Workshop on Constructive Side-Channel Analysis and Secure Design*, pages 55–70. Springer, 2016.

[12] Anjana Rajan, J Jithish, and Sriram Sankaran. Sybil attack in IOT: Modelling and defenses.

In *2017 international conference on advances in computing, communications and informatics (ICACCI)*, pages 2323–2327. IEEE, 2017.

[13] Zachry Basnight, Jonathan Butts, Juan Lopez Jr, and Thomas Dube. Firmware modification attacks on programmable logic controllers. *International Journal of Critical Infrastructure Protection*, 6(2):76–84, 2013.

[14] Ang Cui, Michael Costello, and Salvatore Stolfo. When firmware modifications attack: A case study of embedded exploitation. 2013.

[15] Boldizsár Bencsáth Levente Buttyán Tamás Paulik. XCS based hidden firmware modification on embedded devices. 2011.

[16] Eugene Y Vasserman and Nicholas Hopper. Vampire attacks: draining life from wireless ad hoc sensor networks. *IEEE transactions on mobile computing*, 12(2):318–332, 2011.

[17] Chiara Pielli, Federico Chiariotti, Nicola Laurenti, Andrea Zanella, and Michele Zorzi. A game-theoretic analysis of energy-depleting jamming attacks. In *2017 International Conference on Computing, Networking and Communications (ICNC)*, pages 100–104. IEEE, 2017.

[18] Elisa Bertino and Nayeem Islam. Botnets and Internet of Things Security. *Computer*, 50(2): 76–79, feb 2017. doi: 10.1109/mc.2017.62. URL https://doi.org/10.1109%2Fmc.2017.62.

[19] Manos Antonakakis, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J Alex Halderman, Luca Invernizzi, Michalis Kallitsis, et al. Understanding the MIRAI botnet. In *26th {USENIX} security symposium ({USENIX} Security 17)*, pages 1093–1110, 2017.

[20] DDoS on Dyn Impacts Twitter, Spotify, Reddit; Krebs on Security, oct 2016. URL https://krebsonsecurity.com/2016/10/ddos-on-dyn-impacts-twitter-spotify-reddit/. Accessed: 2023-01-27.

[21] Joan Daemen and Vincent Rijmen. AES proposal: Rijndael. 1999.

[22] R. Shirey. Internet Security Glossary, Version 2. Technical report, aug 2007. URL https://doi.org/10.17487%2Frfc4949.

[23] Alfred J. Menezes. *Handbook of Applied Cryptography*. Taylor & Francis Inc, 11 1997. ISBN 9780849385230.

[24] Ronald L. Rivest. The MD5 Message-Digest Algorithm. RFC 1321, RFC Editor, April 1992. URL http://www.rfc-editor.org/rfc/rfc1321.txt.

[25] Hung-Yu Chien. Dynamic Public Key Certificates with Forward Secrecy. *Electronics*, 10 (16):2009, aug 2021. doi: 10.3390/electronics10162009. URL https://doi.org/10.3390%2Felectronics10162009.

[26] Soline Blanc, Abdelkader Lahmadi, Kévin Le Gouguec, Marine Minier, and Lama Sleem. Benchmarking of lightweight cryptographic algorithms for wireless IoT networks. *Wireless Networks*, 28(8):3453–3476, jul 2022. doi: 10.1007/s11276-022-03046-1. URL https://doi.org/10.1007%2Fs11276-022-03046-1.

[27] Susovan Chanda, Ashish Kumar Luhach, Waleed Alnumay, Indranil Sengupta, and Diptendu Sinha Roy. A lightweight device-level Public Key Infrastructure with DRAM based Physical Unclonable Function (PUF) for secure cyber physical systems. *Computer Communications*, 190:87–98, 2022.

[28] Benamar Kadri, Mohammed Feham, and Abdallah Mhamed. Lightweight PKI for WSN μPKI. *The Journal of Security and Communication Networks*, 10(2):135–141, 2010.

[29] Mohsen Toorani and A Beheshti. LPKI-a lightweight public key infrastructure for the mobile environments. In *2008 11th IEEE Singapore International Conference on Communication Systems*, pages 162–166. IEEE, 2008.

[30] Loïc Champagne. Replacing Public Key Infrastructures (PKI) by blockchain IoT devices security management. 2021.

[31] Michelle S Henriques and Nagaraj K Vernekar. Using symmetric and asymmetric cryptography to secure communication between devices in IoT. In *2017 International Conference on IoT and Application (ICIOT)*, pages 1–4. IEEE, 2017.

[32] Darshana Pritam Shah and Pritam Gajkumar Shah. Revisting of elliptical curve cryptography for securing Internet of Things (IOT). In *2018 Advances in Science and Engineering Technology International Conferences (ASET)*, pages 1–3. IEEE, 2018.

[33] Roel J. Wieringa. *Design Science Methodology for Information Systems and Software Engineering*. Springer Berlin Heidelberg, 2014. doi: 10.1007/978-3-662-43839-8. URL https://doi.org/10.1007%2F978-3-662-43839-8.

[34] What is a Systematic Review? URL https://guides.temple.edu/c.php?g=78618&p=4178713. Accessed: 2023-1-13.

[35] Captain John Adams, Hafiz T A Khan, Robert Raeside, and David I White. *Research methods for graduate business and social science students*. SAGE Publications, Thousand Oaks, CA, August 2007.

[36] Christopher Greer, Martin Burns, David Wollman, Edward Griffor, et al. Cyber-physical systems and internet of things, 2019.

[37] Johana A. Manrique, Johan S. Rueda-Rueda, and Jesus M.T. Portocarrero. Contrasting Internet of Things and Wireless Sensor Network from a Conceptual Overview. In *2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*. IEEE, dec 2016. doi: 10.1109/ithings-greencom-cpscom-smartdata.2016.66. URL https://doi.org/10.1109%2Fithings-greencom-cpscom-smartdata.2016.66.

[38] Keping Yu, Liang Tan, Caixia Yang, Kim-Kwang Raymond Choo, Ali Kashif Bashir, Joel JPC Rodrigues, and Takuro Sato. A blockchain-based Shamirs threshold cryptography scheme for data protection in industrial internet of things settings. *IEEE Internet of Things Journal*, 2021.

[39] A. Liu and P. Ning. TinyECC: A configurable library for elliptic curve cryptography in wireless sensor networks. pages 245–256, 2008. doi: 10.1109/IPSN.2008.47. URL https://doi.org/10.1109/IPSN.2008.47.

[40] W. Shi and P. Gong. A new user authentication protocol for wireless sensor networks using elliptic curves cryptography. *International Journal of Distributed Sensor Networks*, 2013, 2013. doi: 10.1155/2013/730831. URL https://doi.org/10.1155/2013/730831.

[41] M. Elhoseny, H. Elminir, A. Riad, and X. Yuan. A secure data routing schema for WSN using Elliptic Curve Cryptography and homomorphic encryption. *Journal of King Saud University - Computer and Information Sciences*, 28(3):262–275, 2016. doi: 10.1016/j.jksuci.2015.11.001. URL https:/doi.org/10.1016/j.jksuci.2015.11.001.

[42] P. Szczechowiak, A. Kargl, M. Scott, and M. Collier. On the application of Pairing Based Cryptography to Wireless Sensor Networks. pages 1–12, 2009. doi: 10.1145/1514274.1514276. URL https://doi.org/10.1145/1514274.1514276.

[43] M. Khari, A.K. Garg, A.H. Gandomi, R. Gupta, R. Patan, and B. Balusamy. Securing Data in Internet of Things (IoT) Using Cryptography and Steganography Techniques. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 50(1):73–80, 2020. doi: 10.1109/TSMC.2019.2903785. URL https://doi.org/10.1109/TSMC.2019.2903785.

[44] Firas Albalas, Majd Al-Soud, Omar Almomani, and Ammar Almomani. Security-aware CoAP application layer protocol for the internet of things using elliptic-curve cryptography. *Power (mw)*, 1333:151, 2018.

[45] Rosheen Qazi, Kashif Naseer Qureshi, Faisal Bashir, Najam Ul Islam, Saleem Iqbal, and Arsalan Arshad. Security protocol using elliptic curve cryptography algorithm for wireless sensor networks. *Journal of Ambient Intelligence and Humanized Computing*, 12(1):547–566, apr 2020. doi: 10.1007/s12652-020-02020-z. URL https://doi.org/10.1007%2Fs12652-020-02020-z.

[46] T Daisy Premila Bai, K Michael Raj, and S Albert Rabara. Elliptic curve cryptography based security framework for Internet of Things (IoT) enabled smart card. In *2017 World Congress on Computing and Communication Technologies (WCCCT)*, pages 43–46. IEEE, 2017.

[47] Aakanksha Tewari and Brij B Gupta. A lightweight mutual authentication protocol based on elliptic curve cryptography for IoT devices. *International Journal of Advanced Intelligence Paradigms*, 9(2-3):111–121, 2017.

[48] Wenbo Shi and Peng Gong. A New User Authentication Protocol for Wireless Sensor Networks Using Elliptic Curves Cryptography. *International Journal of Distributed Sensor Networks*, 9 (4):730831, apr 2013. doi: 10.1155/2013/730831. URL https://doi.org/10.1155%2F2013%2F730831.

[49] Hsiu-Lien Yeh, Tien-Ho Chen, Pin-Chuan Liu, Tai-Hoo Kim, and Hsin-Wen Wei. A secured authentication protocol for wireless sensor networks using elliptic curves cryptography. *Sensors*, 11:4767–4779, 2011.

[50] Kishore Rajendiran, Radha Sankararajan, and Ramasamy Palaniappan. A Secure Key Predistribution Scheme for WSN Using Elliptic Curve Cryptography. *ETRI Journal*, 33(5):791–801, oct 2011. doi: 10.4218/etrij.11.0110.0665. URL https://doi.org/10.4218%2Fetrij.11.0110.0665.

[51] Rolf Blom. An optimal class of symmetric key generation systems. In *Workshop on the Theory and Application of Cryptographic Techniques*, pages 335–338. Springer, 1984.

[52] J Louw, G Niezen, TD Ramotsoela, and Adnan M Abu-Mahfouz. A key distribution scheme using elliptic curve cryptography in wireless sensor networks. In *2016 IEEE 14th International Conference on Industrial Informatics (INDIN)*, pages 1166–1170. IEEE, 2016.

[53] Song Ju. A lightweight key establishment in wireless sensor network based on elliptic curve cryptography. In *2012 IEEE International Conference on Intelligent Control, Automatic Detection and High-End Equipment*, pages 138–141. IEEE, 2012.

[54] Oriol Pinol Pinol, Shahid Raza, Joakim Eriksson, and Thiemo Voigt. BSD-based elliptic curve cryptography for the open Internet of Things. In *2015 7th International Conference on New Technologies, Mobility and Security (NTMS)*, pages 1–5. IEEE, 2015.

[55] Zhe Liu and Hwajeong Seo. IoT-NUMS: evaluating NUMS elliptic curve cryptography for IoT platforms. *IEEE Transactions on Information Forensics and Security*, 14(3):720–729, 2018.

[56] Carlos Andres Lara-Nino, Arturo Diaz-Perez, and Miguel Morales-Sandoval. Lightweight elliptic curve cryptography accelerator for internet of things applications. *Ad Hoc Networks*, 103:102159, 2020.

[57] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Decentralized business review*, page 21260, 2008.

[58] Mathijs Hoogland. A Distributed Public Key Infrastructure for the IoT. 2018.

[59] Ankush Singla and Elisa Bertino. Blockchain-based PKI solutions for IoT. In *2018 IEEE 4th International Conference on Collaboration and Internet Computing (CIC)*, pages 9–15. IEEE, 2018.

[60] Sebastian Magnusson. Evaluation of Decentralized Alternatives to PKI for IoT Devices: A literature study and proof of concept implementation to explore the viability of replacing PKI with decentralized alternatives, 2018.

[61] Mustafa Al-Bassam. SCPKI: A smart contract-based PKI and identity system. In *Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts*, pages 35–40, 2017.

[62] Lorenzo Pintaldi. *Implementation of a Blockchain-based Distributed PKI for IoT using Emercoin NVS and TPM 2.0*. PhD thesis, Politecnico di Torino, 2022.

[63] Guozhen Xiao, Mingxin Lu, Lei Qin, and Xuejia Lai. New field of cryptography: DNA cryptography. *Chinese Science Bulletin*, 51:1413–1420, 2006.

[64] Harsh Durga Tiwari and Jae Hyung Kim. Novel Method for DNA-Based Elliptic Curve Cryptography for IoT Devices. *ETRI Journal*, 40(3):396–409, jun 2018. doi: 10.4218/etrij.2017-0220. URL https://doi.org/10.4218%2Fetrij.2017-0220.

[65] Mohammed Abbas Fadhil Al-Husainy, Bassam Al-Shargabi, and Shadi Aljawarneh. Lightweight cryptography system for IoT devices using DNA. *Computers and Electrical Engineering*, 95:107418, 2021.

[66] Dustin Moody, Rene Peralta, Ray Perlner, Andrew Regenscheid, Allen Roginsky, and Lily Chen. Report on Pairing-based Cryptography. *Journal of Research of the National Institute of Standards and Technology*, 120:11, jan 2015. doi: 10.6028/jres.120.002. URL https://doi.org/10.6028%2Fjres.120.002.

[67] Binbin Yu and Hongtu Li. Anonymous authentication key agreement scheme with pairing-based cryptography for home-based multi-sensor internet of things. *International Journal of Distributed Sensor Networks*, 15(9):155014771987937, sep 2019. doi: 10.1177/1550147719879379. URL https://doi.org/10.1177%2F1550147719879379.

[68] Michael Burrows, Martin Abadi, and Roger Needham. A logic of authentication. *ACM Transactions on Computer Systems (TOCS)*, 8(1):18–36, 1990.

[69] Yansong Gao, Said F Al-Sarawi, and Derek Abbott. Physical unclonable functions. *Nature Electronics*, 3(2):81–91, 2020.

[70] Zeeshan Siddiqui, Jiechao Gao, and Muhammad Khurram Khan. An Improved Lightweight PUF-PKI Digital Certificate Authentication Scheme for the Internet of Things. *IEEE Internet of Things Journal*, 2022.

[71] Michael Schukat and Pablo Cortijo. Public key infrastructures and digital certificates for the Internet of Things. In *2015 26th Irish signals and systems conference (ISSC)*, pages 1–5. IEEE, 2015.

[72] Nouf Aljadani and Tahani Gazdar. A New distributed PKI for WSN-Based Application in Smart Grid. In *The 4th International Conference on Future Networks and Distributed Systems (ICFNDS)*, pages 1–5, 2020.

[73] Filip Forsby, Martin Furuhed, Panos Papadimitratos, and Shahid Raza. Lightweight x. 509 digital certificates for the internet of things. In *Interoperability, Safety and Security in IoT: Third International Conference, InterIoT 2017, and Fourth International Conference, SaSeIot 2017, Valencia, Spain, November 6-7, 2017, Proceedings 3*, pages 123–133. Springer, 2018.

[74] Francesco Marino, Corrado Moiso, and Matteo Petracca. PKIoT: A public key infrastructure for the Internet of Things. *Transactions on Emerging Telecommunications Technologies*, 30 (10), jul 2019. doi: 10.1002/ett.3681. URL https://doi.org/10.1002%2Fett.3681.

[75] Bayu Anggorojati and Ramjee Prasad. Securing communication in inter domains Internet of Things using identity-based cryptography. In *2017 International Workshop on Big Data and Information Security (IWBIS)*, pages 137–142. IEEE, 2017.

[76] Daniel Diaz-Sanchez, Andrés Marín-Lopez, Florina Almenárez Mendoza, Patricia Arias Cabarcos, and R Simon Sherratt. TLS/PKI challenges and certificate pinning techniques for IoT and M2M secure communications. *IEEE Communications Surveys & Tutorials*, 21(4): 3502–3531, 2019.

[77] Nicholas Jansma and Brandon Arrendondo. Performance comparison of elliptic curve and RSA digital signatures. *nicj. net/files*, 2004.

[78] The Case for Elliptic Curve Cryptography - NSA/CSS — web.archive.org, jan 2009. URL https://web.archive.org/web/20090117023500/http://www.nsa.gov/business/programs/elliptic_curve.shtml. Accessed: 2023-05-31.

[79] Lucianna Kiffer, Dave Levin, and Alan Mislove. Stick a fork in it: Analyzing the Ethereum network partition. In *Proceedings of the 16th ACM Workshop on Hot Topics in Networks*, pages 94–100, 2017.

[80] Pooja Singh and RK Chauhan. A survey on comparisons of cryptographic algorithms using certain parameters in wsn. *International Journal of Electrical & Computer Engineering (2088-8708)*, 7(4), 2017.

[81] Jun Jiang and Zhixiang Yin. The advantages and disadvantages of DNA password in the contrast to the traditional cryptography and quantum cryptography. In *Proceedings of The Eighth International Conference on Bio-Inspired Computing: Theories and Applications (BICTA), 2013*, pages 307–316. Springer, 2013.

[82] Jens Groth. Short Pairing-Based Non-interactive Zero-Knowledge Arguments. In *Asiacrypt*, volume 6477, pages 321–340. Springer, 2010.

[83] Dan Boneh and Matt Franklin. Identity-based encryption from the Weil pairing. In *Advances in CryptologyCRYPTO 2001: 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19–23, 2001 Proceedings*, pages 213–229. Springer, 2001.

[84] Eric Zavattoni, Luis J Dominguez Perez, Shigeo Mitsunari, Ana H Sánchez-Ramırez, Tadanori Teruya, and Francisco Rodríguez-Henríquez. Software implementation of an attribute-based encryption scheme. *IEEE Transactions on Computers*, 64(5):1429–1441, 2014.

[85] Zhengjun Cao and Lihua Liu. On the Disadvantages of Pairing-based Cryptography. Cryptology ePrint Archive, Paper 2015/084, 2015. URL https://eprint.iacr.org/2015/084.

[86] Tyler Cultice and Himanshu Thapliyal. PUF-based post-quantum CAN-FD framework for vehicular security. *Information*, 13(8):382, aug 2022. doi: 10.3390/info13080382. URL https://doi.org/10.3390%2Finfo13080382.

[87] Kamal Y. Kamal and Radu Muresan. Mixed-signal physically unclonable function with CMOS capacitive cells. *IEEE Access*, 7:130977–130998, 2019. doi: 10.1109/access.2019.2938729. URL https://doi.org/10.1109%2Faccess.2019.2938729.

[88] Roel Maes and Vincent van der Leest. Countering the effects of silicon aging on SRAM PUFs. In *2014 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*. IEEE, may 2014. doi: 10.1109/hst.2014.6855586. URL https://doi.org/10.1109%2Fhst.2014.6855586.

[89] Elaine Barker. Recommendation for Key Management:. Technical report, may 2020. URL https://doi.org/10.6028%2Fnist.sp.800-57pt1r5.

[90] Darrel Hankerson and Alfred Menezes. Elliptic curve discrete logarithm problem. In *Encyclopedia of Cryptography and Security*, pages 397–400. Springer US, 2011. doi: 10.1007/978-1-4419-5906-5_246. URL https://doi.org/10.1007%2F978-1-4419-5906-5_246.

[91] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, nov 1976. doi: 10.1109/tit.1976.1055638. URL https://doi.org/10.1109%2Ftit.1976.1055638.

[92] Neal H Walfield and Werner Koch. TOFU for OpenPGP. In *Proceedings of the 9th European Workshop on System Security*, pages 1–6, 2016.

[93] M. Cooper, Y. Dzambasow, P. Hesse, S. Joseph, and R. Nicholas. Internet X.509 Public Key Infrastructure: Certification Path Building. RFC 4158, RFC Editor, September 2005.

URL http://www.rfc-editor.org/rfc/rfc4158.txt. http://www.rfc-editor.org/rfc/rfc4158.txt.

[94] C. Bormann and P. Hoffman. Concise Binary Object Representation (CBOR). RFC 7049, RFC Editor, October 2013.

[95] High-performance, IEEE 802.15.4 wireless system-on-chip with up to 256 Kbytes of embedded Flash memory. URL https://www.farnell.com/datasheets/1781170.pdf. Accessed: 2023-06-07.

[96] Ken MacKay. micro-ecc, 2014. URL https://github.com/kmackay/micro-ecc.

[97] Luca Vigano. Automated security protocol analysis with the AVISPA tool. *Electronic Notes in Theoretical Computer Science*, 155:61–86, 2006.

[98] Ueli Maurer. Information-theoretic cryptography. In *Advances in Cryptology — CRYPTO' 99*, pages 47–65. Springer Berlin Heidelberg, 1999. doi: 10.1007/3-540-48405-1_4. URL https://doi.org/10.1007%2F3-540-48405-1_4.

[99] Danny Dolev and Andrew Yao. On the security of public key protocols. *IEEE Transactions on information theory*, 29(2):198–208, 1983.

Appendix A

HLPSL models

A.1 Node enrollment (Section 5.2)

```
role role_N(N:agent,S:agent,K:symmetric_key,SND,RCV:channel(dy))
played_by N
def=
        local
            State:nat,
        R:text,C:text
        init
            State := 0
        transition
            1. State=0 /\ RCV(start) =|>
                   State':=1
                   /\ R':=new()
                   /\ SND({R'}_K)
            2. State=1 /\ RCV({C'}_K) =|>
                   State':=2
end role

role role_S(S:agent,N:agent,C:text,K:symmetric_key,SND,RCV:channel(dy))
played_by S
def=
        local
                State:nat,
        Nonce:text,R:text
        init
                State := 0
        transition
                1. State=0 /\ RCV({R'}_K) =|>
                   State':=1
                   /\ SND({C}_K)
                   /\ secret(C,sec_1,{N,S})
end role

role session(N:agent,S:agent,C:text,K:symmetric_key)
def=
        local
                SND1,RCV1,SND2,RCV2:channel(dy)
        composition
```

```
                role_S(S,N,C,K,SND1,RCV1) /\ role_N(N,S,K,SND2,RCV2)
end role

role environment()
def=
        const
            k:symmetric_key,
        node,server:agent,
        c:text,
        sec_1:protocol_id
        intruder_knowledge = {node,server}
        composition
                session(node,server,c,k)
end role

goal
        secrecy_of sec_1
end goal

environment()
```

A.2 Subsequent communication (Section 5.2.1)

```
role role_N(N:agent,S:agent,Kn:public_key,Ks:public_key,SND,RCV:channel(dy))
played_by N
def=
        local
                State:nat,Nonce:text,Request,Response:text
        init
                State := 0
        transition
                1. State=0 /\ RCV(start) =|>
                   State':=1
                   /\ Nonce':=new()
                   /\ Request' := new()
                   /\ SND({Nonce'.Request'}_Ks)

                2. State=1 /\ RCV({Nonce.Response'}_Kn) =|>
                   State':=2
                   %% Verify nonce
                   /\ request(N,S,auth_1,Nonce)
end role

role role_S(S:agent,N:agent,Kn:public_key,Ks:public_key,SND,RCV:channel(dy))
played_by S
def=
        local
                State:nat,Nonce:text,Request,Response:text
        init
                State := 0
        transition
                1. State=0 /\ RCV({Nonce'.Request'}_Ks) =|>
                   State':=1
                   /\ Response' := new()
                   /\ SND({Nonce'.Response'}_Kn)
```

```
                    /\ secret(Response',sec_1,{N,S})
                    /\ witness(S,N,auth_1,Nonce')
end role

role session(N:agent,S:agent,Kn:public_key,Ks:public_key)
def=
        local
                SND1,RCV1,SND2,RCV2:channel(dy)
        composition
                role_S(S,N,Kn,Ks,SND1,RCV1) /\ role_N(N,S,Kn,Ks,SND2,RCV2)
end role

role environment()
def=
        const
                kn,ks:public_key,
                node,server:agent,
                sec_1,auth_1:protocol_id
        intruder_knowledge = {node,server,kn,ks}
        composition
                session(node,server,kn,ks)
end role

goal
        secrecy_of sec_1
        authentication_on auth_1
end goal

environment()
```