UPPSALA
UNIVERSITET

Digital Comprehensive Summaries of Uppsala Dissertations from the Faculty of Science and Technology 2230

Public Key Infrastructure and its applications for resource-constrained IoT

JOEL HÖGLUND

ACTA
UNIVERSITATIS
UPSALIENSIS
UPPSALA
2023

ISSN 1651-6214
ISBN 978-91-513-1696-3
URN urn:nbn:se:uu:diva-495320

Dissertation presented at Uppsala University to be publicly examined in Häggsalen, Ångströmlaboratoriet, Lägerhyddsvägen 1, Wednesday, 15 March 2023 at 13:15 for the degree of Doctor of Philosophy. The examination will be conducted in English. Faculty examiner: Professor Andrei Gurtov (Linköping University).

Abstract
Höglund, J. 2023. Public Key Infrastructure and its applications for resource-constrained IoT. *Digital Comprehensive Summaries of Uppsala Dissertations from the Faculty of Science and Technology* 2230. 49 pp. Uppsala: Acta Universitatis Upsaliensis. ISBN 978-91-513-1696-3.

The Internet of Things (IoT) is rapidly expanding and IoT devices are being deployed in security-critical scenarios, such as in critical infrastructure monitoring and within e-health, and privacy-sensitive applications in hospitals and homes. With this, questions of security and safety become paramount. The overall theme of the research presented here is to bridge some of the identified gaps in IoT security, with a particular focus on enabling Public Key Infrastructure (PKI) functionality for constrained IoT devices. The contributions of this dissertation are made through six research papers that address identified shortcomings and challenges. The focus is on protocols, mechanisms, and efficient encodings rather than specific cryptographic solutions. The work to improve the state-of-art regarding PKI for IoT includes enrollment, revocation and trust transfer. We design and implement integrated lightweight certificate enrollment solutions for IoT devices and new compact certificate formats. This brings the total communication costs of session establishment and enrollment operations down to feasible levels for constrained IoT devices. An improved design is made to benefit from application layer security, enabling end-to-end communication capable of proxy traversal. To handle revocation of trust, we propose and design lightweight certificate revocation. We show how significant performance improvements compared with existing solutions can be made without sacrificing functionality or compromising security. To address the long-time maintainability of IoT systems, we design a lightweight schema for trust transfer, which allows control of IoT deployments to shift between service providers in a highly automated manner.

In addition to improving PKI functionality, we propose mechanisms for secure storage and updates, which complement and strengthen the overall IoT security landscape. We show that standard-based application-layer security mechanisms can be extended to enable secure storage and communication, reducing the memory required for cryptographic solutions and the latency when sending sensor data onto the network. In our last contribution, we propose a design for secure software updates. Based on the existing ACE framework, we use token-based access control to fulfil the need for both authentication and authorisation security services.

We have been working with industry partners to share our work in the shape of new standards for a better potential for industrial impact. In summary, several new building blocks required to create, maintain and support secure PKIs capable of including constrained IoT devices are proposed, forming important steps towards making IoT devices first-class Internet citizens.

Keywords: IoT, PKI, cybersecurity, security, asymmetric cryptography, Contiki-NG

Joel Höglund, Department of Information Technology, Box 337, Uppsala University, SE-75105 Uppsala, Sweden.

© Joel Höglund 2023

ISSN 1651-6214
ISBN 978-91-513-1696-3
URN urn:nbn:se:uu:diva-495320 (http://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-495320)

List of papers

This thesis is based on the following papers, which are referred to in the text by their Roman numerals.

I Joel Höglund, Samuel Lindemer, Martin Furuhed, and Shahid Raza. "PKI4IoT: Towards public key infrastructure for the Internet of Things". In: *Computers & Security* 89 (2020). DOI: https://doi.org/10.1016/j.cose.2019.101658

II Joel Höglund and Shahid Raza. "LICE: Lightweight certificate enrollment for IoT using application layer security". In: *IEEE Conference on Communications and Network Security, CNS 2021, Tempe, AZ, USA, October 4-6, 2021*. IEEE, 2021. DOI: 10.1109/CNS53000.2021.9705036

III Joel Höglund, Martin Furuhed, and Shahid Raza. "Lightweight certificate revocation for low-power IoT with end-to-end security". In: *Journal of Information Security and Applications* 73 (2023). DOI: https://doi.org/10.1016/j.jisa.2023.103424

IV Joel Höglund, Simon Bouget, Martin Furuhed, Göran Selander, John Mattsson, and Shahid Raza. *AutoPKI: Public Key Infrastructure for IoT with Automated Trust Transfer*. Submitted. 2023

V Joel Höglund and Shahid Raza. "BLEND: Efficient and blended IoT data storage and communication with application layer security". In: *2022 IEEE International Conference on Cyber Security and Resilience (CSR)*. 2022, pp. 253–260. DOI: 10.1109/CSR54599.2022.9850290

VI Joel Höglund, Anum Khurshid, and Shahid Raza. "AC-SIF: ACE Access Control for Standardized Secure IoT Firmware Updates". In: *International Conference on Emerging Security Information, Systems and Technologies*. 2022

Reprints were made with permission from the publishers.

Additional Publications

Peer reviewed papers

- Joel Höglund, Shahid Raza, and Martin Furuhed. "Towards Automated PKI Trust Transfer for IoT". in: *2022 IEEE International Conference on Public Key Infrastructure and its Applications (PKIA)*. 2022, pp. 1–8. DOI: 10.1109/PKIA56009.2022.9952223
- EunSeong Boo, Shahid Raza, Joel Höglund, and JeongGil Ko. "FDTLS: Supporting DTLS-Based Combined Storage and Communication Security for IoT Devices". In: *16th IEEE International Conference on Mobile Ad Hoc and Sensor Systems, MASS 2019, Monterey, CA, USA, November 4-7, 2019*. IEEE, 2019, pp. 127–135
- Ye Liu, Thiemo Voigt, Niklas Wirström, and Joel Höglund. "EcoVibe: On-Demand Sensing for Railway Bridge Structural Health Monitoring". In: *IEEE Internet of Things Journal* 6.1 (2019), pp. 1068–1078. DOI: 10.1109/JIOT.2018.2867086
- Shahid Raza, Simon Duquennoy, Joel Höglund, Utz Roedig, and Thiemo Voigt. "Secure communication for the Internet of Things—a comparison of link-layer security and IPsec for 6LoWPAN". in: *Security and Communication Networks* 7.12 (2014), pp. 2654–2668. DOI: https://doi.org/10.1002/sec.406

Standardisation work

- John Preuß Mattsson, Göran Selander, Shahid Raza, Joel Höglund, and Martin Furuhed. *CBOR Encoded X.509 Certificates (C509 Certificates)*. Internet-Draft draft-ietf-cose-cbor-encoded-cert-05. IETF Secretariat, 2023. URL: https://www.ietf.org/archive/id/draft-ietf-cose-cbor-encoded-cert-05.txt

Contents

Acknowledgement

Personal Acknowledgements
I would like to thank my main supervisor Shahid Raza for supporting me along the journey of my dissertation work and sharing his insights into cybersecurity, including many interesting discussions around security topics. My co-supervisor Thiemo Voigt for feedback and academic insights. My closest manager Joakim Eriksson for being a supportive unit leader. My co-authors Simon Bouget, Anum Kurshid and Samuel Lindemer, and our industry partners Martin Furuhed from Nexus Group and Göran Selander and John Mattsson from Ericsson for good collaborations. My colleagues, current and former, at both the Connected Intelligence and the Cybersecurity unit at RISE for their peer support, Niclas, Nicolas, Zhitao and many more. My thesis students whom I learnt a lot from, through supervising them. All of my RISE colleagues at the Kista office, for creating an enjoyable work environment in general and for a great variety of lunch discussions in particular. The people at the department of Information Technology at Uppsala University, who cared also for industry doctoral students. The management and lab leaders of SICS, including my first manager Sverker Janson, who together created the institute and workplace that I joined many years back, a work environment that made us feel like we were between academia and industry in a very positive way, which inspired me to stay and add my contributions to the field.

I want to express my gratitude to my family, my parents, Boel and Åke, who taught me that there are always more things to be learned for someone who is curious, and who together with my sister Rakel, always have supported me and my choices in life. And shown that one doesn't necessarily have to take the shortest path between degrees.

Finally, I am grateful for my blues dance community friends, who have offered the best possible complement to my academic endeavours, aiding the movement of thoughts by the movement of the body.

Funding Acknowledgements
The research done in this thesis has been funded primarily through the Swedish Foundation for Strategic Research (SSF) industrial PhD program. The work in this thesis is also funded by the following EU H2020 projects SECREDAS (GA No. 783119), ARCADIAN-IoT (GA No. 101020259), CONCORDIA (GA No. 830927) and by Vinnova through the STACK project (GA No. P123800021).

Joel Höglund
Stockholm, January 2023

List of Acronyms

ACE	Authentication and Authorization for Constrained Environments
AS	Authorization Server
ASN.1	Abstract Syntax Notation One
CA	Certificate Authority
CoAP	Constrained Application Protocol
CMS	Certificate Management System
COSE	CBOR Object Signing and Encryption
CRL	Certificate Revocation List
CWT	CBOR Web Token
DTLS	Datagram Transport Layer Security
EDHOC	Ephemeral Diffie-Hellman Over COSE
EST	Enrollment over Secure Transport
IETF	Internet Engineering Task Force
IKE	Internet Key Exchange
MAC	Message Authentication Code
OCSP	Online Certificate Status Protocol
OIDs	object identifiers
OSCORE	Object Security for Constrained RESTful Environments
PKI	Public Key Infrastructure
PQC	Post-Quantum Cryptography
PSK	Pre-shared Key
RATS	Remote ATtestation procedureS
RP	Relying Party
RS	Resource Server
SUIT	Software Updates for Internet of Things
TCP	Transmission Control Protocol
TEEs	Trusted Execution Environments
TLS	Transport Layer Security
UDP	User Datagram Protocol
VA	Validation Authority
WSN	Wireless Sensor Network

Part I:
Dissertation Summary

1. Introduction

The Internet of Things (IoT) has been hyped as a concept for more than a decade, but the last couple of years have shown a rapid increase in actual deployments and everyday usage. IoT solutions are increasingly used in security-critical scenarios such as infrastructure monitoring and within e-health, as well as in privacy-sensitive applications in hospitals and homes. With the fast growth of IoT deployments, questions of security and safety while maintaining scalability become paramount.

Media attention has been given to cybersecurity attacks on critical infrastructure, such as the STUXnet attack [54]. Multiple reports of potential privacy endangerment have been made regarding security flaws in smart assistants [4] and other connected devices in homes, such as toys [19]. In addition, IoT devices that do not have any security critical role can be hacked and, once an attacker has gained control of a number of devices forming a botnet, in turn, used to attack other Internet targets [5]. These examples have helped to increase the awareness of new vulnerabilities when the world is becoming increasingly connected and of the need for strong cybersecurity.

In the early days of IoT, security solutions were absent, insufficient, or, at best provided as custom-made proprietary solutions, creating vendor lock-ins and preventing large-scale interoperability. To a large degree, the lack of security solutions, specifically standard-based security solutions, was due to resource constraints. In parallel, there has been an interest both from academia and industry in bringing standards to constrained devices, due to its potential benefits. With suitable open standards, it becomes easier to deploy and manage IoT devices also in heterogeneous systems, it opens up markets for a more extensive diversity of service providers and it contributes to raising the programming abstraction levels by offering higher-level interfaces and APIs build on top of the standardised lower protocol layers.

Starting before 2010, there were embedded wireless platforms capable of running the standard IP, with extensive optimisations, which was an important step towards the Internet of Things that we see today [18]. A typical example of an early platform is the widely used Tmote Sky device. It has an 8 MHz Texas Instruments MSP430 microcontroller, 10 kB RAM, and 48 kB flash memory [40]. This setup is sufficient for supporting optimised IP-based communication, and in addition, it has been used in numerous lab experiments with different security solutions. But the memory and the computational capacity needed to support standard-based security solutions, which meet today's stringent requirements and still have room for an actual application, are not there.

Looking at present-day IoT platforms, there now are 32-bit ARM Cortex-M4 based CPUs, such as the nRF52840, with 256 kB RAM and 1 MB flash [41]. This has expanded the possibilities for adapting advanced security solutions while still being practically usable. These resource-constrained devices with from 50 kB of RAM, which should be prepared to operate on limited battery power for an extended duration of time, belong to the so-called class 2 type of devices, as defined in "Terminology for Constrained-Node Networks" by the Internet Engineering Task Force (IETF) [11].

Within the broad scope of IoT, two disparate trends can be seen. One is the development of really constrained devices, for example, using passive radios that transmit through passive reflection and modulation of incoming RF signals (so-called backscatter) or supported by energy harvesting mechanisms [57]. These devices will always need to rely on more powerful devices to handle important security aspects.

The other trend is the development of more capable IoT devices, where it is possible to perform advanced crypto operations while maintaining low power consumption. The work presented here focuses on these, compared with backscatter devices, powerful but still constrained devices. There are still several constraints regarding computational resources, available energy, and communication capacity to adhere to. General constraints can result from an overall desire to keep costs low, energy budgets will need to be kept while operating on batteries, and bandwidth limitations will remain, especially when operating in harsh environments with radio noise. An important observation is that while the trend towards stronger computational capabilities is likely to continue, the mentioned motives for keeping resource usage at a minimum will still be valid.

To go beyond custom-made proprietary security solutions and avoid creating completely isolated IoT subnetworks, IoT security systems should benefit from the solutions already developed, tested, and used on the rest of the Internet. One of the core security services is authentication, the act of verifying the identity of someone you want to interact with. Thanks to developments in asymmetric cryptography already during the 1970s, it has been possible to create a system with public-private key pairs, where the public keys can be freely distributed. By knowing the public key of a user or device, another user can validate whether a signature was made by the corresponding private key. Through encapsulating public keys in digital certificates, data structures that are signed by a trusted party, it becomes possible for two parties to directly and securely authenticate each other without an external third party, as long as the certificate signatures can be traced back to a shared trusted root. The complete system needed to manage the authentication services and their artefacts, the certificates, keys, policies, and roles, forms a PKI.

This system, with public keys encapsulated in signed certificates, has evolved to form the basis of almost all Internet communication. For human Internet users, these details are typically hidden. The web browsers come with a set

of installed trusted root certificates. The root certificates allow the browser to authenticate any server with a certificate issued and signed by a Certificate Authority (CA), as long as the signature of the CA can be traced back to one of the already trusted roots.

Existing PKI solutions have been too complex and resource-consuming for IoT devices to handle. Given the large potential advantages of including IoT devices in PKIs, PKI for IoT has been an interesting area of research. As can be seen reflected in the title of the thesis, "Public Key Infrastructure and its applications for resource-constrained IoT", a substantial part of the research work has been focused on enabling PKI functionality for IoT devices belonging to the above-mentioned class 2 type of constrained devices. PKI mechanisms are needed for most of the IoT lifecycle stages, from the initial deployment, over normal operations, to the final decommissioning and revocation of capabilities. The overall theme of the research conducted as part of this dissertation has been to bridge some of the identified gaps in IoT security, with the ultimate goal *to bring IoT devices closer to being first-class Internet citizens*.

The emphasis has been on protocols, procedures, mechanisms, and efficient encodings rather than specific cryptographic solutions. Examples include investigating and proposing solutions that utilise Constrained Application Protocol (CoAP) over Datagram Transport Layer Security (DTLS) rather than HTTP over Transport Layer Security (TLS) and make use of the shifts towards security for layers higher up in the protocol stack, which enables more complete end-to-end security and allows more fine-grained access control. In parallel with the PKI development, other secure services that rely on lightweight key management have been investigated. An overview of the thesis contributions in their context is shown in figure 1.1.

By adhering to the latest and upcoming IoT standards and proposing new ones, the developed solutions are ensured to be compatible with what is considered by academia and industry through the standardisation bodies to be relevant best cryptographic practices for the future.

The overall goal can be seen through the decomposition into the research challenges presented below.

1.1 Research Challenges

Research Challenge 1, RC1: Current PKI enrollment solutions developed for the regular Internet are infeasible for IoT because of the resource constraints present in IoT. The challenge is *providing sufficiently lightweight IoT alternatives which also fit with existing echo systems*. In practice, this means to analyse the needs of IoT in relation to PKI services, to find the critical functional subsets needed to provide meaningful PKI mechanisms, and to provide the desired services while maintaining the high security guarantees expected from modern PKIs.

Research Challenge 2, RC2: With increasing demands for absolute end-to-end security, older security solutions have become insufficient since previous transport layer protocols do not provide protection for individual applications or proxy traversal. There have been proposals for application layer communication protocols suitable for IoT (EDHOC and OSCORE [52, 53]), which need to be complemented with secure key distribution protocols to be independently usable for PKI. The challenge is to *adapt and make efficient use of new security building blocks to provide the desired end-to-end enrollment functionality in a resource-efficient manner*.

Research Challenge 3, RC3: Digital certificates come with a prescribed valid lifetime, but computers and devices can be compromised at any time. A revocation service that allows checking the validity of certificates is an expected PKI service and is inevitable for any security-critical scenario. Existing solutions such as Certificate Revocation List (CRL) usage and the Online Certificate Status Protocol (OCSP) have not been developed for recently standardised IoT formats and protocols. The challenge is *to move beyond existing solutions for certificate revocation, which are infeasible for IoT because of overhead both in terms of memory/storage and communication*.

Research Challenge 4, RC4: For IoT systems to be economically valuable for all of their expected lifetimes, the long-time maintainability of the IoT deployments needs to be addressed. This includes mechanisms for secure change of control and ownership between different system owners or operators. To support interoperability and minimise the adaption needs of service providers the solution should be based on PKI mechanisms as far as possible. Because of the scale, the solution needs to operate with minimal human intervention. The challenge is *designing interoperable mechanisms for performing the transfer of IoT control in a highly automated yet secure manner*.

Research Challenge 5, RC5: For IoT systems, where storing data securely inside the device is necessary, a separate set of potentially heavy-weight crypto mechanisms is needed. For IoT devices, having two separate crypto mechanisms for storage and communication causes unacceptable overhead. In addition, the crypto operations needed when data at rest need to be decrypted and encrypted again before being sent to the network incurs undesired delays. Given lightweight solutions for communication security, it ought to be possible to do better. The challenge is *providing a solution that incurs minimal overhead for IoT devices, keeps well-tested security properties intact, and does not compromise standard compliance and interoperability when the stored data is sent*.

Research Challenge 6, RC6: Another mechanism that is needed for the long-term maintenance of deployed IoT systems is functionality for secure

updates. The update mechanism itself needs to be secure, or else it could easily be turned into an attack vector. Hence the solution cannot rely on authentication alone. The challenge is *solving the secure update authorisation problem for IoT in a resource-efficient, scalable and interoperable manner*.

1.2 Methodology

The research conducted as part of the dissertation has mainly been experimental. The three recurring steps on a general level are: formulating a research question based on an exploration of the research challenge, the design of a potential solution, and experimental validation of the stated solution. Applied to the IoT security domain, this corresponds to: The current gaps in, and shortcomings of, existing IoT security solutions have been investigated, including the exploration of possible partial solutions which could serve as input. After the investigation stage, a more concrete problem has been formulated as the basis for new solutions. New protocols and mechanisms to fit the identified gaps have been designed. The designs have been either prototyped or more completely implemented to the point that they can be evaluated based on the identified security and resource usage requirements and, whenever possible, compared with the state-of-art, using relevant IoT hardware.

For the concrete hardware experiments, mainly two representative IoT platforms have been used. For Paper I, Paper III and Paper V, Arm Cortex M3-based devices, Zolertia Fireflies, have been used. The platform provides a 32 MHz MCU, 32 kB RAM and 512 kB of Flash. For Paper II and Paper IV, the newer nRF52840-DK platform was used. In general, the experiments test a subset of the full functionality expected of a complete IoT deployment, where the security solutions form a part. Therefore it has been important to find relevant target platforms where the evaluations show the possibilities to also run applications in parallel with the tested security mechanisms. To provide the server side in experiments where IoT devices interact with more powerful Internet devices, both Raspberry Pi 3 devices and regular desktop computers have been used, either filling the role of an edge device or a regular Internet server.

1.3 Contributions

The recurring theme has been to improve the state-of-art for IoT security. Below is the description of how the work has addressed the research challenges presented above.

Lightweight Certificate Enrollment for IoT

In Paper I we designed and implemented an integrated lightweight certificate enrollment solution for IoT devices, which supports fully automated protec-

tion of enrollment sessions. In addition, we provided IoT certificate profiling with domain specific certificate compression mechanisms and lightweight encoding of X.509 certificates. Together the improvement brings the total communication costs of session establishment and enrollment operations down to levels that are feasible for constrained IoT devices. The proposed solutions were tested and evaluated using relevant target IoT hardware.

In Paper II a new design was made to enable enrollment for devices using the new and latest proposed application-layer security communication solutions, for usage with OSCORE and EDHOC. We designed, implemented, and evaluated LICE (LIghtweight Certificate Enrollment for IoT using application layer security). We advanced the usage of efficient protocol data encodings by proposing optimised CBOR encodings for the EST enrollment operations. We demonstrated that a complete key establishment and certificate enrollment data exchange can be brought down to 800 bytes of data. This is less than a third of the data being transferred compared with existing EST-coaps solutions. By utilising application layer security solutions, the enrollment protocol follows upcoming standards and can traverse proxies, achieving real end-to-end security. In addition, the functionality is required to avoid already constrained devices needing to support DTLS and OSCORE in parallel.

Through these contributions we have addressed RC1 and RC2.

Lightweight Certificate Revocation

After a detailed analysis of the existing OCSP protocol specification, we proposed and designed TinyOCSP in Paper III as a response to the identified RC3. TinyOCSP leverages recent IoT standards and is a lightweight revocation alternative to the original OCSP. We showed how significant performance improvements can be made without sacrificing functionality or compromising security.

We implemented the protocol for relevant state-of-the-art IoT hardware with low-power radio communication, in parallel with the original OCSP, and evaluated the performance benefits in terms of total energy usage and communication overhead. In addition, we proposed CRL compression mechanisms using Bloom filters. We analysed and calculated the conditions where Bloom filter compression is beneficial, verified the theoretical results through simulations and present the optimal parameters.

Low-latency Secure Storage and Communication

Through the design of BLEND in Paper V we address RC5. We showed that standard-based application layer security mechanisms can be extended to enable a solution for both secure storage and secure communication. The design is able to reduce the memory required by preventing the need for separate crypto management for storage and communication, and to reduce the latency when sending sensor data to the network. The system has been implemented and evaluated to show its suitability for IoT.

Automated Trust Transfer
To address the challenges of long-time maintainability of IoT systems (RC4) we designed AutoPKI, a lightweight schema for trust transfer in Paper IV, which allows control of IoT deployments to shift between service providers in a highly automated manner. By using the proposed application layer security enrollment solutions, and signed token concepts from the Authentication and Authorization for Constrained Environments (ACE) framework, both the added overhead for IoT devices and the needed modifications for servers already using PKI for IoT solutions are kept minimal.

The limited overhead of the proposed solution was verified through a feasibility study using a prototype implementation for constrained IoT devices. A security analysis was done to demonstrate that the schema meets the stated security requirements.

Automated Secure IoT Software Updates
To address RC6 we have proposed AC-SIF, a firmware manifest design and update architecture in Paper VI. Based on the ACE framework and the recommendations for the upcoming Software Updates for Internet of Things (SUIT) standards, our solution can fulfil the need for both authentication and authorisation functionality. Through the use of CBOR Web Tokens for Proof-of-Possession, we are able to decouple the IoT devices from having to keep track of a potentially large number of update providers, contributing to automated secure updates for IoT.

Through the dissertation work done and presented here, several new building blocks needed to create, support, and maintain secure PKIs which can span across constrained IoT devices and full-scale Internet infrastructure have been proposed. Important steps towards making IoT devices first-class Internet citizens have been taken.

While the challenges we have addressed always are meant to improve the state-of-art from an academic standpoint, we have also strived to include requirements relevant to the industry. For the core PKI contributions presented, the solutions have been validated with our industry partners to be in line with the expected upcoming needs of certificate authorities. In addition, we have been working together with industry partners to share our work in the shape of new standards, for it to have a better potential for industrial impact.

1.4 Dissertation Structure

The two parts of the dissertation are disposed within the following structure: Part I contains an extensive summary of the dissertation work, subdivided into six chapters. In Chapter 2, I present and discuss background information about the basis of this dissertation. In Chapter 3, the six publications constituting

the core of my work are summarised. An overview of related work is given in Chapter 4. In Chapter 5, I present some concluding remarks and an outlook on future challenges and possible continuations of the work. Finally Chapter 6 contains a high-level summary in Swedish. Part II contains a reprint of the six papers included in the dissertation.

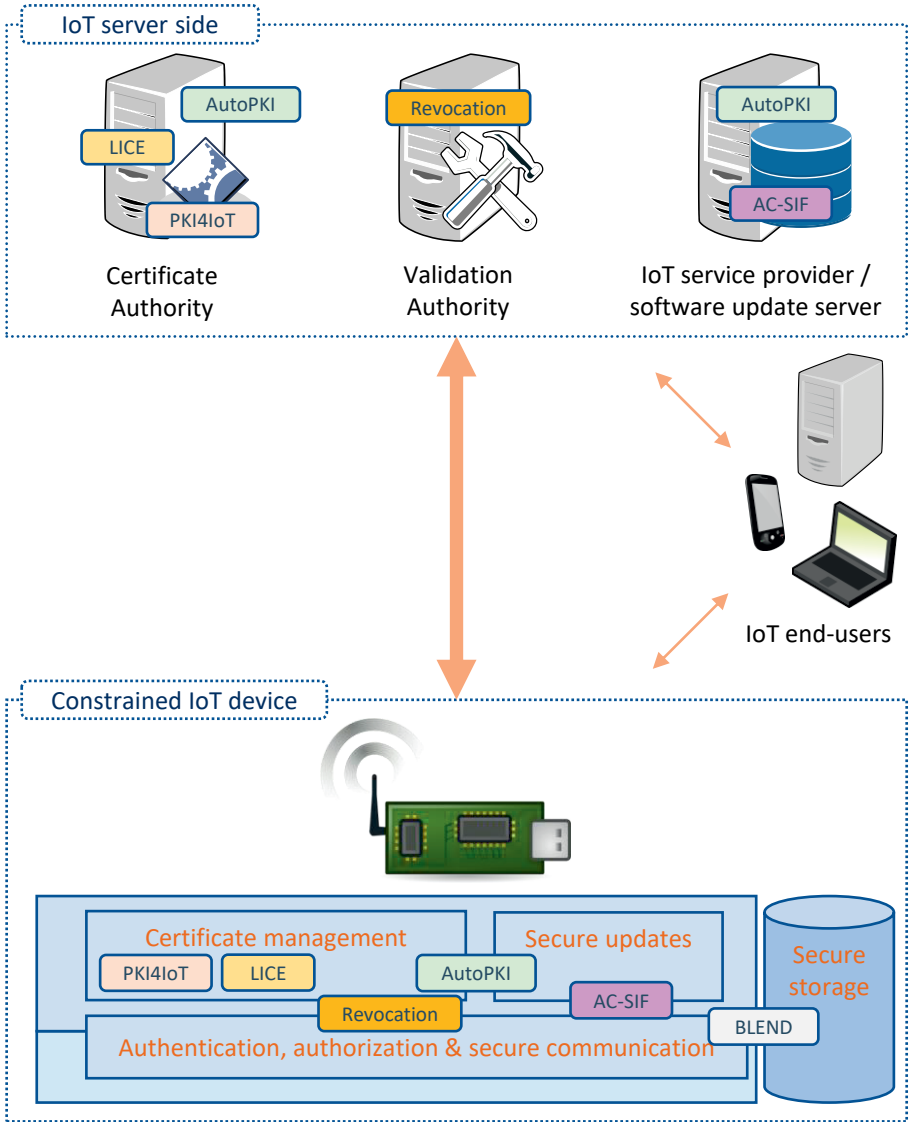


Figure 1.1. Public Key Infrastructure and its applications for resource-constrained IoT. The diagram presents a high-level view of the dissertation contributions in their IoT and server context. The interactions between servers are not shown; for instance, the compound protocols AutoPKI and AC-SIF involve more than one server type in their server-side operations.

2. Background

This chapter provides background on the security mechanisms and some of the most important standards and their underlying technologies which are required for achieving the results in this dissertation.

2.1 Feasibility of Security Protocols

An essential basis for modern PKI solutions is the existence of secure and authenticated key exchanges. An influential work in this area is the work around the so-called SIGMA family of protocols, described in Krawczyk's paper from 2003 [34].

The starting point for the SIGMA work is the following three requirements: First, to provide perfect forward secrecy through a Diffie-Hellman-based secure key-exchange protocol. Second, to provide public-key authentication through the use of digital signatures. Finally, to optionally provide identity protection for the protocol participants. One important observation is that the identity protection functionality can be in conflict with the authentication of protocol participants. To authenticate yourself you need to some degree to reveal your identity, which you might be unwilling to do unless you already trust your counterpart. As a consequence, different protocols might do different trade-offs, for instance, which party is first to reveal its identity, affecting the number of protocol messages needed to achieve the authentication phase.

Perfect forward secrecy is the condition that old secure session keys established between two parties, and later erased, should be protected and impossible to regenerate by an attacker, even if any long-term secret belonging to the parties has been revealed. The SIGMA work gives theoretical underpinnings to the security guarantees of protocols used for PKI, from the older Internet Key Exchange (IKE) [30] to the recent EDHOC [53] key-establishment protocol.

When constructing compound security services, these fundamental security services are assumed to act as reliable building blocks such that the security analysis can focus on if and how the composition might introduce new weaknesses. At the same time, it should highlight the need to form contingency plans in the event that any weakness is found in the fundamentals of a cipher suit or a particular implementation. In those cases, the affected software components must be immediately replaced. Alternatively, if the security breach has already happened, the trust in the affected entities should be

revoked, shielding them from other parts of the system. The threat of discovering new vulnerabilities highlights the need for both secure update solutions and revocation services feasible for IoT.

2.2 PKI Concepts

This section expands on constituent parts and services which are essential for the creation and operation of a PKI. A starting observation is that an overarching goal of a PKI is to enable actors to establish trust between each other, through the PKI security services.

Security Services and PKI

In order to establish and maintain trust from a system perspective, two essential security services are required: authorisation and authentication. Authorisation ensures that actors are only able to perform actions for which they are authorised. To establish an authorisation service, a secure authentication service is a prerequisite. This is because the authorisation of actions requires the authentication of actors. The primary function of an authentication service is to establish a trustworthy binding between an entity and a public key.

Establishment of Trust in PKIs

Public Key Infrastructures utilise authentication based on publicly accessible keys encapsulated within digital certificates. These certificates are issued by a CA, which acts as a trusted third party that verifies the identity of the entity requesting the certificate. The CA, in turn, is identified by its own certificate, which can either be self-signed or signed by another CA, creating a hierarchical structure with a self-signed root CA at the apex. This system enables the verification of certificate chains up to the top nodes, which are assumed to be trusted [25].

To be able to establish trust in these certificate chains, the authenticating party must have access to the self-signed root node certificates. For IoT devices, this necessitates the inclusion of the necessary root certificates in a trust store through either factory pre-programming or explicit secure and trusted enrollment operations. In standard computers interacting with the Internet through web browsers, these PKI operations are contained and hidden through lists of trusted root certificates kept and maintained by the web browsers, acting as PKI proxies between clients and servers.

The communication needs of the IoT device, and the complexity of the CA hierarchies will determine how many root certificates must be provided to the device trust store. For performance reasons, additional certificates can be added to later enable authentication through certificate references, for protocols where this is possible, in which case it is sufficient for the communicating parties to only send hashes of certificates.

X.509 Certificates

The standard format for how to encapsulate the public keys needed for conventional PKIs is X.509 [14]. The latest version, v3, is the most flexible, allowing extensions that can be used to specify the desired capabilities or constraints that should be associated with the certificate, such as whether it belongs to a CA, or which key usage operations should be permitted. The expected X.509 encoding format is Abstract Syntax Notation One (ASN.1), together with object identifiers (OIDs) to label and identify the many fields needed. ASN.1 is an old standard that is defined to be both human and machine-readable [58] As a result, the resulting encoding is not compact, but rather lengthy, where the headers of nested fields grow rapidly in size.

2.3 Communication Protocols for Lossy Wireless Networks

The work presented in this thesis is primarily concerned with protocols and mechanisms which operate on the transport layer or above in the network protocol stack. Yet many of the constraints which affect wireless IoT devices are due to limitations stemming from the physical radio medium, communicating over a Wireless Sensor Network (WSN), and the resulting impact on the protocol stack.

UDP and Lower Network Stack Layers

Below the application layer the User Datagram Protocol (UDP) is the predominant protocol to be used in IoT for the transport layer. It has lower resource usage compared with the Transmission Control Protocol (TCP), and by leaving out the functionality to handle data losses, the higher layers can choose when resending is really needed, making UDP better suited for constrained lossy networks. For the network layer, especially for IoT devices that should be globally available, IPv6 is the alternative that thanks to its large address space avoids address scarcity, and can operate without the need for NAT solutions [15].

6LoWPAN is used for wireless IoT devices communicating over 802.15.4 radio networks, where the maximum radio packet size is only 127 bytes. The protocol handles fragmenting and reassembly of IP packets sent over lossy radio links [26]. Since 6LoWPAN does not support already fragmented IP packets, higher layers need to ensure that data fits within a single datagram.

Constrained Application Protocol (CoAP)

The lightweight alternative to HTTP for IoT is the Constrained Application Protocol (CoAP) [13]. It is designed for constrained devices, and since it is targeting machine-to-machine communication, it does not need to include

human-readable tags. It has been augmented by block-wise transfers, which allows CoAP implementations to handle sending and receiving larger chunks of data without relying on IP fragmentation.

2.4 Security Standards and IoT

The security standards described below are either existing solutions, improved solutions for IoT, or new standards primarily designed for IoT. This thesis relies on the existence of these protocols for building the PKI for IoT, for secure software updates for IoT, and for secure storage and communication of IoT data.

2.4.1 PKI Protocols

Enrollment over Secure Transport (EST)
The EST protocol is a commonly used standard for certificate management. It specifies the use of Certificate Management System (CMS) messages over HTTPS for secure certificate enrollment [45]. The minimum functionality required by EST is the ability to securely transfer the following protocol data between a client and a certificate authority: CA certificates, which are used to establish trust between the client and other CAs. These certificates are typically stored in a client-side trust anchor database, commonly referred to as a trust store. The transfer of CA certificates is crucial to enable secure communication between the client and server with certificates signed by other CAs. Certificate enrollment requests from clients to the CA, and the resulting enrolled certificate back to the client.

Online Certificate Status Protocol (OCSP)
OCSP is an online alternative to using certificate revocation lists, which allows clients to check the status of a certificate of a server before deciding to establish a secure session with it [50]. A client, in OCSP terminology called Relying Party (RP), sends a request to a Validation Authority (VA) which replies with signed certificate status information. The messages are ASN.1-encoded and transported over HTTP. If the protocol is used without an optional cryptographic nonce bound to a specific request, an intercepted old and no longer valid reply could be re-sent by an attacker, thus creating a replay attack. With a nonce included in the exchange, the attack can be prevented.

2.4.2 Transport and Application Layer Security Solutions

Datagram Transport Layer Security (DTLS)
DTLS is a well-established alternative to TLS for scenarios when it is advantageous to run the secure session on top of UDP instead of TCP. The DTLS

specification encompasses the functionality both to perform key establishment using either Pre-shared Key (PSK) or certificate-based asymmetric cryptography, and to encrypt/decrypt messages using symmetric cryptography in the established secure session [48]. The certificate-based version is used to secure the communication in Paper I, with a special focus on the profile of DTLS specifically addressing the needs of IoT [56]. Since the protocol operates on the transport layer, data from the CoAP layer is encrypted in full, making it impossible for a proxy to handle the message without being able to fully decrypt it. The data decryption will reveal also the application layer data and break end-to-end security, despite the proxy only needing access to certain CoAP parameters.

OSCORE and EDHOC

Object Security for Constrained RESTful Environments (OSCORE) is a recently standardised application-layer protocol for the protection of CoAP messages [52]. It is designed to be lightweight and suitable for IoT. Since the protocol works with data on the application layer it enables secure data communication between CoAP devices over proxies also with HTTP endpoints, through CoAP-to-HTTP mapping operations. The protocol needs to be complemented with functionality for key establishment, which is not part of the OSCORE specification.

Ephemeral Diffie-Hellman Over COSE (EDHOC) is a proposed key exchange protocol, at the time of writing in the last stages of review before being accepted as an RFC. A design goal is to be useful for constrained scenarios [53]. The data exchange can be extremely compact, needing only three messages to perform mutual authentication and establish shared key material in its minimal form. A clear use case is to establish key material for an OSCORE security session, but the low overhead can make it suitable for other mutual authentication scenarios.

These application layer protocols are used as building blocks in Paper II, Paper IV and Paper V.

2.4.3 Efficient Data Encodings and the ACE Framework

Concise Binary Object Representation (CBOR)

CBOR is a data serialisation format designed with the aim to provide a compact representation of data while minimising resource overhead during the encoding and decoding process [12]. It supports a variety of basic data types, as well as maps and arrays. Additionally, it allows for the representation of binary byte strings, enabling its use as a wrapper for various forms of binary data. CBOR is meant to be self-describing, in which case no pre-defined schema is needed during the decoding process.

Authentication and Authorization for Constrained Environments (ACE)
ACE is a framework for IoT, designed to provide authorisation for devices wanting to access resources from servers. It uses new compact schemes and protocols for data encoding and crypto operations, CBOR, COSE, OAuth 2.0 and CWTs [51]. A client which wishes to access protected resources requests an access token from an Authorization Server (AS), which, if the request is authorised, encodes the granted permissions in a CBOR Web Token (CWT). A token is a serialised data object, used to encode *claims* about a *subject*, which are guaranteed by an *issuer*. CWTs use the CBOR Object Signing and Encryption (COSE) format specification for encryption, signatures and to augment data with a Message Authentication Code (MAC). The authorisation server binds the token to secret key material known by the client. The token can then be presented to the Resource Server (RS) by the requesting device [29]. The framework is included in the solution proposed in Paper VI.

3. Summary of Papers

3.1 Paper I

Joel Höglund, Samuel Lindemer, Martin Furuhed, and Shahid Raza. "PKI4IoT: Towards public key infrastructure for the Internet of Things". In: *Computers & Security* 89 (2020). DOI: https://doi.org/10.1016/j.cose.2019.101658

Summary

In this paper, we propose the combination of DTLS with an early version of the compact CBOR encoded certificates, together with tests of the new lightweight enrollment protocol, EST-coaps. The non-destructive compact certificate encoding allows existing CAs to issue standard X.509 for IoT devices. The certificates are converted to and from the compact CBOR format by a radio gateway. The gateway acts as an edge device on constrained networks, in an integrity-preserving way, which does not require the device performing the certificate conversion to be trusted. We implemented these protocols within the Contiki embedded operating system and evaluated their performance on an Arm Cortex-M3 platform. The results showed that it is possible to introduce the compact certificate encoding in a way transparent to the server side. The IoT device can decrease the amount of data needed to be sent during the costly handshake operation, which is a large benefit in lossy radio networks.

Reflections

This work marks an important step toward bringing Internet-grade security to IoT devices, by creating the conditions for moving from pre-shared keys to digital certificates, and demonstrating that it is practically feasible.

This work gave me valuable insights into the dependency of security solutions on the lower layers of the stack, the challenges with lossy radio networks, and the value in trimming down every possible byte needed to be sent over the radio. If I had redone the presentations in the paper today, I would have presented the timing graph with a focus on the median rather than the mean. The mean was clearly affected by occasional packet losses and obscured the differences between the test cases. This we improved upon for the following Paper II.

My Contribution

I am the main author of the paper. I started the work with an existing prototype EST implementation, which was conceptualised and formulated by Shahid Raza. I designed and implemented the combined system where lightweight certificate handling was integrated and tested and evaluated the entire system. All the experiments were performed by me, and I wrote the first version of the manuscript all the paper authors took part in refining.

3.2 Paper II

Joel Höglund and Shahid Raza. "LICE: Lightweight certificate enrollment for IoT using application layer security". In: *IEEE Conference on Communications and Network Security, CNS 2021, Tempe, AZ, USA, October 4-6, 2021*. IEEE, 2021. DOI: 10.1109/CNS53000.2021.9705036

Summary

In this paper, we designed and evaluated a new compact enrollment protocol utilising the recent advancements in application layer security. Through new and proposed standards, it is possible for IoT devices to implement more lightweight solutions for application layer security which has the potential to enable real end-to-end security, capable also of traversing proxies. We presented LICE, which fills an important missing niche before certificate-based security can be used with new IoT standards such as OSCORE and EDHOC. We implemented the protocol on IoT hardware and evaluated it using low-power radio devices in representative noisy network environments. Compared with the previous state-of-art using EST-coaps over DTLS, when using LICE the complete combined secure session establishment and enrollment operations can be completed using less than 800 bytes, less than one-third of the data used for EST-coaps.

Reflections

The proposed EDHOC key establishment protocol has at the time of writing still not been completely finalised as a standard, and the latest draft version contains some differences regarding how auxiliary data is transported in the handshake messages. Details of our implementation would need to be updated, but the total differences in cryptographic operations performed are minor and unlikely to change the performance more than marginally.

We have used the crypto algorithms recommended by the standards to be supported for the target IoT platforms. It deserves to point out that while these recommendations can be upgraded to point to stronger crypto suits with longer key lengths, the same protocol can still be utilised.

My Contribution

I am the main author of the paper. I am the main responsible for the design, and I have done the implementation and performed the experiments. I wrote the first version of the manuscript. The work was done under the active supervision of Shahid Raza.

3.3 Paper III

Joel Höglund, Martin Furuhed, and Shahid Raza. "Lightweight certificate revocation for low-power IoT with end-to-end security". In: *Journal of Information Security and Applications* 73 (2023). DOI: https://doi.org/10.1016/j.jisa.2023.103424

Summary

In this paper, we designed and evaluated a lightweight alternative to the state-of-art concerning certificate revocation status checking. This is a required functionality in a PKI for IoT, to allow also constrained devices to ensure that a device with a given certificate is still to be trusted. We implemented the Online Certificate Status Protocol (OCSP) on constrained IoT hardware, as it is the most used standard for certificate validation on the Internet. This showed that the protocol has too high of a resource overhead for the target IoT environments. As a solution, we developed and implemented TinyOCSP, a lightweight alternative that utilises compact IoT protocols such as CoAP and CBOR. Through our experiments, we have demonstrated that TinyOCSP required 41% less energy for validating eight certificates compared to OCSP on an ARM Cortex-M3 SoC. Additionally, we observed that the transactions in bytes needed by TinyOCSP were at least 73% smaller than those needed by OCSP.

Reflections

This piece of work covers an important aspect of the IoT life cycle: the ability to verify the trust in external servers at a shorter time perspective than the valid lifetime of a certificate, which can be very long. Different IoT scenarios will have different requirements in terms of how frequent trust verifications need to be done. In some cases, the server endpoints will be considered always trusted, but in others, the IoT device can be required to always get an up-to-date validation of a communication endpoint for each new communication session. Since also the relatively lightweight revocation status check adds further communication and processing, the trade-off between strict security and resource usage needs to be done for the target scenario.

My Contribution

I am the main author of the paper. Shahid Raza conceptualised this paper and I together with Shahid Raza supervised a master thesis and we together formulated the design of this paper. The thesis student implemented the design. I have also enhanced the initial design and worked on the security validation and analysis.

3.4 Paper IV

Joel Höglund, Simon Bouget, Martin Furuhed, Göran Selander, John Mattsson, and Shahid Raza. *AutoPKI: Public Key Infrastructure for IoT with Automated Trust Transfer*. Submitted. 2023

Summary

In this paper, we identified the increasing need to provide well-defined mechanisms for securely shifting the control of IoT deployments from one service provider to another. This is required in order to prevent vendor lock-in and guarantee long-time support and maintainability, as IoT deployments grow in both size and number. We proposed AutoPKI, a lightweight mechanism for securely transferring control between two IoT service providers or operators. The solution utilises CBOR Web Tokens to encapsulate the agreements between service providers, which must be met during the operation, together with existing lightweight certificate enrollment. We show that the overhead for the involved IoT devices is small and that the number of manual steps is minimised. We analyse the fulfilment of the security requirements, and for a subset of them, we demonstrate that the desired security properties hold through formal verification in Tamarin.

Reflections

This is a paper that puts the enrollment work in a larger context of new standards and protocols and business and policy decisions which still require manual input. Hence the work serves to illustrate both the advantages of the newly proposed PKI mechanisms and the several remaining sectors of IoT security, which would benefit from continued standardisation work to further reduce the need for human intervention in IoT security operations.

My Contribution

I am the main author of the paper. I have conceptualised and formulated the initial design. The formal analysis was made by Simon Bouget, which led

to refinements of the design. I have done the prototype implementation to validate the design and measure the overhead. I wrote the first version of the manuscript, with formal modelling input from Bouget. The paper was finalised together with the other authors. Shahid Raza has supervised this work.

3.5 Paper V

Joel Höglund and Shahid Raza. "BLEND: Efficient and blended IoT data storage and communication with application layer security". In: *2022 IEEE International Conference on Cyber Security and Resilience (CSR)*. 2022, pp. 253–260. DOI: 10.1109/CSR54599.2022.9850290

Summary

In this paper, we presented BLEND, a mechanism for when IoT devices need to store sensor data securely, but without adding extra overhead for when data needs to be sent onto the network. The combination of secure storage and communication security is achieved by storing data as pre-computed encrypted network packets. Compared with local secure storage methods, the need for separate cryptographic operations for secure storage and communication is eliminated, and the communication latency is significantly reduced. The evaluation demonstrated that BLEND is capable of reducing the latency to get securely stored data ready for sending from 630 microseconds down to 110 microseconds per packet.

One of the key advantages of BLEND is that it does not require modifications to the secure communication standard reused also for secure storage, and it can therefore preserve the underlying protocols' security guarantees. This is an important consideration when deploying secure communication systems for critical infrastructure.

Reflections

The results in this paper illustrate that actual improvements for a given performance metric are tightly tied to the specific hardware used. At the same time, I see it as an illustration of the rich set of compound services that can be designed and realised when reliable PKI for IoT solutions are available, further contributing to the IoT security infrastructure.

My Contribution

I am the main author of the paper. I have done the design, with inspiration from previous work done by Shahid Raza and others, implemented our design

and performed the experiments. I wrote the first version of the manuscript. The final manuscript version was written together with Shahid Raza.

3.6 Paper VI

Joel Höglund, Anum Khurshid, and Shahid Raza. "AC-SIF: ACE Access Control for Standardized Secure IoT Firmware Updates". In: *International Conference on Emerging Security Information, Systems and Technologies*. 2022

Summary

In this paper, we identified the lack of standard-based remote secure update solutions for IoT, something critical for the long time maintainability of IoT deployments, and to prevent vendor lock-ins. We provided an update architecture design, which is capable of achieving end-to-end security between the IoT devices and the update authors, without requiring a per-author-based trust anchor provisioning by the manufacturer. The proposed solution follows the existing recommendations from the new SUIT standard group and adds functionality for the authorisation of update authors through the usage of CBOR Web Tokens.

Reflections

Some of the SUIT related standard proposals are still being updated, which means there is potential to further address shortcomings regarding complexity and insufficient authorisation mechanisms we identified in the earlier versions. Our work contributes a security building block that highlights the interplay between PKI and other security mechanisms. The secure update architecture requires secure key management, and in turn, improves the security and maintainability of the existing systems.

My Contribution

I am the main author of this paper. This work is built upon a research problem conceptualised and formulated by Shahid Raza. I, with close interaction with security researchers at RISE have developed the design with significant input from Anum Khurshid.

4. Related Work

This chapter contains an overview of the related work to put the results of the dissertation in context within the state-of-art in the included research fields and relevant target industry developments.

4.1 Certificate Based Authentication

The idea of utilising certificates also for IoT security is not new; the advantages and problems with authentication based on certificates have been discussed for more than a decade, as illustrated by several surveys [1, 28, 55]. Industrial actors have pointed out the benefits of PKI solutions for IoT, but mainly proposing the usage of standard Internet protocols for less constrained devices [16, 31]. The immaturity of existing standards and the challenges related to limited resources in the IoT and PKI were specifically identified in an IoT security overview by Khan and Salah from 2018 [33].

The utilisation of automated and efficient certificate management has been identified as crucial in several fields, where e-health and the automotive industry deserve to be mentioned specially. These industries place a significant emphasis on privacy and safety, as well as compliance with strict standards for legal and safety purposes [2, 17, 20]. The requirements and conditions (including real-time constraints and specialised hardware) of the automotive industry make the results obtained there not directly applicable to the broader Internet of Things domain. However, the use cases do emphasise the importance of standard-based interoperability and optimised resource usage.

The conclusion of the existing state-of-art work is that there long has been an active interest in bringing stronger authentication solutions and ideally full-fledged PKI capabilities to a wider range of also more constrained IoT devices. Yet the lack of suitable standards regarding mechanisms for lightweight key management has been an unsolved issue, hindering the developments in the area. Our contributions in Paper I and Paper II serve to close this gap.

X.509 Alternatives

There have been alternative proposals to X.509 for certificate formats. One proposal is implicit certificates, with a potentially smaller memory footprint, where the public key is not explicitly contained in the certificate, but can be reconstructed from the certificate using the public key of the CA. Experiments using implicit certificates for IoT devices have been demonstrated by

Park [43]. In addition, the IEEE 1609.2 standard defines implicit certificate usage for Wireless Access in Vehicular Environments [27]. The uptake and applicability for a wider range of IoT scenarios seem to have been limited, illustrating the difficulty to gain widespread usage for solutions that are not in line with the other infrastructure standards. In comparison, our proposed efficient certificate encoding schema can offer substantial memory savings while being well-aligned with existing standards (see Paper I and Paper IV).

4.2 Revocation, and Efficient Techniques for Revocation List Encoding

How to efficiently share the status of a device through propagating the revocation status of the associated certificate has been extensively studied. OCSP has been established as the main standard for systems where a new certificate validity update is produced for each new request. Other proposed revocation systems share the revocation data through lists or other data structures, which carry varying overheads.

Revocation Lists and Techniques for Compact Revocation Status Encoding

CRL is an old technique still used where the associated overhead is acceptable. In addition to X.509 CRLs and delta-CRLs, Google maintains a curated list of revoked certificates for Chrome, called CRLSet. CRLSet has been shown by a private company investigation to detect less than 2% of the revoked certificates on the Web [21]. The underlying assumption to make this acceptable is that the majority of websites with revoked certificates are very unlikely to ever be visited by regular web users. Since CRLs carry a large overhead, proposals for more compact encodings have been proposed. CRL compression through Bloom filters has been proposed by various researchers for Vehicular Ad hoc Networks [47], advanced metering infrastructures [46] and Web browsers [35]. These research papers have not been formalised in any standardised protocol, but they demonstrate a potential for high compression ratios using Bloom filters. In Paper III we present details for which scenarios where using Bloom filters for CRL compression is beneficial.

Distributed Revocation State

An alternative to the explicit methods is to share certificate status in a distributed fashion across networks of devices, as proposed by Li et al. and Wright et al. [38, 59]. These solutions break end-to-end security between the device wanting to validate a certificate (the relying party) and the validation authority, thus not meeting the same general requirements. Our proposed solution in Paper III keeps the security properties of OCSP while providing a lightweight format usable also for IoT.

4.3 Ownership Transfer

The area of transference of trust between two service providers, with the goal to enable different IoT service providers to shift responsibilities between each other, is related to the more specific issue of ownership transfer. The field of ownership transfer for IoT has been examined with the goal of privacy protection and through custom non-PKI-based solutions.

Privacy protection is the focus of Khan et al. in [32], where a solution for automatically detecting ownership changes of smart home devices, based on user profiles, is presented. Gunnarsson and Gehrmann focus on ensuring forward and backward security between the former and new owner in [22]. The solution is based on symmetric keys and a trusted third party, avoiding the need for PKI support or standard compliance.

The conclusion is that there is a lack of scalable solutions which utilise interoperable PKI functionality. The absence motivated our work in Paper IV.

4.4 Secure Storage

Compared with the area of secure communication, the field of secure storage has seen much fewer standardisation efforts. Relevant efforts can be found in several related areas. Designs for custom deployments have been proposed based on Blockchain solutions [42, 61]. These are dependent on custom server infrastructure and not suitable for those IoT devices which are less powerful than cell phones or routers.

The field of Trusted Execution Environments (TEEs), such as ARM's TrustZone, is another related area. Secure storage solutions for Android-based devices using TrustZone have been proposed by Li et al. [36]. A use case for TEEs with relevance for IoT is to construct secure key storages, as seen in work by Pinto and Santos and Hein et al., as well as in Android OS documentation [3, 23, 44]. These solutions are generally expensive to implement and maintain per stored byte, and not suitable as a general IoT storage system for constrained devices.

Previous research has presented two solutions for combining secure communication with secure storage, FUSION by Bagci et al. [7] and FDTLS by Boo et al. [10], using IPsec and DTLS. Relevant insights include the importance to optimise memory hardware usage, to minimise costly flash memory operations. A key disadvantage of the proposed solutions is the reliance on PSK, pre-shared keys. The many security weaknesses and single-point-of-failure built into pre-shared password solutions are parts of the problems that modern key management solutions seek to alleviate. The transport layer protocols used have large headers, which, especially in 802.15.4 radio networks, severely limits the space left for IoT data. In addition, the proposals break the protocol specifications for how session keys should be generated with random data.

The identified shortcomings have motivated our work in Paper V where we propose a design taking advantage of application layer security and lightweight key management, that avoids the endangering of security which follows the removal of random data in previous solutions.

4.5 Secure Software Updates

A secure update architecture for IoT needs to provide solutions for how the update software and the associated metadata are specified and securely distributed. To prevent vendor lock-ins and further interoperability, the solution benefits from being standard-based and using established credential management mechanisms.

For systems with few resource constraints, package managers have been a commonly used solution for how to solve the issue of software update distribution. Depending on the target platform, there are systems such as RPM, dpkg, and commercial app stores. To establish trust and verify the validity of updates, solutions similar to those used for web browsers can be used, where the trust anchors required to verify updates with PKI operations, such as code signing, need to be pre-installed in the operating system.

Update architectures can introduce severe security vulnerabilities, and it has been argued by Samuel et al. that a single signature is insufficient for a recipient to trust a new update [49]. To increase the security level, a scheme where at least a predefined number, out of a total pool of trusted signers, have signed an update could be utilised. A version with two trusted signers was proposed by Asokan et al., relying on lists of authorised authors and their trust anchors to be pre-shared with the target devices [6].

A proposal for IoT software updates is presented by Xue et al., where non-constrained controller devices support less powerful IoT devices [60]. For networks with devices that are themselves too constrained to act as fully independent endpoints this is a possible solution which is complementary to the efforts to bring PKI capabilities to the IoT devices that could utilise them.

The conclusion based on existing state-of-art is that there is a need for a standard based more lightweight secure update architecture for IoT, a challenge we address in Paper VI. Our proposed design aligns with proposed SUIT standards [39] and additionally proposes a solution for authorisation of update providers, which removes the need for pre-shared lists of authorised authors.

5. Conclusions and Future Work

In this section, I present some concluding remarks before discussing some future IoT security challenges and possible future directions of work.

5.1 Conclusions

Technological development has been rapid in the area of both IoT and security in the last decade. Yet the combination of the areas, to provide secure IoT solutions, has been less obvious. A limiting factor is the slow process to develop and agree on open standards, a bureaucratic process that often takes many years from the initial proposal to the accepted standard. Quite often large tech companies create their own infrastructure and ecosystems, with their own security solutions. This creates a situation where third parties either are shut out or have to pay fees to contribute to the proprietary infrastructures.

Based on the above perspectives, I believe it is important for academia together with the parts of the industry that are prepared to work actively for interoperability, to go further and show the potential of new security solutions. In this thesis, we have proposed and evaluated several new building blocks, with a common goal to further the security for IoT. This has been demonstrated by providing lightweight mechanisms for certificate enrollment and compact certificate encodings in Paper I and Paper IV. Together with the proposals for lightweight certificate revocation in Paper III, we have made steps towards standard-based key management which is practically available for IoT. By proposing mechanisms for combined secure storage and communication in Paper V we help make more security-sensitive scenarios feasible. Finally, by proposing solutions for secure transfer of trust and secure updates in Paper VI we contribute to the long time maintainability of IoT systems. Together they form steps towards the creation of a scalable security infrastructure, capable of handling heterogeneous IoT devices, bringing IoT devices closer to being full Internet citizens.

5.2 Future Challenges and Future Work

Future Challenges

Concerning some current aspects barely touched upon in the previous sections:

Artificial Intelligence. The full impact of the rapid development of AI with respect to the area of computer security is very hard to predict. Some observations can still be made. Until we have AI systems that can formally prove the claims they make, or at least give very strict explanations, it is unlikely that human-made security solutions are replaced with entirely machine-made ones. AI systems might be trained to serve as useful tools for security and safety development, for instance, to act as powerful attackers and try to hack existing systems to find real-world vulnerabilities. Other AI security-related areas are surveillance and different forms of detection systems. These are all developments that further highlight the need for secure communication in general and stress the importance of a robust security infrastructure. Finally, AI tools for code generation and code testing can help to speed up practical security developments.

Quantum cryptography. Quantum computing is seen as an imminent threat to all Internet security since important asymmetric algorithms used in today's PKIs could be compromised with the advent of post-test-scale quantum computing. The predictions of when quantum computers might be sufficiently powerful to be real threats to current asymmetric algorithm keys vary greatly. Nevertheless, both academia and state agencies are looking into ways of alleviating the potential problem, and developing Post-Quantum Cryptography (PQC) [8]. Within IETF there are proposals and active discussions on quantum robust algorithms, using novel hashing mechanisms instead of for instance elliptic curves (such as Leighton-Micali Hash-Based Signatures [37]). The proposed application layer standard, EDHOC, acknowledges and discusses the potential need to in the future replace the current algorithms with PQC, and their usage has already been specified for COSE [24].

Regarding certificates specifically, it is less problematic since a certificate data structure can easily hold key material belonging to hash-based keying mechanisms, which at least currently, are believed to be quantum resilient. It is worth noticing that the algorithm which is specified for COSE usage only securely can be used for a fixed number of signing operations [24]. This could lead to a demand for frequent certificate updates and highlights the need for compact certificates and lightweight enrollment operations.

Future Work

There are many open possible future directions for how to continue to strengthen IoT security infrastructure. Some that I think are worth mentioning are:

Looking into how the newly published proposal for a Remote ATtestation procedureS (RATS) architecture [9] best could be combined with lightweight PKI solutions to improve the possibilities to offer secure IoT deployments and further automate maintenance.

Our solution for revocation does not address OCSP stapling, a mechanism which might be adaptable also for IoT to remove the reliance on an always reachable validation authority and possibly further reduce the overhead.

There are still low-level details related to bootstrapping of IoT devices in multi-hop networks where manual or insecurely automated deployment steps would benefit from further standardisation and developments making use of PKI functionality.

The ongoing interest in extremely constrained batteryless IoT devices leads to new vulnerabilities which are even harder to address. Finding efficient ways for the more capable IoT devices to coordinate security services for the most constrained devices is an important task, where the new PKI functionality should be taken advantage of as far as possible.

One of the practical details I plan to continue attending to is pushing the C509 draft to an accepted RFC. This could include adding expansions to cover the encoding of revocation operations.

6. Summary in Swedish

Sakernas internet, även kallat IoT, det nätverk av kommunicerande enheter som oftast saknar traditionella användargränssnitt och är begränsade i beräkningskapacitet, har vuxit kraftigt de senaste tio åren. I allmänhetens medvetande är troligen de många smarta prylar för styrning och kontroll i privata hem, eller för träning och motion, mest framträdande. IoT-lösningar används alltmer även i säkerhetskritiska tillämpningar, som infrastruktur och sjukvård. I samband med det kraftigt ökande användande har behoven av skalbara säkerhetslösningar vuxit i motsvarande grad. Kända cyberangrepp som Stuxnet-attackerna på Irans elinfrastruktur har fått mycket mediauppmärksamhet, men även till synes säkerhetsmässigt obetydliga smarta prylar kan hackas och användas mot andra mål i cyberattacker genom så kallade botnets.

Under framväxten av sakernas internet har säkerhetslösningarna ofta släpat efter, varit otillräckliga och många gånger bundna till ett specifikt företag, något som skapar inlåsningseffekter och förhindrar skapandet av säkra nätverk med samverkande enheter från olika leverantörer.

Till stor del har bristen på säkerhetslösningar speglat bristen på kapacitet hos IoT-enheterna. Samtidigt har det funnits ett intresse från både forskarvärlden och industrin att skapa fler standarder även för enheter med begränsade resurser. Det skulle ha en rad fördelar, som att göra dem enklare att använda även i heterogena system, för att göra det möjligt för fler och även mindre företag i branschen att kunna konkurrera, och det höjer abstraktionsnivån för dem som ska skapa tjänster för systemen genom att skapa enhetliga programmeringsinterface mot standardiserade protokoll.

Två parallella trender inom IoT som kan urskiljas är å ena sidan en utveckling mot extremt resursbegränsade enheter utan batterier som samlar sin energi från omgivningen. Dessa kräver än så länge extremt anpassade lösningar, och kan inte direkt integreras med övriga internet. Den andra trenden är mot mer kraftfulla enheter, som med rätt anpassningar har förutsättningar för att integreras med säkerhetslösningar som används på övriga internet. Det är värt att påpeka att behovet av optimeringar kommer att kvarstå även när resurserna blir fler: för att hålla nere hårdvarukostnaderna, för att förlänga tiden vid batteridrift och för att hantera utmaningarna för trådlösa uppkopplingar i utmanande radiomiljöer.

För att undvika beroende av enskilda företag bör IoT-säkerhet dra nytta av de lösningar som redan används på resten av internet. En av de mest grundläggande säkerhetstjänsterna är autentisering, att verifiera identiteten hos någon du behöver interagera med. Tack vare lösningar för kryptografi som bygger

på tvådelade nycklar, en publik och en hemlig, har det varit möjligt att bygga upp system där de offentliga nycklarna fritt kan distribueras. Det är tillräckligt att känna till den publika nyckeln som hör till en användare eller enhet för att en annan användare ska kunna bekräfta om en signatur har skapats med den tillhörande hemliga nyckeln. Genom tillgång till digitala certifikat, där publika nycklar kapslats in och som signeras av en betrodd tredje part, blir det möjligt för två parter att säkert autentisera varandra. Hela systemet för att hantera tjänsterna för autentisering och tillhörande delar, certifikat, nycklar och policyer, bildar en så kallad Public Key Infrastructure, PKI. Systemet med publika nycklar distribuerade genom certifikat har blivit grunden för nästan all internetkommunikation. För vanliga användare är dessa säkerhetsdetaljer oftast dolda.

Befintliga lösningar för PKI har varit för komplexa och resurskrävande för att kunna användas för IoT, men fördelarna med att låta även IoT-enheter bli en del av säkerhetsinfrastrukturen har gjort det till ett intressant forskningsområde.

Temat för arbetet som presenteras i denna avhandling har varit att överbrygga delar av de befintliga gap vi identifierat i det nuvarande sakernas internet gällande befintliga säkerhetslösningar. Fokus är på protokoll, mekanismer och effektiva format snarare än nya kryptografiska algoritmer.

Flera av bidragen syftar till att förbättra tillgången till säkra PKI-lösningar för IoT. Det inkluderar lösningar för att registrera nya certifikat, en av de mest grundläggande PKI-operationerna. Tillsammans med nya mer kompakta format för att lagra och hantera certifikat minskas den resursanvändning som krävs för en IoT-enhet att koppla upp sig och registrera ett nytt certifikat till mer hanterbara nivåer. Ytterligare förbättringar handlar om att skapa lösningar som gör det möjligt att tillhandahålla komplett punkt-till-punkt-säkerhet som låter datatrafik passera proxy-routrar.

Förutom att registrera nya certifikat behövs också mekanismer för att återkalla befintliga sådana, t ex om en dator hackats och inte längre ska användas. Ett av bidragen föreslår en resurseffektiv lösning för att låta IoT-enheter undersöka statusen hos eventuellt återkallade certifikat.

Ett av bidragen syftar till att föreslå nya metoder för att effektivt lagra sensordata lokalt på IoT-enheter, så att data omedelbart och säkert kan skickas ut på nätverket utan extra operationer för kryptering. Vi demonstrerar hur detta kan minska resursbehovet och minska fördröjningarna för datainsamling.

För att förbättra möjligheterna för mer långsiktigt underhåll av IoT-system föreslår vi effektiva metoder för att flytta kontrollen mellan olika operatörer, med ett minimum av manuellt arbete.

En annat bidrag för att förbättra möjligheterna att upprätthålla säkerheten för IoT-system över tid, till exempel i ljuset av nyupptäckta brister som kräver åtgärder av befintliga installationer, är en standardbaserad arkitektur för säkra mjukvaruuppdateringar för IoT. Genom kompakta representationer av meta-

data skapas möjligheter för resurseffektiv accesskontroll för IoT-enheter som kan verifiera om en uppdatering är avsedd för dem och har en giltig utgivare.

Utöver de bidrag som har publicerats i vetenskapliga artiklar har vi samarbetat med kollegor i industrin för att dela resultaten i form av nya standarder, för att öka chanserna till industriellt genomslag.

Sammanlagt har vi genom dessa bidrag föreslagit en rad nya byggstenar för att skapa och upprätthålla säkra PKI-system som klarar att hantera begränsade IoT-enheter. Vi har därigenom tagit viktiga steg mot att göra även IoT-enheter till fullvärdiga deltagare på ett säkert internet.

Bibliography

[1] Fahd A. Alhaidari and Ebtesam J. Alqahtani. “Securing Communication between Fog Computing and IoT Using Constrained Application Protocol (CoAP): A Survey”. In: *J. Commun.* 15 (2020), pp. 14–30.

[2] M. N. Aman, U. Javaid, and B. Sikdar. “A Privacy-Preserving and Scalable Authentication Protocol for the Internet of Vehicles”. In: *IEEE Internet of Things Journal* 8.2 (2021), pp. 1123–1139. DOI: 10.1109/JIOT.2020.3010893.

[3] *Android keystore system*. 2020. URL: https://developer.android.com/training/articles/keystore (visited on 05/31/2021).

[4] Deeksha Anniappa and Yoohwan Kim. “Security and Privacy Issues with Virtual Private Voice Assistants”. In: *2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC)*. 2021, pp. 0702–0708. DOI: 10.1109/CCWC51732.2021.9375964.

[5] Manos Antonakakis et al. “Understanding the Mirai Botnet”. In: *Proceedings of the 26th USENIX Conference on Security Symposium*. SEC’17. Vancouver, BC, Canada: USENIX Association, 2017, 1093–1110.

[6] N. Asokan, T. Nyman, N. Rattanavipanon, A. Sadeghi, and G. Tsudik. “ASSURED: Architecture for Secure Software Update of Realistic Embedded Devices”. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 37.11 (2018), pp. 2290–2300. DOI: 10.1109/TCAD.2018.2858422.

[7] Ibrahim Ethem Bagci, Shahid Raza, Utz Roedig, and Thiemo Voigt. “Fusion: coalesced confidential storage and communication framework for the IoT”. In: *Security and Communication Networks* 9.15 (2016), pp. 2656–2673.

[8] William Barker, William Polk, and Murugiah Souppaya. *White paper: Getting Ready for Post-Quantum Cryptography: Exploring Challenges Associated with Adopting and Using Post-Quantum Cryptographic Algorithms*. Tech. rep. NIST CSWP 15. 100 Bureau Drive, Gaithersburg, MD 20899: National Institute of Standards and Technology, 2021. URL: https://doi.org/10.6028/NIST.CSWP.04282021.

[9] H. Birkholz, D. Thaler, M. Richardson, N. Smith, and W. Pan. *Remote ATtestation procedureS (RATS) Architecture*. RFC 9334. RFC Editor, 2023. URL: https://www.rfc-editor.org/info/rfc9334.

[10] EunSeong Boo, Shahid Raza, Joel Höglund, and JeongGil Ko. "FDTLS: Supporting DTLS-Based Combined Storage and Communication Security for IoT Devices". In: *16th IEEE International Conference on Mobile Ad Hoc and Sensor Systems, MASS 2019, Monterey, CA, USA, November 4-7, 2019*. IEEE, 2019, pp. 127–135.

[11] C. Bormann, M. Ersue, and A. Keranen. *Terminology for Constrained-Node Networks*. RFC 7228. RFC Editor, 2014. URL: http://www.rfc-editor.org/rfc/rfc7228.txt.

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

[13] C. Bormann and Z. Shelby. *Block-Wise Transfers in the Constrained Application Protocol (CoAP)*. RFC 7959. RFC Editor, 2016.

[14] 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, 2008. URL: http://www.rfc-editor.org/rfc/rfc5280.txt.

[15] Stephen E. Deering and Robert M. Hinden. *Internet Protocol, Version 6 (IPv6) Specification*. RFC 2460. RFC Editor, 1998.

[16] DigiCert. *PKI: The Security Solution for the Internet of Things*. Tech. rep. DigiCert Inc., 2017.

[17] C. Doukas, I. Maglogiannis, V. Koufi, F. Malamateniou, and G. Vassilacopoulos. "Enabling data protection through PKI encryption in IoT m-Health devices". In: *2012 IEEE 12th International Conference on Bioinformatics Bioengineering (BIBE)*. 2012, pp. 25–29. DOI: 10.1109/BIBE.2012.6399701.

[18] Adam Dunkels, Joakim Eriksson, Niclas Finne, Fredrik Österlind, Nicolas Tsiftes, Julien Abeillé, and Mathilde Durvy. "Low-power IPv6 for the Internet of Things". In: *2012 Ninth International Conference on Networked Sensing (INSS)*. 2012, pp. 1–6. DOI: 10.1109/INSS.2012.6240537.

[19] David Emery. *'My Friend Cayla' Doll Records Children's Speech, Is Vulnerable to Hackers*. Snoopes, 2017. URL: https://www.snopes.com/news/2017/02/24/my-friend-cayla-doll-privacy-concerns/ (visited on 12/20/2022).

[20] Thanassis Giannetsos and Ioannis Krontiris. "Securing V2X Communications for the Future: Can PKI Systems Offer the Answer?" In: *Proceedings of the 14th International Conference on Availability, Reliability and Security*. ARES '19. Canterbury, CA, United Kingdom: Association for Computing Machinery, 2019.

[21] Steve Gibson. *An Evaluation of the Effectiveness of Chrome's CRLSets*. [Online; accessed 22-January-2023]. 2014. URL: https://web.archive.org/web/20221220170220/https://www.grc.com/revocation/crlsets.htm.

[22] Martin Gunnarsson and Christian Gehrmann. "Secure ownership transfer for the Internet of Things". eng. In: *Proceedings of the 6th International Conference on Information Systems Security and Privacy*. Ed. by Steven Furnell, Paolo Mori, Edgar Weippl, and Olivier Camp. Vol. 1. SciTePress, 2020, pp. 33–44. DOI: 10.5220/0008928300330044.

[23] Daniel Hein, Johannes Winter, and Andreas Fitzek. "Secure Block Device – Secure, Flexible, and Efficient Data Storage for ARM TrustZone Systems". In: *2015 IEEE Trustcom/BigDataSE/ISPA*. Vol. 1. 2015, pp. 222–229. DOI: 10.1109/Trustcom.2015.378.

[24] R. Housley. *Use of the HSS/LMS Hash-Based Signature Algorithm with CBOR Object Signing and Encryption (COSE)*. RFC 8778. RFC Editor, 2020. URL: https://www.rfc-editor.org/info/rfc8778.

[25] Russell Housley, Warwick Ford, Tim Polk, and David Solo. *Internet X.509 Public Key Infrastructure Certificate and CRL Profile*. RFC 2459. RFC Editor, 1999.

[26] J. Hui and P. Thubert. *Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks*. RFC 6282. RFC Editor, 2011. URL: http://www.rfc-editor.org/rfc/rfc6282.txt.

[27] *IEEE 1609.2-2016 - IEEE Standard for Wireless Access in Vehicular Environments*. 2016. URL: https://standards.ieee.org/standard/1609_2-2016.html.

[28] Qi Jing, Athanasios V. Vasilakos, Jiafu Wan, Jingwei Lu, and Dechao Qiu. "Security of the Internet of Things: perspectives and challenges". In: *Wireless Networks* 20.8 (2014), pp. 2481–2501. DOI: 10.1007/s11276-014-0761-7.

[29] M. Jones, L. Seitz, G. Selander, S. Erdtman, and H. Tschofenig. *Proof-of-Possession Key Semantics for CBOR Web Tokens (CWTs)*. RFC 8747. RFC Editor, 2020.

[30] C. Kaufman. *Internet Key Exchange (IKEv2) Protocol*. RFC 4306. RFC Editor, 2005.

[31] Keyfactor. *PKI: The Solution for Building Secure IoT Devices*. Tech. rep. Keyfactor, Inc, 2020.

[32] Md Sakib Nizam Khan, Samuel Marchal, Sonja Buchegger, and N. Asokan. "chownIoT: Enhancing IoT Privacy by Automated Handling of Ownership Change". In: *Privacy and Identity Management. Fairness, Accountability, and Transparency in the Age of Big Data :* vol. 547. 2018, pp. 205–221. DOI: 10.1007/978-3-030-16744-8_14.

[33] Minhaj Ahmad Khan and Khaled Salah. “IoT security: Review, blockchain solutions, and open challenges”. In: *Future Generation Computer Systems* 82 (2018), pp. 395–411. DOI: https://doi.org/10.1016/j.future.2017.11.022.

[34] Hugo Krawczyk. “SIGMA: The ‘SIGn-and-MAc’ Approach to Authenticated Diffie-Hellman and Its Use in the IKE Protocols”. In: *Advances in Cryptology - CRYPTO 2003*. Ed. by Dan Boneh. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 400–425.

[35] James Larisch, David Choffnes, Dave Levin, Bruce M. Maggs, Alan Mislove, and Christo Wilson. “CRLite: A Scalable System for Pushing All TLS Revocations to All Browsers”. In: *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 539–556. DOI: 10.1109/sp.2017.17. URL: https://doi.org/10.1109/sp.2017.17.

[36] Xiaolei Li, Hong Hu, Guangdong Bai, Yaoqi Jia, Zhenkai Liang, and Prateek Saxena. “DroidVault: A Trusted Data Vault for Android Devices”. In: *2014 19th International Conference on Engineering of Complex Computer Systems*. 2014, pp. 29–38. DOI: 10.1109/ICECCS.2014.13.

[37] D. McGrew, M. Curcio, and S. Fluhrer. *Leighton-Micali Hash-Based Signatures*. RFC 8554. RFC Editor, 2019.

[38] Xin Li Minmei Wang Chen Qian and Shouqian Shi. “Collaborative Validation of Public-Key Certificates for IoT by Distributed Caching”. In: *Proceedings of IEEE INFOCOM*. Paris, France, 2019, pp. 92–105.

[39] Brendan Moran, Hannes Tschofenig, Henk Birkholz, and Koen Zandberg. *A Concise Binary Object Representation (CBOR)-based Serialization Format for the Software Updates for Internet of Things (SUIT) Manifest*. Internet-Draft draft-ietf-suit-manifest-17. IETF Secretariat, 2022. URL: https://www.ietf.org/archive/id/draft-ietf-suit-manifest-17.txt.

[40] Moteiv. *Tmote Sky, Ultra low power IEEE 802.15.4 compliant wireless sensor module*. [Online; accessed 23-January-2023]. 2006. URL: https://web.archive.org/web/20070205095928/http://www.moteiv.com/products/docs/tmote-sky-datasheet.pdf.

[41] Nordic Semiconductor. *nRF52840 System-on-Chip, Multiprotocol Bluetooth 5.2 SoC supporting Bluetooth Low Energy, Bluetooth mesh, NFC, Thread and Zigbee*. [Online; accessed 23-January-2023]. 2021. URL: https://web.archive.org/web/20210731190637/https://www.nordicsemi.com/Products/nRF52840.

[42] Baraka William Nyamtiga, Jose Costa Sapalo Sicato, Shailendra Rathore, Yunsick Sung, and Jong Hyuk Park. "Blockchain-Based Secure Storage Management with Edge Computing for IoT". In: *Electronics* 8.8 (2019). DOI: 10.3390/electronics8080828.

[43] Chang-Seop Park. "A Secure and Efficient ECQV Implicit Certificate Issuance Protocol for the Internet of Things Applications". In: *IEEE Sensors Journal* 17.7 (2017), pp. 2215–2223. DOI: 10.1109/JSEN.2016.2625821.

[44] Sandro Pinto and Nuno Santos. "Demystifying Arm TrustZone: A Comprehensive Survey". In: *ACM Comput. Surv.* 51.6 (2019).

[45] M. Pritikin, P. Yee, and D. Harkins. *Enrollment over Secure Transport*. RFC 7030. RFC Editor, 2013.

[46] K. Rabieh, M. M. E. A. Mahmoud, K. Akkaya, and S. Tonyali. "Scalable Certificate Revocation Schemes for Smart Grid AMI Networks Using Bloom Filters". In: *IEEE Transactions on Dependable and Secure Computing* 14.4 (2017), pp. 420–432. DOI: 10.1109/TDSC.2015.2467385.

[47] Maxim Raya, Daniel Jungels, Panos Papadimitratos, Imad Aad, and Jean-Pierre Hubaux. "Certificate Revocation in Vehicular Networks". In: *Laboratory for computer Communications and Applications (LCA)* (2006).

[48] E. Rescorla and N. Modadugu. *Datagram Transport Layer Security Version 1.2*. RFC 6347. RFC Editor, 2012. URL: http://www.rfc-editor.org/rfc/rfc6347.txt.

[49] Justin Samuel, Nick Mathewson, Justin Cappos, and Roger Dingledine. "Survivable Key Compromise in Software Update Systems". In: *Proceedings of the 17th ACM Conference on Computer and Communications Security*. CCS '10. Chicago, Illinois, USA: ACM, 2010, pp. 61–72. DOI: 10.1145/1866307.1866315.

[50] S. Santesson, M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams. *X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP*. RFC 6960. RFC Editor, 2013. URL: http://www.rfc-editor.org/rfc/rfc6960.txt.

[51] Ludwig Seitz, Goeran Selander, Erik Wahlstroem, Samuel Erdtman, and Hannes Tschofenig. *Authentication and Authorization for Constrained Environments (ACE) using the OAuth 2.0 Framework (ACE-OAuth)*. Internet Draft draft-ietf-ace-oauth-authz-24. IETF Secretariat, 2019. URL: http://www.ietf.org/internet-drafts/draft-ietf-ace-oauth-authz-24.txt.

[52] G. Selander, J. Mattsson, F. Palombini, and L. Seitz. *Object Security for Constrained RESTful Environments (OSCORE)*. RFC 8613. RFC Editor, 2019.

[53] Goeran Selander, John Mattsson, and Francesca Palombini. *Ephemeral Diffie-Hellman Over COSE (EDHOC)*. Internet-Draft draft-ietf-lake-edhoc-03. IETF Secretariat, 2020.

[54] Stuxnet. *Stuxnet — Wikipedia, The Free Encyclopedia*. [Online; accessed 10-January-2023]. 2023. URL: https://en.wikipedia.org/w/index.php?title=Stuxnet&oldid=1131553616.

[55] R. T. Tiburski, L. A. Amaral, E. de Matos, D. F. G. de Azevedo, and F. Hessel. "Evaluating the use of TLS and DTLS protocols in IoT middleware systems applied to E-health". In: *2017 14th IEEE Annual Consumer Communications Networking Conference (CCNC)*. 2017, pp. 480–485. DOI: 10.1109/CCNC.2017.7983155.

[56] H. Tschofenig and T. Fossati. *Transport Layer Security (TLS) / Datagram Transport Layer Security (DTLS) Profiles for the Internet of Things*. RFC 7925. RFC Editor, 2016.

[57] Nguyen Van Huynh, Dinh Thai Hoang, Xiao Lu, Dusit Niyato, Ping Wang, and Dong In Kim. "Ambient Backscatter Communications: A Contemporary Survey". In: vol. 20. 4. 2018, pp. 2889–2922. DOI: 10.1109/COMST.2018.2841964.

[58] Wikipedia contributors. *ASN.1 — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=ASN.1&oldid=1108950664. [Online; accessed 22-January-2023]. 2022.

[59] Rebecca N. Wright, Patrick D. Lincoln, and Jonathan K. Millen. "Efficient Fault-tolerant Certificate Revocation". In: *Proceedings of the 7th ACM Conference on Computer and Communications Security*. CCS '00. Athens, Greece: ACM, 2000, pp. 19–24. DOI: 10.1145/352600.352605.

[60] Nian Xue, Daojing Guo, Jie Zhang, Jihao Xin, Zhen Li, and Xin Huang. "OpenFunction for Software Defined IoT". In: *2021 International Symposium on Networks, Computers and Communications (ISNCC)*. 2021, pp. 1–8. DOI: 10.1109/ISNCC52172.2021.9615751.

[61] Lijing Zhou, Licheng Wang, Yiru Sun, and Pin Lv. "BeeKeeper: A Blockchain-Based IoT System With Secure Storage and Homomorphic Computation". In: *IEEE Access* 6 (2018), pp. 43472–43488. DOI: 10.1109/ACCESS.2018.2847632.

Acta Universitatis Upsaliensis

Digital Comprehensive Summaries of Uppsala Dissertations from the Faculty of Science and Technology 2230

Editor: The Dean of the Faculty of Science and Technology

A doctoral dissertation from the Faculty of Science and Technology, Uppsala University, is usually a summary of a number of papers. A few copies of the complete dissertation are kept at major Swedish research libraries, while the summary alone is distributed internationally through the series Digital Comprehensive Summaries of Uppsala Dissertations from the Faculty of Science and Technology. (Prior to January, 2005, the series was published under the title "Comprehensive Summaries of Uppsala Dissertations from the Faculty of Science and Technology".)

Distribution: publications.uu.se
urn:nbn:se:uu:diva-495320

ACTA
UNIVERSITATIS
UPSALIENSIS
UPPSALA
2023

Paper I

Computers & Security 89 (2020) 101658

Contents lists available at ScienceDirect

Computers & Security

journal homepage: www.elsevier.com/locate/cose

PKI4IoT: Towards public key infrastructure for the Internet of Things

Joel Höglund [a], Samuel Lindemer [a], Martin Furuhed [b], Shahid Raza [a,*]

[a] RISE Research Institutes of Sweden Isafjordsgatan 22, Kista 16440, Stockholm
[b] Technology Nexus Secured Business Solutions, Sweden, Telefonvägen 26, Hägersten 12626, Stockholm

ARTICLE INFO

Article history:
Received 20 June 2019
Revised 3 October 2019
Accepted 31 October 2019
Available online 5 November 2019

Keywords:
Security
CBOR
IoT
PKI
Digital certificates
Enrollment
Embedded systems
Contiki

ABSTRACT

Public Key Infrastructure is the state-of-the-art credential management solution on the Internet. However, the millions of constrained devices that make of the Internet of Things currently lack a centralized, scalable system for managing keys and identities. Modern PKI is built on a set of protocols which were not designed for constrained environments, and as a result many small, battery-powered IoT devices lack the required computing resources. In this paper, we develop an automated certificate enrollment protocol light enough for highly constrained devices, which provides end-to-end security between certificate authorities (CA) and the recipient IoT devices. We also design a lightweight profile for X.509 digital certificates with CBOR encoding, called *XIOT*. Existing CAs can now issue traditional X.509 to IoT devices. These are converted to and from the XIOT format by edge devices on constrained networks. This procedure preserves the integrity of the original CA signature, so the edge device performing certificate conversion need not be trusted. We implement these protocols within the Contiki embedded operating system and evaluate their performance on an ARM Cortex-M3 platform. Our evaluation demonstrates reductions in energy expenditure and communication latency. The RAM and ROM required to implement these protocols are on par with the other lightweight protocols in Contiki's network stack.

© 2019 The Authors. Published by Elsevier Ltd.
This is an open access article under the CC BY-NC-ND license.
(http://creativecommons.org/licenses/by-nc-nd/4.0/)

1. Introduction

Public key infrastructure (PKI) is ubiquitous throughtout a wide variety of networked systems for centralized credential management and key distribution. The Internet of Things (IoT) has been slow to adopt PKI due to reasons both economic and technical. Instead, embedded systems often rely on pre-shared keys (PSK), which become problematic when those systems are connected to the Internet and become globally adressable. The keys must be installed before deployment, and because centralized resources must share a key with each device in order to communicate, a single server compromise can put the entire network at risk. Moreover, many basic security guarantees such as proof-of-origin, access control, non-repudiation and authentication are simply not possible with PSK systems.

Regardless of the simplicity of each individual device, the IoT presents great security risks due to its sheer scale, which is likely to increase by billions of devices in the next few years. In the Mirai attack of 2016, for example, hackers created a 600,000-device botnet of unsecured embedded systems (primarily surveillance cameras), which were then used to launch DDoS attacks. In Germany, parents were told to destroy the doll “My Friend Cayla”, because it

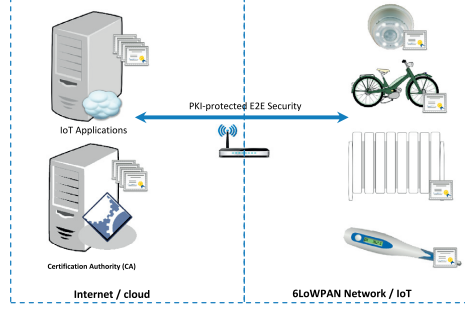


Fig. 1. An illustration of PKI-protected IoT setup with end-to-end security between IoT devices and back-end service, without intermediate trusted gateways.

* Corresponding author.
E-mail addresses: joel.hoglund@ri.se (J. Höglund), samuel.lindemer@ri.se (S. Lindemer), martin.furuhed@nexusgroup.com (M. Furuhed), shahid.raza@ri.se (S. Raza).

https://doi.org/10.1016/j.cose.2019.101658
0167-4048/© 2019 The Authors. Published by Elsevier Ltd. This is an open access article under the CC BY-NC-ND license. (http://creativecommons.org/licenses/by-nc-nd/4.0/)

contained an unsecured Internet-connected microphone, and was classified as an illegal surveillance device (Emery, 2017). These examples are trivial compared to the risks associated with Internet-connected patient monitoring systems and medical implants. In summary, any device, regardless of size or function, must have strong authentication mechanisms when connected to the Internet, and those mechanisms are all based on PKI and digital certificates. PINs and passwords are extremely weak in comparison.

IoT devices are often basic sensors or actuators with stringent resource constraints. In order to participate in a PKI, they must have mechanisms for initial enrollment (i.e., obtaining the first certificate and key pair), re-enrollment and certificate verification. These are not inherently complex operations, but the existing standards for each of them were not designed for battery-powered devices with tens of kilobytes of RAM and low-power radios. In recent years, several lightweight profiles of traditional Web protocols have been standardized to enable IPv6 networking on highly-constrained devices. This began with 6LoWPAN in 2007, which compresses IPv6 packets for low-power radio networks. Later CoAP and CBOR were introduced, which are lightweight profiles of HTTP and JSON, respectively. In this paper, we build upon these technologies to enable PKI for IoT, or *PKI4IoT*.

With very few exceptions, digital certificates are issued in the X.509 ASN.1 format. This is not a particularly compact encoding scheme, but due to its monopoly on modern PKI implementations, the solutions developed in this paper are designed work with it. An important distinction should be drawn between reinventing PKI and making constrained devices compatible with existing PKI. We believe the latter is the better, more feasible approach; migrating the entire Internet to a new, compact certificate format is unlikely to occur within the next decade.

Our contributions in this work constitute a major step towards PKI4IoT. The main contributions of the paper are as follows.

- The design of an integrated lightweight certificate enrollment for IoT devices, with support for automated zero-touch protection of enrollment sessions.
- The profiling and lightweight encoding of X.509 certificates.
- The implementation of lightweight enrollment for resource-constrained devices, specifically the integration of EST and domain specific certificate compression mechanisms which maintains the end to end security guarantees.
- Evaluation of the PKI protocols in a multi-hop IoT test bed using real IoT hardware.

The rest of this paper is organized as follows: Section 2 gives our threat model and assumptions. Section 3 presents related work. Section 4 presents the PKI4IoTlife cycle. Section 5 presents the enrollment and profiling plus compression (5.1, 5.2), and puts PKI4IoTin context with respect to the establishment of trust and practical feasibility for IoT (5.3, 5.4). Section 6 presents the implementation. Section 7 gives evaluation results. Section 8 gives further security considerations before concluding the paper.

2. Threat model and assumptions

2.1. Trust base

We consider the hardware of the constrained device, and the local software to be trusted. We assume there is at least one CA capable of generating certificates which can be trusted by the device through an initial trust store. We assume the NIST hash and crypto recommendations for which types of algorithms and key length to use for the upcoming decade the will stay valid.

2.2. Attack model and security guarantees

An attacker can eavesdrop all communication between the involved entities. They can block or modify messages in transit, or store and replay any message sent, with the goal to masquerade as a trusted entity, or to get hold of any secret message content in plaintext. A communication system should withstand this type of attacker, and still offer authentication and confidentiality through secure communication. For a single communication session, mechanisms to detect modified or replicated/replayed message attacks are needed. For a complete communication system, secure and efficient key management mechanisms are needed, including key provisioning and key revocation. Providing initial secure and efficient building blocks for key management that can withstand these types of attackers are our main contributions.

2.3. Limitations

The proposed system does not itself protect against denial-of-service attacks, in which the perpetrator floods a victim with erroneous request; however, PKI4IoT makes it very hard to an attacker to take control of an IoT device in the first place. For DoS protection, additional firewall mechanisms should be deployed. As mentioned, since we assume the hardware to be trusted, an attacker who could gain physical access to the device could potentially extract a private key, compromising the system. To prevent this, additional secure hardware should be deployed. To ensure software security, measures such as fuzzing or formal analysis should be used.

3. Related work

With growing interest in IoT and the accelerating process of digitalization, IoT security has received considerable attention. The existing security paradigm for resource-constrained devices are still based on pre-shared key solutions, or PSK, where devices have a secret phrase pre-installed which is used to create the keys for secure connections. This avoids the higher computational cost of public key operations, but is more vulnerable. A leaked shared secret requires updating all devices in the system, whereas a compromised certificate can be individually revoked.

3.1. PKI for IoT

Industry actors have identified the need for a PKI which encompasses the IoT, for IoT products and services to be sustainably successful (Research, 2016) but existing state-of-the-art solutions are severely restricted. Modern public key solutions can mainly be grouped in two categories. One, such as described in the DigiCert whitepaper, are mainly comprised of “big” IoT devices with relatively few resource constraints (DigiCert, 2017). These devices can run standard X.509-based protocols with little or no modifications. Others, like Managing, rely on offloading certificate management operations to gateways or other powerful devices. This has been the previous state-of-the-art in research (Raza et al., 2016; Ting et al., 2018). While the delegation approach reduces the load on the resource-constrained devices, it breaks the end-to-end security and introduces additional entities that need to be trusted by all parties. Initial attempts to provide Internet standard PKI solutions for IoT has been presented in Raza et al. (2017), in which the authors show that running DTLS over CoAP is feasible, and that the existing X.509 certificate format could be used, albeit with high overhead.

3.2. Existing enrollment solutions

There are existing proposed standards for certificate enrollment which are designed for, and hence suitable for, non-constrained

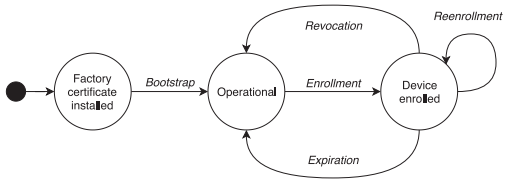


Fig. 2. PKI4IoTlife cycle overview, illustrating the IoT device life cycle events, from manufacturing firmware installation, over initial enrollment to normal operation state, with the possibilities of re-enrollment and revocation or expiration.

devices: SCEP (Gutmann, 2019), CMC (Schaad and Myers, 2008), CMP (Adams et al., 2005), and EST (Pritikin et al., 2013), the newest. Out of these EST has the best support for automated operations, while being relatively lightweight.

3.3. X.509 alternatives

There are alternatives to X.509 certificates. So-called implicit certificates have a smaller memory footprint. Security solutions based on implicit certificates have been shown to establish secure connections between IoT devices (Park, 2017). Until now, there has not been any security infrastructure that allows implicit certificates to be used when a IoT device wants to communicate with an arbitrary Internet device. Recently, implicit certificates certificates are being tested for V2X communication in the US. The IEEE 1609.2 standard defines implicit certificates for use by Wireless Access in Vehicular Environments (WAVE) devices (Ieee, 2016), but the uptake and applicability across the IoT domain remains unclear. There have been previous attempts at creating more compact ECC-based certificates which could use either implicit ECQV or ECDSA signatures (Ford and Poeluev, 2015). They present similar size reductions to the ones presented in our results, but have not been demonstrated in any PKI. This highlights the requirement to keep X.509 compatibility – a requirement that is likely to remain valid for several years to come.

3.4. Compression mechanisms

There are efforts to reduce the overhead of sending certificate chains by allowing on-the-fly compression during the initial handshake, using general purpose compression mechanisms (Ghedini and Vasiliev, 2018). Requiring additional compression mechanisms would create further overhead for constrained devices. More importantly, with certificate-domain knowledge, more compact representations are possible in relation to general purpose compression.

4. PKI4IoT Life-cycle

This section explains the life cycle events that need to be supported by a PKI for IoT. For long-running connected IoT deployments, the system needs to be able to evolve, including security upgrades. Additionally, in order to scale to billions of IoT devices, fully automatized security management is needed. Fig. 2 gives an overview of the relevant life cycle events for PKI4IoT, and the relationship between them.

4.1. Bootstrapping

A recent survey of secure bootstrapping mechanisms relevant for IoT highlights the lack of agreed-upon standards, and even definitions (Sarikaya et al., 2018). One of several cited definitions defines bootstrapping in the context of IoT as “the process by which the state of a device, a subsystem, a network, or an application changes from not operational to operational”. The vagueness of the term is reflected in the wide variety of current solutions, often times dependent on hard-coded initial parameters. These solutions have problems both in terms of security and scalability. While the solutions presented in this paper cover part of the bootstrapping stage, a purely standard based solution encompassing all network layers remains to be defined.

4.2. Certificate enrollment and renewal

Even if obstacles related to initial bootstrapping are overcome, a device must have ways of acquiring and verifying valid security keys for the continued operation. In order to achieve fully automated key management for IoT, new solutions for certifying the security keys must be found. While there are enrollment solutions for traditional Internet devices, they are not adjusted to resource-constrained IoT devices and existing IoT protocols. Since no PKI is stronger than its weakest link, secure key management is crucial.

4.3. Certificate revocation

A security infrastructure must handle revocation of trust, if a previously trusted entity for any reason should no longer be granted access to certain assets. Traditional solutions with blacklists for revoked certificates, CRLs, causes unacceptable overhead for constrained devices and networks. Alternatives include protocols such as Online Certificate Status Protocol, OCSP, with less memory footprint but which instead requires more communication steps. Additionally, certificates carry a final validity date which gives a limit after which the responsible certificate authority could deny certificate renewal. For some scenarios, short certificate validity periods could give uncertainty bounds.

Out of the above presented functions, this paper focuses on providing secure certificate enrollment (and re-enrollment), while reducing the certificate-handling overhead connected with existing X.509 certificates.

5. PKI4IoTCore components

In the following the challenge of a secure and lightweight PKI for IoT is addressed in two steps. First a solution for secure enrollment is presented, and then how to reduce the certificate usage overhead. The resulting PKI should be practically usable for IoT scenarios, hence the final subsection shows how the proposed new mechanisms are compatible with efforts of removing human intervention from the certificate management.The following entities are relevant for the scenarios described below:

Factory CA an entity that issues the manufacturer-installed certificate for the IoT device.

Enrollment CA the entity that the IoT device contacts to do enrollment, or re-enrollment.

The IoT device in the scenarios we consider, the IoT device is deployed with a pre-installed certificate from the factory CA, the knowledge of which enrollment CA to contact, and the means to verify the identity of the enrollment CA.

6LoWPAN border router For the case where the certificates used in the handshake are profiled and thus compressible, the border router transparently compresses the certificates entering the low power radio network and decompresses them as they leave the radio network. Fig. 3 shows the presented actors in their respective domains.

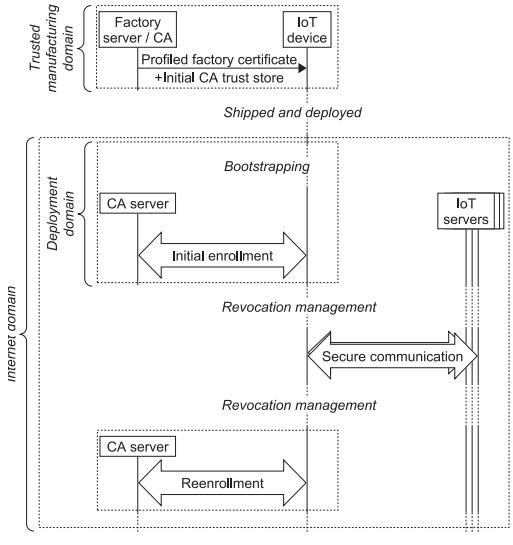


Fig. 3. PKI4IoTfrom a communication perspective, showing the IoT device life cycle events in their respective domains: in factory environment, during deployment, and post-deployment, including communication, certificate re-enrollment and revocation.

5.1. X.509 certificate enrollment for IoT

One of the most urgent challenges to tackle is the provisioning of new certificates to devices as they are being deployed. The mechanisms to do so must be secure themselves, and preferably as close to existing PKI solutions as possible to reduce the barriers for industrial adoption. At the same time, the mechanisms must work on resource-constrained devices, often without any input device. Fig. 3 highlights the entire PKI4IoTprocess with a focus with pre- and post-deployment operations.

5.1.1. Pre-enrollment in manufacturer environment

To enable a true PKI solution for IoT where certificates are used also for initial authentication, an IoT device needs to have certificates pre-installed. The minimal initial truststore needs to contain a factory certificate the device can use to identify itself plus the corresponding private key, and at least one trusted CA certificate that can be used to validate the server to be used for enrollment.

This pre-loading of keys and certificates are preferably done in a trusted manufacturer environment, for example when installing the initial firmware to be used.

5.1.2. Automated enrollment protocol design

In a typical enrollment process, a client sends a certificate signing request (CSR) and gets a response with a newly-created certificate. Either the requesting client or the certificate authority server (CA) can generate a public-private key pair. In the former case, the client generates its own key pair and sends a CSR containing the public key to the CA. The CA in turn generates a certificate by binding the public key with the identity and other necessary information, and sends the digitally signed certificate back to the client. Unlike certificate enrollment done with traditional Internet hosts, we cannot expect the enrollment of IoT devices to involve any human out-of-band operation. A fully automated enrollment process is crucial for enabling PKI for IoT.

We have previously presented a solution for basic enrollment functionality for constrained devices in He et al. (2019). In this work we build upon the previous solutions. We integrate lightweight certificate handling, add functionality for re-enrollment and verify the enrollment process with an off-the-shelf third-party CA software provider that implements the server-side support.

We have chosen to design the enrollment protocol in line with the Enrollment over Secure Transport protocol (EST) that is used for enrollment over HTTP. Unlike previous enrollment solutions for conventional internet (SCEP, CMC), EST does not require additional crypto operation implementations for the sake of enrollment and relies only on the transport layer security, which is particularly useful for resource-constrained IoT devices.

Enrollment layer. The basic enrollment services are provided through EST servers responding to client calls to */crts*, */sen* and */sren*. These are abbreviated versions of the corresponding calls for EST over HTTP: */cacerts*, */simpleenroll* and */simplereenroll*. The path is shortened to a minimum to save additional overhead. The */crts* request serves to update the client with additional CA certificates needed for subsequent certificate verification. The */sen* request contains the initial certificate signing request, CSR, to apply for a certificate from the enrollment CA. Finally, the */sren* is used for subsequent certificate renewals.

CoAP layer. CoAP is a lightweight UDP-based protocol specifically designed for constrained IoT networks. The enrollment process sends CoAP messages of type confirmable, which provides reliability on top of the potentially lossy links. HTTP GET and POST requests in EST are mapped to corresponding GET and POST requests in CoAP, with the corresponding mapping of return values and error codes. Where needed, a couple of new CoAP content formats are introduced, in line with the proposed standard. Since EST messages can grow large, support for block-wise transfer is needed. All payload data is sent in binary format (van der Stok et al., 2019).

DTLS layer. The enrollment message exchange is secured through the creation of a DTLS session between the IoT device and the enrollment CA. The session is created through a certificate-based handshake, where client and server certificates are exchanged. The message flow of a DTLS handshake and subsequent enrollment operations are shown in Fig. 4.

Lower layers. Below DTLS are the UDP and IP layers. For IoT devices that should be globally available, IPv6 is the future-proof alternative that avoids address scarcity or the need for NAT solutions. For IoT devices communicating over 802.15.4 radio networks, 6LoWPAN is used to handle IP packets over the lossy radio links. 6LoWPAN does its own fragment and reassembly of packets and does not support already fragmented IP packets, hence the need for the enrollment process to make use of CoAP block-wise transfer to ensure that DTLS records fit within a single datagram.

5.2. Lightweight X.509-compliant certificates for IoT

The above presented enrollment solution makes it possible for IoT nodes to safely acquire certificates in an automated manner. The current Internet PKI is centred around the use of certificates, predominantly in the X.509 format, with some limited exceptions such as Card Verifiable Certificates, CVC, used for smart cards. The X.509 certificates are relatively heavy, causing significant overhead when existing PKI solutions are being ported to more resource-constrained IoT devices. The structure of existing X.509 certificates is given in Fig. 5. Previous work has shown that the X.509 format can also be used on battery-powered devices, but naively using existing X.509 certificates causes too much overhead (Raza et al., 2017). Hence there is a need to optimize the certificate handling for the IoT.

The structure of the X.509 certificate format is shown in Fig. 5. The main parts are: (i) Information about the subject, the issuer

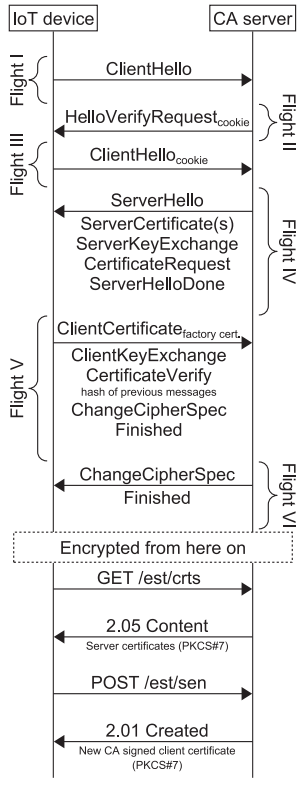


Fig. 4. Enrollment with certificate based DTLS-handshake.

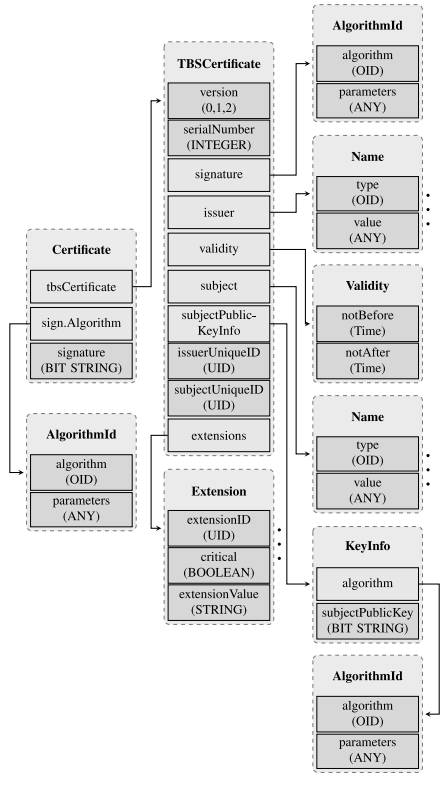


Fig. 5. X.509 certificate structure with compound and primitive fields.

and details such as validity dates (ii) the public key of the subject and the algorithm used (iii) a Certificate Authority (CA) signature and the algorithm used plus, optionally (iv) extensions.

There are DTLS usage guidelines specified for resource-constrained IoT devices (Tschofenig and Fossati, 2016). This gives recommendations on which cipher suites to use, and serves as a source for current best practices. Below, we present actions to reduce the space needed for PKI certificates: profiling, efficient encoding and omission of implied fields.

5.2.1. An X.509 certificate profile for IoT

Based on the knowledge of current protocols and IoT device constraints, we propose the following certificate profile, which we call the XIOT profile, which can be applied to reduce certificate sizes without breaking X.509 compatibility. A summary is given in Table 1.

Version number. The X.509 standard has not moved beyond version 3 since 2008. With the introduction of certificate extensions, new certificate fields can be added without breaking the format, making version changes less likely. Therefore the XIOT profile fixes the version number to 3.

Serial number. The serial number together with the identity of the CA is the unique identifier of a certificate. The standard does not specify the signedness, but following the DTLS IoT guideline, the XIOT profile requires an unsigned integer.

Signature algorithm. For the profile, the signature algorithm is fixed to ECDSA with SHA256.

Issuer. This is used to identify the issuing CA through a sequence of name-value pairs. The IoT profile is restricting this to one pair, common name and associated string value. The requirement is that for global usage, the common name must uniquely identify the CA. Alternatively, for IoT deployments where the trust and communication relations are be of limited scope, a non-unique common name could be used.

Table 1
Summary of XIOT profile field restrictions.

Field	Value
Version	3
Serial number	Unsigned integer
Signature	ecdsaWithSHA256
Issuer	CommonName containing CA name as UTF8String
Validity	UTCTime in format YYMMDDhhmmss
Subject	EUI-64 as UTF8String
Subject public key info	ecPublicKey followed by secp256r1 and 64 byte uncompressed ECC public key
Issuer and subject unique Id	Not present
Extensions	Any extension
Signature algorithm	ecdsaWithSHA256
Signature	ECDSA-Sig-Value

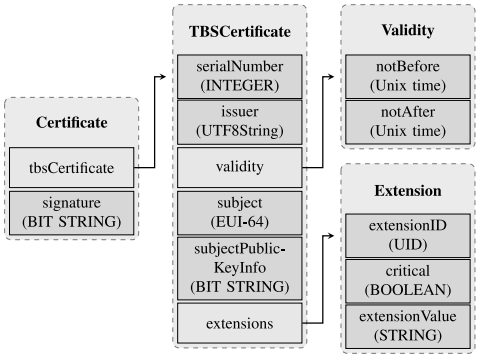


Fig. 6. A XIOT profiled and encoded X.509 certificate, where all fields with known values have been removed.

```
certificate = [
    serial_number  : uint,
    issuer         : text,
    validity       : [notBefore: uint,
                      notAfter: uint],
    subject        : text / bytes,
    public_key     : bytes,
    ? extensions   : [+ extension],
    signature      : bytes
]

extension = [
    oid            : int,
    ? critical     : bool,
    value          : bytes
]
```

Fig. 7. The XIOT CBOR profile of X.509 expressed in Concise Data Definition Language (CDDL).

```
0x30                // Sequence
0x22                // Size 34
0x31                // Set
0x20                // Size 32
0x30                // Sequence
0x1E                // Size 30
0x06                // OID
0x03                // Size 3
0x55 0x04 0x03      // 2.5.4.3
0x0C                // UTF8 string (commonName)
0x17                // Size 23
0x30 0x31 0x2D 0x32 0x33 0x2D 0x34 0x35
0x2D 0x36 0x37 0x2D 0x38 0x39 0x2D 0x41
0x42 0x2D 0x43 0x44 0x2D 0x45 0x46
// Value "01-23-45-67-89-AB-CD-EF"
```

Fig. 8. The serial number field of an X.509 certificate encoded in ASN.1 DER (36 bytes).

```
0x48                // Byte array, Size 8
0x01 0x23 0x45 0x67 0x89 0xAB 0xCD 0xEF
// Value 0x0123456789ABCDEF
```

Fig. 9. The serial number field of an X.509 certificate after XIOT profiling and CBOR encoding (9 bytes).

Validity. X.509 certificates need a 'not before' and a 'not after' date. The profile mandates using the UTCTime-format, YYMMDDhhmmss, for the most compact representation.

Subject. The subject section has the same format as the issuer, identifying the receiver of the public key through a sequence of name-value pairs. This section can be restricted to a single pair, subject name and associated (unique) value. For an IoT device, the MAC-derived EUI-64 serves this purpose well.

Subject public key info. This section contains the public key and identifies the key algorithm. For IoT devices, elliptic curve cryptography (ECC) algorithms can fulfill security requirements with shorter key lengths and less computational overhead compared with older RSA-based cipher suits. A 256 bit ECC key, corresponding to a 3072 bit RSA key, would meet NIST based recommendations for many years ahead. A reasonable restriction following the DTLS IoT profile recommendation is fixing the algorithm to *secp256r1*, sometimes refereed to as *prime256v1*. (For a longer motivating discussion see Forsby, 2017.)

Extensions. To maintain forward compatibility, the profile does not restrict the use of extensions. By the X.509 standard, any device must be able to process eight extensions types. Since only four of them are critical for IoT, the profile makes the other four optional. The following extensions remain critical: - Key Usage - Subject Alternative Name - Basic Constraints - Extended Key Usage

Certificate Signature algorithm. This field duplicates the info present in the signature algorithm field. Fixed to ECDSA with SHA256.

Certificate signature. The field corresponding to the signature done by the CA private key. For the IoT profile, this is restricted to ECDSA signatures.

5.2.2. Encoding and compression

By combining the use of CBOR encoding (Bormann and Hoffman, 2013) instead of ASN.1 together with the knowledge of the XIOT profile restrictions, we are able to reduce the IoT certificates size by more than 50%. The resulting certificate structure is shown in Fig. 6, where all the fields which are known by the profiling have been omitted. The CBOR template corresponding to the certificate profile is shown in Fig. 7.

For the presently-used DTLS v1.2 protocol, in which the handshake messages are unencrypted, the actual encoding and compression can be done transparently at a 6LoWPAN border router, which allows the server side to remain unmodified. For DTLS v1.3, the encoding needs to be done fully end-to-end, by adding additional functionality to the server. A handshake with an unmodified server and a compressing/decompressing gateway is shown in Fig. 10.

Below we go through the relevant certificate fields, and the expected size reductions achieved by the compression. Figs. 8 and 9 give a detailed illustration of how the CBOR encoding achieves increased compactness.

Version number. Assuming a fixed version 3 flag, this field is omitted and recreated when needed. This saves 5 bytes.

Serial number. With no known structure, the savings only come from the CBOR encoding. Encoding overhead is reduced by one byte.

Signature algorithm. The signature algorithm is known from the profile and is omitted in the encoding. This saves 12 bytes.

Issuer. Following the profile restriction to allow only common name, the common name type field is omitted. The total field overhead goes from 13 bytes to one byte.

Validity. This is encoded as UnixTime, which reduces the size from 32 to 11 bytes for a 'not before'-'not after' validity pair.

Subject A subject identified by an EUI-64, based on a 48bit unique MAC id, can be encoded using only 6 bytes with CBOR. This

is a reduction down from 36 bytes for the corresponding ASN.1 encoding.

Subject public key info. The algorithm identifier is known from the profile restrictions and is omitted. One of the public key ECC curve point elements can be calculated from the other, hence only one curve point is needed (Jivsov, 2014). These actions together reduce size from 91 to 35 bytes.

Extensions. Some minor savings can be achieved by the more compact CBOR encoding.

Certificate signature algorithm. Since this is fixed by the profile restrictions, it can be omitted, saving 12 bytes.

Signature. By omitting unneeded ASN.1 information, the overhead for sending the two 32-bit values is reduced from 11 to 2 bytes.

5.3. PKI4IoTchain of trust

A key reason for the success of existing PKI is that it can be used to establish trust between actors without previous shared information. For IoT devices, not only the individual certificate causes overhead, but also the number of certificates needed for trust establishment.

5.3.1. Challenges

A client that wants to verify the authenticity of a server can check if the public certificate is signed by an authority the client already trusts. In this case the single server certificate is sufficient. But often a chain of certificates are needed; the server certificate is signed by a server CA, signed by a shared trusted root CA, possibly with additional intermediate CA:s in between. In addition, specific use cases can have further constraints such as the use of pseudonym CA:s, capable of issuing a large number of pseudonymous certificates for privacy reasons in PKI for the automotive industry.

Devices with powerful CPUs and memory have no problems receiving, handling and checking long chains of certificates up to a shared common trusted root. For IoT devices handling long chains might not be feasible. The exact needs, and hence the optimization possibilities will differ between different use cases.

5.3.2. Solutions

To address the overhead of trust establishment, we revisit the trust relationships presented in the beginning of 5 to discuss which certificates need to be exchanged as part of the initial DTLS handshake.

The general case. The most general case is where the enrollment CA has no previous direct trust relationship with the factory CA. Our current solution can already handle this scenario by having multiple root CA certificates pre-installed, but further investigations on the most efficient solutions for the general case are deferred to future work.

A typical IoT case. For IoT deployments, it is relevant to consider scenarios where the factory CA and the enrollment CA have certificates issued by the same root CA. Alternatively, the factory CA and the enrollment CA have been issued by the same intermediate CA. In both of these cases, the IoT device could already has been given the relevant issuing CA and root CA certificates in its initial truststore.

DTLS handshake optimizations. Based on the above scenario, we propose DTLS handshake optimizations for the case where there is a previous trust relationship between the IoT device and the server. By the TLS standard, the handshake certificate messages from either the server or client side are supposed to contain the full chain of certificates between the endpoint and the root CA. Only the self-signed certificate that specifies the root certificate authority may be omitted from the chain, because the remote end already has it Dierks and Rescorla (2008). We suggest relaxing this requirement, by allowing the certificate messages to only contain the missing server certificates.

We suggest that this procedure can be formalized as follows: The client can advertise the preferred server side authentication method through the ClientHello message. Currently, if the X.509 method is selected, the server will send the full chain of certificates. By changing the TLS Certificate Types flag from indicating the standard X.509 to X.509_SINGLE, the client can indicate that it only expects the server's own certificate, since the client already has the other certificates needed to validate it.

5.4. PKI4IoT: Economical considerations

Besides the practical necessities for automating all aspects of PKI for IoT, there are also economic incentives. For IoT providers to accept the costs of individual certificates for a large number of devices, the prices must be lower than prices for the certificate services currently available. The manual verification steps of current PKI solutions form bottlenecks in terms of both CA performance/capacity and certificate cost. To be able to provide cheap but secure IoT devices, a zero-touch PKI solution is needed.

With our proposed system, based on pre-installed factory certificates and provisioning of new certificates using EST-CoAP, the issuing of certificates can be fully automated. This is crucial for reducing the cost of a single certificate. The CA needs to be capable to issue X.509 certificates following the XIOT certificate profile for the resulting issued certificates to be compressible, but no other modifications are needed.

6. PKI4IoT: Implementation

PKI4IoThas been developed in parallel with the evolution of the EST over CoAPs draft in IETF (van der Stok et al., 2019). The target environment is the Contiki operating system, which provides protocol support for IPv6, UDP and CoAP. The TinyDTLS library has provided the needed cryptographic functions, and acted as a starting point for the entirely certificate-based DTLS version becoming part of PKI4IoT. In addition, we have added point decompression support from the μECC library,[1] and CBOR functionality from CN-CBOR.[2] When the device receives a compressed certificate as part of the handshake message exchange, as seen in Fig. 10, the DTLS component will use the new XIOT component to reconstruct the original certificate. The XIOT component is also used to compress the device certificate before sending it. The relevant parts of the device software stack is illustrated in figure 10.

Before initial enrollment, the device uses its pre-installed factory certificate. After enrollment it will have the certificate assigned by the CA during the enrollment process stored in its certificate store. This means the factory certificate is normally only used once. Consequently it is less crucial that the CA used to issue the factory certificate follows the IoT profile, the main savings follow from using a compressible certificate assigned by the enrollment process.

To support DTLS certificate based handshake, and the EST-CoAP enrollment operations, a certificate store component handles saving certificates to node flash memory, using a modified version of the Contiki file system Coffee.

Since the enrollment messages might not fit in the standard MTU frames of 1280 bytes, the Contiki CoAP block-wise transfer functionality is important.

[1] https://github.com/kmackay/micro-ecc.

[2] https://github.com/cabo/cn-cbor.

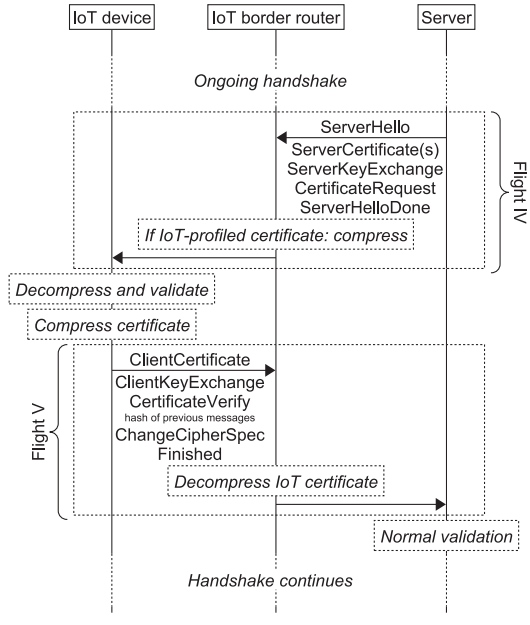


Fig. 10. DTLS handshake with certificate compression and decompression.

The upcoming standard compliant parts of EST-CoAP have been interop tested with two server implementations, from partners and external IETF members.

7. PKI4IoT: Evaluation

The evaluation is done to demonstrate both the feasibility of the proposed integrated design, and the resulting savings in terms of communication and energy usage overhead. The implementation, which handles both regular X.509 certificates and the proposed profiled and compressed version, is tested in both single and multip-hop scenarios.

7.1. Experimental setup

For the experiments on ARM Cortex M3-based CC2538 devices, Zolertia Fireflies have been used. The relevant characteristics are as follows: a current draw of 7.12 mA at 16 MHz clock speed without radio activity, and 20 mA or 24 mA with radio in receive or transmit mode, respectively. The platform also provides hardware support for AES and ECC cryptographic operations. Complete specifications can be found on the Firefly wiki and in the CC2538 datasheet (Instruments, 2015; Zolertia, 2018).

For all scenarios, one Firefly node is directly connected to a host computer, serving as a low-power radio network border router. The device is only forwarding the packets to a Linux process running on the host computer, and hence neither memory nor processing power is limited for the border router. The border router can compress and decompress certificates on the fly without significant delays. The main test enrollment server is running on the host computer. Tests with remote servers are also conducted. These networking delays are negligible compared with the low power radio network delays.

The DTLS communication has been tested both in single-hop and multi-hop scenarios. In all scenarios a Firefly acting as a client endpoint contacts the test server through the border router. For the multi-hop scenarios another Firefly node acts as an intermediate router, preventing any direct access between the border router and the end client.

7.2. Micro benchmarks

7.2.1. HTTPs and CoAPs header sizes

CoAP is designed for minimal overhead, and when doing enrollment, the CoAP overhead is very predictable. The requests contain a server path. This could be as small as 5 bytes for the cacerts request, */crts*, and 4 for the simple enroll request, */sen*. These are the settings we suggest to use, but the standard also allows longer hierarchical paths.

In contrast, EST over HTTP offers a wide range of possible configuration options, each with their own set of parameters. We have compared the HTTP header overhead from the samples given in the standard (Pritikin et al., 2013) with samples from a test server, and determined expected minimal bounds. Table 2 shows the results for the comparison of the protocol overhead when doing enrollment over CoAP and corresponding operations over HTTP. These are the values before encryption, which in turn adds computational overhead as well as minor (D)TLS headers. While the overhead caused by the HTTP headers are acceptable for powered devices with high capacity network connections, for con-

Table 2
Header sizes for EST calls over HTTP and CoAP.

Message		Header size (B)	
		CoAP	*HTTP*
cacerts	request	6–26	> 160
	response	3	> 118
enroll	request	8–28	> 235
	response	3	> 125

Table 3
Typical certificate size ranges.

Crypto	Size
RSA 2048	0.8kb–3kb
ECC no explicit profile	296–2000
ECC profiled	296–350
ECC compressed	140–160

strained devices and lossy networks the CoAP savings could be crucial.

7.2.2. Certificate sizes

Table 3 presents expected certificate sizes for the recommended RSA and ECC key sizes. These size values are for certificates in binary representation. In base64 encoding they would be 33% bigger. RSA-based certificates with the recommended 2048-bit keys and sha256 signatures have a lower size bound around 800 bytes, where only common name is included, and minimal extensions. Due to the usage of extensions, no upper bound can be determined, but commonly-found web certificates range between 1.5 kB and 3 kB. In contrast, a profiled ECC-based certificate is less than 50% of the minimal RSA certificate, and the compression reduces the size with another 50%.

It is important to note that these reductions are based on the domain-specific knowledge on which fields can be dropped and recreated, as seen in 6. Using gzip/deflate on certificates can reduce non-profiled certificates up to 30%, but it cannot further reduce the size of already-profiled certificates. (Figs. 1 and 11)

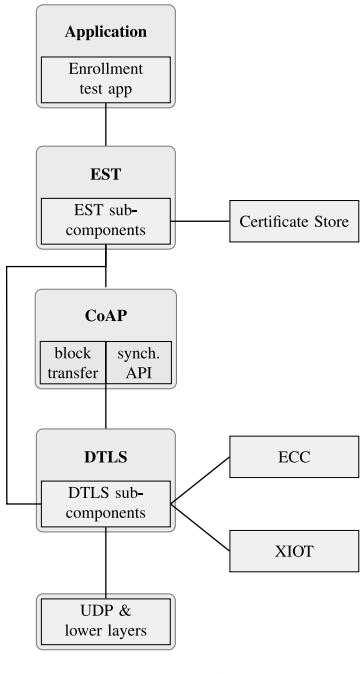


Fig. 11. The constrained device software stack.

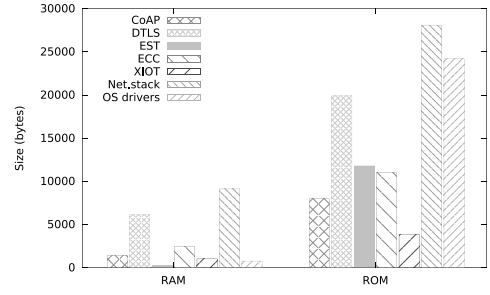


Fig. 12. Comparison of IoT client software module memory usage.

7.2.3. Memory overhead

The current lightweight certificate implementation adds 3.7 kB of ROM and 1.1 kB of RAM, shown as the XIOT component in the diagram. Seen in comparison to the memory footprint of other protocols, the ip stack and other operating system modules in Fig. 12 the memory footprint of the added XIOT component is relatively small. Some details are worth noting: The memory requirements for the device netstack will vary depending on the device usage requirements, such as the need for concurrent connections and number of neighbours. Specifically, if the resource-heavy handshake operation sets the bounds for the needed radio queues and resend buffers, the introduction of the compression mechanism could allow for shorter queues. These savings are in the range of a few hundred bytes up to 1.5 kB of RAM, which more than offsets the RAM memory usage of the XIOT component.

7.3. Energy consumption

One of the main limiting factors for battery powered devices is energy usage. Knowing, understanding and limiting the energy usage for any new functionality is crucial if the mechanism should be possible to use in real deployments. Below we show how reduced communication needs corresponds to reduced energy needs.

7.3.1. Energy consumption modeling

Using the Energest timer system in Contiki we measure the time spent for relevant protocol stages and active peripheral. This allows us to calculate the consumption based on current and voltage levels from the CC2538 hardware datasheets (Instruments, 2015).

Listening for radio data is almost as expensive as sending. Hence, it is beneficial to minimize the total data traffic, not just the amount of data the sensor needs to send. To benefit from the traffic reductions, an efficient radio duty cycling protocol is needed.

It is worth observing that since CPU usage is less than the energy cost of radio usage, doing some more local computation to reduce the data transfer can be beneficial.

Based on the discussion on certificate chains in 5.3.2, we consider two main scenarios. In one general case, a single root certificate pre-installed in the client trust store is not enough to verify the server certificate. Thus, the server needs to send a chain of certificates as part of the Server Certificate message in the handshake procedure in order to establish the chain of trust. In one typical IoT case, the node has already obtained the necessary certificates to verify the signature of the server certificate. In the latter case, the Server Certificate message only needs to contain the missing server certificate.

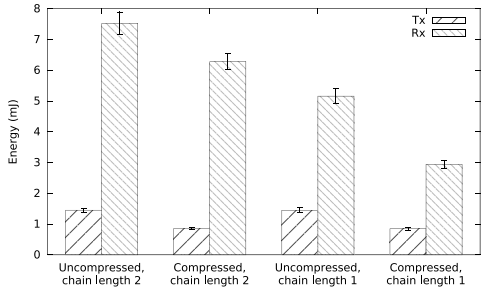


Fig. 13. Comparison of energy consumption for certificate message transmissions.

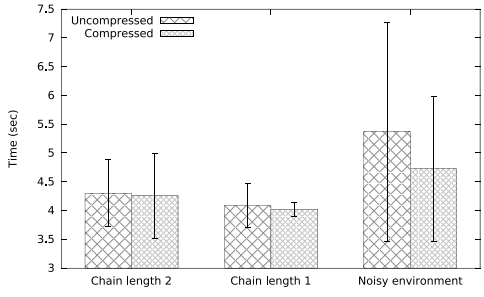


Fig. 14. DTLS handshake completion time in a Zolertia Firefly IoT device with the Contiki.

7.3.2. Energy measurement results

In the experiment, we measure and calculate the consumption for the parts of the handshake procedure corresponding to certificate exchanges. Each experiment is repeated at least 200 times. The detailed energy consumption results are shown in Fig. 13. As expected, the highest energy consumption is found for the case where a certificate chain of length 2 is sent. Here, the communication cost is 9 mJ. Just compression reduces the consumption with 20%. The case where the certificate chain can be shortened, and only the server certificate is needed, results in a 26% reduction. The best results are achieved when combining the methods, resulting in a 58% reduction. These results are focused on the client end node. Taking also the energy consumption of the relaying nodes into account in a multi hop network would yield higher total transmission savings, but the results would look very different between different radio configurations and environments.

7.4. Round trip times

Round trip times are standard measurements used to determine the usability of a system. Servers need to adjust their timeout delays based on expected transmission times. With a timeout shorter than is necessary, a constrained device might not be able to complete the DTLS handshake. On the other hand, long timeouts occupy server resources and limits the number of connections that can be handled in a time period. For the DTLS handshake in a clean radio environment, the local node cryptographic processing is the most time-consuming activity. As can be seen in Fig. 14, the certificate compression only very marginally improves the expected handshake time for our two-hop test setup. The same lab test setup running during office hours with more neighbouring WiFi traffic shows different numbers, where more frequent radio retransmissions make the version with compressed certificates come out ahead, with average savings of 12%. The figures illustrate that for IoT scenarios in lossy radio environments, small savings get amplified.

8. PKI4IoT: Security considerations

The major components of the proposed PKI are based on existing standards, which have been subject to detailed security analyses. Here we highlight the relevant findings, analyze missing parts, and assess if the new proposed solutions risk exposure new vulnerabilities.

A security solution may have vulnerabilities both in the protocol design and the implementation. We do not claim to address all potential code vulnerabilities associated with the embedded operating system and its parts; such efforts are ongoing in parallel by related researchers and Contiki developers. We focus here on the protocol vulnerabilities, and the risks introduced by new proposed components.

Protocol risks. DTLS is specifically designed to offer security for CoAP-based communication, suitable for resource-constrained devices. While the presently-used DTLS version 1.2 does not explicitly forbid the known insecure MD5 hashing algorithm, later recommendations and those specifically the presented XIOT profile do not allow MD5 or SHA1 to be used. The choice of using elliptic curve cryptography is motivated by the lower overhead compared with RSA. For long term usage, one should consider the risks of current cryptho methods rendered obsolete by advances in quantum computing. A study done at Microsoft Research show that ECC algorithms might be more susceptible to quantum computing attacks compared with RSA, but any such attack on the relevant recommended key lengths is still far out of reach (Roetteler et al., 2017).

Long lived CA certificates. The EST over CoAP protocol inherits the characteristics of the plain EST protocol that was designed for HTTP-based communication, which include security weaknesses (Pritikin et al., 2013; van der Stok et al., 2019). Some vulnerabilities, such as plaintext exposure of client passwords through Basic authentication, are made obsolete by the draft standard (van der Stok et al., 2019). One remaining issue is the usage of an implicit truststore, which in our scenarios is pre-installed in the node at the manufacturer premises. Until there are accepted revocation mechanisms in place that can be used for IoT devices, the potentially long-lived CA certificates in the implicit truststore need to be trusted by the node for as long as the are kept in the store. A risk reduction strategy is to overwrite the pre-installed CA certificates as part of the enrollment process. This reduces the risk of rouge CAs, but introduces new complexity if the node is resold and needs to be reset to factory settings.

The border router. In the case of DTLS 1.2 certificates, compression/decompression during the handshake can take place at a border router at the edge of 6LoWPAN network. It is important to observe that this does not require the border router to be a trusted entity, since the communication is verified end-to-end between client and server. The border router may waste node resources by modifying messages, causing a handshake to fail, but the malicious behavior of a 6LoWPAN border router is easily detectable by intrusion detection systems.

Initial device time. An ongoing design challenge is how to reliably get the initial time when a resource-constrained device is bootstrapping. This is partially addressed by the proposed BRSKI draft (Pritikin et al., 2019). There, it is noted that a device doing bootstrapping will need to be flexible with the certificate lifetime (i.e., the “not before” and “not after” times) of the certificate presented to it, since no secure time has been established. By intro-

ducing the concept of "current reasonable date" (CRD), a device which has access to the compile time of its firmware can establish a lower bound for the "not before" date.

The corresponding time-related issue on the server side is that a server responding to a bootstrap request can accept a special value, "99991231235959Z" by the X.509 standard, for the "notAfter" field of the presented factory certificate, which can be used to indicate that a certificate has no well-defined expiration date (Cooper et al., 2008).

The current test implementation uses a trusted in-network server as time source, leaving a more refined procedure for future work.

New functionality. The new component for certificate encoding and decoding utilizes a CBOR parser and ECC library functions, and is not expected to add further vulnerabilities. The compact encoding format, CBOR, is designed to be easily parsed on constrained devices. A simple parser reduces the attack surface. A strict decoding mode is necessary to distinctly decode the certificate at hand (Bormann and Hoffman, 2013). A rogue server could send faultily-compressed certificates, which would make the node spend energy trying to decompress it, but this wasted effort is less than the full energy cost of validating a certificate with correct format but invalid signature. Hence this new attack is less probable than already existing possibilities for attacks on the *availability* security service.

IoT devices as attackers. The introduction of billions of IoT devices opens up possibilities for new types of DDoS attacks, with IoT nodes as attackers. PKI4IoTis a major step in bringing strong security to IoT and preventing nodes from becoming compromised. However, these devices may still be physically cloned and hacked. A new and improved firewall compatible with IoT protocols may still be needed to mitigate attacks to the global Internet from IoT devices and vice versa. In addition, servers should employ basic protection strategies such as a back-off after potentially malicious failed connection attempts in order to limit the impact of an active attacker.

9. Conclusion

In this paper we have presented challenges for enabling PKI for IoT, and new important PKI building blocks as answers to two of those challenges: secure enrollment and certificate overhead reduction. We have shown that they are capable of successfully performing their tasks, securely performing initial enrollment as well as re-enrollment, and reducing the X.509 overhead for the target IoT scenarios. These contributions are bringing functional full-fledged PKI closer to real IoT deployments. For maximum impact and interoperability across different vendors we are pushing both enrollment and lightweight certificates as standards in IETF, where the enrollment protocol draft is close to being accepted as an official RFC.

To make the PKI more complete, new solutions for certificate revocation and status checking are being designed, and more detailed chain of trust scenarios for IoT devices in vehicle communication and health care domains are being investigated.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research has partly been funded by The Swedish Foundation for Strategic Research (SSF), Vinnova, Formas and Energimyndigheten under the Strategic Innovation Program for IoT (SIP-IoT) SecureCare project; partly by the H2020 ECSEL SECREDAS (grant ID: 783119) project; and partly by the H2020 CONCORDIA (Grant ID: 830927).

References

Adams, C., Farrell, S., Kause, T., Mononen, T., 2005. Internet x.509 Public Key Infrastructure Certificate Management Protocol (cmp). RFC 4210, RFC editor.

Bormann, C., Hoffman, P., 2013. Concise Binary Object Representation (cbor). RFC 7049, RFC Editor.

Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., Polk, W., 2008. Internet x.509 public key infrastructure certificate and certificate revocation list (crl) profile. RFC 5280, RFC Editor.

Dierks, T., Rescorla, E., 2008. The Transport Layer Security (tls) Protocol version 1.2. RFC 5246, RFC Editor.

DigiCert, 2017. PKI: The security solution for the internet of things. Tech rep, DigiCert Inc.

Emery D. my friend cayla doll records childrens speech, is vulnerable to hackers. 2017. https://www.snopes.com/news/2017/02/24/my-friend-cayla-doll-privacy-concerns/.

Ford, W., Poeluev, Y., 2015. The machine-to-machine (m2m) public key certificate format. Internet-Draft draft-ford-m2mcertificate-00, IETF Secretariat. http://www.ietf.org/internet-drafts/draft-ford-m2mcertificate-00.txt.

Forsby F. Digital certificates for the internet of things. Master's thesis, KTH, Network and Systems engineering 2017. http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-217120.

Ghedini, A., Vasiliev, V., 2018. Tls certificate compression. Internet-Draft draft-ietf-tls-certificate-compression-04, IETF Secretariat. http://www.ietf.org/internet-drafts/draft-ietf-tls-certificate-compression-04.txt.

Gutmann, P., 2019. Simple certificate enrolment protocol. Internet-Draft draft-gutmann-scep-14, Internet Engineering Task Force, work in Progress. https://datatracker.ietf.org/doc/html/draft-gutmann-scep-14.

He, Z., Furuhed, M., Raza, S., 2019. Indraj: Certificate Enrollment for Battery-powered Wireless Devices. In: Proceedings of the 12th ACM Conference on Security and Privacy in Wireless and Mobile Networks. ACM.

Ieee. 1609.2–2016 - ieee standard for wireless access in vehicular environments. 2016. https://standards.ieee.org/standard/1609_2-2016.html.

Instruments, T., 2015. CC2538 Powerful wireless microcontroller system-on-chip for 2.4-GHz IEEE 802.15.4. 6LoWPAN, and ZigBee Applications. http://www.ti.com/product/CC2538#.

Jivsov, A., 2014. Compact representation of an elliptic curve point. Internet-Draft draft-jivsov-ecc-compact-05, IETF Secretariat. http://www.ietf.org/internet-drafts/draft-jivsov-ecc-compact-05.txt.

Managing. the internet of things (iot) device authentication life cycle. https://www.deviceauthority.com/assets/INTEL_Device-Authority_Solution-Brief.pdf.

Park, C., 2017. A secure and efficient ecqv implicit certificate issuance protocol for the internet of things applications. IEEE Sensors J. 17 (7), 2215–2223. doi:10.1109/JSEN.2016.2625821.

Pritikin, M., Richardson, M., Behringer, M., Bjarnason, S., Watsen, K., 2019. Bootstrapping remote secure key infrastructures (brski). Internet-Draft draft-ietf-anima-bootstrapping-keyinfra-18, IETF Secretariat. http://www.ietf.org/internet-drafts/draft-ietf-anima-bootstrapping-keyinfra-18.txt.

Pritikin, M., Yee, P., Harkins, D., 2013. Enrollment Over Secure Transport. RFC 7030, RFC Editor.

Raza, S., Helgason, T., Papadimitratos, P., Voigt, T., 2017. Securesense: end-to-end secure communication architecture for the cloud-connected internet of things. Future Gener. Comput. Syst. 77, 40–51. doi:10.1016/j.future.2017.06.008.

Raza, S., Seitz, L., Sitenkov, D., Selander, G., 2016. S3k: Scalable security with symmetric keys - DTLS key establishment for the internet of things. IEEE Trans. Autom. Sci. Eng. 13 (3), 1270–1280. doi:10.1109/TASE.2015.2511301.

Research, C., 2016. Public key infrastructure for the internet of things. Tech rep, Certified Security Solutions, Inc.

Roetteler M., Naehrig M., Svore K., Lauter K.. Quantum resource estimates for computing elliptic curve discrete logarithms. 2017.

Sarikaya, B., Sethi, M., Garcia-Carillo, D., 2018. Secure iot bootstrapping: a survey. Internet-Draft draft-sarikaya-t2trg-sbootstrapping-05, IETF Secretariat. http://www.ietf.org/internet-drafts/draft-sarikaya-t2trg-sbootstrapping-05.txt.

Schaad, J., Myers, M., 2008. Certificate Management Over cms (cmc). RFC 5272, RFC editor.

van der Stok, P., Kampanakis, P., Richardson, M., Raza, S., 2019. Est over secure coap (est-coaps). Internet-Draft draft-ietf-ace-coap-est-10, IETF Secretariat. http://www.ietf.org/internet-drafts/draft-ietf-ace-coap-est-10.txt.

Ting, P., Tsai, J., Wu, T., 2018. Signcryption method suitable for low-power iot devices in a wireless sensor network. IEEE Syst. J. 12 (3), 2385–2394. doi:10.1109/JSYST.2017.2730580.

Tschofenig, H., Fossati, T., 2016. Transport Layer Security (tls) / Datagram Transport Layer Security (dtls) Profiles for the Internet of Things. RFC 7925, RFC Editor.

Zolertia S.L. Zolertia firefly platform. 2018. https://github.com/Zolertia/Resources/wiki/Firefly.

Paper II

2021 IEEE Conference on Communications and Network Security (CNS)

LICE: Lightweight certificate enrollment for IoT using application layer security

1st Joel Höglund
RISE Research Institutes of Sweden
Stockholm, Sweden
joel.hoglund@ri.se

2nd Shahid Raza
RISE Research Institutes of Sweden
Stockholm, Sweden
shahid.raza@ri.se

***Abstract*—To bring Internet-grade security to billions of IoT devices and make them first-class Internet citizens, IoT devices must move away from pre-shared keys to digital certificates. Public Key Infrastructure, PKI, the digital certificate management solution on the Internet, is inevitable to bring certificate-based security to IoT. Recent research efforts has shown the feasibility of PKI for IoT using Internet security protocols. New and proposed standards enable IoT devices to implement more lightweight solutions for application layer security, offering real end-to-end security also in the presence of proxies.**

In this paper we present LICE, an application layer enrollment protocol for IoT, an important missing piece before certificate-based security can be used with new IoT standards such as OSCORE and EDHOC. Using LICE, enrollment operations can complete by consuming less than 800 bytes of data, less than a third of the corresponding operations using state-of-art EST-coaps over DTLS. To show the feasibility of our solution, we implement and evaluate the protocol on real IoT hardware in a lossy low-power radio network environment.

***Index Terms*—Public Key Infrastructure, PKI, IoT, digital certificate, enrollment, CBOR, OSCORE, EDHOC, embedded systems**

I. Introduction

The Internet of Things continues to grow rapidly, making fully automated IoT security one of the most important challenges to prevent serious threats to the whole Internet infrastructure. New IoT devices are becoming more capable of performing cryptographic operations, but the perceived overhead and lack of standards have been obstacles to move beyond pre-shared key solutions and create public key infrastructure (PKI) solutions that include IoT. Using OSCORE, IoT devices can communicate securely in a standardized manner using application layer security, which allows messages to traverse proxies to offer true end-to-end security [1]. OSCORE together with EDHOC [2] for key establishment have the potential to offer lightweight solutions for establishing secure sessions.

a) Problem: Application layer security solutions that support digital certificates are being standardized but are not practical to use unless PKI support for these standards is available. The most important building block for a PKI is the functionality to do enrollment. For the enrollment to scale to billions of devices, it needs to be fully automated, requiring no out-of-band operations, and lightweight to meet IoT constraints. Existing solutions can perform certificate enrollment using CoAP over DTLS [3], [4], but these solutions are not adapted to be used with the new application layer security protocols. Besides true end-to-end security, application layer security can offer lightweight sessions where different applications use different credentials. In theory, services in a constrained IoT device could use certificates to setup multiple DTLS-sessions with different endpoints. In practice, DTLS-sessions are costly in terms of memory, and IoT devices can rarely afford more than a single DTLS-session at once.

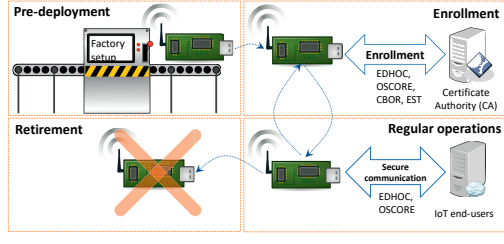


Fig. 1. An IoT device life cycle: factory setup, initial enrollment, normal operation/communication mode, back to re-enrollment or retirement. The focus of this paper is the enrollment phase.

OSCORE does not exclude the usage of pre-shared keys, which can offer the smallest footprint for the most constrained devices. But for the growing number of more capable IoT devices, real PKI solutions are needed for them to become first-class Internet citizens.

b) Challenges: IoT devices are resource constrained compared with computers on the Internet. If they are battery powered, every byte needed to be sent over radio counts towards the total energy budget. To be interoperable with mainstream Internet devices and services, they need to use standard based communication mechanisms. Existing protocols designed for less constrained devices have lengthy encoding formats, creating substantial overhead both in terms of memory and communication.

c) Contributions: This paper provides a lightweight enrollment protocol for IoT, based on optimizing EST-coaps [3] for usage with OSCORE and EDHOC. Through our proposed enrollment protocol we provide an important building block towards enabling PKI for IoT. Without LICE devices would

978-1-6654-4496-5/21/$31.00 ©2021 IEEE

need to support DTLS and OSCORE in parallel, completely removing the advantages of moving to a more lightweight set of communication protocols. Utilizing application layer security solutions, the enrollment protocol follows upcoming standards and achieves real end-to-end security. Through efficient secure session establishment, together with CBOR [5] encoding for EST enrollment operations, we bring the enrollment data exchange down to 800 bytes of data. This is less than a third of the data being transferred compared with existing EST-coaps solutions.

The core contributions of the paper are as follows:

- We provide LICE, a lightweight and secure enrollment protocol exploiting the novel OSCORE and EDHOC protocols that can traverse proxies without breaking end-to-end security.
- We propose CBOR-encoding for the LICE EST operations necessary for application layer enrollment.
- We provide an implementation for resource constrained devices.
- We evaluate and compare LICE with previous state of art solutions for IoT enrollment, EST-coaps over DTLS.

The rest of this paper is organized as follows: related work is presented in section II. Section III gives a brief presentation of relevant background. Section IV introduces the assumed threat model. Section V presents the enrollment protocol design and VI details the proposed CBOR encodings. The details of the implementation and evaluation are provided in sections VII and VIII respectively. Section IX discusses further security considerations and section X concludes the paper.

II. Related Work

Within the field of IoT security, benefits and challenges with certificate based authentication and related PKI requirements have been discussed for a long time [6]–[9]. The area has gained considerable industry attention, but mainly addressing devices powerful enough to run standard Internet protocols [10], [11]. An IoT security overview from 2018 highlights the challenges with resource constraints related to PKI and the immaturity of existing standards [12].

There are two application areas where the need for automated and efficient certificate management have been highlighted: E-health and the automotive industry. These are both areas where privacy and safety concerns are paramount [13], as well as strict standard compliance for both legal and safety reasons [14], [15]. Results from the automotive scenarios are not easily transferable to the general IoT domain, but they highlight the requirements of standard based interoperability and resource efficiency.

A conclusion is that there is significant interest in PKI solutions for constrained devices, but that standardized lightweight key management is mainly an unsolved issue.

Existing standards for certificate enrollment have been designed for non-constrained devices. SCEP, CMC, CMP, and the original EST protocol belong to this category [16]–[19]. EST over secure CoAP (EST-coaps) provides relatively lightweight enrollment operations using DTLS over UDP [3]. EST-coaps has been evaluated in [20] and [4]. In [4] some steps are taken to reduce the verbose ASN.1 encoding format, but the more compact encoding is only used inside a low power radio network, transparent to end servers on the wired Internet. The solution proposed in this paper is a missing piece before the new application layer security protocols can be scaled to billions of IoT devices using certificate-based cryptography.

III. Necessary Technology Background

This section provides a quick overview of the technologies necessary to understand the paper.

A. EST

The original Enrollment over Secure Transport protocol specifies how to use Certificate Management over CMS messages for enrollment, using HTTPS [19]. The required minimum functionality of EST is to allow the secure transfer of: CA certificates needed for the client trust anchor database (also called trust store), certificate enrollment requests and the resulting enrolled certificate. The proposed EST-coaps is defined to use DTLS over UDP [3]. It replaces the base64 message encoding used in original EST with binary message encodings.

B. CBOR

The Concise Binary Object Representation is a data format designed with the explicit goals to be compact and possible to encode and decode with low resource overhead [5]. Besides basic data types, maps and arrays it supports binary byte strings, making it possible to use as wrapper for any binary data. Pure CBOR encoded data should be self-describing, possible to decode without a schema.

C. OSCORE

Object Security for Constrained RESTful Environments is a newly standardized application-layer protocol for protection of CoAP messages [1]. It uses COSE [21] and CBOR functionality for encryption and encoding. Besides end-to-end encryption it provides integrity and replay protection. It is designed handle CoAP-mappable HTTP, hence enabling CoAP devices to communicate securely over proxies also with HTTP endpoints. OSCORE does not define key establishment, which needs to be provided by additional mechanisms.

D. EDHOC

Ephemeral Diffie-Hellman Over COSE is a proposed key exchange protocol, designed to be useful for constrained scenarios [2]. Like OSCORE it builds upon COSE cryptography and uses CBOR encoding. Besides mutual authentication it provides identity protection and perfect forward secrecy. Establishing the key material needed for OSCORE is one possible use case, but the protocol can be used on its own for other mutual authentication scenarios. In its minimal form it only needs three messages to perform mutual authentication and establish shared key material. In addition, the handshake messages can carry auxiliary data (presented as `AD_1-3` from the proposed

standard draft), with some restrictions regarding the offered protection. An extra fourth message can be added to carry auxiliary data, removing the need to setup a full OSCORE session if no further data exchange is needed.

IV. Threat model

We base our threat model for the IoT enrollment communication scenarios on the Dolev-Yao threat model. By the model it is assumed that any communication between the involved entities can be eavesdropped by an attacker, which is capable of modifying and re-sending any message. This means the system must be able to withstand replay attacks and still offer authentication and confidentiality services, preventing unauthorized access to any secret content. On the other hand, it is assumed that nothing about a plaintext message can be learned from the ciphertext for an attacker without access to the keying material (within the considered time frames).

Trustbase and limitations: We assume there is at least one accessible Certificate Authority (CA), which can be reached by the IoT device and that they can perform mutual authentication.

For the involved hash and crypto algorithms and their key lengths, we assume that the NIST recommendations will be valid for the relevant future. We also assume that the existing and proposed standards used are not compromised.

This work does not address potential backdoors in the IoT devices, and considers the software stack and hardware to be trusted. Research on software security, trusted execution environments (such as TrustZone) and secure storage are complementary to this work.

V. LICE: Application Layer Enrollment Protocol

This section details the LICE protocol and presents different design options for different usage scenarios.

A. Involved entities

The following entities are relevant for the scenarios the design needs to cater for:

Factory Certificate Authority (Factory CA). The entity that issues the manufacturer-installed certificate for the IoT device.

Enrollment Certificate Authority (CA). The entity an IoT device contacts to perform enrollment or re-enrollment. This CA needs to be able to verify the certificates issued by the factory CA.

The IoT device. In the scenarios we consider, the IoT device is deployed with a pre-installed certificate from the factory CA, the knowledge of which enrollment CA to contact, and an initial trust store to verify the identity of the enrollment CA.

B. Protocol requirements

The most fundamental enrollment protocol operation consists of a certificate signing request (CSR), and a reply with a newly issued certificate. The certificate is a token of trust needed to become part of a public key infrastructure. Before issuing a certificate the recipient of the enrollment request, the trusted CA, must be able to verify the binding between a privately kept secret key (Private Key) and the public counterpart present in the certificate signing request (Public Key).

To update or replace its factory installed trust store, the IoT device needs additional certificates. Hence an operation for secure transfer of certificates is required.

Another basic operation is the possibility to renew a certificate that is about to expire, an operation which is close to identical with the initial enrollment operation.

To perform these operations the enrollment protocol requires a secure channel, where the involved parties are mutually authenticated before any sensitive data is exchanged. This in turn requires a key exchange for secure session establishment. Both the key exchange phase and any subsequent messages must be transported such that replay-attacks can be detected and prevented.

A full-fledged PKI will need to address issues of certificate revocation, which is not addressed in this paper. It is however worth noticing that certificates with sufficiently short validity time can be seen as an alternative to active revocation, especially for resource-constrained IoT devices. Until a revocation mechanism is in place, and instead short certificate lifetimes are used to compensate, it becomes extra important to make the enrollment operations lightweight.

C. Choice and motivation of protocol building blocks

We have chosen the EST enrollment protocol as a starting point. It has proven to be adaptable for usage with DTLS in resource constrained environments, such as the low-power and lossy networks we primarily target, while providing the required basic enrollment operations [4]. When the new enrollment protocol is designed using the latest crypto building blocks, using application layer security and compact CBOR encoding schemes, it has the potential to become more lightweight compared with older DTLS solutions using ASN.1 encoding.

OSCORE can be used for the secure channel, complemented with EDHOC for the key establishment. Both OSCORE and EDHOC contain mechanisms to counter replay-attacks through sender sequence numbers and transcript hashes, respectively.

From the security of the underlying SIGMA schema it follows that as long as the included components keep their security guarantees, the resulting protocol will provide the desired security services. The SIGMA-I variant used as the basis for EDHOC has the additional benefit of identity protection. The identity of the initiator is protected from active attacks, while the responder has passive attack protection [22]. For IoT scenarios, the constrained and possibly mobile devices are generally more vulnerable than a server, hence assigning the role of initiator to the IoT device is the default choice.

For devices prepared to run OSCORE, cryptographic functionality usable for EDHOC and libraries for CBOR are already available, lowering the added overhead.

Based on these requirements and building blocks we propose a lightweight enrollment solution capable of end-to-end security, including Internet proxy traversal. The following sections presents two different design options.

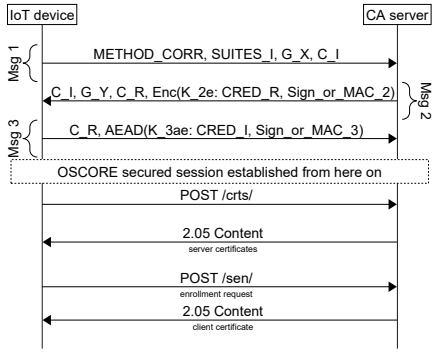


Fig. 2. *Version I:* Enrollment *over* OSCORE, protected with EDHOC secured session.

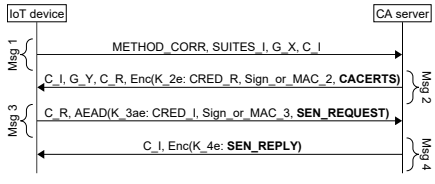


Fig. 3. *Version II:* Enrollment embedded in the EDHOC protocol, the version LICE proposes to use. The auxiliary data fields are used to carry the enrollment messages.

D. Enrollment over EDHOC-OSCORE

A basic approach using application layer security is a design where EDHOC is executed without any modifications. Once the secure session is established between an IoT device and a CA, the enrollment operations are performed. This approach has the benefit of being agnostic to the key establishment method, and does not risk altering the security guarantees offered by the key establishment method used. Only three messages are needed for the devices to mutually authenticate, and calculate a shared secret which can be used to create further key material needed for OSCORE. This solution requires that both OSCORE and EDHOC are implemented and present in the IoT device being enrolled.

Running EDHOC to completion and continuing with OSCORE can be seen as the equivalence of performing a DTLS handshake, and thereafter protecting EST messages with the DTLS Record protocol.

The scenario is shown in Fig. 2. It consists of a total of four round trips, since also message three will be acknowledged with an empty reply.

E. Enrollment using EDHOC

A more optimized alternative compared to the above approach is to reuse the key establishment messages to carry EST payload. The proposed EDHOC protocol has hooks for an application to provide auxiliary data which can be used to carry EST messages in a standard compliant manner. Combining EDHOC with enrollment has the potential to further reduce the number of messages, and the total bytes sent. Care must be taken to ensure that the protocol security guarantees are not jeopardized. Specifically, data sent as part of an EDHOC exchange cannot be considered protected until both parties have authenticated each other. (The shared secret is computed, and used, before the mutual authentication is finished.) In addition it creates a binding to the key establishment protocol by the requirement to handle application data.

The compact alternative, pushed by LICE for IoT, proposes to use the first EDHOC message as the `cacerts` request. The `AD_2` in message two is used to carry the `cacerts` reply. `AD_3` in the third message is used for the enrollment request. If the request is accepted, `AD_4` in the added fourth message carries the CA reply with the newly enrolled certificate information. The message exchange scenario is shown in Fig. 3.

One of the main advantages of using the proposed version II solution is to avoid strong bindings between EDHOC and OSCORE. While EDHOC can be used by OSCORE for key establishment, it is a standalone protocol and can be used by other security protocols. In those cases when EDHOC is used for other purposes, having the enrollment protocol bound with the combined usage of OSCORE and EDHOC will require a full-fledged implementation of OSCORE only for the purpose of enrollment.

1) Security considerations when sending auxiliary data: The `cacerts` request does not carry any data of its own, and is identified by the target URL alone. Hence the only information leaked to an outside observer is the fact that the client is initiating contact with an EDHOC endpoint. Using `AD_2` in message two to carry root certificates to update the client trust store means they will be sent before the CA has finished authenticating the client. The certificates in the message are meant to be publicly available, and handling the request is computationally lightweight, hence the risk for the CA to handle the request is small. For the client it is vital to finish authenticating the CA through the complete processing of message two before adding the received certificates to its trust store. Any data sent by the IoT device as `AD_3` in message three should only be processed by the server after it has finished the authentication by verifying the included signature.

F. Choice of crypto keys

Early EDHOC proposals included the capabilities to be used with either pre-shared keys or asymmetric keys such as X.509 certificates. Pre-shared keys do not scale well and causes security vulnerabilities. If a server containing the pre-shared key of several IoT devices is hacked the entire IoT deployment is compromised. Hence LICE is designed for using EDHOC with certificates. The first secure session establishment is carried out by factory installed certificates. As a result of using certificate based authentication, the `Sign_or_MAC`-fields in the diagrams (Fig. 2 and 3) will be used to carry signatures.

Further design choices are whether to send full certificates as part of the handshake or to only send references, carried in the CRED field in Fig. 3. If the reference format is chosen, the EDHOC proposed format is to include a CBOR encapsulated hash of the full certificate. Sending only a reference can greatly reduce the amount of data to transfer, but the procedure requires the parties to already have the referenced certificates stored locally. For many IoT scenarios where the devices are given pre-installed factory certificates to use for authenticating a CA for initial enrollment, this is a reasonable assumption. LICE proposes the usage of certificate hashes, but allows full certificates for deployments where the solution using hashes only is unavailable.

For IoT devices, elliptic curve cryptography (ECC) based solutions have become a de facto standard. ECC can offer strong cryptographic guarantees already at shorter key lengths compared with RSA and results in less overhead both in terms of memory and computation. LICE uses ECC, primarily with the NIST P-256 curve, which is well supported among existing crypto implementations, and is seen as sufficiently secure for the foreseeable future.

G. Layering design considerations

1) Enrollment layer and transfer using CoAP: For the design *Version I* the same URI paths which are proposed for EST-coaps are used, `/crts` for the `cacerts` operation and `/sen` for the `simpleenroll` operation. In the same manner the CoAP response codes are reused. To indicate a successful operation, the content response code *2.05* is used for `cacerts` and the created response code *2.01* is used for `simpleenroll`. To indicate errors, the applicable *4.xx* or *5.xx* codes are used.

For the compact design, *Version II*, the server offers an alternative endpoint, `/edhoc_est`. Since the CoAP response codes are (partially) used by the EDHOC protocol, any enrollment error code from the server is placed in the auxiliary data field. Any single byte reply is interpreted as an error code, whereas any longer field is interpreted as the payload of a successful operation. By this design the core of the EST application is oblivious to the protocol version used, and can easily be used for either.

For all protocol variants CoAP confirmable messages are used to increase reliability when transporting protocol data. This is in line with the recommendations for EDHOC.

2) UDP, IPv6 and 6LoWPAN: For IoT deployments in wireless sensor networks, the expected lower layers of the stack will consist of UDP, IPv6 and 6LoWPAN. For the primarily targeted low power 802.15.4 radio networks, the expected total frame size is only 127 bytes. While 6LoWPAN can handle larger network layer packets through fragmentation, this causes larger overhead and potential security weaknesses compared with if the fragmentation is done by the higher layers. LICE is designed to support block-wise transfer, which enables the CoAP layer to handle the fragmentation of large packets.

VI. LICE: CBOR ENCODINGS OF EST MESSAGES

One of the main objections to using PKI solutions for IoT is that the overhead is prohibitive for constrained devices. Since CBOR is gaining widespread usage as an encoding standard it is an acceptable requirement that CA:s which target the growing IoT market should be capable of handling CBOR encoded messages and certificates. Based on this development, LICE proposes to use CBOR encodings of EST messages, extending size reductions previously shown for certificates to also include enrollment message payloads.

A. Encoding for distribution of CA certificates

The `cacerts` *request* itself has no payload, and hence no overhead to remove. In EST, the ASN.1 encoding of the PKCS#7 formatted *reply* comes with a large header and footer. If CBOR is used, a chain of certificates can be encoded as a CBOR array, with a total of max three bytes overhead per certificate. This array construction can be used even if the certificates themselves are kept in ASN.1 format, although CBOR encoding of the certificates reduces the size further [23].

```
CertificateChain = [
  +CBORCertificate / ASN.1_Certificate
]
```

B. Encoding for enrollment requests

In EST the certificate signing request is formatted according to the PKCS#10 format, which is a format with flexibility and extensive options. For the IoT scenarios we consider, the information a device needs to send to the CA server as part of the enrollment operations is very limited. The device needs to present its subject information, the public part of the key it wishes to enroll, the key type and a request signature as proof of possession of the secret key. For IoT devices we propose using an EUI-64 based on the device MAC address as the subject information. The decisions on which key usage rights to include in the certificate, to be encoded as certificate extensions, are made by the CA and thus this information does not need to be part of the request. The resulting Concise Data Definition Language (CDDL) template needed to specify an IoT CBOR certificate request is:

```
CBORCertificateRequest = (
  Request,
  subjectSignatureValue,
)

Request = (
  type : int,
  subject : bytes,
  subjectPublicKey : bytes,
  subjectPublicKeyAlgorithm :
    AlgorithmIdentifier
)
subjectSignatureValue : bytes

AlgorithmIdentifier = int /
  [ algorithm: ~oid, ? parameters: bytes ]
```

The `type` field is used with a `0` to indicate a request for a natively signed CBOR certificate. A natively signed

CBOR certificate is meant to be used as such, to communicate with other devices and services capable of handling CBOR certificates. Alternatively, a `1` is used to indicate a request for a CBOR encoded certificate that can be reconstructed into a standard X509 certificate. This is needed for communicating with devices not equipped to handle CBOR certificates.

For commonly used public key algorithms a single integer is sufficient for identification. If needed the field can be expanded to a CBOR array, to accommodate for more algorithms based on OIDs, potentially including explicit algorithm parameters.

For the IoT scenarios we target, an ECC P-256 public key is a typical choice. The consequences for the request fields are the following:

The `subjectPublicKey` field carries the minimal public key information. An ECC public key consists of a (X,Y) curve point pair. When the curve is known, the Y point can be calculated from the X coordinate together with the sign bit of Y. Hence it is sufficient to include only this information in the request field.

For a P-256 key, the `signatureValue` field carries the two 32 bit values that make up the signature. When the curve is known, no extra info or padding is needed.

C. Encoding for enrollment replies

To transfer the newly issued certificate back to the IoT device, EST is using PKCS#7. Instead we propose to let the payload of the EST server reply consists entirely of the enrolled CBOR encoded certificate, with no extra headers or footers besides a minimal CBOR byte string wrapping.

VII. Implementation

We develop a LICE implementation as modules in C, that can easily be adapted for available operating systems for embedded systems such as Contiki NG and Zephyr [24], [25]. The LICE implementation contains EHDOC, OSCORE, EST, and CBOR encoding and decoding.

For a stand alone version which can be ported to any platform, we have adapted the experimental OSCORE support in libcoap. For the constrained hardware implementation we made an OpenThread based version which we deploy on nRF52840-DK, an Arm Cortex-M4 board with 802.15.4-radio. OpenThead is an open-source implementation of Thread, with support for DTLS and CoAP. As of spring 2021, the OpenThread CoAP libraries lack support for OSCORE. To evaluate the overhead of OSCORE we perform the corresponding key derivation and encryption/decryption directly on the CoAP payload, with added bytes to compensate for the CoAP headers.

Choice of crypto algorithms. Both EDHOC and OSCORE need an AEAD algorithm and a HMAC-based key derivation function (HKDF) to establish a secure session. In addition, EDHOC needs to specify which Elliptic-curve Diffie–Hellman (ECDH) curve to use for the ephemeral keys and shared secret generation, plus an algorithm and curve to use for the signatures.

For our implementation we have focused on support for AES-CCM-16-64-128 and SHA-256 for AEAD and HKDF, respectively. The curve25519 is chosen for ECDH. Finally ECC P-256 is the main choice for the signature algorithm and curve [26].

These crypto components are included in the cipher suites proposed for standard EDHOC, targeting IoT scenarios. They are selected for their strong cryptographic properties while being relatively lightweight.

For the primitive crypto operations we have reused functionality from MbedTLS [27], curve25519-donna and uECC, as well as the crypto libraries available in the nRF_SDK [28], which enables the use of hardware acceleration for some of the crypto operations.

VIII. Evaluation

A. Method and overview

We use an experimental research methodology where we evaluate the impact of one particular variable of a phenomenon while keeping other parts of the system setup static. This ensures the results can be properly attributed to a specific system change. We present both micro benchmarks showing the individual aspects, as well as basic system tests. We include versions where ASN.1 is used in the comparisons, both to demonstrate the benefits of CBOR and to show compatibility with legacy deployments where no or minimal updates to the CA server side are done.

B. Experimental setup

For the hardware experiments we setup a local EST test server on a Raspberry Pi 3B+ which also acts as an OpenThead gateway. For DTLS we use the `ECDHE-ECDSA-AES128-CCM8` suite, with the P-256 curve. This is a commonly used configuration for IoT (for instance a mbedTLS default), similar to the setup for EDHOC. For fair comparisons any other cipher suit configuration is disabled, which prevents lengthy suit identifiers being advertised through DTLS handshake messages.

C. Micro benchmarks

1) Handshake message sizes: The individual EDHOC handshake message sizes for a regular unmodified EDHOC execution are shown in table I. In a setup where both the IoT client and the CA have access to each others public credentials, only the certificate references need to be included, and the messages sizes can be kept minimal. Compared with DTLS, the corresponding handshake using DTLS 1.2 and the same certificates, the total handshake message size is 1716 bytes when using the X.509 certificate format or 1369 bytes when using CBOR certificates.

2) Enrollment message sizes: Table II shows the EST payload for the enrollment related operations. By using CBOR encoding of the EST messages we achieve a reduction with more than 50%.

3) Total enrollment cost: To see the full effect of the enrollment operation for the relevant possible configurations, we calculate the total enrollment cost in bytes. The numbers for a full enrollment, including the `cacerts` operation, is shown in table III. For the case where all the necessary CA

TABLE I
EDHOC MESSAGE SIZES, USING ORIGINAL X.509, CBOR CERTIFICATES OR REFERENCE ONLY

Message	CoAP size (B)		
	X.509	*CBOR*	*Ref. only*
Message 1	51	51	51
Message 2	422	274	123
Message 3	399	251	110
Total size	*872*	*567*	*284*

TABLE II
EST MESSAGE SIZES, USING ASN.1 AND CBOR ENCODINGS

Message			EST payload size (B)	
			ASN.1	*CBOR*
cacerts	request	empty	0	0
	response	header/footer	49	3
		CA cert.	375	208
enroll	request	requestInfo	141	47
		signature alg.	10	1
		signature	74	64
	response	header/footer	49	0
		Client cert.	316	138
		Total size	*1014*	*458*

certificates are already present in the device trust store, only the simple enrollment operation is needed. The numbers for those enrollment scenarios are included as the 'b'-alternatives in Fig. 4.

It is clear from the data that there are substantial savings to be made to switch from ASN.1 encoding to CBOR. For the cases where the handshake is done with reference only, but the IoT device needs to update its trust store with one further certificate, the total operation cost is less than 800 bytes (Version II). When only enrollment is needed, the entire enrollment operation can be done with less than 600 bytes of application data.

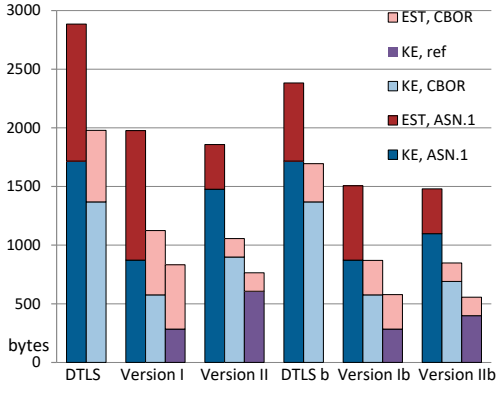


Fig. 4. Combined cost in bytes of handshake and enrollment messages, including and excluding (b) the cacerts operation

4) Memory requirements: The basic crypto functionality shared by OSCORE and EDHOC takes approximately 6 KB of ROM. EDHOC, without optimisations, adds 5 KB more of code.

The dynamic memory requirements will depend on the enrollment scenario. For embedded environments where malloc is not available or deemed unsuitable it is necessary to pre-allocate static data buffers large enough to handle incoming EDHOC and EST messages. In version I, where no additional data is sent in EDHOC, less than 2 KB of RAM is needed when using CBOR and only sending certificate references. If using ASN.1 and including the certificates in the messages, the RAM usage is at least 5 KB or more depending on certificate sizes. This is expected for this legacy case, and is comparable to the OpenThread RAM usage when using DTLS.

The EST component requires less than 10 KB of ROM. The RAM usage depends on how long certificate chains the component needs to be capable of handling. For the most confined case, where total of three certificates are involved, 2 KB is sufficient. As a comparison, for the nRF52840 test hardware a default OpenThead instance running only the OpenThead test program uses at least 263 KB of ROM and 69 KB of RAM. This means that the EST memory footprint is less than 4% of the ROM and less than 3% of the RAM usage for a typical OpenThread application.

5) Computational effort, handshake: To calculate the overhead of an EDHOC key establishment we measure the different crypto operations involved. The table IV shows the cost in time for performing the operations on our target hardware, with the range of the data sizes that occur during the handshake for the protocol versions. On the client side, the main computationally intensive operations during the key establishment are all related to public key operations: the generation of keys, shared secrets and signatures, plus signature verification. Hashing and AES-CCM crypto operations are lightweight in comparison. The key derivation (HKDF) is cheap when it is used for the most common nonce and key output sizes, 13 and 16 bytes. Such nonce and key generation is performed a total of seven times during the handshake. In addition, in the processing of the second message, one key long enough to cover the entire length of the message is generated. The maximum time, 9.4 ms (*), corresponds to the completely non optimized case when a full ASN.1 encoded certificate is sent, together with EST payload. For the version II using CBOR and certificate references, 3.4 ms is needed.

It should be noted that the significantly higher cost for shared secret generation (**) is a result of the X25519-curve operations not being fully supported in terms of hardware acceleration on the hardware used, as opposed to the corresponding operation done with P-256 keys. X25519 type key pairs are on the other hand cheaper to generate.

In a DTLS 1.2 handshake with the selected cipher suit the corresponding high cost operations are performed. The main differences lies in DTLS using fewer key derivation operations,

TABLE III
TOTAL COST OF ENROLLMENT IN BYTES. THE KEY ESTABLISHMENT MESSAGES IN VERSION II CARRY PARTS OF THE EST DATA

CoAP payload, bytes	DTLS		EDHOC / OSCORE					
	X.509	*CBOR*	*Version I*			*Version II*		
			X.509	CBOR	ref.	X.509	CBOR	ref.
Key establishment*	1716	1369	872	576	284	1476	899	607
Enrollment only	1169	610	1104	548	548	382	157	157
Total sum	*2885*	*1979*	*1976*	*1124*	*832*	*1858*	*1056*	*764*

TABLE IV
COMPUTATIONAL COST OF CRYPTOGRAPHIC OPERATIONS AND MESSAGE PROCESSING FOR THE INITIATOR DURING HANDSHAKE

Operation		*Input size, byte*	*Time*
SHA-256		212–878	99–112 μs
AES-CCM encrypt		201–655	160–261 μs
AES-CCM decrypt		290–420	230–280 μs
		Output size, byte	
hkdf nonce and key generation		13–16	361–363 μs
hkdf content protection		81–874	1.3–9.4 ms*
		Curve	
public key pair generation		P-256	19 ms
		X25519	14.6 ms
shared secret computation		P-256	18.2 ms
		X25519	121 ms**
signature generation		P-256	20.3 ms
signature verification		P-256	21 ms
compressed key reconstruction		P-256	6.5 ms
sum of all CBOR parsing			1.4 ms
total EDHOC	*Version I, CBOR*	P-256	92 ms
		X25519	191 ms
	Version II, CBOR	P-256	94 ms
		X25519	193 ms
	Version I, ref.	P-256	86 ms
		X25519	184 ms
	Version II, ref.	P-256	88 ms
		X25519	186 ms

but more messages and less compact encoding of flags and options.

6) Computational effort, EST: The result of the `cacerts` operation is a certificate chain, which will be subjected to certificate path validation. Path validation includes verifying the certificate signatures, making this the computationally most costly operation in the processing of the reply.

For the enrollment, the device generates a new key-pair and one signature to sign the request. Upon receiving the reply, one signature from the CA is checked. For our choice of EDHOC crypto algoritms, these are the same type of key and signature operations with the same cost as during the handshake.

Additionally, if CBOR certificate encoding is used, there is an extra encoding/decoding cost. Encoding carries a minimal overhead, but decoding might include recalculating the compressed public key. This is needed for path validation, with a calculation time of 6.5 ms per certificate. For the enrollment operation, the client does not need to recalculate the compressed public key, as it is the same as it has generated and stored locally. If the CA for any reason is trying to trick the device with a faulty (non CBOR native) certificate, signature verification will fail and the certificate will not be accepted.

The EST related costs are not dependent on the choice of secure session used to carry the messages.

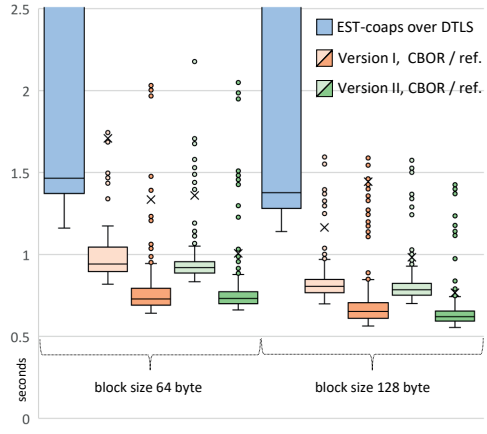


Fig. 5. Box plots showing median time and quartile data for the combined handshake and enrollment operations

D. System tests

To evaluate the actual performance of the protocol we test the most relevant versions against unmodified EST-coaps over DTLS. To minimize the 6loWPAN fragmenting we use CoAP block transfer. The deployment environment is noisy, reached by several wifi networks, and occasional bluetooth traffic.

The results of the test runs with at least 1500 iterations per test item are shown in Fig. 5. No outliers have been removed from the data, hence the average times are considerably larger than the expected median operation times. It is clear that DTLS struggles with significant message losses and is prone to recurring long delays, due to the error prone fragmentation of handshake messages, which is avoided by our protocol. The cropped upper quartiles extend to nine seconds. For a particular deployment setting, if occasional long delays should be avoided, one could evaluate an initial deployment to tweak time-outs and increase the resend rate, but besides the extra efforts an increased resend rate also comes with an increased

energy cost. Also the EDHOC based enrollment runs suffer from long delays, but as can be seen the differences are significant. By avoiding the lengthy handshake packets and using more compact encodings, the entire enroll operation can be performed with a median time between 0.62 and 0.81 seconds, when using a 128 byte block size. This is between 45% and 58% of the median time needed for the DTLS alternative. Despite frequent message losses, a bigger 128 byte block size shows a bit better overall performance, since decreased fragmenting overhead out-weights the losses.

Looking closer at the EDHOC versions, no significant time differences can be seen when comparing version I and II, which is to be expected because the relatively small differences in size. For an environment where OSCORE needs to be available either protocol version can be used, whereas version II can be used independently.

E. Comparison with previous state-of-art

The previous state-of-art for IoT enrollment is represented by the work done in [20] and [4]. The EST-coaps version presented in [20] corresponds to the DTLS X.509 test cases evaluated here in terms of overhead for handshake and enrollment payloads. In [4] steps are taken to introduce CBOR encoding of certificates. In that work the encoding is only used during the handshake inside the low power radio network, relying on a custom radio gateway for on-the-fly conversion between the encoding formats, making it hard to directly compare, but from the point of view of the IoT device this corresponds to a payload cost between the original X.509 and the CBOR versions of DTLS given in table III.

F. Evaluation Summary

The micro benchmarks and systems tests show that it is feasible to execute certificate enrollment protected with application layer security on IoT devices in a lossy low-power radio network with acceptable overhead. LICE offers improved performance compared with certificate enrollment protected with DTLS.

IX. Security Considerations

Both proposed versions of the protocol are built on a minimal EDHOC message exchange, where the server starts executing the protocol before checking the client reachability status. Since network addresses can be spoofed, in some scenarios this can be considered a potential denial-of-service attack threat. To prevent this an additional echo-request can be sent from the server, forcing the client to demonstrate reachability before further server processing happens.

To limit the reliance on a potentially outdated factory installed trust store, especially in the absence of suitable revocation mechanisms, it is recommended to only use the initial trust store once, for the first authentication with the enrollment CA, until it has been updated by a successful `cacerts` operation. In a larger PKI perspective questions related to transfer of trust and mechanisms that enable reselling and re-deployment of devices should be addressed.

While this work does not address IoT software security specifically, it should be noted that the reduced complexity of CBOR parsers and encoders can make them easier to keep secure compared with more complex software for handling ASN.1.

While IoT devices are becoming increasingly capable, asymmetric cryptographic operations are still expensive. Whenever energy is of concern, they should be used sparingly. Once a secure context is established, a constrained device and the connected endpoint can keep the context active as long as deemed safe and useful, needing only symmetric crypto operations for onward communications. Questions of determining suitable context life-times are related to general questions of trust, certificate validity periods and certificate revocation mechanisms. With potentially long lived security contexts, it becomes important to ensure that the context lifetime is never longer than the validity period of the credential upon which the context was created. In addition, a trust revocation mechanism for IoT should ideally ensure the termination of relevant security contexts.

It is worth emphasising a key point of the SIGMA schema used; as long as the included components keep their guarantees, a correctly implemented protocol will provide the desired security services. If any weakness is found in the fundamentals of a cipher suit or in a particular implementation, the corresponding component must be immediately replaced or updated. This highlights the need for secure updates, another security service which benefits from standardized PKI for IoT solutions.

X. Conclusion

We have provided one of the most important building blocks needed to bring PKI to IoT, a fully automated certificate enrollment protocol. LICE is built on the new ecosystem of emerging and proposed application layer security standards, creating a solution which can traverse proxies and offer true end-to-end security. Using compact CBOR-based encodings and standard cryptographic components, we have demonstrated certificate enrollment with a significantly lower cost, using less than one third of the data for communication compared with corresponding operations using the previous state-of-art EST-coaps.

Acknowledgment

This research has partly been funded by The Swedish Foundation for Strategic Research (SSF), by the H2020 ECSEL SECREDAS project (grant ID: 783119) and by the Sweden's Innovation Agency Vinnova through the ITEA 3 STACK project.

References

[1] G. Selander, J. Mattsson, F. Palombini, and L. Seitz, "Object security for constrained restful environments (oscore)," Internet Requests for Comments, RFC Editor, RFC 8613, July 2019.

[2] G. Selander, J. Mattsson, and F. Palombini, "Ephemeral diffie-hellman over cose (edhoc)," Working Draft, IETF Secretariat, Internet-Draft draft-ietf-lake-edhoc-03, December 2020.

[3] P. van der Stok, P. Kampanakis, M. Richardson, and S. Raza, "Est over secure coap (est-coaps)," Working Draft, IETF Secretariat, Internet-Draft draft-ietf-ace-coap-est-18, January 2020.

[4] J. Höglund, S. Lindemer, M. Furuhed, and S. Raza, "PKI4IoT: Towards Public Key infrastructure for the Internet of Things," *Computers & Security*, p. 101658, 2019.

[5] C. Bormann and P. Hoffman, "Concise binary object representation (cbor)," Internet Requests for Comments, RFC Editor, RFC 7049, October 2013.

[6] M. Schukat and P. Cortijo, "Public key infrastructures and digital certificates for the internet of things," in *2015 26th Irish Signals and Systems Conference (ISSC)*, 2015, pp. 1–5.

[7] Q. Jing, A. V. Vasilakos, J. Wan, J. Lu, and D. Qiu, "Security of the internet of things: perspectives and challenges," *Wireless Networks*, vol. 20, no. 8, pp. 2481–2501, Nov 2014.

[8] R. T. Tiburski, L. A. Amaral, E. de Matos, D. F. G. de Azevedo, and F. Hessel, "Evaluating the use of tls and dtls protocols in iot middleware systems applied to e-health," in *2017 14th IEEE Annual Consumer Communications Networking Conference (CCNC)*, 2017, pp. 480–485.

[9] F. A. Alhaidari and E. J. Alqahtani, "Securing communication between fog computing and iot using constrained application protocol (coap): A survey," *J. Commun.*, vol. 15, pp. 14–30, 2020.

[10] DigiCert, "PKI: The Security Solution for the Internet of Things," DigiCert Inc., Tech. Rep., 2017. [Online]. Available: https://www.digicert.com/resources/fact-sheet/pki-the-security-solution-for-the-internet-of-things.pdf

[11] Keyfactor, "PKI: The Solution for Building Secure IoT Devices," Keyfactor, Inc, Tech. Rep., 2020.

[12] M. A. Khan and K. Salah, "Iot security: Review, blockchain solutions, and open challenges," *Future Generation Computer Systems*, vol. 82, pp. 395–411, 2018.

[13] C. Doukas, I. Maglogiannis, V. Koufi, F. Malamateniou, and G. Vassilacopoulos, "Enabling data protection through pki encryption in iot m-health devices," in *2012 IEEE 12th International Conference on Bioinformatics Bioengineering (BIBE)*, Nov 2012, pp. 25–29.

[14] T. Giannetsos and I. Krontiris, "Securing v2x communications for the future: Can pki systems offer the answer?" in *Proceedings of the 14th International Conference on Availability, Reliability and Security*, ser. ARES '19. New York, NY, USA: Association for Computing Machinery, 2019.

[15] M. N. Aman, U. Javaid, and B. Sikdar, "A privacy-preserving and scalable authentication protocol for the internet of vehicles," *IEEE Internet of Things Journal*, vol. 8, no. 2, pp. 1123–1139, 2021.

[16] P. Gutmann, "Simple certificate enrolment protocol," Internet Requests for Comments, RFC Editor, RFC 8894, September 2020.

[17] J. Schaad and M. Myers, "Certificate management over cms (cmc)," Internet Requests for Comments, RFC Editor, RFC 5272, June 2008.

[18] C. Adams, S. Farrell, T. Kause, and T. Mononen, "Internet x.509 public key infrastructure certificate management protocol (cmp)," Internet Requests for Comments, RFC Editor, RFC 4210, September 2005.

[19] M. Pritikin, P. Yee, and D. Harkins, "Enrollment over secure transport," Internet Requests for Comments, RFC Editor, RFC 7030, October 2013.

[20] Z. He, M. Furuhed, and S. Raza, "Indraj: Digital certificate enrollment for battery-powered wireless devices," in *Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks*, ser. WiSec '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 117–127.

[21] J. Schaad, "Cbor object signing and encryption (cose)," Internet Requests for Comments, RFC Editor, RFC 8152, July 2017.

[22] H. Krawczyk, "Sigma: The 'sign-and-mac' approach to authenticated diffie-hellman and its use in the ike protocols," in *Advances in Cryptology - CRYPTO 2003*, D. Boneh, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 400–425.

[23] S. Raza, J. Hoglund, G. Selander, J. Mattsson, and M. Furuhed, "Cbor encoding of x.509 certificates (cbor certificates)," Working Draft, IETF Secretariat, Internet-Draft draft-mattsson-cose-cbor-cert-compress-06, January 2021. [Online]. Available: https://tools.ietf.org/html/draft-mattsson-cose-cbor-cert-compress-06

[24] T. Vu Chien, H. Nguyen Chan, and T. Nguyen Huu, "A comparative study on operating system for wireless sensor networks," in *2011 International Conference on Advanced Computer Science and Information Systems*, 2011, pp. 73–78.

[25] Linux Foundation Project, "Zephyr project," https://www.zephyrproject.org/, 2020.

[26] A. Langley, M. Hamburg, and S. Turner, "Elliptic curves for security," Internet Requests for Comments, RFC Editor, RFC 7748, January 2016.

[27] ARMmbed, "Mbed tls," 2021. [Online]. Available: https://github.com/ARMmbed/mbedtls

[28] Nordic Semiconductor, "nrf5-sdk," 2021. [Online]. Available: https://www.nordicsemi.com/Software-and-Tools/Software/nRF5-SDK-for-Thread-and-Zigbee

Paper III

Journal of Information Security and Applications 73 (2023) 103424

Contents lists available at ScienceDirect

Journal of Information Security and Applications

journal homepage: www.elsevier.com/locate/jisa

Lightweight certificate revocation for low-power IoT with end-to-end security

Joel Höglund [a,*], Martin Furuhed [b], Shahid Raza [a]

[a] RISE Research Institutes of Sweden, Isafjordsgatan 22, Kista, Stockholm, 16440, Sweden
[b] Nexus Group, Telefonv. 26, Stockholm, 12626, Sweden

ARTICLE INFO

Keywords:
IoT security
Revocation
OCSP
PKI
X.509

ABSTRACT

Public key infrastructure (PKI) provides the basis of authentication and access control in most networked systems. In the Internet of Things (IoT), however, security has predominantly been based on pre-shared keys (PSK), which cannot be revoked and do not provide strong authentication. The prevalence of PSK in the IoT is due primarily to a lack of lightweight protocols for accessing PKI services. Principal among these services are digital certificate enrollment and revocation, the former of which is addressed in recent research and is being pushed for standardization in IETF. However, no protocol yet exists for retrieving certificate status information on constrained devices, and revocation is not possible unless such a service is available.

In this work, we start with implementing the Online Certificate Status Protocol (OCSP), the de facto standard for certificate validation on the Web, on state-of-the-art constrained hardware. In doing so, we demonstrate that the resource overhead of this protocol is unacceptable for highly constrained environments. We design, implement and evaluate a lightweight alternative to OCSP, TinyOCSP, which leverages recently standardized IoT protocols, such as CoAP and CBOR. In our experiments, validating eight certificates with TinyOCSP required 41% less energy than validating *just one* with OCSP on an ARM Cortex-M3 SoC. Moreover, validation transactions encoded with TinyOCSP are *at least* 73% smaller than the OCSP equivalent.

We design a protocol for compressed certificate revocation lists (CCRL) using Bloom filters which together with TinyOCSP can further reduce validation overhead. We derive a set of equations for computing the optimal filter parameters, and confirm these results through empirical evaluation.

1. Introduction

The Internet of Things (IoT) is projected to grow to billions of devices in the coming years, and with this growth comes great security challenges. In small deployments of constrained devices with controlled access, such as wireless sensor networks, pre-shared key (PSK) solutions are the current state-of-the-art for security. However, PSK cannot be scaled to billions of globally identifiable devices, because the keys must be distributed out-of-band. Moreover, the keys are symmetric, so a server compromise puts all connected devices at risk, some of which may be used in safety-critical applications.

Public key infrastructure (PKI) provides centralized key management and mutual authentication for a wide range of networked systems. As constrained devices become connected to the Internet-at-large, the transition from PSK to PKI security is inevitable, and there has been an ongoing effort to facilitate this transition [1]. This entails designing new lightweight protocols to enable certificate enrollment, re-enrollment and validation on constrained devices. These procedures are illustrated in Fig. 1. A constrained device can act as either an end entity and/or a relying party, depending on the context.

The greatest barrier to implementing these procedures in the IoT is the minimal computing resources available to many of the devices. For cheap sensors and actuators still widely used in industrial deployments, RAM is often on the order of tens of kilobytes, and communication is often over lossy wireless channels. In the worst case, devices are battery-powered and infrequently recharged. Under these conditions, all network protocols must be remarkably efficient.

Devices with sufficient computing resources have two options for retrieving certificate status information: download an X.509 certificate revocation list (CRL), or request the status of individual certificates with the Online Certificate Status Protocol (OCSP), see [2]. A CRL containing hundred or thousands of entries can be on the scale of tens or hundreds of kilobytes [3], which is far too large for some constrained devices. OCSP exchanges are on the order of hundreds of bytes, but this does not mean it is suitable for the Internet of Things.

* Corresponding author.
E-mail addresses: joel.hoglund@ri.se (J. Höglund), martin.furuhed@nexusgroup.com (M. Furuhed), shahid.raze@ri.se (S. Raza).

https://doi.org/10.1016/j.jisa.2023.103424

Available online 23 January 2023
2214-2126/© 2023 The Authors. Published by Elsevier Ltd. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

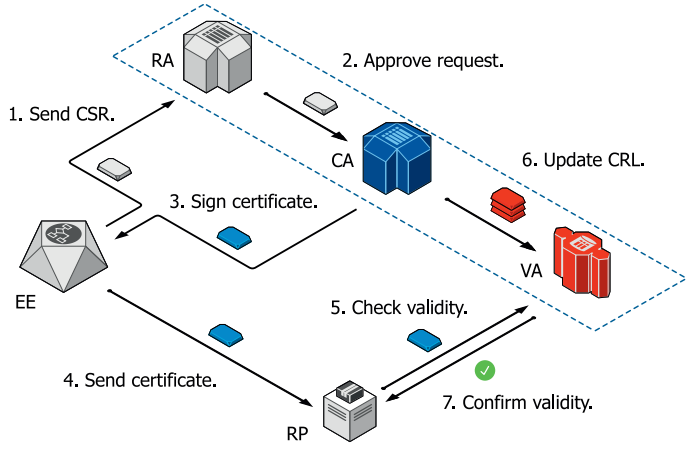


Fig. 1. The principle actors and procedures in a public key infrastructure: A registration authority (RA) verifies the identity of an end entity (EE) before a certificate authority (CA) signs and issues its certificate. Relying parties (RPs) can verify the status of this certificate using either OCSP or a CRL download from a validation authority (VA)[1] server.

Since the standardization of OCSP in 1999, many new technologies for constrained applications have been introduced.

The development of lightweight versions of Web protocols for the IoT began around 2007 with the introduction of 6LoWPAN [4,5]. This enabled IPv6 networking on highly constrained devices. Two IPv6 network stacks for highly constrained environments are illustrated in Fig. 2. Both of these utilize the Constrained Application Protocol (CoAP), which is essentially a profile of HTTP. CoAP messages are usually encoded with the Concise Binary Object Representation (CBOR) encoding scheme [6,7], which is essentially a compact version of JSON. Our solution for certificate validation on constrained devices, TinyOCSP, is designed for use in CoAP network stacks.

The three primary contributions of this work are presented in the following sections:

3 We critically analyze the specification of the OCSP protocol and explain how significant performance improvements can be made without sacrificing functionality or compromising security.
4 We design a new protocol called TinyOCSP, which leverages recent IoT standards, as well as our findings in Section 3. We implement this on state-of-the-art IoT hardware with low-power radios alongside the original OCSP protocol, and evaluate the performance benefits in terms of energy use and message overhead.
5 We integrate CRL compression using Bloom filters with TinyOCSP to further reduce validation overhead. Furthermore, we derive equations to compute the optimal Bloom filter parameters, and verify these results through simulation.

The rest of this paper is organized as follows:

2 Related work and existing approaches to overhead reduction in digital certificate revocation
6 Security considerations relevant to certificate validation on constrained node networks
7 Concluding remarks

[1] "Validation authority" is a generic term for a server hosting a range of PKI services. In the context of OCSP, the VA is often referred to as an *OCSP responder*, as per RFC 6960. We use the term "VA" in this paper to refer to both OCSP responders and CRL distribution points for simplicity.

2. Related work

Digital certificate revocation is a fairly simple operation from a network administrator's point of view. However, a certificate is not *practically* disabled until all relying parties are notified of this change in authorization. The efficient distribution of revocation state is a difficult problem and has been the focus of many related studies.

OCSP remains uncontested in the realm of online validation systems (i.e., systems which generate new validity proofs for every request received). Other revocation systems can be categorized according their underlying data structures: Bloom filters, Merkle trees, hash chains, lists and various distributed architectures.

2.1. Bloom filters

The use of Bloom filters for CRL compression has been proposed for VANETs by Raya et al. [12], advanced metering infrastructures in [13] and Web browsers in [14]. This particular data structure is of interest due to its very high compression ratio in applications where one must determine if an item (e.g., a certificate) is a member of a set (e.g., a CRL), and the expected outcome is non-membership. However, these systems have only been demonstrated in research settings and are not associated with any standardized protocols. A thorough treatment of Bloom filters is given in Section 5.

Our contribution to this research area is a generally applicable set of equations, which can be used to determine if Bloom filters have utility in any given application area. Existing studies have conducted experiments for specific use cases, which may not provide accurate predictions of performance in other use cases.

Additionally, we demonstrate that CRL compression with Bloom filters is feasible on devices with tens of kilobytes of RAM. This is an important distinction, because existing works in this area target more powerful IoT devices. For example, one recent paper presented an implementation of Bloom filters for CRL compression on the Raspberry Pi 2, which the authors referred to as an "IoT device" in [15]. However, that platform has one gigabyte of RAM, and would therefore not be classified as constrained according to IETF terminology, see [16].

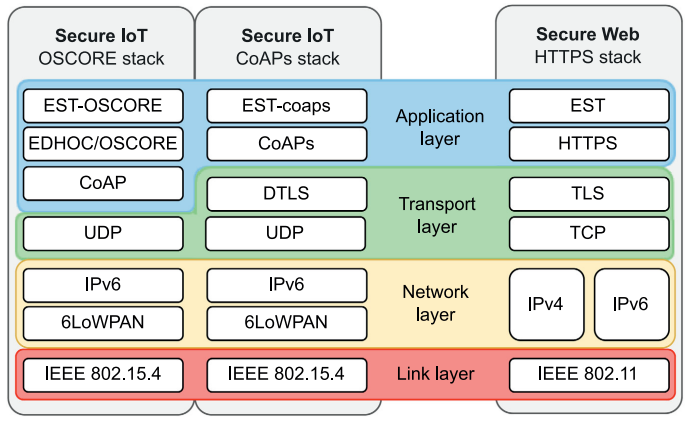


Fig. 2. Two networks stacks available for constrained devices, and their Web counterparts. The Object Security for Constrained RESTful Environments (OSCORE) [8] stack only encrypts CoAP payloads, whereas the CoAPs stack uses DTLS to encrypt entire CoAP messages. EST-OSCORE, EST-coaps and EST are all certificate enrollment protocols with end-to-end security (i.e., no trusted intermediary is relied upon). See [9–11].

2.2. Merkle trees

Certificate Revocation Trees (CRT), first proposed in 1998 in [17], store revoked certificates in a Merkle tree. This allows relying parties to query the status of an individual certificate, much like OCSP. The validity or revocation proof consists of a list of $\lceil \log_2(n) \rceil$ hashes, where n is the number of revoked certificates. (This is a property of Merkle trees, which will not be thoroughly examined in this paper.) One of the hashes included in every proof is the root node of the tree, which has been signed by the VA. CRTs would be generated periodically, and since the VA uses the same signed data structure to service every request, the system can potentially reduce the computational load on VA servers. Marginal improvements to the system have been proposed through the use of skip lists by Goodrich et al. [18] and 2–3 trees by Naor and Nissim [19].

This approach does not address the challenges faced by highly constrained networks. The energy cost to relying parties must be minimized, while the cost to servers is of secondary concern. Consider the case where 1000 certificates have been revoked, and a CRT is constructed with SHA-256 hashes. A validity proof must then contain $32 \times \lceil \log_2(1000) \rceil = 320$ bytes, plus a signature and metadata. It will be shown in Section 4.3 that this is larger than an OCSP response.

2.3. Hash chains

The use of hash chains for certificate validation was first proposed by Micali [20], and later given the name NOVOMODO by Micali [21]. The approach works, in principle, by appending two hashes to a certificate upon its issuance. The CA generates two secrets, S_1 and S_2. S_1 is hashed once, and will be revealed by the CA if the certificate is revoked. The certificate has a lifetime T and the CA will update its status every interval of time t. The second secret S_2 is hashed T/t times, forming the hash chain $H_{T/t}(S_2)$. At each update interval, the CA reveals the next hash in the chain, thus informing relying parties that the certificate is still valid. For example, if a certificate is issued for 365 days, the CA will reveal $H_{300}(S_2)$ after 65 days. The relying party can then hash this value 65 times and compare it with the value $H_{365}(S_2)$, which is embedded in the certificate.

The advantage of NOVOMODO is that VA responses (i.e., the hashes) do not need signing. However, researchers have observed that the hash chains of millions of certificates can amount to several terabytes [22], and these must either be stored by the CA, transferred to VA servers, or generated dynamically as requests are received. The number of certificates issued to the IoT will be an order of magnitude higher than the Web, so NOVOMODO is not a feasible solution. Moreover, relying parties cannot request fresh information, because the update intervals are hard-coded into the certificates.

2.4. Revocation lists

Aside from X.509 CRLs and delta-CRLs, Google Chrome's CRLSet is the only list-based revocation in widespread use today. This is essentially a curated list of revoked certificates from many TLS website CRLs. One study found that CRLSet detects less than 2% of the revoked certificates on the Web, see [23]. This system is built on the assumption that most revoked certificates belong to websites that are rarely, if ever, visited by end users. It is not clear how such a system could be transposed to the Internet of Things, as the algorithm for generating the CRLSets is not publicly available.

2.5. Distributed revocation state

Some authors have proposed distributed systems for sharing certificate status information across networks of devices, see [24,25]. These approaches break end-to-end security between the VA and RPs, and are therefore generally not applicable.

3. OCSP optimizations

The Online Certificate Status Protocol (OCSP) defines two ASN.1-encoded messages: a request sent by a relying party, which indicates one or more certificates to be validated, and a response returned by the validation authority, which contains digitally signed certificate status information. This standard is now 20 years old, and it is likely that constrained M2M applications were not seriously considered in its drafting. We follow the approach applied by other successful adaptations of Web standards for constrained environments, such as CoAP and 6LoWPAN. This begins with identifying *outdated*, *unnecessary* or *inefficient* features in an existing standard. Here, we present our analysis of OCSP under these criteria.

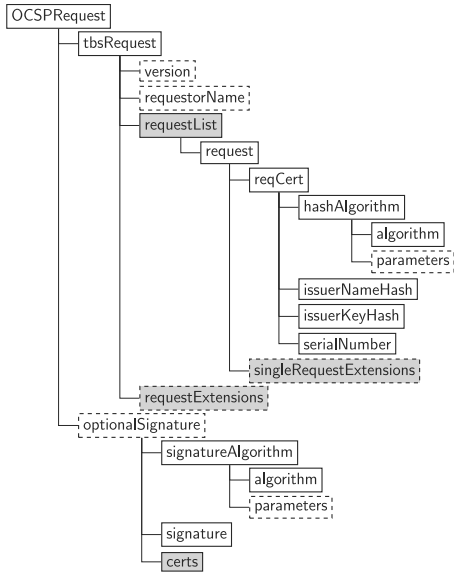


Fig. 3. The encoded data structure of an OCSP request. Gray boxes indicate an ASN.1 *SEQUENCE of SEQUENCE*, which can contain any number of duplicates of its nested contents. Dashed boxes are optional to include.

3.1. Request encoding

We observed significant room for improvement in the encoding of certificate identifiers in OCSP request messages. Each certificate is identified in the `reqCert` field (see Fig. 3). This contains a hash of the issuing CA's name, a hash of the issuing CA's public key, and a serial number. The `hashAlgorithm` field is mandatory. Surprisingly, SHA-1 is still widely used in OCSP implementations, even though collision attacks have been demonstrated against the algorithm [26]. SHA-1 is the default hash function for OCSP certificate identifiers in the latest version of OpenSSL [27]. We also found that the latest version of Mozilla Firefox uses SHA-1 in OCSP requests, despite having dropped support for TLS certificates containing SHA-1 signatures in 2017, see [28].

The total ASN.1 encoded size of the CA identifier is 53 bytes with SHA-1, although it can be larger with other hash functions, such as SHA-256. And yet, there is an X.509 certificate extension for this same purpose called an authority key identifier (AKID). RFC 5280 §4.2.1.1 recommends an AKID of only 8 bytes in length to uniquely identify the CA associated with a certificate. If this field were used as the certificate authority identifier in OCSP requests, relying parties could simply copy the AKID from a certificate, which would improve performance and simplicity. A note on security properties: If the AKID is compromised in the sense that a collision is possible, it could be used as a extremely cumbersome denial of service attack vector, by invalidating a valid certificate with the same AKID as a revoked one. The change will never lead to a compromised and revoked certificate being accepted as a valid one.

The nested encoding structure of OCSP allows messages to be extensible, but adds significant encoding overhead. Simply switching from ASN.1 to Concise Binary Object Representation (CBOR) encoding would improve OCSP. CBOR array tags only indicate the number of *data items* that follow, so they can often be encoded with just one byte. Switching the encoding format does not affect the security properties.

3.2. Response encoding

We discovered that OCSP responses contain several fields which are simply disregarded by relying parties, and can therefore be removed without any negative security consequence. The identity of the VA is specified in the required `responderID` field, even though relying parties must already know the VA's identity through its certificate. Otherwise, it would not be possible to verify the signature on responses. The `responderID` field can contain either the subject name string from the VA's certificate, or the VA's subject key identifier (SKID). It is peculiar that a choice is given for this field, because the SKID is much more concise than the name string.

OCSP appears to contain redundant timestamps, some of which do not provide meaningful information. The response contains two required timestamps, `producedAt` and `thisUpdate`. The latter indicates when the VA last updated its store of revocation information. In modern PKI, there is very little latency between the revocation of a certificate and the propagation of this information to all VA servers in the PKI. In any case, the relying party simply needs to know whether or not the response is fresh enough to be accepted based on its own security policy. `producedAt` does not provide any useful information in this regard.

There is an optional `nextUpdate` field for the case where the VA receives periodic updates, which is very rarely the case in modern PKI. This information is unlikely to be of practical use for a relying party's security policy. All of these timestamps are encoded in ASN.1 `GeneralizedTime` strings. This format is more verbose than alternatives such as POSIX time (i.e., seconds since the beginning of the year 1970), which can be represented as a 4-byte integer.

The `responseType` field can be assumed to be of type `id-pkix-ocsp-basic` without security implications or loss of applicability for the target IoT scenarios.

The required `signatureAlgorithm` field can also be removed without compromising security or losing relevant functionality. The requesting party either has a copy of the VA certificate already, or can obtain it through the OCSP response. X.509 certificates state which signature algorithm the associated key pair is authorized to perform.

3.3. Extensions

OCSP offers a number of extensions, some which are valid for requests and some for replies. All extensions are optional to implement for both clients and servers. The list includes a nonce, detailed explanations for a certificate's revocation, preferred signature algorithms and CRL download locations. We argue that with the exception of a nonce, which is useful for replay attack detection and prevention, these extensions are not necessary for constrained M2M environments, where the drawbacks of added extension handling complexity outweigh the benefits. Relying parties simply need to know whether or not to reject a certificate, and have to delegate for instance advanced auditing functionality to other less constrained system entities.

3.4. Redundant message echoing

The most notable performance drawback we have identified in OCSP is the inclusion of request data in the response. The entire `reqCert` request field is echoed back in the `certID` response field (see Fig. 4). All extensions are also echoed back. On first inspection, this may appear to be necessary, because the response signature is computed over the entire `tbsResponseData` field. If the VA does not sign a nonce, it serves no purpose. If the VA does not sign the certificate identifiers, there is no way to prove which certificates the revocation statuses belong to.

We circumvent the redundancy problem entirely with our new protocol, TinyOCSP. No information included in the request is echoed back in the response. Instead, the VA signs the *concatenation* of the

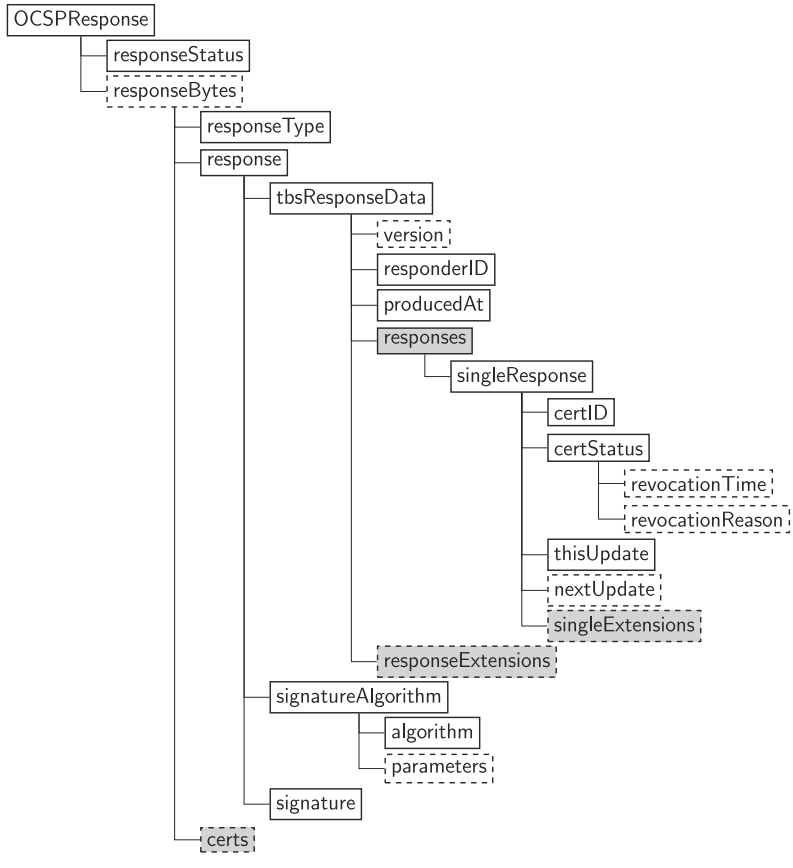


Fig. 4. The encoded data structure of an OCSP response. The `certID` field contains a full copy of the `reqCert` field from the instigating request.

request and the authenticated response data. This ensures that the association between the certificate identifiers and their statuses remains authenticated, even though the response only contains the statuses.

A concluding overall remark after the OCSP analysis is that it is very clear that the protocol has been designed for, and used by, non constrained devices with relatively good connectivity. For these scenarios the inefficient encodings and potentially complex extension handling are no major drawbacks, and does not need further optimization. A comparable case is TLS, which has been complemented rather than replaced with DTLS and other more lightweight protocols better suited for IoT communication.

4. TinyOCSP: A lightweight alternative

Based on our findings in the preceding section, we devised a lightweight version of OCSP, called TinyOCSP. We define a request message sent by an RP, and the response sent by a VA. TinyOCSP messages are serialized with the Constrained Binary Object Representation (CBOR) encoding scheme, rather than ASN.1. This has a few advantages in constrained applications:

- CBOR is the recommended serialization layer for CoAP.
- CBOR is more concise than ASN.1.
- Embedded operating systems are unlikely to include a generalized ASN.1 parser (i.e., not just for X.509 certificate handling).

In other words, TinyOCSP is much easier to implement on modern constrained devices than OCSP. At the same time it keeps the critical content and structure from the original OCSP, lowering the threshold for servers to complement existing OCSP systems, to implement and start using TinyOCSP.

4.1. TinyOCSP request encoding

The certificates of the issuing CA are identified in TinyOCSP using the X.509 authority key identifier (AKID). RFC 5280 recommends using an 8- or 20-byte value for this field, although any method for generating a unique identifier is acceptable. This means that all end entities in the PKI must include the X.509 AKID extension in their certificates, so that they can be identified in requests. Using the AKID value to identify the issuing CA eliminates the hash operations required by OCSP requests and hence the need of the `hashAlgorithm` field. The removal of these hashes is of negligible security concern. The purpose of the certificate issuer identifier in OCSP is to handle cases where a single VA is authorized to validate certificates issued by more than one CA. It is extremely unlikely for two CAs to have the same 8-byte

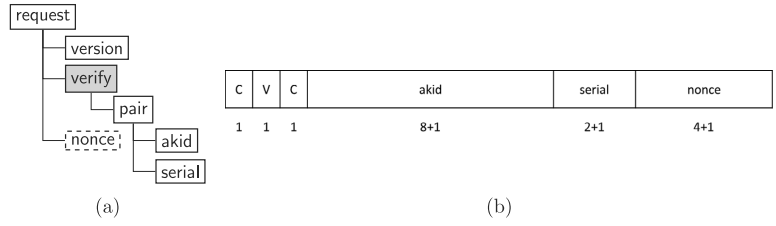


Fig. 5. The CBOR-encoded data structure of an TinyOCSP request.
(a) Gray boxes indicate an array containing any number of duplicates of its nested contents. Dashed boxes are optional to include.
(b) Shown as message payload in bytes, for querying the status of one certificate, including an optional nonce and CBOR headers.

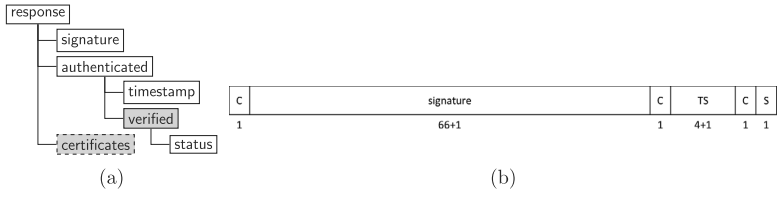


Fig. 6. The CBOR-encoded data structure of an TinyOCSP response.
(a) The `verified` field contains certificate status codes in order corresponding to the `verify` field from the instigating request.
(b) Shown as message payload in bytes, for reporting the status of one certificate, including CBOR headers.

AKID if they have followed the recommended practices in RFC 5280. As detailed in 3.1, the risk of collision does not break any security properties of the system. An TinyOCSP request is illustrated in Fig. 5.

The exclusion of the optional request signature, together with the accompanying `requestorName` field, saves both message bytes and the computationally expensive signature computation. At the same time it does remove an option for denial of service protection for the OCSP server, which should be taken into account when setting up the server protection.

It is crucial that highly constrained devices are able to indicate whether or not they need the VA certificate appended to a validation response, but OCSP has no mechanism to do this. This certificiate inclusion would be problematic to have as a default in highly constrained environments, as a single X.509 certificate can be hundreds of bytes. We address this issue by including a version code in TinyOCSP, which can be used to request the VA certificate. This addition carries no security implications.

4.2. TinyOCSP reply encoding

TinyOCSP drastically reduces the size of validation responses by removing any and all data already known by the RP. Instead, the VA signs the *concatenation* of the entire request and the `authenticated` field in the response. This also allows the relying party to include a nonce of any size in its request without increasing the response size. Since only the concatenation with the original request data, which is still kept and easily checked by the requesting device, will produce a valid signature, this proposed reduction of the data sent will not impact the security guarantees negatively (see Fig. 6).

Responses contain a POSIX timestamp, which totals 5 bytes after CBOR encoding. By replacing the `producedAt` timestamp by the `thisUpdate` timestamp, the requesting client will get the most significant timing information to determine if the status information is sufficiently recently updated. What is lost is the ability to automatically determine whether the response was pre-computed by the server. By including a nonce in the request, the client can ensure that the server produces a new response. The `verified` array maps certificate revocation reason codes (see RFC 5280 §5.3.1) to the certificates indicated in the request. These are integers in the range 0 to 10, which can be encoded in 1 byte with CBOR. The signature algorithm need not be declared in the response, as this can be found in the VA certificate. An adversary trying to misuse the lack of an explicit algorithm declaration would render the reply useless, but does not risk letting a revoked certificate pass.

4.3. Size comparison

The size of OCSP responses vary depending on the validation authority's configuration. Fig. 7 shows the size of encoded OCSP and TinyOCSP validation messages in a best-case OCSP configuration scenario. The VA uses its 8-byte subject key identifier in the `responderID` field, rather than the more verbose subject identifier string option. Signatures are generated with the *prime256v1* ECDSA algorithm, which is the de facto standard for constrained applications. (RSA requires significantly larger signatures for equivalent security.)

The OCSP messages for validating a single certificate total 358 bytes. The corresponding TinyOCSP messages total only 96 bytes, a 73% reduction. Moreover, TinyOCSP performs *significantly* better when several certificates are aggregated in one request. This is because each validation adds only one byte (i.e., one additional `status` field) to the TinyOCSP response, whereas OCSP echoes the entire certificate identifier back.

4.4. Implementation

In the context of battery-powered devices, energy use is a top importance metric of performance. In low power wireless personal area networks (LoWPANs), packet loss is high, and the maximum payload size before packet fragmentation is needed is low (81 bytes for devices running 6LoWPAN over IEEE 802.15.14). As a result, marginal increases in payload size can result in disproportionately higher energy use.

We have implemented both OCSP and TinyOCSP on modern constrained hardware to evaluate their performance under realistic IoT conditions. We selected the Zolertia Firefly Rev. B prototype board,

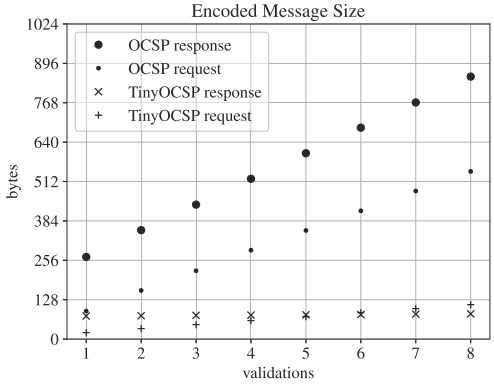


Fig. 7. Encoded message sizes in OCSP and TinyOCSP. All requests contain a 4-byte nonce. The certificates identified in the requests have 2-byte serial numbers and the same issuing CA, which has an 8-byte key identifier. The VA certificate is not appended to the responses.

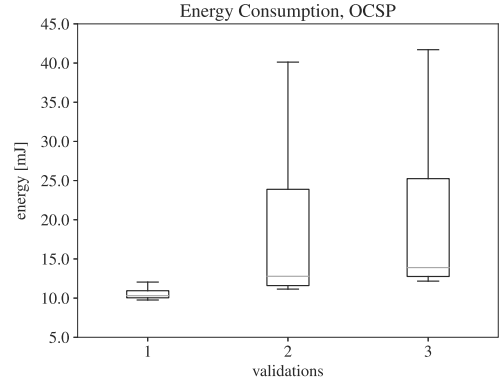


Fig. 8. Distribution of energy consumption over one hundred OCSP transactions. These data were estimated using Contiki-NG's *Energest* module.

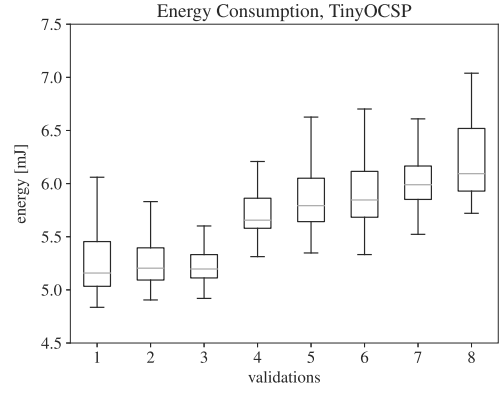


Fig. 9. Distribution of energy consumption over one hundred TinyOCSP transactions. These data were estimated using Contiki-NG's *Energest* module.

which uses the Texas Instruments CC2538 system-on-chip. This SoC features a 32-MHz ARM Cortex M3 microcontroller, 512 KB ROM and hardware acceleration for both ECC and SHA-256. Although this device has 32 KB of RAM, only 16 KB are retained in low power modes, which are used extensively in IoT applications.

Both certificate validation protocols were implemented on the Contiki-NG embedded operating system. We use Contiki-NG's implementation of the following network protocols: TSCH MAC layer, RPL routing, 6LoWPAN adaptation layer, IPv6 and CoAP. We flash one Firefly with Contiki-NG's RPL border router example code, which is then connected to an Apple desktop computer via serial interface. This desktop computer runs the VA server code. A second Firefly is flashed with the client code running our OCSP and TinyOCSP implementations. This device communicates with the RPL border router via IEEE 802.15.4 2.4-GHz radio transmitting at 7 dBm output power.

Contiki-NG's *Energest* module provides an API for recording the number of clock cycles spent in the various CC2538 radio and CPU states. The typical current draw in each of these states, according to the device datasheet, are as follows: radio receiving 20 mA, radio transmitting 34 mA, CPU on 13 mA, CPU low power mode 0.6 mA, and CPU deep low power mode 1.3 μA. The Firefly supplies 3.2 V to the CC2538 chip when powered via USB. We use these values to estimate energy consumption in our experiments with Eq. (1).

$$E = V \times I \times \text{cycles}/f_{\text{clock}} \tag{1}$$

For both OCSP and TinyOCSP, we evaluated the use case described in Section 4.3. All certificate serial numbers are 2 bytes, all AKIDs are 8 bytes and a 4-byte nonce is included. The VA is configured to provide the most concise OCSP responses allowed by the specification. Messages for both protocols are transported over CoAP.

In our experiments, the certificate validation protocols perform one hundred transactions with the VA for a range of certificate counts. *Energest* monitoring is initiated when request encoding begins and is terminated when the response signature has been verified. This means that re-transmissions due to random packet loss and failed integrity checks are captured in the measurements.

4.5. Results

The median energy consumed by TinyOCSP for one, two and three validations was 50%, 59% and 63% less than OCSP, respectively. Validating eight certificates with TinyOCSP required 41% less energy than validating a single certificate with OCSP, see Fig. 9. OCSP had a significantly higher upper standard deviation for two and three validations than any of the other test cases, see Fig. 8. This is most likely due to random packet loss and re-transmissions, as the median values still follow a linear upward trend, as one would expect. On lossy wireless links, reducing message overhead by even a few hundred bytes can significantly reduce radio use, which is often the greatest consumer of energy on wireless devices. For larger multihop networks the cost of packet losses and re-transmissions become further aggravated.

4.6. Memory discussion

The available RAM on embedded platforms is extremely scarce. To put this in perspective, consider that Contiki-NG's *hello-world* example code occupies 11 of the 16 kilobytes of available RAM on the CC2538. In order to enable CoAP and TSCH, the network protocol buffers must be reduced to the minimum allowable values. Once this is done, there are roughly 2 kilobytes of RAM available for any application-specific code.

In order to verify the signatures on responses, both protocol implementations allocated a buffer in RAM large enough to store the signed data and the signature. For OCSP, this corresponds to the response size, and for TinyOCSP, this corresponds to the size of the concatenation of request and response. With a 256 byte buffer, our TinyOCSP implementation could handle at least eight validations at a time. Even with a 512 byte buffer, our OCSP implementation could only handle three.

TinyOCSP comfortably fits in the test platform's available RAM with roughly one kilobyte to spare. Our OCSP implementation is minimal – only enough to generate a request with no extensions and parse the corresponding response – yet it fills all of the test platform's remaining RAM. This is not to say that these implementations are highly optimized, but in practical terms, it demonstrates that OCSP *should not* be used on devices with stringent memory constraints.

4.7. Energy discussion

Savings of 50% or more on energy consumption for a single protocol is a significant improvement for highly constrained devices. There are numerous application areas where battery-powered, infrequently-recharged devices are used in the Internet of Things [5]. Some of these include:

- Industrial process automation (e.g., chemical plants, waste-water treatment, refineries, oil platforms, etc.)
- Urban ecosystem monitoring (e.g., air quality, weather, traffic conditions, waste, parking lot occupancy, etc.)
- Home automation (e.g., wireless door locks, radiator valve drivers, window openers, etc.)
- Structural health monitoring (e.g., bridges, power stations, office buildings, etc.)
- Container tracking

Replacing batteries on a large network of IoT devices, or on devices which are difficult to access, incurs a high labor cost. Reducing the energy consumption of frequently-performed operations can extend their lifetimes by weeks or months.

Simple wireless sensors and actuators with months- or years-long deployments often initiate connections with a few resource servers upon activation, which will then be kept alive for its entire lifetime. These devices spend most of their operating life in a dormant low-power state, only waking up periodically to send a measurement or request instruction. An appropriate security policy in these scenarios would be to periodically check the revocation status of these servers' certificates. These devices are therefore likely to send hundreds of validation messages on a single charge.

5. CRL compression

Online validation with TinyOCSP provides a fresh, deterministic proof of a certificate's revocation status. Several researchers have experimented with Bloom filters (BF), a probabilistic data structure, as a complementary system to online validation protocols [12–15]. With this approach, relying parties download a compressed certificate revocation list (CCRL), which is a Bloom filter generated by the VA. A revoked certificate will never be mistakenly identified as valid by this CCRL. However, some valid certificates will *appear* to be revoked according to a false positive rate p. As a result, all certificates identified as *revoked* by the CCRL must be double-checked with a deterministic validation protocol (e.g., OCSP or TinyOCSP). This approach aims to reduce the number of online checks, thereby improving performance.

Related works in this area have demonstrated that the CCRL approach is feasible in some controlled use cases. Our contribution is a set of equations for computing the optimal Bloom filter parameters, and for predicting its performance in *any* use case. We then design a CCRL validation protocol, predict its performance with our framework, and verify those predictions through simulations.

5.1. Bloom filters

A Bloom filter is essentially a bitmap of a predetermined size m. All bits are initially set to 0. A value is added to the data structure by computing its hash with k unique hash functions. Each output represents the index of a bit, and these bits are all set to 1. Algorithm 1 depicts this algorithm applied to CRL compression. Checking the membership of an arbitrary value in the set represented by the BF follows a similar procedure. The value is hashed with each of the k hash functions, and the values of the corresponding bits are retrieved. If any bits are 0, the value is *definitely not* a member of the set. If all bits are 1, it *might be* a member of the set. An increase in the number of hash functions k does not automatically mean that the total size m must be increased. Although with more hash functions the likelihood that all test bits for a given membership test will be set to one is increased, which in turn generates a larger set of potential memberships to be further validated.

Algorithm 1 Procedure for generating an m-bit compressed CRL (CCRL) with a Bloom filter using k hash functions.

```
function COMPRESS(CRL, m, k)
    for i ← 0 to m do
        CCRL[i] ← 0
    end for
    for c in CRL do
        for i ← 0 to k do
            j ← hash_i(c) mod m
            CCRL[j] ← 1
        end for
    end for
    return CCRL
end function
```

Algorithm 2 depicts the procedure for validating a set of certificates U with a CCRL. Each certificate which cannot be definitively validated by the BF is added to a set V. This set of certificates is then passed to the online backup (e.g., OCSP or TinyOCSP) to confirm whether or not they are revoked.

Algorithm 2 Procedure for validating a set of certificates U with an m-bit CCRL generated with k hash functions. The set of certificates V which are not definitively validated must then be re-checked with an online validation protocol.

```
function VALIDATE(CCRL, U, k)
    V ← ∅
    for c in U do
        for i ← 0 to k do
            j ← hash_i(c) mod m
            if CCRL[j] = 0 then
                break
            end if
            if i = k then
                V ← V ∪ c
            end if
        end for
    end for
    ValidateOnline(V)
end function
```

5.2. False positive rate

The false positive rate p of a Bloom filter is the likelihood that all bits indicated by the k hash functions are 1 for an arbitrary input value. This depends on the size m of the filter (in bits), the number of values n

which have been added to the filter, and the number of hash functions k. The upper bound on this value was derived in [29] and is given by Eq. (2).

$$p \leq \left(1 - e^{\frac{k(n+0.5)}{m-1}}\right)^k \approx \left(1 - e^{-kn/m}\right)^k \tag{2}$$

In the context of this problem, n is the number of certificates on the CRL, which cannot be known in advance. (Certificate revocation only occurs when an unforeseen complication forces the network administration to end a certificate's validity before the expiration date.) The problem at hand is selecting the optimal values of k and m for a given n. In other words, the selection of optimal values of k and m creates the most efficient filter encoding of the expected probabilities of certificate revocation.

5.3. Quantifying validation cost

We approach this problem by defining a cost function for Bloom filter retrieval $C_{\mathrm{BF}}(m)$ and for online validation $C_{\mathrm{OV}}(v)$, where v is the number of validations to be performed. Our metric for cost is the total size of the information exchange between the relying party and validation authority. (We consider only the information in these protocols themselves, not the underlying network layers, which will vary between applications.) We then define the total cost of the CCRL validation procedure as a function $C_{\mathrm{CCRL}}(v, k, m, n)$.

Consider the case of validating $v = 2$ certificates with the CCRL procedure. Retrieving the Bloom filter incurs a fixed cost of C_{BF}. Assuming perfect hash functions, the membership checks for each certificate in the BF are independent events, statistically speaking. The expected value of C_{CCRL} is, then, the sum of the fixed cost plus the sum of each possible outcome of the membership checks multiplied by the cost of the outcomes.

$$\begin{aligned} C_{\mathrm{CCRL}}(2, k, m, n) = {} & C_{\mathrm{BF}}(m) \\ & + C_{\mathrm{OV}}(2)p^2 \\ & + C_{\mathrm{OV}}(1)p(p-1) \\ & + C_{\mathrm{OV}}(1)(p-1)p \end{aligned}$$

The likelihood that no additional online validations are needed is equal to the likelihood that all membership checks provide a definitive *not revoked* status, $(p-1)^v$. We extrapolate this approach to an arbitrary number of validations v, which gives us Eq. (3).

$$\begin{aligned} C_{\mathrm{CCRL}}(v, k, m, n) = {} & \sum_{i=1}^{v} C_{\mathrm{OV}}(i)\binom{v}{i}p^i(1-p)^{v-i} \\ & + C_{\mathrm{BF}}(m) \end{aligned} \tag{3}$$

The question of whether the CCRL procedure outperforms online validation acting alone can be restated as the following inequality:

$$C_{\mathrm{CCRL}}(v, k, m, n) < C_{\mathrm{OV}}(v) \tag{4}$$

When the above holds true, then the CCRL procedure will, on average, require a smaller data transfer between RP and VA than simply validating all certificates with the online procedure.

5.4. Bloom filter retrieval

In order to proceed with this analysis, we introduce a hypothetical protocol for Bloom filter retrieval so that $C_{\mathrm{BF}}(m)$ can be derived. For simplicity, we make this as similar as possible to TinyOCSP. The request and response encodings are shown in Figs. 10 and 11. The length of a data structure encoded in CBOR is deterministic and simpler to compute than the length of an ASN.1-encoded structure. Eq. (5) gives $C_{\mathrm{BF}}(m)$, in bits, based on this hypothetical protocol.

$$C_{\mathrm{BF}}(m) = 82 \times 8 + m \tag{5}$$

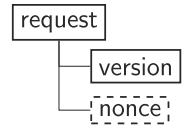


Fig. 10. Hypothetical Bloom filter retrieval request CBOR encoding.

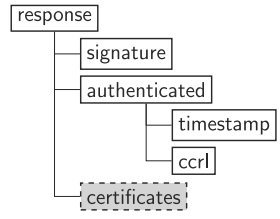


Fig. 11. Hypothetical Bloom filter retrieval response encoding. The `ccrl` field is a byte array containing the Bloom filter.

Eq. (5) holds true for $23 < m/8 < 2^8$. Within these bounds, the CBOR encoding of the `ccrl` field header is two bytes. We make the assumption that m is a multiple of 8; otherwise padding bits would be required to encode the Bloom filter. This equation also accounts for a 4-byte nonce included in the request.

5.5. Equal-cost boundaries

Consider the use case described in Section 4.3. Under those assumptions, we derive cost functions for both OCSP and TinyOCSP, which are shown in Eqs. (6) and (7). C_{OCSP} is an approximation which is accurate to within a few bytes; C_{TinyOCSP} is exact for $v < 24$. These functions compute to the sum of the request and response sizes (see Fig. 7).

$$C_{\mathrm{OCSP}}(v) \approx (148 \times v + 246) \times 8 \tag{6}$$

$$C_{\mathrm{TinyOCSP}}(v) = (14 \times v + 82) \times 8 \tag{7}$$

These cost functions can be substituted for C_{OV} in Eq. (3). We now apply an equation solver to compute the equal-cost boundary defined by Eq. (4). Fig. 12 illustrates this boundary for OCSP when $k = 1$ (which is in the following section shown to be the optimal value for k). This was computed with Python's *SciPy* library.

We observe that given a sufficiently small CRL and enough validations, the CCRL procedure will outperform OCSP, on average. The same can be said when CCRL is combined with TinyOCSP, although with quite different boundaries, as shown in Fig. 13. One would need to be validating dozens of certificates simultaneously for CCRL to outperform TinyOCSP, assuming a few hundred revocations have occurred.

These plots must be viewed within the context of CRL sizes in real-world applications. The frequency of certificate revocation is unpredictable, because one of the most common reasons to revoke certificates is a security breach. According to one study, less than 2% of TLS Web server certificates were revoked in early 2014, which jumped to over 8% later that year when the Heartbleed bug in OpenSSL was discovered [30]. More research must be done to establish whether similar revocation rates can be expected in the IoT.

5.6. Simulation

In order to verify the cost of CCRL operations predicted by Eq. (3), we implemented Algorithms 1 and 2 on an Apple desktop computer and

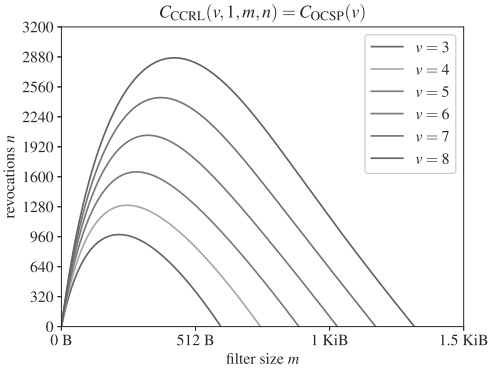


Fig. 12. The boundaries where both sides of Eq. (4) are equal for $k = 1$. For values of n below these curves, validating certificates with the CCRL procedure incurs, on average, a smaller message overhead than using OCSP for v validations.

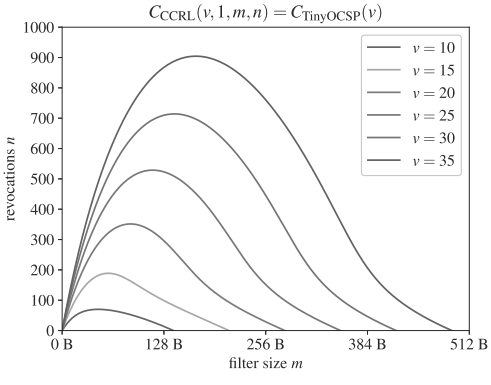


Fig. 13. Equal-cost boundaries for CCRL and TinyOCSP when $k = 1$, computed from Eq. (4) with Python's *SciPy* library equation solver.

ran simulations. We consider the case where OCSP is used as the online validation backup for $v = 8$ simultaneous validations. The predicted equal-cost boundary for this case with $k = 1$ is shown in Fig. 12.

We approach this by selecting combinations of Bloom filter sizes m and CRL sizes n such that the expected equal-cost curve will be captured in the simulations (i.e., the axis ranges on Fig. 12). For each of these combinations, Algorithm 1 is executed once, and Algorithm 2 is executed 100 times with the newly created CCRL. All certificate identifiers for both CCRL generation and validation are randomly generated. We record the mean transaction size (i.e., C_{CCRL}) for each input combination.

The results of this simulation with $k = 1$ are displayed as a heatmap in Fig. 14. The color of each pixel represents the mean observed C_{CCRL} divided by $C_{\text{OCSP}}(8)$. This normalizes the heatmap, such that a white pixel indicates parity with the performance of OCSP used independently (i.e., equal cost).

Thus far, we have only discussed the case of $k = 1$ hash function for the Bloom filter. We established that this is always the optimal value for CCRL by running additional simulations. Increasing the number of hash functions reduces the maximum value of n on the equal-cost boundary, thereby limiting the number of revocations that can occur before the

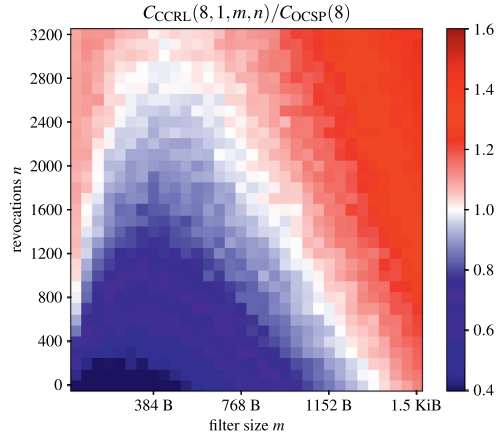


Fig. 14. Simulation results of the CCRL procedure used in conjunction with OCSP with $v = 8$ certificates, $k = 1$ hash function and 930 combinations of m and n, executed 100 times each. The mean transaction size represented by each pixel is scaled by $C_{\text{OCSP}}(8)$, such that a value of 1.0 indicates parity with OCSP for $v = 8$ certificates.

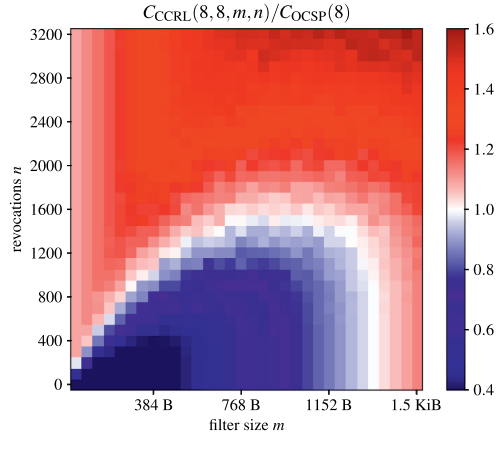


Fig. 15. Simulation results of the CCRL procedure used in conjunction with OCSP for $v = 8$ certificates, $k = 8$ hash functions.

CCRL procedure becomes ineffective. This result is illustrated in Fig. 15, which depicts the same simulation run using $k = 8$ hash functions. We observe that the maximum allowable CRL size has been approximately halved.

These simulations demonstrate that Eq. (3) is an accurate predictor of the message overheads incurred by the CCRL procedure. They also confirm that there is a limit to the utility of the CCRL approach, as defined by Eq. (4). As long as this inequality holds true, CCRL will, on average, improve the performance of the online validation system used.

6. Security considerations

TinyOCSP and CCRL have been designed to improve the performance of online certificate validation, thus enabling fully-fledged PKI

credential management on even highly constrained Internet-connected devices. It should be duly noted, however, that dependence on recent certificate status information introduces new attack vectors. These risks can be mitigated, and ultimately the advantages of phasing out pre-shared keys in favor of PKI far outweigh the challenges.

This section provides a discussion of potential vulnerabilities to online certificate validation systems in the IoT. These apply to all protocols discussed in this work, including those referenced in Section 2.

6.1. Denial of Service (DoS) attacks

Upon visiting a secure website, a Web browser receives the server's certificate via the TLS handshake. It will then wait to connect until it has validated the certificate with either OCSP or a recently-downloaded CRL. This system is resilient to DoS attacks against validation authorities for two reasons: (1) there are many redundant VA servers and (2) the CRL acts as an offline backup if all VAs become unreachable.

In contrast, highly constrained devices do not have enough memory to store full CRLs, and thus have no offline backup. (CRL compression with Bloom filters is non-deterministic and thus cannot be used in isolation.) If devices cannot access any validation authorities, a choice must be made to either accept or reject all certificates. In the former case, an attacker that has successfully compromised a private key could impersonate an authorized device in the organization by launching a DoS attack on all available VAs. In the latter case, a denial of service attack on all available VAs would effectively paralyze the network, because the nodes would be unable to verify any credentials.

These risks should be viewed within the context of denial of service attacks more generally. Although PKI *does* introduce additional attack vectors, virtually all networked systems are already vulnerable to DoS attacks. Modifications to existing firewalls may be advisable when implementing TinyOCSP, but this particular risk should not be the deciding factor in choosing between PKI or PSK-based security solutions.

6.2. Replay attacks

The inclusion of a nonce in the validation request message is optional in OCSP, TinyOCSP and CCRL because it incurs a trade off between performance and security. On the Web, for example, validation authorities receive many identical requests for certificates belonging to popular websites. VA servers can conserve resources by pre-computing responses for these certificates periodically. A single response can then be copied to each requesting party without generating new signatures.

The use of pre-computed responses creates a window of vulnerability, during which a revoked certificate will still appear valid until the VA generates a fresh response. This must be considered when configuring VA servers, regardless of the validation protocol.

An RP can force the VA to uniquely generate a response simply by including a nonce in the request, as the VA must compute a signature over this value. However, this should remain an optional feature in TinyOCSP and CCRL, as it may be possible for an IoT network to overwhelm a VA under normal operation due to the sheer number of relying parties. This is only speculation, as certificate validation in the IoT has not yet seen deployment.

6.3. Network time synchronization

Highly constrained devices are susceptible to clock drift and poor clock rate stability, both of which make network time synchronization challenging [31]. This has implications for online certificate validation, because relying parties depend on the response timestamp to determine the "freshness" of the information. If it appears that the response was generated a significant amount of time before the request, or at some point in the future, the relying party should reject it.

Unless a sufficiently accurate network time synchronization protocol is in place, devices must be configured to accept a potentially wide range of timestamp values. Alternatively, they could be configured to rely only on signed nonces and disregard the timestamp altogether. This would prevent the VA from using pre-computed responses, as discussed in Section 6.2, but this may be perfectly acceptable for many IoT deployments.

7. Conclusions

This paper has presented TinyOCSP, a lightweight alternative to OCSP. We have designed, implemented and evaluated this protocol on state-of-the-art IoT hardware with low-power radio communication. We have shown that TinyOCSP can validate at least eight certificates simultaneously with a message buffer of less than 256 bytes, whereas OCSP requires a buffer larger than this to validate *just one* certificate. The message overhead for a single validation has been reduced by, at minimum, 73%, which corresponded to a median energy reduction of 50% on constrained hardware in our experiments. For three simultaneous certificate validations, the message overhead reduction and energy savings amounted to 81% and 63%, respectively. With the introduction of this new protocol, it is now possible to implement fully fledged PKI credential management in the Internet of Things.

We have also introduced a complementary, probabilistic certificate validation protocol based on compressed certificate revocation lists (CCRL). We have demonstrated that this approach, when used in conjunction with TinyOCSP, can further reduce overhead. The equations presented in Section 5 will be a useful tool for future researchers in this area, as well as for IoT network administrators in the near future. For interoperability across different vendors and the promotion of widespread usage, we plan to push TinyOCSP for IETF standardization.

CRediT authorship contribution statement

Joel Höglund: Conceptualization, Software, Validation, Formal analysis, Investigation, Writing – original draft, Writing – review & editing, Visualization. **Martin Furuhed:** Conceptualization, Writing – original draft, Writing – review & editing. **Shahid Raza:** Conceptualization, Methodology, Validation, Resources, Writing – original draft, Writing – review & editing, Supervision, Project administration, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

This work was partly supported by the Swedish Foundation for Strategic Research (SSF) institute PhD program, and by the H2020 CONCORDIA (GA No. 830927) and ARCADIAN-IoT (GA No. 101020259) projects.

References

[1] Hummen R, Ziegeldorf JH, Shafagh H, Raza S, Wehrle K. Towards viable certificate-based authentication for the web of things. In: ACM workshop on hot topics on wireless network security and privacy, co-located with ACM WiSec 2013. Budapest, Hungary; 2013, p. 0–5.

[2] Santesson S, Myers M, Ankney R, Malpani A, Galperin S, Adams C. X.509 internet public key infrastructure online certificate status protocol - OCSP. RFC 6960, RFC Editor; 2013, URL http://www.rfc-editor.org/rfc/rfc6960.txt.

[3] Liu Y, Tome W, Zhang L, Choffnes D, Levin D, Maggs B, et al. An end-to-end measurement of certificate revocation in the web's PKI. In: Proceedings of the 2015 internet measurement conference. 2015, p. 183–96. http://dx.doi.org/10.1145/2815675.2815685.

[4] Montenegro G, Hui J, Culler D, Kushalnagar N. Transmission of IPv6 packets over IEEE 802.15.4 networks. RFC 4944, RFC Editor; 2007, http://dx.doi.org/10.17487/RFC4944, URL https://rfc-editor.org/rfc/rfc4944.txt.

[5] Vasseur JP. Interconnecting smart objects with IP: the next internet. Morgan Kaufmann; 2010.

[6] Shelby Z, Hartke K, Bormann C. The constrained application protocol (CoAP). RFC 7252, RFC Editor; 2014, URL http://www.rfc-editor.org/rfc/rfc7252.txt.

[7] Bormann C, Hoffman P. Concise binary object representation (CBOR). RFC 7049, RFC Editor; 2013.

[8] Selander G, Mattsson J, Palombini F, Seitz L. Object security for constrained restful environments (OSCORE). In: Internet-draft draft-ietf-core-object-security-15. IETF Secretariat; 2018, URL http://www.ietf.org/internet-drafts/draft-ietf-core-object-security-15.txt.

[9] Selander G, Raza S, Furuhed M, Vucinic M. Protecting EST payloads with OSCORE. In: Internet-draft draft-selander-ace-coap-est-oscore-02. IETF Secretariat; 2019, URL http://www.ietf.org/internet-drafts/draft-selander-ace-coap-est-oscore-02.txt.

[10] van der Stok P, Kampanakis P, Kumar S, Richardson M, Furuhed M, Raza S. EST over secure CoAP (EST-coaps). In: Internet-draft draft-ietf-ace-coap-est-06. IETF Secretariat; 2018, URL http://www.ietf.org/internet-drafts/draft-ietf-ace-coap-est-06.txt.

[11] Pritikin M, Yee P, Harkins D. Enrollment over secure transport. RFC 7030, RFC Editor; 2013.

[12] Raya M, Jungels D, Papadimitratos P, Aad I, Hubaux J-P. Certificate revocation in vehicular networks. In: Laboratory for computer communications and applications. EPFL; 2006.

[13] Rabieh K, Mahmoud MMEA, Akkaya K, Tonyali S. Scalable certificate revocation schemes for smart grid AMI networks using bloom filters. IEEE Trans Dependable Secure Comput 2017;14(4):420–32. http://dx.doi.org/10.1109/TDSC.2015.2467385.

[14] Larisch J, Choffnes D, Levin D, Maggs BM, Mislove A, Wilson C. Crlite: A scalable system for pushing all TLS revocations to all browsers. In: 2017 IEEE symposium on security and privacy. IEEE; 2017, p. 539–56. http://dx.doi.org/10.1109/sp.2017.17.

[15] Li Duan, Yong Li, Lijun Liao. Flexible certificate revocation list for efficient authentication in IoT. In: Proceedings of the 8th international conference on the internet of things. New York, NY, USA: ACM; 2018, p. 7:1–8. http://dx.doi.org/10.1145/3277593.3277595, URL http://doi.acm.org/10.1145/3277593.3277595.

[16] Bormann C, Ersue M, Keränen A. Terminology for constrained-node networks. RFC 7228, RFC Editor; 2014, http://dx.doi.org/10.17487/RFC7228, URL https://rfc-editor.org/rfc/rfc7228.txt.

[17] Kocher PC. On certificate revocation and validation. In: Proceedings of the second international conference on financial cryptography. London, UK, UK: Springer-Verlag; 1998, p. 172–7, URL http://dl.acm.org/citation.cfm?id=647502.728330.

[18] Goodrich MT, Tamassia R, Schwerin A. Implementation of an authenticated dictionary with skip lists and commutative hashing. In: Proceedings DARPA information survivability conference and exposition II, Vol. 2. 2001, p. 68–82. http://dx.doi.org/10.1109/DISCEX.2001.932160.

[19] Naor M, Nissim K. Certificate revocation and certificate update. IEEE J Sel Areas Commun 2000;18(4):561–70. http://dx.doi.org/10.1109/49.839932.

[20] Micali S. Efficient certificate revocation. Technical report, Cambridge, MA, USA: Massachusetts Institute of Technology; 1996.

[21] Micali S. NOVOMODO: Scalable certificate validation and simplified PKI management. In: Proceedings of the 1st annual PKI research workshop, Vol. 15. 2002.

[22] Lim TL, Lakshminarayanan A. On the performance of certificate validation schemes based on pre-computed responses. In: IEEE GLOBECOM 2007 - IEEE global telecommunications conference. 2007, p. 182–7. http://dx.doi.org/10.1109/GLOCOM.2007.42.

[23] Gibson S. An evaluation of the effectiveness of chrome's CRLSets. Gibson Research Corportation; 2014, URL https://www.grc.com/revocation/crlsets.htm.

[24] Wang M, Qian C, Li X, Shi S. Collaborative validation of public-key certificates for IoT by distributed caching. In: Proceedings of IEEE INFOCOM. Paris, France; 2019, p. 92–105.

[25] Wright RN, Lincoln PD, Millen JK. Efficient fault-tolerant certificate revocation. In: Proceedings of the 7th ACM conference on computer and communications security. New York, NY, USA: ACM; 2000, p. 19–24. http://dx.doi.org/10.1145/352600.352605, URL http://doi.acm.org/10.1145/352600.352605.

[26] Stevens M, Bursztein E, Karpman P. Announcing the first SHA1 collision. Google Security Blog; 2017, URL https://security.googleblog.com/2017/02/announcing-first-sha1-collision.html.

[27] Foundation OS. OSCP manpage. 2018, URL https://www.openssl.org/docs/manmaster/man1/ocsp.html.

[28] Jones J. The end of SHA-1 on the public web. Mozilla Security Blog; 2017, URL https://blog.mozilla.org/security/2017/02/23/the-end-of-sha-1-on-the-public-web/.

[29] Goel A, Gupta P. Small subset queries and bloom filters using ternary associative memories, with applications. In: Proceedings of the ACM SIGMETRICS international conference on measurement and modeling of computer systems. New York, NY, USA: ACM; 2010, p. 143–54. http://dx.doi.org/10.1145/1811039.1811056, URL http://doi.acm.org/10.1145/1811039.1811056.

[30] Zhang L, Choffnes D, Levin D, Dumitraş T, Mislove A, Schulman A, et al. Analysis of SSL certificate reissues and revocations in the wake of heartbleed. In: Proceedings of the 2014 conference on internet measurement conference. New York, NY, USA: ACM; 2014, p. 489–502. http://dx.doi.org/10.1145/2663716.2663758, URL http://doi.acm.org/10.1145/2663716.2663758.

[31] Mani SK, Durairajan R, Barford P, Sommers J. A system for clock synchronization in an internet of things. 2018, CoRR abs/1806.02474, arXiv:1806.02474.

Paper IV

AutoPKI: Public Key Infrastructure for IoT with Automated Trust Transfer

Joel Höglund[a], Simon Bouget[a], Martin Furuhed[b], John Preuß Mattsson[c], Göran Selander[c], Shahid Raza[a]

[a] *RISE Research Institutes of Sweden, Isafjordsgatan 22, 16440 Kista, Stockholm*
[b] *Technology Nexus Secured Business Solutions, Telefonvägen 26, 12626 Hägersten, Stockholm*
[c] *Ericsson, Torshamnsgatan 21, 164 83 Stockholm, Sweden.*

Abstract

IoT deployments grow in numbers and size, which makes questions of long-time support and maintainability increasingly important. Without scalable and standard-compliant capabilities to transfer the control of IoT devices between service providers, IoT system owners cannot ensure long-time maintainability, and risk vendor lock-in. We propose AutoPKI, a lightweight protocol to update the IoT PKI credentials and shift the trusted domains, enabling the transfer of control between IoT service providers, building upon the latest IoT standards for secure communication and efficient encodings. We show that the overhead for the involved IoT devices is small and that the overall required manual overhead can be minimized. We analyse the fulfilment of the security requirements, and for a subset of them, we demonstrate that the desired security properties hold through formal verification using the Tamarin prover.

Keywords: security, IoT, PKI, digital certificates, enrollment, embedded systems, Contiki

1. Introduction

IoT deployments are rapidly increasing, both in numbers and in fields of use, including for safety and security-critical applications. While there has been a related fast development of accompanying security solutions, there is currently a lack of services for long time robustness and secure management.

Security solutions, such as secure communication and authentication, have been adapted to suit also relatively resource-constrained Internet of Things devices. Based on these more primitive cryptographic mechanisms, more complex services such as key establishment and certificate enrollment for IoT have been proposed, and are getting standardised. Together they form the basis of creating a Public Key Infrastructure, PKI, capable of encompassing the Internet of Things.

One of the important application areas for a functional PKI for IoT is the creation of services for long time support and maintainability of IoT deployment. This is an area that is becoming ever more important with the spread of IoT deployments. For a sustainable IoT ecosystem, there must be mechanisms for trust transfer, how to efficiently update the IoT PKI credentials and shift the trusted domains in order to handle when the responsibilities of maintenance of IoT devices are shifted from one service provider to another. To amend the current gap where no well-defined protocol exists, in this work, we map out and propose a solution with references to existing protocols together with new proposals where there currently are no specified mechanisms.

To have a real-world impact, the proposed solution must, in addition to stringent security requirements, be scalable, resource-efficient, and as far as possible build on agreed standards. To achieve scalability, the overhead in terms of manual labour must be kept minimal. Based on the above description, the concrete problem formulation becomes: what is the minimal procedure needed, in terms om manual in-

Email addresses: joel.hoglund@ri.se (Joel Höglund), simon.bouget@ri.se (Simon Bouget), martin.furuhed@nexusgroup.com (Martin Furuhed), john.mattsson@ericsson.com (John Preuß Mattsson), goran.selander@ericsson.com (Göran Selander), shahid.raza@ri.se (Shahid Raza)

tervention, to securely shift the operation of one IoT device from one service provider to another?

The criteria for a complete and successful transfer of trust is when all involved IoT devices have enrolled and received new operational certificates, making them recognized as valid participants of the target organization PKI while meeting all the requirements defined for the proposed protocol.

The main contributions of the paper are as follows:

- A design of a lightweight schema for trust transfer, which allows the control of IoT deployments to be shifted between service providers with minimal manual overhead.
- A feasibility study using a prototype implementation for constrained IoT devices.
- A theorem prover based security analysis for critical protocol security requirements.

The rest of this paper is organized as follows: Section 2 presents a brief discussion of vital concepts for the proposed protocol. Section 3 presents related work. Section 4 gives a motivating scenario. Section 5 presents our threat model and assumptions. Section 6 formalizes the requirements of the proposed protocol. Section 7 explains the lifecycle operations of a PKI-enabled IoT device, including new PKI optimizations which make the security functionality sufficiently lightweight, and optional security mechanisms which are needed for the strongest security guarantees. Section 8 presents the detailed scenario for AutoPKI together with our proposal for formalizing the steps into a protocol with a maximal level of automation. Section 9 presents the results of the feasibility evaluation, including a discussion from the business perspective (9.4). In section 10 we present the security assessment of the requirements, before concluding the paper.

2. Towards Automated PKI: background technologies and challenges

This section introduces concepts and mechanisms which are important for the creation of a PKI for IoT, which are referred to in the rest of the paper.

2.1. Security services and PKI

When designing more complex security services, encapsulating lower-level mechanisms as basic security services provides a useful abstraction.

To establish and maintain trust from a system perspective, authentication and authorization are two key security services.

An authentication service provides the necessary trustworthy binding between an entity and, in the case of a PKI, a public key. This functionality in turn can be used to create an authorization mechanism, which ensures that an authenticated actor can perform exactly the actions they are entitled to and no other actions.

The full system needed to manage the authentication services and their artefacts; certificates, keys, policies and roles, forms a Public Key Infrastructure, PKI.

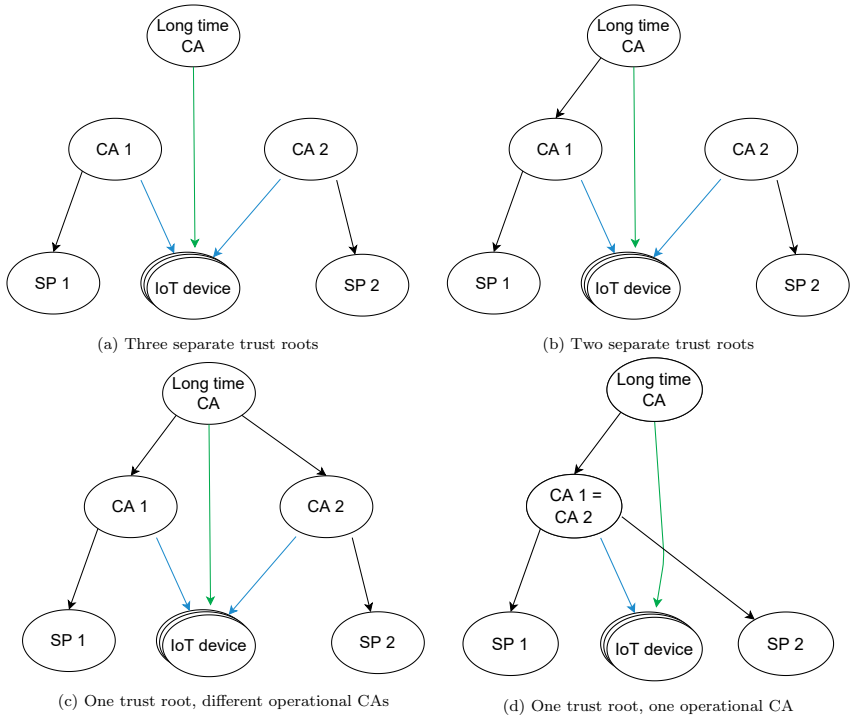
2.2. PKI hierarchies

A cornerstone of PKIs is authentication through publicly available keys, keys encapsulated in certificates signed by a certificate authority, CA. The certificate authorities are identified by their certificates which can be either self-signed or signed by another CA. This system of signed certificates forms hierarchies up to the self-signed top/root CAs. The resulting chains of certificates can be verified up to the top nodes, but the self-signed root nodes need to be already trusted [1].

The party traversing the certificate chain and performing the authentication must have access to the top node certificates, in a trusted manner. Practically for IoT devices, this means they should be equipped with the necessary root certificates in their local trust stores. The placement can happen before deployment through factory pre-programming, and/or dynamically through enrollment operations[1].

The IoT devices act as leaf nodes at the lowest layers of the CA hierarchies, together with the service provider servers, with which the devices need to communicate. Figure 1 illustrates some alternatives for CA hierarchies, depending on the trust relations between the entities, involving two different service providers and IoT devices. For the task of securely transferring control of IoT deployments, the implications of the different CA hierarchies illustrated in 1 are the following: If the trust hierarchies are completely separated, as in 1a, the IoT device needs to be equipped with a root certificate for CA1 in advance of

[1]The last years' advancements in both IoT capabilities and solutions targeting IoT have made PKI enrollment solutions feasible for IoT as well [2, 3].



(a) Three separate trust roots

(b) Two separate trust roots

(c) One trust root, different operational CAs

(d) One trust root, one operational CA

Figure 1: Different options for CA hierarchies. All arrows represent certificate issuing, green arrows for factory certificates, blue arrows for operational IoT certificates

the first enrollment, to be able to authenticate CA1. Correspondingly the device must be updated with a root certificate needed to authenticate CA2 in advance of the trust transfer. In the case where the CA1 is a sub-CA of the permanent CA, as shown in 1b, it is sufficient to provide an update with the a root certificate for CA2. For the relationships shown in 1c and 1d, all entities can be authenticated with only prior access to the certificate of the permanent CA.

There can be performance reasons to go beyond the minimal requirements for which root certificates that must be added to the IoT device trust store. By providing additional certificates from the servers with which the IoT device needs to communicate, certificate reference-based authentication can be enabled. This allows the communicating parties to send hashes of certificates, which the counterpart already possesses, rather than full certificates and certificate chains. This type of reference-based public key authentication is for instance supported in EDHOC key establishment [4].

2.3. The concept of trust in PKI

From the PKI perspective, trust is something that can be established between two or more parties with help from the infrastructure and the concept of trusted root nodes [1]. The more general discussion on trust is a large topic with a multitude of overlapping definitions. Two perspectives of relevance for this work are: A *systems perspective*, defined as trust that the system will provide the desired services, without unintended or undesired side effects. Complemented by an *organisational perspective*, defined as trust that the involved parties will live up to the obligations they have agreed to, formalized through one or more contracts. Without the latter, the involved parties will not reach the former, the confidence in the system. For much more in-depth discussions on the concept of trust see [5]. In order

to automatize and scale up the number of operations it becomes crucial to provide mechanisms such that all relevant obligations can be tracked and audited to the degree deemed necessary, with low overhead.

3. Related Work

Ownership Transfer: The closely related area of ownership transfer for IoT has been studied from the perspectives of single-user privacy protection and custom non PKI-based solutions. In [6] the focus is on the privacy and protection of smart home device data. A custom solution for creating user profiles, and automatically detecting ownership changes for individual devices is presented. Compared with our efforts, this is on the opposite end of standard compliance, where automatization is used not for reducing costs and handling scale, but for the convenience of individual users and end-user privacy protection.

In [7] a custom non-standard solution is proposed, where the authors specifically do not assume PKI support from the devices. Their focus is on ensuring forward and backward security between the former and new owners. The solution is based on symmetric keys and a trusted third party. Despite the differences in assumptions concerning PKI support and standard compliance, they investigate a similar scenario as we do, and some of their requirements have relevance to our solution as well.

PKI alternatives: For scenarios with IoT devices that are not capable of running PKI mechanisms at all, a number of different custom-made solutions have been proposed. Even for the most constrained devices such as RFID, there are proposals for mutual authentication which, although the master secrets are non-replaceable, include mechanisms to avoid replay attacks [8]. For devices with more capabilities [9] presents a hierarchical model, as well as a comparison with other similar solutions.

These solutions do not offer real end-to-end security, introduce complex intermediaries, and are currently not being standardised. Parts of the solutions which are proposed specifically for local Wireless Sensor Networks (WSNs), could be used complementary with full-fledged PKI mechanisms to solve issues related to initial bootstrapping and initial link layer security key distribution.

4. E-health use case: IoT ownership change and AutoPKI

To introduce the trust transfer problem, and give a motivating example that illustrates some of the involved actors, we present a brief high-level use case where the proposed protocol applies.

A municipality wants to invest in e-health solutions to strengthen its elderly care monitoring capabilities. The goal is to equip beds with a number of wireless sensors to detect movement and rise an alarm if the person in the bed is on the brink of falling out. Since the municipality lacks the operational resources themselves, they procure the purchase, installation, and operation from an external service provider, **SP1** hereafter. To prevent vendor lock-in, the municipality demands that open standards must be used and that the capabilities to shift the service provider must be ensured. The monitoring system must also be easy to integrate with existing systems in the municipality for the handling of personnel and access to personal data.

After some time of operation, the municipality wants to upgrade their system for personnel access. As the new solution would require costly modifications to work with the existing IoT service provider they decide to swap service providers with someone already capable of interacting with the new personnel access system.

The municipality does a new procurement and instructs the original service provider to hand over operations to the selected new service provider, **SP2**. The new service provider securely gains control of the IoT devices and continues the monitoring services.

For the interactions involved in the handover to be both secure and efficient in terms of minimal manual efforts, a protocol is needed. In the following, we present an enabling PKI environment, details on the required interactions, and how our proposed protocol fulfills desired security properties while enabling a high degree of automatization.

5. System and threat model

The main targets of the proposed protocol are IoT deployments with device-to-server communication as the most common communication pattern. We consider IoT devices that are constrained in terms of both bandwidth and computational resources. They are computationally powerful enough to perform

asymmetric crypto operations, but to keep energy budgets limited, computationally expensive operations must be used sparsely. In addition, devices are often communicating using radio, over wireless low-power networks, which adds packet size constraints and the need to handle packet losses.

We assume the Dolev-Yao adversarial model [10], where the potential attackers have full access to the network. They can eavesdrop any message being sent, record messages, and inject both old and modified messages into ongoing communication. Regarding the IoT devices themselves it is assumed that they are not physically tampered with. Regarding the cryptographic functions used, it is assumed that they cannot be broken within the relevant time span.

As a baseline, we assume that the involved service providers establish mutual trust, in such a way that they will not actively attack the counterpart. Unless prevented, they might still be interested in gathering leaked data. We return to these assumptions in relation to remote attestation (section 8.3) and our formal analysis, where we also consider the case where SP1 acts as an attacker (section 10).

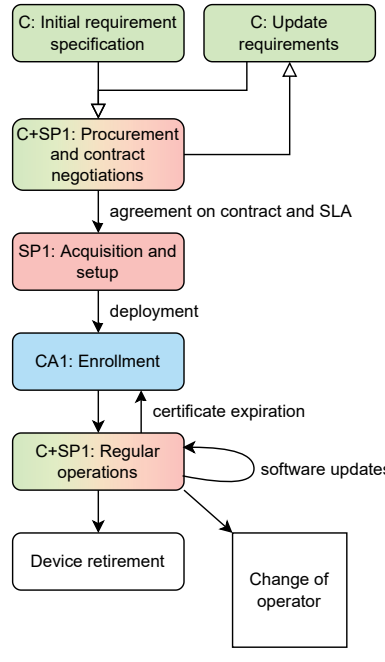


Figure 2: The IoT life cycle

6. Requirements

Based on the above description of challenges and threats we arrive at the following requirements for a trust transfer protocol.

FR1: Integrity protection. If the protocol terminates, we are certain that an attacker has not been able to modify any protocol message received by the IoT device.

FR2: Man-in-middle resistance. An adversary cannot use eavesdropped traffic to successfully hijack a transfer protocol session.

FR3: Forward security. The old service provider shall not get access to any private data which can compromise the privacy of the new service provider and its onward operations.

FR4: Backward security. The new service provider shall not get access to any private data belonging to the old service provider, which is not explicitly agreed to be shared.

These functional requirements make statements about the desired state at the end of a completed protocol run. In addition, we identify the following non-functional requirements, related to scalability and interoperability:

NFR1: Automatization. The protocol must offer the desired functionality with a minimum of manual intervention.

NFR2: Resource efficiency. The protocol must allow all operations directly involving the IoT devices to be sufficiently lightweight to run on relatively resource-constrained devices.

NFR3: Standard compliance. To be feasible for adoption by the industry, the protocol must build upon existing and ongoing standardization efforts wherever possible.

7. AutoPKI life cycle

The main enabler for an IoT device to gain access to a number of crucial security services is to be part of a PKI. It is necessary for the goal of offering standard-based interoperability and preventing vendor lock-in.

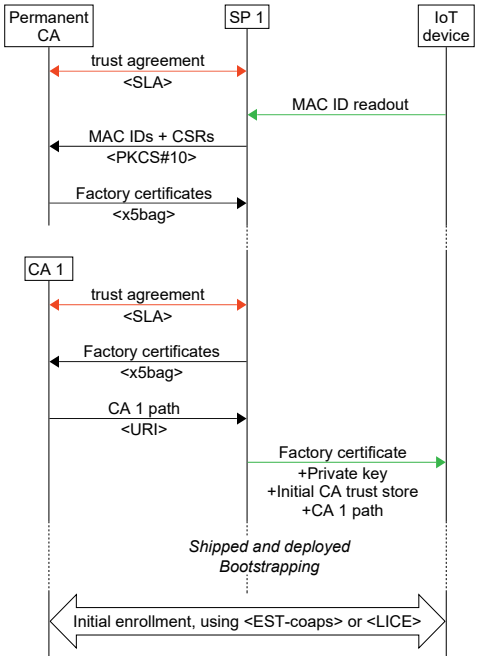


Figure 3: The IoT device's initial life cycle stages, showing the standards used for setup and enrollment. Red arrows correspond to operations where manual intervention is expected. Green arrows are deployment-specific, while black arrows are standard-based and fully automated.

Making resource-constrained IoT devices parts of a PKI is a nontrivial task. To give the context for how the task can be achieved, we present existing and proposed solutions for how an appropriate environment for trust transfer can be created. We cover the first stages in the PKI for IoT life cycle, while adhering to existing standards for all steps wherever possible. A high-level overview of the life cycle is shown in figure 2. A more detailed diagram of the initial life cycle phases is given in figure 3.

Scope and limitations. We address the issues directly related to public key management, needed to guarantee the required security services. In addition, a deployment might have other functional requirements such as downtime constraints which need to be treated separately and be factored in when scheduling the actions to be performed.

Involved actors. In the first steps of the life cycle description the following actors and roles (briefly mentioned in 4) are relevant to specify: **CA**: A well-established and reliable certificate authority (Permanent CA): A certificate authority that can be trusted for an extended period of time, suitable for providing long-lived trust root(s) to the initial device truststore.

SU: The IoT service user, who is also the system owner (owner/user). This is the actor (company or organization) who uses the IoT system to achieve a goal. The goal can be internal, as a service end user, or as a part of providing services to other third parties.

SP1: The initial IoT service provider; the company which is in charge of configuring the IoT devices, installing them and, initially, maintaining them.

CA1: The initial operational CA; the certificate authority with which SP1 has made an agreement to provide operational certificates, including certificate renewals when needed. It can be the same as the permanent CA.

7.1. Procurement, SLAs and Smart Contracts

The starting point for the scenario is that a company or an organization, SU, has identified a need that can be fulfilled with an IoT system. The IoT system needs to be clearly specified, ordered, deployed, and thereafter maintained. The deployment could be within the SU's own premises, or within any other area where they have obligations to perform monitoring or offer services that can be aided by the IoT installation.

As part of the procurement process, the SU specifies service-level agreement conditions that must be met. In this work, we focus on those directly related to PKI and trust management. This includes specifying that the chosen IoT service provider must be able to transfer the role of system maintainer to a new service provider without breaching agreed security guarantees. The demands could also specify additional criteria for minimal service disruptions during any system update.

SLAs and Smart Contracts. In line with the efforts to lessen the burden of manual intervention in any software service operation, service level agreements, SLAs, can be used to formalize contractual agreements in a manner suitable for automated checking [11]. Specifically, to lessen the dependency on additional trusted third parties, smart contracts (SC)

running on blockchain infrastructure have been proposed for the automatizing and monitoring of service-level agreements. This type of solution could potentially further remove the need for human involvement. Early proposals such as [12] considered cloud environments, while newer work address also IoT scenarious. For example, in [13] the authors propose a Hyperledger Fabric-based system for SLA compliance assessment, with applications for IoT. Smart contracts themselves cannot directly access data outside of their blockchain environment, hence a solution for monitoring service parameters will depend on so-called oracles, data feeds that connect the contracts to off-chain information [14].

The field of using SCs for SLA monitoring is an active area of research, where more work is needed before the solutions have reached industry maturity. From the perspective of our trust transfer proposal, details on how SLAs are monitored and acted upon are outside the scope.

An IoT provider who accepts the required conditions gets the order. Together the SU and the IoT service provider, hereafter SP1, formalize the requirements in a contract containing the agreed upon service level agreement. Besides quality of service specifications, the parties clarify the service endpoints to be used for accessing services and data.

7.2. Device acquisition, factory credential and firmware preparations

The SP1 acquires IoT devices that meet the functional sensing and actuation requirements of the customer, as well as the non-functional requirements regarding security protocol support and update capabilities. The section corresponds to the Acquisition and setup stage of figure 2.

A vital part of a PKI capable of handling IoT devices with minimal manual intervention is how to prepare the devices, such that they can perform initial authentication operations once deployed. To perform mutual authentication the device must be able to identify itself to a server and have means to authenticate the server with which it is communicating.

The practical solution is to pre-program devices with a secret factory key and a factory certificate, plus an initial truststore containing server certificates. For the general case, the device needs both the server certificates forming the certificate chain up to the CA root of the factory certificate, plus additional root certificates to authenticate servers with certificates belonging to other root CAs.

All IoT devices come with unique IDs when they are delivered from the manufacturer. In the following, we assume that the SP1 uses the unique device IDs provided by the manufacturer as the basis for the device names in the factory certificates. The device IDs might be matched between a list of IDs and stickers on the devices, or through QR codes, or extracted through some programming port. The exact measures will depend on the device type at hand.

If the IoT device is equipped with a secure and protected hardware module, it can implement the 802.1AR standard for Secure Device Identities, DevIDs [15]. The hardware requirements make the standard less suitable for the most constrained IoT devices, but for sufficiently capable devices the module can be used to offer also physical tampering protection.

The SP1 has an agreement with a CA that they trust, allowing them to order long-lived factory certificates. This agreement must be compatible with the conditions in the SLA made with the SU regarding the long-time availability of the CA. The IoT factory certificate should have a lifetime corresponding to the lifetime of the IoT device itself. Hence it is extra important to strive for access to an entity that can reply to inquiries about the certificate revocation status for all of the expected device lifetime.

The SP1 generates cryptographic keypairs and creates certificate signing requests, CSRs, for all IoT devices that should receive factory certificates. The requests are communicated to the permanent CA, which creates factory certificates and sends them back. This communication takes place over the regular Internet and is therefore not restricted in terms of bandwidth. The certificate signing requests can therefore be sent using the verbose PKCS#10 standard [16]. Since the targets are IoT devices, it is beneficial if the resulting factory certificates are compact. Using the proposed C509 certificate format results in significantly more compact certificates compared with X509, especially when using ECC crypto keys, offering the strongest cryptographic guarantees at relatively short key lengths [17] (see also section 7.3 below). The CSRs as well as the replies can be sent one by one as needed or collected and sent in batches. All of the communication happens over a TLS-secured communication link. If the key pairs for the factory certificates are generated outside of the

target IoT devices, extra care must be taken to ensure the private keys are not leaked. Preferably they should be kept in a Hardware Security Module, HSM, and destroyed on the server side after being uploaded to the target devices.

It is worth emphasising that the long-term factory certificates should be restricted in terms of operational capabilities, allowing only the authenticating of the device for doing an enrollment operation and special device updates. The initial post-deployment enrollment is what assigns an operational certificate to the device, with the needed capabilities to operate within the SP1 infrastructure. Hence the devices need to be given information on which CA to contact for operational enrollment.

SP1 contacts a CA which will act as the operational CA, CA1. Unless CA1 is the same as the permanent CA, the operational CA needs to be updated about the identities of the devices to which it should be prepared to grant operational certificates to. This is solved by sharing the factory certificates. A proposed format with minimal overhead is an x5bag, in which certificates are wrapped in byte strings, and placed in a CBOR array [18]. In return, the SP1 is given the URI that the IoT devices should contact for the enrollment of operational certificates.

The data exchange between the SP1 and the CA1 can be fully automatised, given a pre-existing contract which specifies the rights for any device, which can authenticate itself using a private key corresponding to one of the shared factory certificates, to request an operational certificate.

At this point, the SP1 is equipped with the data needed to do the initial programming of devices, which provides the device with its initial firmware, including a factory private key, factory certificate, initial truststore, an SP server URI and information on the CA-URI. The initial programming and data transfer to the IoT devices take place in a trusted environment.

The steps covered until this point are illustrated in figure 3 up until "Shipped and deployed".

7.3. C509 Certificates

One of the obstacles to using PKI for IoT has been the prohibitive overhead created by having to handle lengthy X.509 certificates. To reduce the overhead we have proposed a more compact encoding using CBOR, the C509 certificate format [17]. Besides the savings due to CBOR being more compact than ASN.1, the encoding makes use of domain knowledge to extend the savings beyond general compression. It includes compression of elliptic curve points, replacement of long OIDs with short integers and removal of known static fields. The format can either be used natively, if the involved CAs and servers understand the format or in a compatibility mode where the certificate signature verification is done on a reconstructed X509 certificate.

The format of the current C509 version is given by the following CDDL:

Listing 1: C509Certificate

```
C509Certificate = [
   TBSCertificate,
   issuerSignatureValue : any,
]

; The elements of the following group are used in a
    CBOR Sequence:
TBSCertificate = (
   c509CertificateType: int,
   certificateSerialNumber: CertificateSerialNumber,
   issuer: Name,
   validityNotBefore: Time,
   validityNotAfter: Time,
   subject: Name,
   subjectPublicKeyAlgorithm: AlgorithmIdentifier,
   subjectPublicKey: any,
   extensions: Extensions,
   issuerSignatureAlgorithm: AlgorithmIdentifier,
)

CertificateSerialNumber = ~biguint

Name = [ * RelativeDistinguishedName ] / text / bytes

RelativeDistinguishedName = Attribute / [ 2*
    Attribute ]

Attribute = ( attributeType: int, attributeValue:
    text ) //
( attributeType: ~oid, attributeValue: bytes )

Time = ~time / null

AlgorithmIdentifier = int / ~oid /
[ algorithm: ~oid, parameters: bytes ]

Extensions = [ * Extension ] / int

Extension = ( extensionID: int, extensionValue: any )
     //
( extensionID: ~oid, ? critical: true,
  extensionValue: bytes )
```

The initial focus was to encode all certificates following the IoT profile given in RFC 7925 [19]. The

work has been expanded to specify identifiers (using short integers) for a more general set of extensions, attributes and algorithms for keys and signatures. Together these cover a wide range of well-behaved cases, while still allowing more lengthy byte representations of rare cases. Certificates compliant with a number of significant certificate profiles, such as IEEE 802.1AR, CNSA, and RPKI, can be encoded, resulting in a good general RFC 5280 coverage. Consequently, unless the service provider has very specific needs, not only the IoT certificate but also the server certificates that the IoT device needs to handle can be compactly C509 encoded, greatly reducing the overhead for PKI-related communication and certificate handling.

7.4. Deployment and initial enrollment

The device is physically installed in its target environment. Depending on the contract between the SP1 and the SU, this can be done by the SP1, by the SU themselves or by a trusted third party.

A full specification of a concrete deployment needs to address further practical details, such as bootstrapping, seeding of the device time source and if there are policies to use for re-assigning dynamic MAC addresses. These issues are highly dependent on the operator and the deployment scenario. For example, how to securely provision a time source is an open issue. The latest available relevant standards, such as BRSKI, allow IoT devices to ignore the certificate validity periods during initial authentication if the device has not yet been given a reliable current time [20]. In the following, we assume that the deployment-specific bootstrapping issues have been solved.

Upon startup, the IoT device contacts the CA1 to do initial enrollment and be given an operational certificate. The device authenticates itself through the factory certificate which is registered with the CA1. The factory certificate also serves to authorize the request for an operational certificate. Mutual authentication is done as part of establishing a secure channel, using either a DTLS or EDHOC handshake.

The IoT device sends a certificate signing request to the operational CA, using the proposed C509 CBOR format [17], or the less compact PKCS#10-format for legacy systems. The CA replies with an operational certificate, in either C509 or X509 format.

The choice of format depends on whether the enrollment is done following the older EST-coaps [21] or the proposed more flexible EDHOC-based enrollment protocol [3]. If the device is using the proposed compact enrollment protocol the enrollment message exchange can be encapsulated already inside an EDHOC handshake.

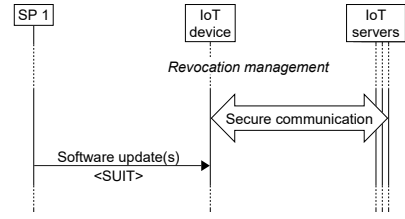


Figure 4: The IoT interactions during normal operations.

IoT devices with sufficient computational resources are capable of generating the key pair themselves, which is the preferred solution whenever available, as the private key never needs to leave the device. For the most constrained devices, a similar enrollment approach is feasible also for requesting a server-generated key pair.

Note on trust: The IoT device trusts the operational CA, given that the device has authenticated it during the handshake, and believes the given CA-URI to be valid.

7.5. Normal operations

After the enrollment, the IoT device is equipped with an operational certificate which is recognized by the servers it needs to communicate with, and has an updated truststore which ensures that the device can perform authentication of all endpoints of relevance.

During normal operations, the SP1 ensures the IoT devices are kept up to date with software upgrades, following the SUIT architecture mechanisms [22]. Before the operational certificate expires, the device will do re-enrollment with the CA1. The normal operations are illustrated in figure 4.

8. IoT Trust Transfer

8.1. Introduction and problem formulation

A motivating high-level use case was given in 4. In general terms, the IoT service user, SU, decides that they want to switch service providers for their IoT services while maintaining their existing deployments and installations. This is the high-level goal which should be achieved with a minimum of service

disruptions and with a minimal need of human intervention.

Today the operations needed for a secure ownership transfer between operators are insufficiently specified. Without clear protocols, the task becomes at best very labour intensive with several manual steps which need to be tailor-made to the specific scenario. At worst, impossible.

In the following we detail the needed steps, referring to existing standards where applicable, and proposing solutions for the missing parts. An illustration of the resulting protocol flow is given in figure 5, and the pseudocode for the main actors is listed in the three procedures below.

8.2. Additional involved actors

In addition to the actors introduced in 7, the following are included.

SP2: A second IoT service provider; the company selected by the SU to overtake the responsibilities to maintain the IoT devices from SP1.

CA2: second operational CA; the certificate authority with which SP2 has made an agreement to provide operational certificates.

8.3. Preparations for operator change

If the need arises for the customer to switch service providers, the initial contract (see 7.1) specifies that the current service provider SP1 needs to contact the designated new service provider, SP2. This step might include manual efforts, in forming a specific contract which specifies the details of transactions which are about to take place. Specifically, it needs to specify a starting date from when SP2 must be ready to start maintaining the IoT devices, within the total allowed time span defined by the SU.

SP1 and SP2 need to agree on the state of the IoT firmware, in particular, which services and which versions of the services the IoT devices will provide at the time of shifting the maintenance responsibilities. A solution to automatize the auditing of the IoT device state is to use remote attestation.

Remote attestation, RA, is an advanced security service that has attracted increased attention over the last couple of years. In remote attestation, a device produces a proof of its current state, regarding software, hardware, or both, which is checked and verified by a trusted third party to be in accordance with the expected output.

Procedure SP1 procedures

Overall prerequisites Existing SLAs between SP1 and SP2, as well as between SP2 and CA2

```
procedure trust_transfer()
Input : SP2 URI, List of certificates
prepare UpdateInfoList
secure_send(SP2, UpdateInfoList)
wait_for(ServerTransferMessage)
receive(ServerTransferMessage)
if valid(outer_signature_SP2) then
    foreach IoT_device ⊂ UpdateList do
        prepare IoTTransferMessage tm:
        signature_SP1 ← sign_SP1 (payload_SP2,
          signature_SP2, fallbackURI)
        tm ← (payload_SP2, signature_SP2,
          fallbackURI, signature_SP1)
        iot_device_update(IoT_device, tm)
    end
else
    abort and rise error
end
end procedure
```

```
procedure iot_device_update(td, tm)
Input : target device td, IoTTransferMessage tm
if final sw updates then
    perform_iot_update(td)
end
send(tm)
end procedure
```

To offer the strongest security guarantees, RA relies on access to a trusted hardware component for the device being attested, such as TPM or Arm TrustZone. More constrained IoT devices do not have access to these dedicated hardware resources. There are also software-based RA solutions and hybrid versions with limited requirements on protected memory areas. There is active research in the area [23] as well as a large ongoing IETF standardisation effort [24].

In addition to agreeing on the RA details, the parties will declare which certificates to be used for signing protocol data.

When the trust relationship is established and a transfer specification contract is formed, the old service provider can share device information with the new service provider.

The information exchange needs to contain the following data items:

- The factory certificates for every involved IoT

Procedure SP2 procedures

```
procedure info_sharing(uil)
Input : UpdateInfoList uil

foreach cert ⊂ uil do
 | assert cert_update_period ⊆ SLA_update_period
end

Parse factory certificates into x5bag collection
secure_send(CA2, x5bag)
wait_for(CA2_path_msg)
receive(CA2_path_msg)

Prepare ServerTransferMessage stm:
foreach cert ⊂ uil do
 | Prepare payload_IoT_ID:
 | set time limits
 | if use RA then
 | | payload_IoT_ID.RA_URI ← RA_URI
 | end
 | payload_IoT_ID.updateURI ← SP2 server URI
 | if update before enrollment then
 | | payload_IoT_ID.updateFlag ← TRUE
 | end
 | payload_IoT_ID.enrollURI ← CA2_path
 | payload_IoT_ID.signature ← sign_SP2
 |  (payload_IoT_ID)
end
Add payloads into stm
sign_SP2 (stm)
secure_send(SP1, stm)
end procedure
```

```
procedure cm_processing(cm)
Input : ConfirmationMessage cm

if valid(outer_signature_cm:IoT.op_key) then
 | if valid(inner_signature_cm:IoT.factory_key)
 |  then
 | | Include IoT in set of valid devices
 | end
end
if ∀ IoT: cm is received then
 | TrustTransfer completed
end
end procedure
```

Optional remote attestation
Receive and validate the results of remote attestation

Optional software updates
provide software updates to requesting IoT devices

Procedure IoT device procedures

```
procedure trust_transfer_start(tm)
Input : IoTTransferMessage tm

if valid(signature_tm:SP1) then
 | parse, save and update: raURI,
 |  updateURI_SP2, enrollURI_CA2,
 |  fallbackURI_SP1, payload.signature_SP2
 | reset
end
end procedure
```

```
procedure trust_transfer_continue()

if update before enrollment then
 | check_for_updates(updateURI_SP2)
end
if use RA then
 | prepare evidence
 | perform RA using raURI
end
if enrollment(enrollURI_CA2) is successful then
 | if valid(payload.signature_SP2) then
 | | spURI ← updateURI_SP2
 | | prepare ConfirmationMessage cm:
 | | inner_sign_cm ← sign_factory_key(IoT_ID)
 | | cm ← sign_new_op_key
 | |  (spURI, inner_sign_cm)
 | | send(SP2, cm)
 | | resume normal operations
 | end
else
 | abort, rollback pointers  contact SP1 using
 |  fallbackURI_SP1
end
end procedure
```

device for which the responsibility of maintenance is about to be transferred from SP1 to SP2.

- The earliest and the latest switch-over time for each involved device.

- Firmware code and/or service description(s) of the software that the IoT device is running. There are several possible alternatives, which are affected by if SP2 is to continue using the same software that is already available on the devices, and to what degree the source code of the components is shared. We propose the state of the device software is shared through sharing references to the relevant SUIT manifests.

- Optionally, if remote attestation is to be performed, SP1 needs to share the information needed for a verifier to evaluate the response from the device being attested.

The mandatory information represented as a CBOR array is specified in CDDL as follows:

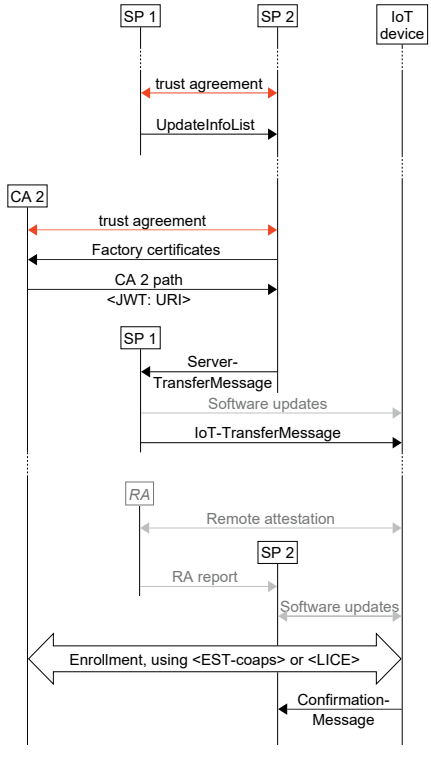


Figure 5: AutoPKI, Operator change. Automated operations in black, optional operations in gray

Listing 2: UpdateInfoList

```
UpdateInfoList = [* DeviceUpdateInfo]

DeviceUpdateInfo = (
  factoryCertificate:   TBSCertificate,
  updateTimeNotBefore:  Time,
  updateTimeNotAfter:   Time,
  versionInfo:          (suit-manifest-seq-number,
                         suit-reference-uri),

)
```

This update information, encoded as an array of pairs, is signed by SP1 using JSON Web Signatures. Described so far are the interactions up until the UpdateInfoList-arrow in figure 5.

The designated SP2, in turn, needs to perform the required actions with an operational CA of choice, that will become responsible for new operational certificates, corresponding to the procedure that SP1 previously carried out together with CA1 before the initial deployment. In short, given an existing trust relationship between the parties, forward the factory certificate list to the CA2, and get a CA-URI token back. In addition to these administrative steps, the SP2 configures an update server endpoint, and prepares a ServerTransferMessage, following the format given below.

Listing 3: ServerTransferMessage

```
TransferMessageList = [* (TransferMessageInfo,
                          Signature)
                        ]

TransferMessageInfo = (
  ResetTimeNotBefore:  Time,
  ResetTimeNotAfter:   Time,
  raURI:               bstr / null,
  updateURI:           (bstr, bool),
  enrollURI:           bstr,
)
```

If remote attestation is used, the ServerTransferMessage contains the remote attestation URI. The updateURI is set to the SP's own update server, with a flag to indicate if devices should contact the update server before the enrollment. We assume the same URI can also be used by the device to report data, hence it will be used to update the main service provider pointer after a successful transfer operation. The CA2 path is set as the enrollURI. This payload is signed, and the transfer info plus signature is included in a list with items for each target IoT device.

8.4. Performing the service provider change

When SP1 has received the ServerTransferMessage from SP2, it parses the set of claims, copies the fields, and adds a fallback URI which is set to the SP1 update server, into individual IoTTransferMessages for each target IoT device. SP1 can, if needed, perform a last remote software update to the target devices. The set of transfer message claims are treated as the payload of COSE_Sign1 objects, which are signed, resulting in signed CBOR Web Tokens sent to each target IoT device.

Listing 4: IoTTransferMessage

```
IoTTransferMessage = (
  ResetTimeNotBefore:  Time,
  ResetTimeNotAfter:   Time,
  raURI:               bstr / null,
  updateURI:           (bstr, bool),
  enrollURI:           bstr,
  SP2_signature:       bstr,
  fallbackURI:         bstr
)
```

After the IoT transfer message has been received and validated, the individual IoT devices reset themselves back to a state agreed upon in the agreement between SP1 and SP2, where the resulting state includes the updated information about the new server endpoints to contact after reset.

Upon restarting, the device will optionally first contact the remote attestation server, to participate in a RA challenge response. Thereafter, depending on the updateURI flag, it can contact the SP2 update server. Thereafter the device does re-enrollment with CA2 The device will receive a new operational certificate, recognized by the relevant SP2 endpoints, as well as additional needed truststore updates.

It should be noted that the device truststore after the last SP1 operation must contain certificates capable of authenticating CA2. Additionally, if remote attestation is used, or the optional pre-enrollment SP2 updates are needed, the trust roots of the RA server and the SP2 update server endpoint must be present in the trust store. The least complex scenario is when the SP2 endpoint can be authenticated by certificates in the IoT truststore in its initial state. This is trivially the case when the CA hierarchies correspond to 1c or 1d. Otherwise, there must be a truststore update operation that is not rolled back by the SP1 reset operation.

In the same way, as in the initial enrollment situation, the IoT device trusts the new CA, given the device is capable of authenticating the server during a secure session establishment.

If any of the steps permanently fails, such as a remote attestation failure, or failure to authenticate with the CA2 or the SP2 update server, the IoT device will use the fallback URI to once more contact the SP1 update server. For completeness, SP1 might now require the device to perform a new remote attestation, to verify its state after the interactions with SP2.

To prevent impersonation attacks, our formal modelling showed the necessity to conclude the trust transfer with a commit phase, using a confirmation message after the successful enrollment. Only at this point the IoT device can validate the SP2 signature contained in the transfer message, redirect the permanently stored local SP pointer from SP1 to SP2 and send a confirmation message to SP2. The confirmation message is constructed by the device by first creating an inner signature, by signing the device id using its factory key. Thereafter signing the SP2-URI and the inner signature by its new operational key for the SP2 domain. Instead of sending two separate signatures, this double signature is sufficient as the payload, by which the IoT device can demonstrate both its identity and having performed a successful enrollment.

8.5. Continued operations

After the new enrollment operations, the device is fully reconfigured as part of the SP2 management domain and will communicate with the SP2 servers based on its new configuration.

8.6. Certificate revocation checking

In the proposed protocol the effort to check the revocation status of IoT device certificates, both operational and long-term factory certificates, is put on the Internet servers. They can handle existing relatively heavy-weight protocols such as OCSP or CRLs. To extend revocation-checking capabilities to constrained devices, more lightweight mechanisms are needed, as proposed in [25].

9. AutoPKI: feasibility study

Based on the protocol design goals, to target resource constrained devices, it is critical to show that the protocol overhead is sufficiently small to match IoT device capabilities. In the following, we validate the proposed building blocks in terms of messaging, computational and memory overhead. Our tests have been performed on the nRF52840-DK platform, which is a relatively powerful but relevant target IoT device with an Arm Cortex-M4, 802.15.4-radio and 256 kB RAM.

9.1. Messaging overhead

To demonstrate the feasibility of the protocol, and the acceptable overhead for IoT devices we calculate the sizes of involved messages and transactions.

As can be seen in table 1 the IoTTransferMessage and the ConfirmationMessage, the protocol messages specifically sent to and from the IoT devices, constitute only a few hundred bytes. This is when using 256-bit keys resulting in 64-byte signatures, plus CWT encapsulation. Since this is small compared with the handshake and enrollment operations, networks and devices which are capable of handling the related PKI operations will have no difficulties with the added AutoPKI messages.

Table 1: Protocol message size in bytes

Message/Operation	CoAP size (B)	
	DTLS, X.509	*EDHOC, C.509*
Protocol specific		
DeviceUpdateInfo	> 400	> 230
Factory certificate	> 320	> 150
IoTTransferMessage	> 287	> 287
ConfirmationMessage	> 90	> 90
Related operations		
Handshake	> 1700	> 575
Enrollment	> 1170	> 550
Total size for an IoT device	> 3247	> 1502

9.2. Computational overhead

With the exception of the remote attestation operations, which are highly dependent on the type of RA performed, the only added operations with significant computational impact for the IoT devices are the signature checks related to the IoT transfer message, and the two signature generations needed for the confirmation message. The signature checking of the COSE_Sign1 is the same type of operation that is performed as part of an EDHOC handshake. On the nRF52840 platform, one signature verification operation takes 21 ms, when the signature is done using the commonly used P-256 curve. The time needed for one signature generation is slightly less, 20 ms. This can be compared with a full EDHOC handshake which needs around 90 ms of active CPU time for the IoT device when using the same ECC curve.

9.3. Memory overhead

The functionality needed for the authentication operations is of the same type that is used for EDHOC and OSCORE. By reusing the crypto libraries, no extra memory footprint will be taken into account for crypto operations, and less than a kilobyte for the transfer message and confirmation message handling. Our implementations of required crypto functionality used by both for OSCORE and EDHOC need approximately 6 kB of ROM, plus 5 kB more of EDHOC specific code, for the nRF52840 platform.

Solutions for remote attestation of IoT devices have been successfully emulated on IoT devices as limited as the old TmoteSky platform with 48 kB ROM, 10 kB of RAM and access to 1MB of flash [26], and could therefore coexist with the required PKI components on more capable devices such as nRF52840.

9.4. Non-functional requirement compliance

The functional requirements are assessed below in 10. Here we focus on evaluating compliance with the non-functional requirements.

NFR1: Automatization. The feasibility analysis illustrates that besides the initial trust agreements and SLA establishments, all other operations can be fully automated. This is a key requirement to enable large-scale IoT deployments with PKI support, through the reduction of the PKI costs per device.

Currently, the pricing models for CA services are complex and dependent on a long range of customer requirements. The requirements can be both security guarantees, such as requirements on dedicated hardware security modules (HSM) and organizational constraints, such as which of the organizational constructs depicted in figure 1 which need to be supported[2] .

Specifically for the cost of individual certificates, for the few CA providers which share any certificate pricing information online, the lowest per certificate cost found is starting from 7.95 USD per year, as of April 2022 [27]. This price range is infeasible for large-scale IoT deployments.

The current situation illustrates the need for continued development towards standards, increased automatization, and reduced costs per device.

NFR2: Resource efficiency. All the needed building blocks have been demonstrated in versions suitable for modern constrained IoT devices. Since the transfer functionality is vital but rarely used, it is crucial to reuse already existing crypto functionality on the device, resulting in a minimal added overhead.

NFR3: Standard compliance. All security critical components are contained within existing or proposed standards. The combination of secure upgrades and remote attestation is still an area where only initial standardisation solutions have been proposed. The modular approach proposed for AutoPKI makes it relatively easy to upgrade parts of the protocol to incorporate for example new remote attestation mechanisms, or new crypto algorithms to be used for authentication or encryption services.

[2]Nexus company policies

10. AutoPKI: Security Assessment

The security assessment of the protocol builds upon the derivations done in the SIGMA paper [28]. A correctly constructed protocol will keep the security properties offered by the individual components, and hence be capable of offering the intended security services as long as the components keep their security guarantees.

We model and formally prove two statements which together cover the requirements FR1 and FR2. To explain the formal verification of the protocol security properties, we first briefly introduce the modelling tool, Tamarin. We then explain the modelling, discuss the results and reason about the assessment of the remaining requirements.

10.1. Introduction to Tamarin Prover

The Tamarin prover is an open-source formal verification modelling tool, designed specifically to aid the verification of communication protocols.[3] The tool operates in the symbolic model, which means that protocol variables are not instantiated with concrete values. Instead, it is the relationship between variables which is evaluated. For example, it is not possible to read out any actual value of a fresh pseudo-random variable, but one knows that it cannot be derived from any other variable. Facts about the state of the world are modelled as a multi-set of logic predicates. Actions, both in the protocol and by a potential attacker are modelled as a set of transition rules for this multi-set of facts. The security properties are modelled as first-order logic formulas. The tool is able to reason about an unbounded number of protocol instances running in parallel, only limited by memory and compute power. A verification run might not terminate, but if it does it results in a 100%-certainty proof that the stated security properties are verified. The details of the tool-generated proof are barely human-readable and not interesting on their own, the value lies in the correspondence between the high-level abstract symbolic model and the concrete protocol being modelled.

10.2. Modelling choices

The Tamarin prover by default has a Dolev-Yao adversary model, introduced in section 5. In short, an adversary has full control of the network and can listen, record, block, delay, and modify all messages. On the other hand, the adversary is in general not capable to break cryptographic functions without the corresponding keys.

In addition, we extend the capabilities of the adversary with the capacity to learn SP1 private keys. This allows us to model SP1 colluding with the adversary. Certificate Authorities are modelled as entities with an identity, the corresponding role 'CA', and the control of a private long-term key, 'ltk_CA'. Additionally, a CA can be a 'Root CA' which provides factory certificates for IoT devices, and an 'operational CA' that controls one or multiple URI.

Certificates are modelled as a public key, signed with a private CA key.

We assume the CAs' public keys are safely known by all involved SPs, i.e. there is no risk for the SPs that the CA public key will be spoofed, manipulated or replaced with a key controlled by the attacker or otherwise compromised. We do not assume that an IoT device enjoys the same privilege, and we explicitly model how and where a device learns the public keys of the CAs that it trusts.

To limit the search space for the prover and ease the proof generation, SP1 and SP2 are modelled as separate roles, with only one instance of each role active in the protocol. We claim this is done without loss of generality, as SP1 can freely collude with the adversary in our model, so adding more instances of SPs would not give any additional capabilities to the adversary, nor enable new attacks.

Similarly, we only allow only one instance of CA to provide factory certificates to the IoT devices. On the other hand, SP1 can own an unbounded number of devices, and an unbounded number of Transfers can happen in parallel.

An IoT device is modelled as an entity with a fixed ID and the 'Device' role, and initially nothing else; the provisioning of the factory certificate is explicitly part of the model.

Since we model the CAs directly with a public/private key pair, and not certificates for themselves, we do not model certificate chains with intermediate CAs. In other words, each CA acts as its own trust root. However, as we mentioned earlier, a CA is allowed to act as both a root/long-time CA and as an operational CA at the same time, which allows the tool to explore all possible combinations with one, two, or three different trust roots for a given instance of transfer.

[3] Available online at https://tamarin-prover.github.io/

Through this setup, the most challenging case (seen in Fig. 1a), can be modelled, and slight simplifications of cases (b), (c) and (d) can be modelled with the certificate chains collapsed. Assuming that the certificate chain validation mechanisms work as intended for the involved parties and do not introduce new vulnerabilities, the properties proven in our model hold in any possible configuration of CAs.

The enrollment process is modelled through one single rule, even though in practice it consists of several message exchanges. This represents an unmodified standard protocol, which is assumed to either run to completion or be fully aborted.

Communications with the IoT device are sent through the adversary-controlled network, as well as communications involving SP1 since it can collude with the adversary. On the other hand, communications between SP2 and its CA are assumed to go through a secure channel (e.g. protected with TLS) and are not sent through the adversary-controlled network in our model.

10.3. Tamarin Results and requirement assessment

In this subsection, we present the properties we have formulated based on the model, that the Tamarin prover has verified to hold true. We then map them back to our initially formulated security properties.

Listing 5: The two Tamarin lemmas we have proved in the model described above

```
lemma Secrecy:
        "All data #i.
        Secret(data) @ i
        ==> not (Ex #j. K(data) @ j)"

lemma Authenticity:
        "All SP2 ID data #i.
        Commit(SP2,ID,data) @ i ==>
        (Ex #j. Running(ID,SP2,data) @ j)"
```

The first lemma, 'Secrecy', means that any data tagged as 'Secret' during an execution of the protocol is indeed unknown to the adversary. Or more formally, if data is tagged secret at a timepoint i, then there does not exist any timepoint j when the adversary knows the data.

The second lemma, 'Authenticity', means that if SP2 can 'Commit' to a transfer with the device ID, ID was indeed 'Running' a transfer, and they both agree on the relevant data exchanged during the transfer. In other words, there is guaranteed to be a correspondence between runs of the protocol executed by SP2 and runs executed by the Device, SP2 cannot 'Commit' to a transfer while being tricked by an attacker.

The 'Secret', 'Running', and 'Commit' tags have been added to the relevant rules in our model to ensure the desired security guarantees:

- the private key of the new operational certificate used in the confirm message received by SP2 is tagged 'Secret'
- this certificate is also included in the data that SP2 'Commits' to.
- the corresponding 'Running' tag is added by the IoT device after a successful enrollment, with the new certificate it just obtained as the data.

An interesting point to note is that, during the proving process, the Tamarin prover initially found an attack with the initial protocol design. Seeing this attack, understanding and patching the corresponding vulnerability, helped us refine our design and specify the precise content of the signature sent by the device in the final confirmation message.

The two lemmas are now automatically provable by Tamarin built-in solver, using the default heuristic, and are guaranteed to hold in any possible execution of the transfer protocol in our model.

Interpretation. Altogether, these two lemmas ensure that, if the transfer protocol concludes properly from the point of view of SP2, and it receives a confirmation message apparently from the IoT device with ID, SP2 is guaranteed that ID was properly transferred and it now has enrolled for an operational certificate that SP2 can trust.

Comparing the Tamarin lemmas to the requirements defined in section 6, our formal model cannot by itself guarantee all the desired properties, but it gives strong evidence that the protocol is sound in principles. Let us study the different requirements individually:

FR1: Integrity protection. The content of the transfer message received by the device is part of the data in the 'Commit'/'Running' tags that the IoT device and SP2 agree upon, and so FR1 is guaranteed in our model.

FR2: Man-in-the-middle resistance. Similarly, our 'Authenticity' lemma with the associated 'Commit'/'Running' tags in the protocol rules ensures that any confirmation message received by SP2 was indeed sent by the transferred device, excluding any possibility of tampering by a man-in-the-middle attacker. So FR2 is guaranteed in our model.

FR3: Forward security. Our model guarantees: (a) from the point of view of SP2, that the device enrolled to, and now trusts, the actual CA2 that issues certificates for SP2 ('Authenticity' lemma); (b) the secrecy of the private key of the new operational certificate that the device enrolled at CA2, *both* from the point of view of the device and the point of view of SP2 ('Secrecy' lemma). However, to fully guarantee FR3, SP2 also needs a proof that the device is honest and does not contain, for instance, a backdoor controlled by SP1, hence the importance of the software update and remote attestation process highlighted in section 8.3. With this additional assumption, SP2 and the device can trust each other and are guaranteed to have a secure access to each other public key through CA2, and FR3 can be guaranteed.

FR4: Backward security. This property relies mostly on SP1 doing its due diligence and erasing all sensitive data from the devices before initiating the transfer proper (while SP1 still has absolute control over the devices) and not so much on the transfer protocol itself. Once again, it highlights the importance of the software update phase that takes place before the transfer protocol itself.

10.4. Further considerations

While the solutions proposed in this paper fills an important gap in terms of security mechanisms for IoT, there are open issues, such as secure time sources and coordination with link-level security solutions, which remain to be fully standardised.

An observation about reachability: In the procedure described here the responsibility to initiate contact, specifically after the factory reset, lies with the IoT devices. For many deployments, this is the only available option, as networks with NAT can cause devices to be unreachable unless the connection is initiated from within the network. While IPv6 is increasing in usage, and could in theory offer all devices globally accessible addresses, there are also security reasons to hide resource-constrained devices from being easily found and attacked, for instance taken down through DOS attacks.

11. Conclusion

When IoT deployments become more common and grow in size, issues of long-time maintenance and the scalability of the security services become critical. Making use of proposed and available PKI solutions suitable for IoT we propose a protocol for the transfer of control of IoT deployments, through the transfer of trust between one operational domain to another, with minimal manual overhead. The solution ensures the possibility of long-time support solutions for IoT deployments, and prevents vendor lock-in. We show that given the integrity of the secure building blocks, the protocol maintains the desired security properties.

Acknowledgment

This research is partially funded by the Swedish SSF Institute PhD grant and by the EU H2020 projects ARCADIAN-IoT (Grant ID. 101020259) and CONCORDIA (Grant ID: 830927).

References

[1] R. Housley, W. Ford, T. Polk, and D. Solo, "Internet X.509 Public Key Infrastructure Certificate and CRL Profile," Internet Requests for Comments, RFC Editor, RFC 2459, January 1999.

[2] J. Höglund, S. Lindemer, M. Furuhed, and S. Raza, "PKI4IoT: Towards public key infrastructure for the Internet of Things," *Computers & Security*, vol. 89, 2020.

[3] J. Höglund and S. Raza, "LICE: Lightweight certificate enrollment for IoT using application layer security," in *IEEE Conference on Communications and Network Security, CNS 2021, Tempe, AZ, USA, October 4-6, 2021*. IEEE, 2021.

[4] G. Selander, J. Mattsson, and F. Palombini, "Ephemeral diffie-hellman over cose (edhoc)," Working Draft, IETF Secretariat, Internet-Draft draft-ietf-lake-edhoc-03, 12 2020.

[5] F. D. Schoorman, R. C. Mayer, and J. H. Davis, "An integrative model of organizational trust: Past, present, and future," *The Academy of Management Review*, vol. 32, no. 2, pp. 344–354, 2007. [Online]. Available: http://www.jstor.org/stable/20159304

[6] M. S. N. Khan, S. Marchal, S. Buchegger, and N. Asokan, "chownIoT: Enhancing IoT Privacy by Automated Handling of Ownership Change," in *Privacy and Identity Management. Fairness, Accountability, and Transparency in the Age of Big Data :*, vol. 547, 2018, pp. 205–221.

[7] M. Gunnarsson and C. Gehrmann, "Secure ownership transfer for the internet of things," in *Proceedings of the 6th International Conference on Information Systems Security and Privacy*, S. Furnell, P. Mori, E. Weippl, and O. Camp, Eds., vol. 1. SciTePress, 2020, pp. 33–44.

[8] M. Safkhani, S. Rostampour, Y. Bendavid, S. Sadeghi, and N. Bagheri, "Improving rfid/iot-based generalized ultra-lightweight mutual authentication protocols," *Journal of Information Security and Applications*, vol. 67, p. 103194, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214212622000758

[9] O. AbuAlghanam, M. Qatawneh, W. Almobaideen, and M. Saadeh, "A new hierarchical architecture and protocol for key distribution in the context of iot-based smart cities," *Journal of Information Security and Applications*, vol. 67, p. 103173, 2022.

[10] D. Dolev and A. Yao, "On the security of public key protocols," *IEEE Transactions on Information Theory*, vol. 29, no. 2, pp. 198–208, 1983.

[11] C. Müller, A. M. Gutierrez, P. Fernandez, O. Martín-Díaz, M. Resinas, and A. Ruiz-Cortés, "Automated validation of compensable slas," *IEEE Transactions on Services Computing*, vol. 14, no. 5, pp. 1306–1319, 2021.

[12] R. B. Uriarte, R. de Nicola, and K. Kritikos, "Towards Distributed SLA Management with Smart Contracts and Blockchain," in *2018 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, 2018, pp. 266–271.

[13] A. Alzubaidi, K. Mitra, and E. Solaiman, "Smart contract design considerations for sla compliance assessment in the context of iot," in *2021 IEEE International Conference on Smart Internet of Things (SmartIoT)*, 2021, pp. 74–81.

[14] A. Beniiche, "A study of blockchain oracles," *ArXiv*, vol. abs/2004.07140, 2020.

[15] "IEEE Standard for Local and Metropolitan Area Networks - Secure Device Identity," *IEEE Std 802.1AR-2018*, pp. 1–73, 2018.

[16] M. Nystrom and B. Kaliski, "PKCS #10: Certification Request Syntax Specification Version 1.7," Internet Requests for Comments, RFC Editor, RFC 2986, November 2000.

[17] J. P. Mattsson, G. Selander, S. Raza, J. Höglund, and M. Furuhed, "CBOR Encoded X.509 Certificates (C509 Certificates)," Working Draft, IETF Secretariat, Internet-Draft draft-ietf-cose-cbor-encoded-cert-03, January 2022.

[18] J. Schaad, "CBOR Object Signing and Encryption (COSE): Header parameters for carrying and referencing X.509 certificates," Working Draft, IETF Secretariat, Internet-Draft draft-ietf-cose-x509-08, December 2020.

[19] H. Tschofenig and T. Fossati, "Transport layer security (tls) / datagram transport layer security (dtls) profiles for the internet of things," Internet Requests for Comments, RFC Editor, RFC 7925, July 2016.

[20] M. Pritikin, M. Richardson, T. Eckert, M. Behringer, and K. Watsen, "Bootstrapping remote secure key infrastructure (brski)," Internet Requests for Comments, RFC Editor, RFC 8995, May 2021.

[21] P. van der Stok, P. Kampanakis, M. Richardson, and S. Raza, "EST-coaps: Enrollment over Secure Transport with the Secure Constrained Application Protocol," Internet Requests for Comments, RFC Editor, RFC 9148, April 2022.

[22] B. Moran, H. Tschofenig, D. Brown, and M. Meriac, "A Firmware Update Architecture for Internet of Things," Internet Requests for Comments, RFC Editor, RFC 9019, April 2021.

[23] S. F. J. J. Ankergård, E. Dushku, and N. Dragoni, "State-of-the-Art Software-Based Remote Attestation: Opportunities and Open Issues for Internet of Things," *Sensors*, vol. 21, no. 5, 2021.

[24] H. Birkholz, D. Thaler, M. Richardson, N. Smith, and W. Pan, "Remote attestation procedures architecture," Working Draft, IETF Secretariat, Internet-Draft draft-ietf-rats-architecture-15, February 2022.

[25] J. Höglund, M. Furuhed, and S. Raza, "Lightweight certificate revocation for low-power iot with end-to-end security," *Journal of Information Security and Applications*, vol. 73, 2023.

[26] E. Dushku, M. M. Rabbani, M. Conti, L. V. Mancini, and S. Ranise, "SARA: Secure Asynchronous Remote Attestation for IoT Systems," *IEEE Transactions on Information Forensics and Security*, vol. 15, 2020.

[27] ComodoSSLstore, "Comodo positive ssl certificate," https://web.archive.org/web/20220420135513/https://comodosslstore.com/positivessl.aspx, April 2022.

[28] H. Krawczyk, "SIGMA: The 'SIGn-and-MAc' Approach to Authenticated Diffie-Hellman and Its Use in the IKE Protocols," in *Advances in Cryptology - CRYPTO 2003*, D. Boneh, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 400–425.

Paper V

2022 IEEE International Conference on Cyber Security and Resilience (CSR)

BLEND: Efficient and blended IoT data storage and communication with application layer security

Joel Höglund, Shahid Raza
RISE Research Institutes of Sweden
Isafjordsgatan 22, 16440 Kista, Stockholm
{joel.hoglund, shahid.raza}@ri.se

***Abstract*—Many IoT use cases demand both secure storage and secure communication. Resource-constrained devices cannot afford having one set of crypto protocols for storage and another for communication. Lightweight application layer security standards are being developed for IoT communication. Extending these protocols for secure storage can significantly reduce communication latency and local processing.**

We present BLEND, combining secure storage and communication by storing IoT data as pre-computed encrypted network packets. Unlike local methods, BLEND not only eliminates separate crypto for secure storage needs, but also eliminates a need for real-time crypto operations, reducing the communication latency significantly. Our evaluation shows that compared with a local solution, BLEND reduces send latency from 630 μs to 110 μs per packet. BLEND enables PKI based key management while being sufficiently lightweight for IoT. BLEND doesn't need modifications to communication standards used when extended for secure storage, and can therefore preserve underlying protocols' security guarantees.

***Index Terms*—Secure storage, communication security, application layer security, OSCORE, EDHOC, IoT**

I. Introduction

IoT is being deployed in extremely heterogeneous and wild scenarios such as agriculture monitoring, battlefields, remote surveillance, power-line monitoring, flood monitoring, and telemedicine. Most of these deployments require data confidentiality and/or integrity while at rest as well as in transit. While traditional Datagram TLS (DTLS) [1] has been extended to IoT, it is still too heavy for many IoT scenarios and lack full end-to-end security across different transport layer technologies. New Application layer protocols, namely OSCORE [2] and EDHOC [3], are specifically designed for resource-constrained IoT and offer full end-to-end security.

In contrast to the active standardization work on enabling secure communication in IoT, the secure storage solutions for IoT have attracted less attention. While a custom-made local secure storage protocol can be developed, it would require new proposals on, for example, key management, choosing encryption and secure hash functions, initialization vectors, etc. Less well tested new solutions are likely to be less secure, and will require additional implementation efforts ultimately requiring more processing and storage resources. Most importantly, a separate secure storage solution will require additional crypto operations when an IoT data is sent to a remote host, which will increase the real-time latency. As shown in Figure 1 (left), before sending a securely storage data, separate secure storage and secure communication solutions will require that the data must be first *decrypted* using one set of security protocol and *encrypted* again with another set of protocols. Such a solution has significant performance overhead and is infeasible for resource-constrained IoT devices.

978-1-6654-9952-1/22/$31.00 ©2022 IEEE

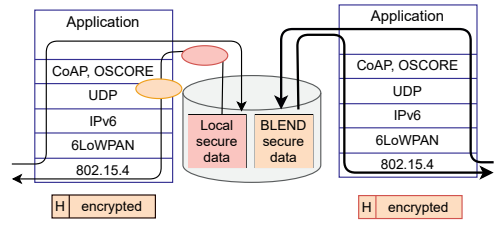


Fig. 1: Retrieving encrypted IoT data and securely sharing it with a remote host, with (right) an without (left) BLEND

In this paper, we propose BLEND that exploits novel application layer security protocols and provide combined secure storage and communication without compromising end-to-end security. BLEND does not require separate protection for storage and for communication, and the stored secure data can be shared with a remote host without any crypto operations during the transmission phase, ultimately reducing the real-time latency significantly; this is depicted in Figure 1 (right). BLEND is particularly advantageous in use cases having hard latency requirements; for example, when a drone cost-effectively collects IoT data from vast smart agriculture deployments or from remote power lines.

The main challenge in providing a combined secure storage and communication is to enable a solution that incurs minimal overhead for IoT devices, keep well-tested security properties intact, and does not compromise standard compliance and interoperability. This can be achieved by extending the use of the newly standardised OSCORE and EDHOC protocols to secure data storage. The core contributions of the paper are as follows: we *(i)* extend standard based application layer security mechanisms and enable combined secure storage and secure communication; *(ii)* provide an implementation of

BLEND for resource constrained devices using Contiki NG; and *(iii)* evaluate BLEND to show its suitability for IoT.

The rest of this paper is organized as follows: related work and relevant background are presented in section II and III, respectively; we present a treat model in Section IV; elaborate our design in section V; provide implementation details in section VI and evaluation in VII; highlight security considerations in Section VIII; and conclude the paper in section IX.

II. Related Work

A. Secure communication

The area of secure communication for resource constrained devices has seen a rapid development the last decade, with the introduction of protocols targeting IoT. Early standards such as IPSec has largely been replaced with DTLS, Datagram Transport Layer Security. Still the protocol overhead is relatively large. Especially for low power radio networks where network radio packets are as small as 127 bytes, and fragmentation can cause delays and security vulnerabilities. This is limiting the usable payload for application layer sensor data down to a maximum of 51 bytes per packet, unless network specific optimizations such as 6lowPAN header compression is used, which limits the general applicability [4]–[6]. Recently new application layer protocols for secure communication have been devised which can reduce the per packet overhead, while supporting crypto algorithms suitable for constrained devices. OSCORE together with with EDHOC for key establishment have the potential to be used for PKI solutions with sufficiently low overhead for IoT. While DTLS has been shown to be feasible for PKI solutions for IoT the cost of key establishment when using standard X509 certificates is high [7], [8].

B. Secure storage

The area of secure storage has seen much less standardisation efforts. Instead several overlapping fields are contributing to the area. Blockchain based research efforts, including [9], [10] design solutions for custom deployments, but mainly address computationally capable end devices such as cellphones or routers, and rely on custom server infrastructure.

Another related area is research on Trusted Execution Environments, TEE, such as ARM's TrustZone. TrustZone functionality has been used as a building block to construct secure storage for Android based devices [11]. An important area for TEE is to enable the creation of secure key storage [12]–[14]. With respect to the more constrained IoT devices the TEE related efforts are complementary to our work on secure storage.

Besides the problem of secure key storage, many of the relatively lightweight cryptographic solutions used in communication protocols can be applied to any data to create a secure sensor data storage. As long as the secure storage only serves local encryption purposes, the need for standardisation is less emphasized.

There are two previous suggestions on how to combine secure communication with secure storage, FUSION and FDTLS [15], [16]. The proposed designs are based on IPsec and DTLS, where promising results in terms of reduced overhead when packets are being sent are shown. An important finding is the need to optimize the storage operations with respect to the memory hardware constraints, such as to write full memory pages to reduce flash handling overhead.

The main shortcomings of these lower layer security approaches are the following: The solutions rely on PSK, preshared keys. This is an outdated mode of key management, with no support for automated key management, including enrollment or revocation. Both IPsec and the DTLS version 1.2 used for the evaluations have large headers, greatly reducing the space available for sensor data when used over low power radio networks. To partly alleviate this, both designs rely on using 6lowPAN header compression, which ties the usage completely to networks where this is available. To allow new connections the protocol is side stepped in terms of removing the randomness used when generating session keys, without analyzing the security implications of this procedure, plus other minor protocol breaking tweaks. Additionally, by relying directly on IPsec or DTLS none of the conveniences offered by CoAP are available for any of the involved parties.

The conclusion is that while several works address some of the issues of secure storage of data for IoT, the existing proposals for coalesced storage and communication have serious shortcomings. We address these shortcomings with a design making use of application layer security.

III. Necessary Background

Object Security for Constrained RESTful Environments is an application-layer protocol specifically designed for IoT security [2]. It protects CoAP messages and builds upon COSE [17] and CBOR functionality for encryption and encoding [18]. The protocol offers replay protection using sequence numbers tied to the security context. Since UDP packets might arrive out of order, the protocol uses a replay windows, such that the receiver keeps a range of currently accepted numbers.

Ephemeral Diffie-Hellman Over COSE (EDHOC) is a proposed key exchange protocol primarily design for OSCORE [3], and shares the usage of COSE and CBOR encoding with OSCORE. It can be used with standard X.509 certificates, or with more compact certificate formats. The security functionality of EDHOC is based on the SIGMA schema, from which it follows that as long as the included components keep their security guarantees, the resulting protocol will provide the desired security services [19].

A successful EDHOC security context establishment will result in the parties agreeing on a Master Secret, a Master Salt, client and recipient IDs, and the crypto algorithms to use. With this information in place, Sender Key, Recipient Key and Common IV can be derived and saved. Once a security context is established, an endpoint is free to act both as server and client, using the same security context for both purposes [2].

IV. Threat model and assumptions

We consider scenarios where an attacker can, with some probability, get physical access to the node and probe the

device permanent flash memory. We discuss both scenarios where we assume that the non-permanent memory is sufficient for key storage and scenarios where a (small) tamper protected memory area exists, which can be used for key storage. For communication, the Dolev-Yao threat model is applicable. An attacker can eavesdrop any communication between the involved entities, and also modify and re-send any message. As a consequence protection for replay attacks are needed, together with authentication and confidentiality services to prevent unauthorized access to any secret content. To generate new keys and perform secure key exchange the devices must have access to a sufficiently strong random number generator. We assume that the standards we use as building blocks are not compromised, but can offer the claimed security guarantees when used together with the recommended crypto algorithm suits.

V. BLEND: DESIGN

A. Requirements

The main requirement is to offer secure storage with low latency for data sending, while keeping the overall overhead low. To preserve security guarantees offered by OSCORE, as few deviations from the protocol usage as possible should be done. Preferably the receiving end of the communication should not need to take any additional steps outside of the regular protocol to receive and decrypt previously stored sensor data. In order to preserve the protocol guarantees, the initial key establishment needs to happen before packets can be precomputed and stored.

B. System building blocks

An EDHOC implementation is needed for key establishment, but requires only standard functionality in terms of key export interfaces to create and retrieve the shared secrets used for the security context.

The OSCORE implementation needs to be augmented with handlers to enable BLEND to precompute packets and send them unmodified at a later point in time. Practically this means allowing retrieval of the byte buffer representing the serialized OSCORE packet and ensuring there are interfaces to control the sequence numbers.

A flash storage abstraction is useful to hide hardware specific details and offer a higher level API. We propose a simple file system like API which allows reading, writing and appending data to files, which are being written out to flash.

C. SecureStorage lifecycle and message flow

The figure 2 illustrates the main events relevant to secure storage operations. After the key establishment both parties have established a secure context, which allows them to act as both clients and servers.

The sensor can thereafter be deployed, and start sensing. Depending on the data generation rate and storage policy, a number of sensor readings might be compiled as the payload for one CoAP packet. The packet is encrypted as a ready to send OSCORE packet and stored onto flash.

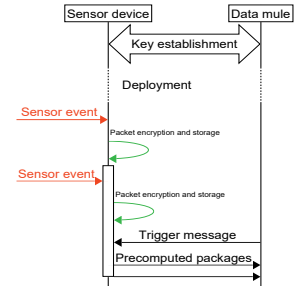


Fig. 2: BLEND overview. An initial key establishment is done before deployment, can be redone later given EDHOC support. Sensor data is encapsulated into precomputed packets, and securely stored until a connection with a data mule, or any other secure endpoint, is available

When the communication link is ready, for instance in the form of a data mule, a trigger command is sent. The trigger message is a CoAP request, protected with the same OSCORE security context as has been previously established through EDHOC. Hence the correct decoding of the trigger message serves both to authenticate the data mule, and to authorize it for accessing the sensor data. The command will cause the sensor device to start sending the stored packages. In sending the stored packages, the sensor device acts as a client. This allows the device to control the sequence numbers included in the packets, reflecting the sequence numbers of the stored packets. To prevent an attacker from stopping a data transfer simply by blocking the trigger message, we require the device to reply with a short no data message in case there is no sensor data to send.

D. Storage overhead trade-offs

The amount of data that the sensor device needs to store locally depends on both the sensor data generation rate and the frequency of data collection from the outside. For sensor devices deployed in low power radio networks, the least amount of overhead is achieved if payloads corresponding to full 802.15.4 frame sized packets are precomputed and stored. If the sensor data generation rate is sufficiently small, the node would need to temporarily store unencrypted sensor data until the amount corresponding to a full payload is gathered. If no temporary plaintext storage is considered acceptable, the device must create encrypted packets for each individual sensor reading.

E. Relation between the layers

An detailed illustration of the relation between the layers is shown in figure 3. In the following we explain the data needed to be included and processed.

F. CoAP packet creation

While the CoAP protocol is versatile, with a range of packet options, we are here interested in a meaningful minimal subset

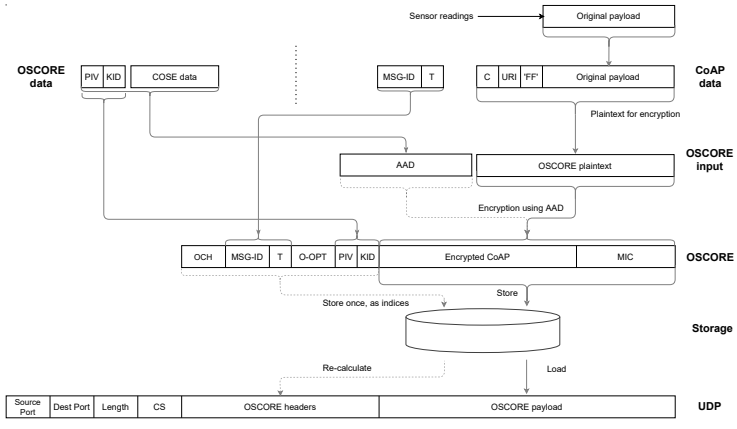


Fig. 3: The relation between the layers when using BLEND

TABLE I: Plaintext data needed to prepare the CoAP packets used in BLEND

Type	*Size*, byte	*Example*	
Version & type	1	'40'	ver.1, confirmable
Code	1	'02'	POST
Message ID	2	'4A 84'	<any id>
Token	1	'84'	
URI path & len	1+path len	'b0'	
Payload marker	1	'FF'	
Payload	6–56	<binary data>	
To be encrypted	*≥ 3+payload len*		

TABLE II: Data contained in the OSCORE packets

Type	*Size*, byte	*Example*	
Version & type	1	'40'	ver.1, confirmable
Code	1	'02'	POST
Message ID	2	'4A 84'	from CoAP
Token	1	'84'	from CoAP
OSCORE flag	2	'93 09'	
Partial IV	1–	'13'	= sequence no
Key ID	1	'42'	= sender id
Payload marker	1	'FF'	
Encrypted payload	9–59	<encrypted CoAP>	
MIC	8		
Packet length	*≥ 21+original sensor data payload len*		

needed to precompute sensor data packets. The table I shows the minimal plaintext data needed to create a CoAP packet, ready to be encrypted for secure storage. In italics are the fields that will be moved to the OSCORE packet. In bold are the fields will be protected through encryption. An observation is that the length of the destination URI directly adds to the packet overhead, but unless otherwise required the empty root path can be used as a valid destination URI.

TABLE III: Previous state-of-art, data contained in a DTLS record packet

Type	*Size*, byte	*Example*
Content type	1	'17' = application data
Version	2	'FEFD' = DTLS 1.2
Epoch	2	'0001'
Sequence number	6	'0000 0000 0001'
Length	2	
Initialization vector	8	
Encrypted payload	6–51	<encrypted raw sensor data>
MIC	8	
Packet length	*≥ 29+original sensor data payload len*	

G. OSCORE packet creation

Given an existing security context and the CoAP packet information, BLEND can encrypt the CoAP payload together with the sensitive header fields, and calculate the correct OSCORE headers. The missing dynamic information needed is the sender sequence number. The sequence number is used as the basis for the partial initialization vector, or Partial IV in COSE terms. The sender ID is used as key ID. ('PIV' and 'KID' in figure 3.) These two items, together with static COSE information on the algorithm used, are used to form the additional authenticated data, AAD, used in encryption. The two items are also used for calculating a nonce used in encryption, and finally they are included in plaintext in the OSCORE packet header.

The table II shows the data present in the resulting OSCORE header. Starting from the original sensor data, the minimal total overhead is 21 bytes. For sequence numbers in the range 255–65535 an extra byte is needed, etc. This flexible sizing is in contrast to the older DTLS standard (shown in table III), where a fixed field of 6 bytes is allocated regardless of the currently needed size.

H. Storage of precomputed packets

For systems with fast flash memory operations, or where energy is of less concern, the prepared OSCORE packet can be saved directly. Where flash operations are slow or energy efficiency is paramount, the OSCORE packet header information can be stored once for a whole series of sensor data packets. Since all the dynamic fields; the message ID, token and sequence number can be assigned in a predictable increasing manner, storing and later retrieving the starting points for the first packet header is sufficient to recalculate the following packet headers. It is this optimized procedure which is shown in figure 3.

I. UPD alternative

Also UPD headers could be precomputed, and the entire UDP databuffer could be stored for minimal processing at the time of sending. Precomputing UDP packets requires the source and destination ports to be known beforehand. A more important drawback is the increased storage overhead, since the UDP headers add another 8 bytes to each precomputed packet, which needs extra time for storage and retrieval.

J. Key management

BLEND relies on devices being able to establish new security contexts upon need. To create a new context with the same endpoint, OSCORE allows existing master secret data to be reused, making the context derivation computationally cheap. This can be used to keep the context sequence number bounded by a fixed length. For key establishment we propose EDHOC to be used. EDHOC offers relatively low overhead while supporting PKI solutions. Low overhead is achieved through using certificate reference based key establishment. This requires relevant certificates to have been securely distributed at an earlier point in time. Certificate distribution is out of scope for this work, but in contrast to solutions based on shared secrets, certificates are meant to be openly shared and could be distributed from any trusted endpoint.

1) Planned secure context updates: If the data collection endpoint is replaced, the sensor device needs to establish a new security context with the new endpoint. For a planned update, a notice can be communicated ahead of time. Depending on the deployment scenario, this might simply be a message to initiate a full new key establishment, which allows the sensor to immediately start using the new context for sensor data storage. For extreme deployments with very limited connectivity and data mules, it might be a notice send during the last data collection round where the old data mule is active. Unless the key establishment can be relayed at that time, the sensor device has to temporarily resort to local storage encryption until a new security context is in place.

2) Unplanned secure context losses: For cases when the data connection endpoint loses the security context, or is lost all together, the following round of data collection with a not previously used endpoint requires re-keying. Any data that has been stored locally using the old security context need to be decrypted by the IoT device and encrypted again.

For IoT devices with access to tamper resistant nonvolatile memory that can be used for key storage, they can store the shared secret data established through the key exchange such that they can recover the security context in case of temporal power losses or restarts.

An IoT device without a secure permanent key storage wants to minimize the storage of security sensitive data to a minimum. Hence there is a risk of losing vital parts of the security context, in case of power losses and unplanned restarts. In the case of context losses the previous stored precomputed sensor data packets become opaque to the device. The stored data can still be sent to the endpoint which has access to the security context and is able to decipher the encrypted packets. Depending on the deployment, the device might report its situation and request a new authentication through a new key exchange before sending the packets from the old security context. In this case the receiving endpoint must keep both contexts in parallel. Alternatively, the setup can be done to allow the IoT device to interpret an incoming message it cannot decipher as the expected trigger message, if it is recovering from a security context loss.

3) Proximity to endpoint: Since EDHOC offers true end-to-end protection it can be used to establish a security context with any reachable remote endpoint, even behind proxies.

K. Re-sending and multiple receivers

The usage of precomputed sensor data packets does not affect resending that happens on lower layers. Lower layer resending will depend on the deployment scenario and radio configuration. As long as a packet has not been received by the other end, the receive window used for the replay detection by the recipient remains unchanged. If on the other hand data has been received the same packet can no longer be resent, as the encryption is affected by the sequence number. For scenarios where either the same receiver wants the same data item more than once, or where multiple receivers are interested in the same data item, extensions of the keying schema must be done. To handle multiple receivers there are proposals for OSCORE group communication, which could be part of an extended secure storage solution [20].

VI. Implementation

We have implemented BLEND in C as a module for the Contiki NG embedded OS [21], that can be adapted for other available operating systems such as Zephyr [22]. The BLEND implementation contains the needed OSCORE libraries, including COSE and CBOR encoding and decoding.

For the basic crypto operations we have reused functionality from the crypto libraries available in Contiki NG, which offers partial crypto operation hardware acceleration for selected target platforms.

a) Secure communication: The secure communication part of BLEND is build using the OSCORE libraries available in an experimental version of Contiki NG, plus our EDHOC implementation for key establishment. To allow reusability of the available code for confirmable CoAP transactions we include a CoAP token in the packets.

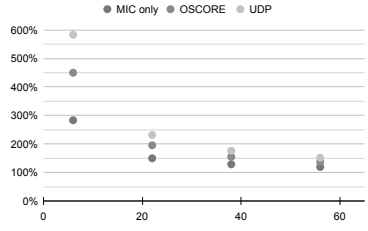


Fig. 4: Storage size relative to the original sensor data size in bytes, for different sensor payloads and storage options

b) Secure storage: The storage part is built on top of the Coffee file system for Contiki, offering a file abstraction for interacting with underlying flash memory. When the optimized packet storage method is used, the dynamic header information needed to recalculate the full headers is recorded at the start of new files, followed by the encrypted part of the packets. The specifics of flash memory block sizes and optimal amount of data to write per file depends on target hardware.

c) Key management and crypto algorithms: We have implemented EDHOC which is used to establish shared secrets, and based on them derive security contexts. Both the EDHOC and the OSCORE standards are flexible in terms of supporting multiple crypto suits. Our implementation is focused on the mandatory SHA-256 for HKDF, HMAC-based Extract-and-Expand Key Derivation Function, and the most commonly used symmetric crypto, AES-CCM-16-64-128. While AES-CCM is a block cipher, it does not require padding of the resulting ciphertext. As a result the length of ciphertext is always the length of the plaintext, plus 8 byte MIC, message integrity code.

VII. Evaluation

We use a quantitative experimental research methodology where we evaluate the impact of one particular variable while keeping other parts of the system setup static, to correctly attribute the performance variations. In the following we present the relevant micro benchmarks illustrating the system performance and overhead.

A. Experimental setup

For the hardware experiments we use Zolertia Firefly nodes, a platform using TI CC2538 ARM Cortex M3 microcontrollers [23]. The nodes are equipped with 32 KB RAM, 512 KB flash, a 2.4GHz 802.15.4 radio for communication and support for hardware acceleration of crypto operations.

B. Storage overhead

The packet storage overhead for different sensor data payloads is shown in figure 4. Storing a ready to send OSCORE packet induces an overhead of 21 bytes, using the configuration presented in tables I and II. If instead a complete UDP packet is stored, the per packet overhead is 29 bytes. If only the starting points for dynamic header data counters are stored once, the overhead quickly shrinks to close to 3 extra CoAP bytes, plus the 8 byte MIC. For our system tests we use file append functionality for storing packets, such that the storage cost of the 6 bytes needed for header recalculations are amortized over 25 precomputed packets.

Depending on the initial sensor data size the resulting storage overhead ranges from 20% for the 56 byte sensor data packets using optimized storage, all the way up to close to 600% for 6 byte sensor data while storing full UDP packets.

In the following experiments the optimised version where needed header data is stored once is used.

C. Latency to get data ready for sending

When the device gets a request to report recorded sensor readings, if local security is used, it needs to read the data from flash, decrypt it with the local key and prepare it for sending. If BLEND is used, the operations needed are reading from flash and, optionally, packet transaction allocation. The two cases are illustrated in figure 1. The total time needed is shown in figure 5a, for when hardware acceleration is available, and in figure 5b without hardware acceleration.

With hardware acceleration BLEND performs around 0.5 ms faster compared with the local security solution. The resulting remaining latency when retrieving a stored packet, recalculating header information and allocating a CoAP transaction is is only between 65 μs and 110 μs.

Without hardware acceleration, with all cryptographic operations done in software, the latency savings are between 0.75 ms and 1.36 ms per packet compared with the local security solution.

D. Total energy usage

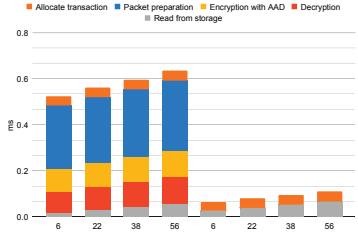
Using the fine grained timer system in Contiki NG we measure the time spent for relevant system operations. This makes it possible to calculate the consumption based on current and voltage levels from the CC2538 hardware datasheets [23]. We use the specified maximum peak current for writing, which means we present the absolute upper bound of energy usage for the flash operations.

The total energy usage is shown in figure 6a and 6b, with and without the usage of crypto hardware acceleration. Using the crypto hardware acceleration the differences are small. Due to relatively slow storage write operations, also a small increase in storage needs can offset crypto savings. Without crypto hardware acceleration, BLEND saves energy for all sensor data sizes.

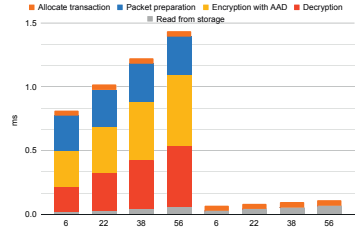
The conclusion is that BLEND performs at least on par with the local storage solution in terms of total energy usage, with a clear advantage for all cases where the crypto operations constitutes a larger proportion of the total work done.

E. Key establishment and comparison with DTLS

1) Key establishment: Using the reference based key establishment option in EDHOC we are able to perform a key establishment using only 284 bytes of application layer

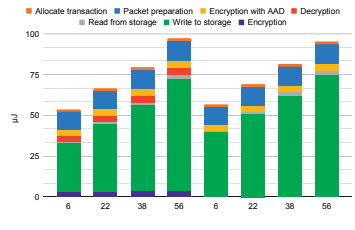


(a) with hardware acceleration

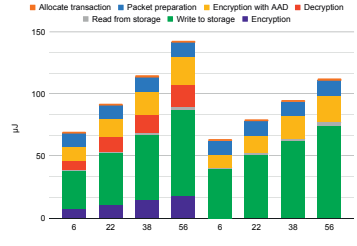


(b) No hardware acceleration

Fig. 5: Latency for packet preparation, with local security and while using BLEND



(a) With hardware acceleration



(b) No hardware acceleration

Fig. 6: Total energy needed for sensor data operations, with local security while using BLEND

data. With DTLS 1.2 and the ECDHE-ECDSA cipher suit corresponding operations use 1.65 kB. This is largely due to lengthy ASN.1 encodings and the need to send full certificates in the handshake. The numbers are based on IoT profiled certificates of only 315 bytes for both parties in the exchange.

2) Packet encryption and overhead: The AES crypto used for OSCORE corresponds to what is commonly used also for DTLS 1.2 in IoT devices. This means the overhead from the crypto operations are directly comparable. An obvious benefit of switching to an OSCORE based solution is the reduced packet overhead. Using the DTLS AES128-CCM8 cipher produces the packet overhead figures given in table III, to be compared with numbers for OSCORE in table II. An OSCORE solution using CoAP saves eight bytes even compared with a DTLS session without CoAP, used to transport raw UDP data. If instead also DTLS is used to provide a CoAPs session, encryption will be performed on the whole CoAP layer packet, which reduces the maximum usable sensor data payload with six more bytes, down to 45 bytes.

F. Memory requirements

The BLEND implementation requires 1.5 kB of ROM and a little less than 0.5 kB of RAM. Figure 7 shows the comparison with the related components in the configuration used. Compared with the size of the total Contiki NG firmware used for the evaluation of around 60 kB ROM and 13 kB of

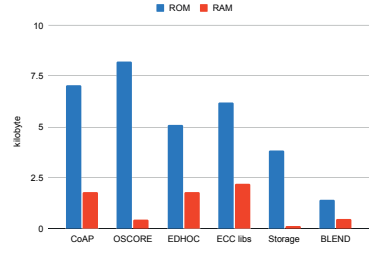


Fig. 7: Memory usage by BLEND and related components

pre-allocated RAM, the BLEND only contributes to 2.5% of the ROM and 3.8% of the RAM.

The numbers shown are when there is memory allocated for two parallel security contexts. Each additional security context adds 143 bytes of RAM.

VIII. Security Considerations

When a protocol designed for securing communication is reused to also protect data at rest, it is important to validate that protocol assumptions are still applicable. This includes the amount of data that can be protected. The OSCORE protocol is designed to allow theoretical maximum sequence numbers

up to 2^{40}-1, but for actual implementations the number will be lower. The implementation used in our evaluation allows sequence numbers up to 4.3 billion. Using 48 byte sensor data packets, to ensure no fragmentation, this corresponds to covering more than 200 GB of data without resetting the sequence counter. This is not a limiting factor for the resource constrained IoT scenarios considered.

The EDHOC protocol relies heavily on the availability of a secure random number generator. For devices with less strong random generators there are proposals on how to incorporate more random material to improve generator quality [24].

If the same master secret data is used to generate multiple secure sessions, forward secrecy is no longer guaranteed [2]. If the long-term secret is leaked, data from previous sessions risk being exposed. This means the multiple session feature should only be used when the risk that previous communication has been eavesdropped is either neglectable, or if the old data no longer is considered secret.

Our only proposed deviation from existing protocol compliance is the suggestion that an IoT device that has lost its secure session could be allowed to send its stored encrypted data without performing mutual authentication and establishing a new secure session. This could enable an attacker to trick the device into sending data, but which the attacker will not be able to decipher, as long as the protocol is not otherwise compromised. To prevent the data from getting lost, the device should keep the data until it has been properly acknowledged through a new security context.

The concerns regarding secure key storage are applicable for any local secure storage solution as well. The local storage needs either a long time key stored in persistent memory, or it needs a secure key management protocol of its own.

IX. Conclusion

We have shown that the new application layer security standard, OSCORE, can be integrated with an IoT storage system, which makes it possible to provide a secure data storage service without compromising any communication security properties or the standard compliance. Our solution, BLEND drastically reduces the latency for sending stored IoT data compared with a local secure storage solution. When combined with EDHOC for performing secure key exchange and establishing the needed security context, BLEND enables a resource efficient way to achieve a complete secure storage and communication solution for IoT.

Acknowledgment

This research is partially funded by the Swedish SSF Institute PhD grant and partly by the EU H2020 ARCADIAN-IoT (Grant ID. 101020259), the ITEA3 Smart, Attack-resistant IoT Networks (Project ID: P123800021) and the H2020 CONCORDIA (Grant ID: 830927) projects.

References

[1] E. Rescorla and N. Modadugu, "Datagram transport layer security version 1.2," Internet Requests for Comments, RFC Editor, RFC 6347, January 2012.

[2] G. Selander, J. Mattsson, F. Palombini, and L. Seitz, "Object security for constrained restful environments (oscore)," Internet Requests for Comments, RFC Editor, RFC 8613, July 2019.

[3] G. Selander, J. Mattsson, and F. Palombini, "Ephemeral diffie-hellman over cose (edhoc)," Working Draft, IETF Secretariat, Internet-Draft draft-ietf-lake-edhoc-09, August 2021.

[4] J. Mattsson, F. Palombini, and M. Vucinic, "Comparison of coap security protocols," Working Draft, IETF Secretariat, Internet-Draft draft-ietf-lwig-security-protocol-comparison-05, November 2020.

[5] J. Hui and P. Thubert, "Compression format for ipv6 datagrams over ieee 802.15.4-based networks," Internet Requests for Comments, RFC Editor, RFC 6282, September 2011.

[6] C. Bormann, "6lowpan-ghc: Generic header compression for ipv6 over low-power wireless personal area networks (6lowpans)," Internet Requests for Comments, RFC Editor, RFC 7400, November 2014.

[7] Z. He, M. Furuhed, and S. Raza, "Indraj: Digital certificate enrollment for battery-powered wireless devices," in *Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks*, ser. WiSec '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 117–127.

[8] J. Höglund, S. Lindemer, M. Furuhed, and S. Raza, "Pki4iot: Towards public key infrastructure for the internet of things," *Computers & Security*, p. 101658, 2019.

[9] L. Zhou, L. Wang, Y. Sun, and P. Lv, "Beekeeper: A blockchain-based iot system with secure storage and homomorphic computation," *IEEE Access*, vol. 6, pp. 43 472–43 488, 2018.

[10] B. W. Nyamtiga, J. C. S. Sicato, S. Rathore, Y. Sung, and J. H. Park, "Blockchain-based secure storage management with edge computing for iot," *Electronics*, vol. 8, no. 8, 2019.

[11] X. Li, H. Hu, G. Bai, Y. Jia, Z. Liang, and P. Saxena, "Droidvault: A trusted data vault for android devices," in *2014 19th International Conference on Engineering of Complex Computer Systems*, 2014, pp. 29–38.

[12] S. Pinto and N. Santos, "Demystifying arm trustzone: A comprehensive survey," *ACM Comput. Surv.*, vol. 51, no. 6, Jan. 2019.

[13] D. Hein, J. Winter, and A. Fitzek, "Secure block device – secure, flexible, and efficient data storage for arm trustzone systems," in *2015 IEEE Trustcom/BigDataSE/ISPA*, vol. 1, 2015, pp. 222–229.

[14] (2020) Android keystore system. [Online]. Available: https://developer.android.com/training/articles/keystore

[15] I. E. Bagci, S. Raza, U. Roedig, and T. Voigt, "Fusion: coalesced confidential storage and communication framework for the iot," *Security and Communication Networks*, vol. 9, no. 15, pp. 2656–2673, 2016.

[16] E. Boo, S. Raza, J. Höglund, and J. Ko, "FDTLS: supporting dtls-based combined storage and communication security for iot devices," in *16th IEEE International Conference on Mobile Ad Hoc and Sensor Systems, MASS 2019, Monterey, CA, USA, November 4-7, 2019*. IEEE, 2019, pp. 127–135.

[17] J. Schaad, "Cbor object signing and encryption (cose)," Internet Requests for Comments, RFC Editor, RFC 8152, July 2017.

[18] C. Bormann and P. Hoffman, "Concise binary object representation (cbor)," Internet Requests for Comments, RFC Editor, RFC 7049, October 2013.

[19] H. Krawczyk, "Sigma: The 'sign-and-mac' approach to authenticated diffie-hellman and its use in the ike protocols," in *Advances in Cryptology - CRYPTO 2003*, D. Boneh, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 400–425.

[20] M. Tiloca, G. Selander, F. Palombini, J. P. Mattsson, and J. Park, "Group oscore - secure group communication for coap," Working Draft, IETF Secretariat, Internet-Draft draft-ietf-core-oscore-groupcomm-14, March 2022.

[21] T. Vu Chien, H. Nguyen Chan, and T. Nguyen Huu, "A comparative study on operating system for wireless sensor networks," in *2011 International Conference on Advanced Computer Science and Information Systems*, 2011, pp. 73–78.

[22] L. F. Project, "Zephyr project," https://www.zephyrproject.org/, 2020.

[23] *Zolertia Firefly platform*, Zolertia S.L, 2018. [Online]. Available: https://github.com/Zolertia/Resources/wiki/Firefly

[24] C. Cremers, L. Garratt, S. Smyshlyaev, N. Sullivan, and C. Wood, "Randomness improvements for security protocols," Internet Requests for Comments, RFC Editor, RFC 8937, October 2020.

Paper VI

AC-SIF: ACE Access Control for Standardized Secure IoT Firmware Updates

Joel Höglund, Anum Khurshid, Shahid Raza
RISE Research Institutes of Sweden
Isafjordsgatan 22, 16440 Kista, Stockholm
{joel.hoglund, anum.khurshid, shahid.raza}@ri.se

***Abstract*—Globally identifiable, internet-connected embedded systems can be found throughout critical infrastructures in modern societies. Many of these devices operate unattended for several years at a time, which means a remote software update mechanism should be available in order to patch vulnerabilities. However, this is most often *not* the case, largely due to interoperability issues endemic to the Internet of Things (IoT). Significant progress toward global IoT compatibility has been made in recent years. In this paper, we build upon emerging IoT technologies and recommendations from IETF SUIT working group to design a firmware update architecture which (1) provides end-to-end security between authors and devices, (2) is agnostic to the underlying transport protocols, (3) does not require trust anchor provisioning by the manufacturer and (4) uses standard solutions for crypto and message encodings. This work presents the design of a firmware manifest (i.e., metadata) serialization scheme based on CBOR and COSE, and a profile of CBOR Web Token (CWT) to provide access control and authentication for update authors. We demonstrate that this architecture can be realized whether or not the recipient devices support asymmetric cryptography. We then encode these data structures and find that all required metadata and authorization information for a firmware update can be encoded in less than 600 bytes with this architecture.**

***Index Terms*—ACE; SUIT; COSE; IoT; security.**

I. Introduction

The need for secure firmware updates in the Internet of Things (IoT) has been apparent for several years. Seen in a longer perspective, the IoT is still in its infancy, and the current situation regarding software updates for IoT is comparable to personal computers in the 1990s [1]. Most embedded systems do not have a system in place for remote software updates, which means device operators must manually download and install them on each device [2]. As a result, many IoT deployments are simply never updated, even after vulnerabilities are found, because the labor cost outweighs the perceived benefit.

The IoT is traditionally characterized by a lack of standards, which incentivizes companies to develop proprietary solutions [3]. For example, Texas Instruments (TI) and Amazon Web Services introduced an update framework specifically for TI devices running Amazon FreeRTOS [4]. This approach leads to *vendor lock-in*, where each manufacturer offers mutually incompatible software ecosystems. This ultimately hurts the industry and consumers: it prevents end users to freely compose networks of devices from different manufacturers, and it creates prohibitively high costs for smaller companies to enter the market and compete, whose only option might be to become sub-providers to providers of proprietary ecosystems. Embedded systems come with a wide range of hardware, operating systems, capabilities and constraints, which should not be a reason for incompatibility. New standards, such as 6LoWPAN [5], DTLS [6], CoAP [7] and OSCORE [8], enable secure IPv6 networking on devices with only tens of kilobytes of RAM, resulting in constrained devices being globally addressed with internet protocols. Although the *content* of firmware updates varies between devices, an industry-wide standard for the distribution of these updates enables the desired interoperability, where the same update infrastructure can serve multiple, or heterogeneous, deployments, instead of requiring several custom solutions. The need for common standards in the area and its challenges is identified within the Internet Engineering Task Force (IETF) standard [9] leading to the formation of the Software Updates for IoT (SUIT) working group. To have long term impact, a secure update framework must support existing embedded systems and systems which have yet to be conceived. The working group describes a firmware update solution consisting of three components: a mechanism for transporting updates, a *manifest* containing metadata about the update, and the firmware image [10]. SUIT suggests the following design requirements for the update architecture: (i) agnostic to firmware image distribution, (ii) friendly to broadcast delivery, (iii) built on state-of-the-art security mechanisms, (iv) not vulnerable to rollback attacks,

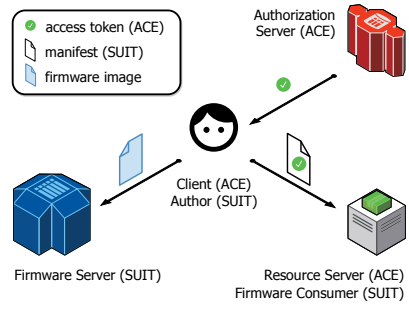


Fig. 1. Our proposed firmware update architecture, combining ACE authorization mechanisms with proposed Software Updates for IoT (SUIT) solutions.

Copyright (c) IARIA, 2022. ISBN: 978-1-68558-007-0

(v) minimal impact on existing firmware formats, (vi) enables robust permissions controls and (vii) diverse modes of operation.

Among the challenges of specifying and implementing an architecture to meet these requirements are how to solve access control and credential management. Without adequate security, an update mechanism becomes an attack vector in itself, and can be used to install malware or simply brick devices. Hence, IoT devices must be able to verify the origin and integrity of the firmware specified in the manifests, and the permissions of the update author. In this paper, we present a solution to this problem based on the Authentication and Authorization for Constrained Environments (ACE) framework. A high level illustration is shown in Figure 1. The main contributions of this work are presented through the following sections:

IV A firmware manifest design and update architecture, based on the ACE framework and SUIT recommendations, to provide both authentication and authorisation mechanisms for secure updates.

V Proposals for the use of CBOR Web Tokens (CWT) for Proof-of-Possession (PoP) in the update architecture.

VI An implementation and evaluation of the manifest and access tokens described in Sections IV and V.

The rest of the paper is organized as follows. The IoT security standards providing the basis of our update architecture are discussed in Section II. Related work is presented in Section III. In Section VII we discuss the security consideration of the proposed architecture, and conclude the paper in Section VIII.

II. Background and Threat Model

This section presents IoT security standards and protocols which form the basis of our proposed update architecture, followed by the assumed threat model.

We briefly summarize the Constrained Application Protocol (CoAP), Concise Binary Object Representation (CBOR), CBOR Object Signing and Encryption (COSE), Public Key Infrastructure (PKI), Authentication and Authorization for Constrained Environments (ACE) and CBOR Web Tokens (CWT).

A. The Constrained Application Protocol (CoAP)

Typical constrained devices are sensors, actuators or both. Heavy computations are offloaded to more powerful devices, while the nodes receive commands, transmit sensor readings and perform periodic tasks. These types of networks are well-suited to RESTful services, but traditional web protocols like HTTP incur an unacceptable overhead for small devices. This has been alleviated by CoAP, a lightweight version of HTTP using binary message encodings rather than human-readable formats and running on top of UDP instead of TCP.

B. CBOR encoding and COSE

In web applications, where computing resources are plentiful and human readability is advantageous, data representations such as XML and JSON have widespread use. For the IoT, CBOR has become the preferred encoding scheme as it is compact, offers lower message overhead and is designed for efficiency [11]. In applications requiring cryptographic operations, COSE is a standard with increasing usage in IoT [12]. COSE provides a standardized format for encryption, signing and Message Authentication Codes (MAC).

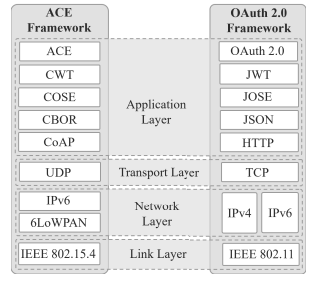


Fig. 2. Network protocols for token-based authentication in the IoT (ACE) along with their web counterparts (OAuth2.0).

C. Public Key Infrastructure (PKI)

PKI provides the basis of authentication and access control in modern networked systems, by managing the distribution and revocation of digital certificates. These certificates rely on asymmetric cryptography, which is computationally demanding for constrained devices. New standards and proposals for lightweight certificate enrollment targeting IoT have provided important PKI building blocks [13][14]. Experimental analyses of these protocols have demonstrated that PKI enrollment is now within the capabilities of constrained devices [15][16]. However, many existing IoT networks still rely on Pre-Shared Keys (PSK), shared with all parties the devices communicate with, or raw public keys (i.e., asymmetric cryptography without attached certificates).

D. The ACE Framework

ACE is an authentication and authorization framework for IoT, built on CBOR, COSE, CoAP and OAuth 2.0 [17]. Clients request access to protected resources from an Authorization Server (AS). If successful, the AS grants the client a token which is bound to a secret key in the client's possession, a specific resource and an expiration date. This token is then used as proof of authorization when accessing the Resource Server (RS). The RS can optionally send an *introspection* request to the AS to confirm the token's validity. A network stack with ACE is shown in Figure 2. In the context of our proposed architecture, the recipient IoT devices act as the RS, as illustrated in Figure 1.

There exists a number of proposals for profiling ACE to be used together with DTLS [18], OSCORE [14] or MQTT [19].

E. CBOR Web Tokens (CWT)

The ACE framework uses CWT instead of their OAuth counterpart, JSON Web Tokens (JWT) [20]. A *token* is essentially a small, serialized object containing *claims* about a

Copyright (c) IARIA, 2022. ISBN: 978-1-68558-007-0

subject, with some cryptographic guarantees generated by the *issuer* (i.e., the AS). The precise encoding of CWT claims are use-case dependent, but all signatures, MACs and encryption are done following COSE format specifications. *Access tokens* are bound to a key known to the token bearer. These are known as Proof-of-Possession (PoP) keys, and the semantics of binding them to CWTs and requesting them through ACE are described in two separate documents, [21] and [22].

F. Threat Model

Our assumptions on the capabilities of an attacker follow the Dolev-Yao adversarial model [23]. An attacker can eavesdrop and record sent messages, and inject messages into the communication. We assume that the adversary cannot break chryptographic functions, and does not have direct access to tampering with the IoT devices.

III. Related Work

Firmware updates can be grouped into two categories: image-based updates and differential updates. A 2017 survey among embedded software engineers found that almost 60% of respondents had a way of remotely updating their products and all of them used systems developed in-house, with a clear preference for image-based updates [2]. Bootloaders that utilize this approach, such as *MCUboot* [24], partition the device ROM into two sections – one for the old image and one for the new – in a way that a backup exists if the new firmware fails to boot. Differential firmware updates are far more diverse, encompassing module-based approaches [25][26], binary patching [27], binary compression [28], and more. Our work regards the secure distribution of firmware updates, and is agnostic to the firmware content or installation method.

A. Update Distribution Architectures

Software updates on systems with relatively few resource constraints are done via package managers, such as RPM or dpkg, and various commercial app stores. The trust anchors required to verify updates with PKI operations, such as code signing, are pre-installed in the operating system. A 2010 paper argued that because update architectures are an attractive target to attackers, recipients should never rely on a single signature [29]. Instead, the authors advocated for a (t, n) signature threshold scheme, whereby a recipient will not accept an update unless t out of n trusted signers have provided a signature. A profile of this scheme for constrained IoT was later proposed in 2018 [30]. Devices would be provisioned with the Original Equipment Manufacturer (OEM) certificate and trust anchors. The OEM would send signed update metadata to a device owner's *domain controller* server. This server would then sign and forward the message to the end devices; hence the update is $(2, 2)$ in the (t, n) notation.

Code signing by firmware update authors presents a problem for the IoT. In order for devices to verify the signatures, they must be provisioned with a list of authorized authors and their trust anchors. Moreover, update authors (for instance the OEM) are likely to be from outside the device owner's organization, and the device's lifetime may exceed that of the validity period of the update author's certificate which was available when initially deploying the IoT device. Our work solves these problems by incorporating a token-granting Authorization Server, which is capable of handling all certificate-based authentication on behalf of the IoT devices.

B. Software defined IoT

An approach to software updates for IoT is presented in [31], where more powerful devices act as controllers for more constrained IoT devices, building upon earlier work to define software defined networks for IoT [32]. This approach can offer solutions for heterogeneous networks which include both more powerful devices and devices which are themselves too constrained to act as fully independent endpoints, but does not address questions of standardisation.

C. Ongoing Standardisation Work

Key points of providing well specified mechanisms for secure software updates, are to achieve long time support capabilities and limit the risks of reliance on proprietary systems. Hence proposals for solutions need to relate to the ongoing standardisation efforts in the area. The SUIT working group within IETF has produced three core documents: one RFC describing the SUIT architecture [33], one RFC on a firmware manifest information model [34] and one draft specifying a proposal for a manifest format [35]. The proposal describes one instantiation of firmware manifests with CBOR/COSE encoding. It includes a new scripting language and recommendations that a series of commands should be embedded in SUIT manifests for firmware installation. This approach has its drawbacks, most notably the steep increase in parser complexity, which is likely to deter some vendors from adopting the standard. Including scripts in the manifest would also introduce new security vulnerabilities. The proposed scripting format contains instructions to verify firmware digests and check update compatibilities. This generates new issues about error handling, and how the device should proceed if an update author neglects to include critical security checks in the installation script. Our work defines a set of procedures to be followed by all manifest recipients; the manifest itself contains no instructions. The SUIT documents do not, however, describe how manifest encryption keys are to be distributed, nor how recipient devices are meant to verify author permissions. With the exception of scripting support, our manifest design follows the recommendations stated in these documents, and extends it by including lightweight solutions for authorization.

A 2019 paper by Zandberg et al. was the first to provide an implementation and performance analysis of a SUIT firmware manifest [36]. The work focused primarily on the RAM, ROM and CPU overhead incurred based on the choice of signing algorithm used for the manifest. Our work, in contrast, is focused specifically on how a SUIT manifest must be encoded to support token-based access control and key distribution, and

Copyright (c) IARIA, 2022. ISBN: 978-1-68558-007-0

considers both PSK and certificate-based use-cases. A recent survey on IoT update solutions shows that the study of SUIT related solutions is so far in its infancy, with only one other work mentioned besides the Zandberg et al. paper [37]. The short paper by Hernández-Ramos et al. discuss update related challenges. They conclude that the SUIT proposals might benefit from being aided by blockchain based mechanisms, which illustrates their complementary approaches [38].

D. Lightweight Machine-to-Machine

The Lightweight Machine-to-Machine (LwM2M) protocol is a device management protocol targeting IoT. The versions of the protocol since 2018 include a firmware update object [39]. This specification is similar to the SUIT model as it supports a *push* or *pull* architecture for firmware metadata, and firmware images can either be packaged with the metadata or retrieved from another server. However, security considerations are explicitly left outside the scope and no threat model is described. Access control, authentication and confidentiality are left entirely to the transport and application layer security mechanisms. This means that LwM2M is not a competitor to the SUIT proposals, but rather a possible framework in which the update solutions could be used. Early attempts in this directions have been reported in [40].

IV. Proposed Firmware Update Architecture

The communication architecture proposed in SUIT is flexible in a way that updates can be triggered either by the devices or the firmware/update authors (i.e., *push* or *pull*). The manifests can be distributed with or without the corresponding firmware images [33]. Our proposed architecture abides by these principles, but deviates in the way authors are authenticated and firmware is verified. SUIT states that a manifest should be directly signed by its author. This requires the provisioning of trust anchors and legitimate author identities. Moreover, the most constrained devices which still rely on symmetric keys (i.e., PSK) lack the ability to verify digital signatures. We approach this as an access control problem and provisioning devices with a list of trusted authors before deployment is insufficient for a number of reasons, such as:

- Author certificates may expire or are revoked.
- Original trusted Update Authors may fail to issue updates (e.g., when devices outlive their warranty).
- Device owners may not want to accept all updates issued by the manufacturer.

Hence we conclude that authentication is not sufficient for authorization. To address these concerns, we propose integrating the SUIT communication model with access control mechanisms provided by the ACE framework. This solution would allow device operators to centrally manage the list of authorized Update Authors (UA), and could be realized entirely using existing standard-based building blocks. Additionally, our proposed architecture can be realized whether or not the recipient IoT devices can verify digital signatures.

Combining SUIT and ACE results in the architecture illustrated in Figure 1. The recipient IoT devices act as *firmware consumers* from the SUIT perspective. Update Authors (UA) in SUIT play the role of the *client* in ACE (i.e., the entity requesting tokens). The client requests access to the firmware/update from the Authorization Server (AS). Finally the firmware updates are stored at, and can be downloaded from, a SUIT *firmware server*.

A. Authorization Tokens

A simple approach to distributing firmware updates with ACE would be to use one of the mentioned proposed profiles of the framework for secure channel establishment (with DTLS, OSCORE or MQTT). With an encrypted and mutually authenticated channel between the Update Author and recipient, manifests and images would not require further signatures or authentication codes. However, to enable a larger range of use-cases, firmware manifests must be standalone verifiable objects [9]. In our proposed update architecture, tokens are issued to the UA simply to authorize the distribution of manifests. The manifests themselves are authenticated and (partially) encrypted, and can be sent over any channel.

An ACE exchange always begins with establishing a security context between the client (i.e., UA) and the Authorization Server (AS). At this time, the AS authenticates the client and verifies their permissions to distribute updates before issuing an *access token*. If a symmetric PoP key is requested, it will be sent to the client over this secure channel. *Access tokens* are not required for the distribution of firmware images. Instead, the manifests contain a secure message digest of the corresponding image. This ensures integrity, and allows devices to retrieve firmware images from another server. The firmware retrieval could take place over an encrypted channel, or a combination of untrusted channels and encryped firmware images, depending on the confidentiality needs. We leave the details of this outside the scope of our architecture.

Our update architecture leverages the ACE framework for the provisioning of CBOR Web Tokens for PoP. There is some flexibility in how these tokens are protected and authenticated with COSE, which is discussed in Section V. The CWT standard defines a set of common claims to include in each token, but leaves the precise meaning of the fields up to the particular use-case. We use four of these and define them as:

`iss` : issuer i.e., the URI of the AS server
`aud` : audience i.e., the recipient device class's UUID
`iat` : issued at i.e, the start of the *access token's* validity
`exp` : expiration i.e., the end of the *access token's* validity

In addition, all tokens contain the confirmation field (`cnf`) which contains the PoP key, following the specification in [21].

B. Manifest Distribution

Our proposed architecture is designed to support both image-based and differential updates with dependencies. In the latter case, recipient devices must parse the dependency list, retrieve corresponding manifests, and parse their dependency lists (illustrated in Figure 3). Installing updates often requires devices to reboot, and potentially lose track of the state in the update process. We propose that devices query a known

Copyright (c) IARIA, 2022. ISBN: 978-1-68558-007-0

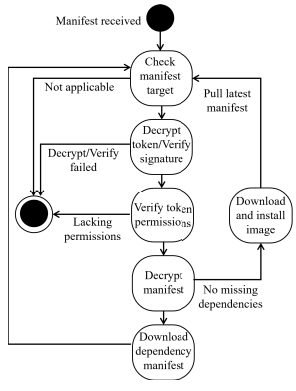


Fig. 3. Procedure followed by recipient devices for manifest dependency tree traversal and firmware update installation.

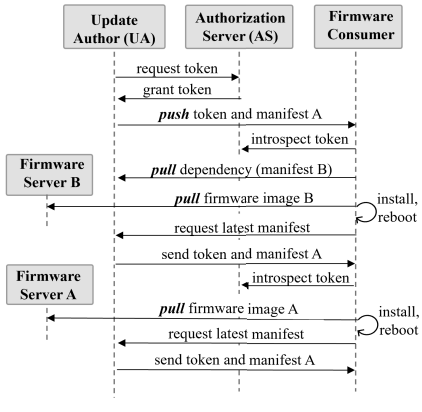


Fig. 4. Update sequence diagram for one possible use-case. The Update Author (UA) establishes a secure channel with the AS before pushing firmware manifest A to a recipient device. Since firmware update A is dependent on firmware update B, the device pulls the dependency list and parses it. Note that token introspection is an optional step.

manifest distributor at startup and request the latest manifest and corresponding access token. The device will know the update is complete when it receives a manifest matching its current firmware.

SUIT describes three categories of update architectures: *server-initiated*, *client-initiated* and *hybrid* updates. The recursive process for dependency installation used in our architecture is categorized as a client-initiated update. Figure 4 depicts interactions between actors for an update with a single dependency. The flow is server-initiated, for the cases where the update author has a known access path to the IoT device, but could easily be turned into client initiated through adding a polling step by the IoT device.

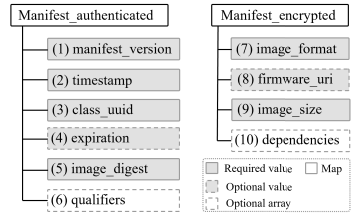


Fig. 5. Our proposed firmware manifest structure. The manifest is encoded as two separate CBOR maps, with the integer key values indicated in parentheses.

C. Manifest Design

We propose encoding firmware manifests as two separate CBOR maps: one containing information about the intended recipient of the update and another containing information about dependencies and image contents. The latter is encrypted, and both are authenticated in a single operation using an Authenticated Encryption with Associated Data (AEAD) algorithm. With this design, it is possible for a UA to broadcast an access token and manifest to a fleet of IoT devices, and any devices to which the update does not apply can quickly ascertain this without performing any cryptographic operations. Hence it is in line with SUIT recommendations to keep the update mechanism broadcast friendly. It is sensible to encrypt information about the image contents, in order to conceal information that is useful to an adversary attempting to gain insights into the software running on devices and its potential vulnerabilities. This includes the dependencies and the exact firmware URI.

In accordance with SUIT's recommendations, device classes representing the target IoT devices are given a 128-bit Universally Unique Identifier (UUID) [41], which is present through the manifest's `class_uuid` field. In our proposed architecture, devices ascertain whether the source of the manifest is authorized to issue updates by comparing this field to the `aud` value in the accompanying *access token*. A timestamp is mandatory in order to prevent rollback attacks, in which an attacker replays an earlier, legitimate firmware manifest with known vulnerabilities. IoT devices must verify that a manifest is issued more recently than their current firmware version. By storing the included timestamp of the current firmware version, a simple ordering check is sufficient to determine the temporal relation between manifests, and does not require access to a well synchronized clock.

The hash of the corresponding firmware image is included in the `image_digest` field. The URI of the firmware server can be specified in the encrypted `firmware_uri` field unless the location is already known to the devices. To handle use-cases where only devices with certain old firmware versions require a patch, the manifests optionally include a `qualifiers` list. This contains a list of firmware digests that a device must already have installed for the update to apply; otherwise it is discarded. The encrypted

Copyright (c) IARIA, 2022. ISBN: 978-1-68558-007-0

TABLE I
CRYPTOGRAPHIC ALGORITHMS EXECUTED BY THE RECIPIENT FOR EACH APPROACH DESCRIBED IN SECTION V.

	AEAD	ECDSA	ECDH	KDF
A	manifest, token			
B	manifest, token	token	manifest	
C	manifest	token	manifest	manifest
D	manifest	token	manifest	

`dependencies` field indicates a list of firmware images which must be installed before installing the present one. This enables differential updates and is handled as per Figure 3.

D. COSE Wrappers

Our proposed manifest is designed for AEAD algorithms, several of which are supported natively by the COSE standard. These algorithms take a Content Encryption Key (CEK), a plaintext and some Additionally Authenticated Data (AAD) as inputs, and produce a ciphertext as output. The unencrypted portion of our manifest design is used as the AAD, and the encrypted portion forms the plaintext. The resulting ciphertext is encapsulated in a `COSE_Encrypt` object. In total, a recipient IoT device will receive three separate CBOR-encoded objects, all of which must be valid in order to accept the update: the token, the AAD, and the COSE-wrapped encrypted manifest data. The `recipients` field in a COSE wrapper is used to encipher the CEK with Key Encryption Keys (KEK) known only to the intended recipients. There are several ways to derive this KEK, which is discussed in further detail in the upcoming sections.

V. AUTHENTICATION OPTIONS

Access control and cryptography in the IoT must be discussed in the context of device capabilities; this ultimately determines the available options. To this end, we group devices into two broad categories: (i) devices that rely entirely on PSK, (ii) devices that possess unique asymmetric key pairs (e.g., digital certificates) and can verify digital signatures. In this section we describe four distinct applications of COSE for protecting firmware manifests and the corresponding access tokens. Only the first option is applicable to devices restricted to only using PSK; the others are applicable wherever asymmetric cryptography is available, where devices are provisioned with certificates via a PKI. The message overhead of each option is analyzed in Section VI.

A. Symmetric PoP Key with PSK

Reliance on PSK for security precludes the use of digital signatures and Diffie-Hellman key exchange algorithms. In addition, since the network's security is based entirely on the secrecy of the PSK, these keys should never be sent to a third party (i.e., an Update Author). We address these constraints by issuing a unique symmetric PoP key with each access token. The key is sent to the author over its secure channel with the AS, and is also included in the `cnf` field of the access token. The token is encapsulated in a `COSE_Encrypt0` object using the network PSK for encryption by the AS, and the manifest is encapsulated in another using the PoP key. It should be noted that this approach is subject to attack vectors not present in the other authentication methods (see Section VII).

B. Symmetric PoP Key

Symmetric PoP keys are an option also where asymmetric cryptography is available. We suggest the following approach, which is not conventional, but well-suited to this particular application. The AS generates the PoP key and encrypts it *with itself* in a `COSE_Encrypt0` object. This is then included in the `cnf` field of the access token, and the token is encapsulated as the payload of a `COSE_Sign1` object signed by the AS. (The following later verification of this signature is what requires asymmetric cryptography capabilities by the receiving IoT device.) The UA distributes the CEK to recipient devices via the `recipients` field in the manifest's COSE wrapper. Recipients can then verify that this CEK is the one contained in the signed token by decrypting the `cnf` field. The motivation for this approach is to avoid including any recipients in the token itself, as this would require the AS to have knowledge of the intended recipients' public keys. The UA must know the recipients' public keys in order to encipher the CEK.

C. Asymmetric PoP Key, Direct Key Agreement

In the case of asymmetric PoP keys, the `cnf` field of the CWT contains the COSE encoding of a public key belonging to the UA. The token is then encapsulated in a `COSE_Sign1` wrapper. The UA now has two options for deriving a CEK for the manifest. The first is through direct key agreement. This type of algorithm applies a key exchange protocol – in this case Elliptic Curve Diffie-Hellman (ECDH) – and a Key Derivation Function (KDF) to generate the CEK directly. The author must use the key pair bound to the token to prove their authorization.

D. Asymmetric PoP Key, Key Wrap

The second asymmetric PoP key approach is to use the key derived through ECDH as a Key Encryption Key (KEK) to encipher a randomly-generated ephemeral CEK. These two approaches have implementation nuances and security considerations which are discussed in Sections VI and VII. Table I summarizes the cryptographic operations that recipient devices much perform in order to process manifests and tokens with each of the four described authentication options.

VI. IMPLEMENTATION

The encoding scheme for each authentication options discussed in the previous section is shown in Table II. In this section, we generate firmware manifests and access tokens for each of the four cases. The purpose of this exercise is both to demonstrate the viability of the proposed architecture, and to evaluate the differences in storage and transmission overhead.

Copyright (c) IARIA, 2022. ISBN: 978-1-68558-007-0

TABLE II
COSE WRAPPERS FOR EACH MANIFEST-TOKEN COMBINATION DESCRIBED IN SECTION V.

	Authentication	Manifest	Token
A	PSK	`COSE_Encrypt0`	`COSE_Encrypt0`
B	Symmetric PoP key	`COSE_Encrypt`	`COSE_Sign1`
C	Asymmetric PoP key	`COSE_Encrypt`	`COSE_Sign1`
D	Asymmetric PoP key	`COSE_Encrypt`	`COSE_Sign1`

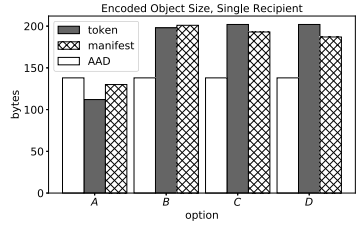


Fig. 6. Encoded sizes of the **manifest**, **token** and **AAD** for each approach in Section V.

A. Profile and Assumptions

For our implementation and analysis, we populate the manifest fields illustrated in Figure 5 with example data. In order to do so we make the following assumptions:

- Images are identified with 32-byte digests.
- Timestamps are represented in relative time.
- The manifest has two qualifiers and two dependencies.
- The firmware server URI is *coaps://example.com*.
- The Authorization Server URI is *coaps://example.com*.

The authenticated and encrypted example manifest components are 138 and 116 bytes, respectively, after CBOR serialization. COSE offers a variety of algorithms with a range of key sizes for each cryptographic operation. For our implementation, we have chosen the following:

- ECDSA signatures with 256-bit keys.
- AES-CCM with 128-bit keys, 64-bit tag and a 13-byte nonce for content encryption.
- AES 128-bit key wrap.
- ECDH Ephemeral-Static (ES).
- HMAC-Based Extract-and-Expand Key Derivation Function (HKDF) with SHA-256.

B. Results and comparison with other SUIT proposals

The update and authentication information is separated into three separate CBOR-encoded objects: the **token**, the encrypted **manifest** data, and the plaintext authenticated manifest data (a.k.a. the additionally authenticated data, or **AAD**). The results are shown in Figure 6. Option A has the smallest total size, with all three CBOR objects totalling 380 bytes. Option C has the largest footprint, totalling 537 bytes. Since the differences are relatively minor, the choice of method should be guided by the offered security properties, as discussed below in VII.

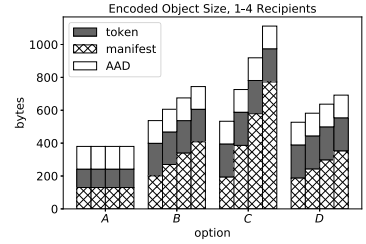


Fig. 7. Encoded sizes of the **manifest**, **AAD** and **token** when the update has multiple recipients.

In the most recent SUIT manifest proposal there are example manifest samples, which allow us to compare our proposals with the draft. In [35] the minimal manifest is only 237 bytes, but for example manifests with content similar to the sample used in our evaluation, the size is between 270 and 400 bytes. The main difference is the addition of our relatively large access tokens, since they are designed to be independent authorization tokens, compliant with ACE requirements. Given this added security functionality we find the added overhead to be clearly acceptable.

In some deployments, it may be preferable for UAs to upload both manifests and firmware images to a dedicated firmware server to be retrieved by devices at a later time. This is feasible within our framework as long as the corresponding access token is stored alongside the manifest. The storage overhead for manifests encoded with multiple recipients is shown in Figure 7. The plot shows the encoded size of the required objects for 1-4 recipients for each authentication method. In Option A, all recipients receive an identical manifest since they possess the same PSK. In Option B and D, the CEK is wrapped, each additional recipient only requires an additional entry in the recipients field of the COSE object. Option C, however, derives a unique CEK for each recipient, which means the manifest must be re-encrypted for every target device, making C the least efficient option for broadcast scenarios.

VII. SECURITY CONSIDERATIONS

The proposals in this paper are founded on well-vetted standards and encryption algorithms. However, there are protocol details that must be fully understood in order to avoid security lapses. A malicious firmware image could permanently disable expensive hardware and compromise an entire network, therefore, great care must be taken to ensure an update distribution mechanism does not become an attack vector in itself.

A. Non-Repudiation

The PSK use-case described in Section V-A precludes any guarantees for the access token. Since the AS uses a symmetric key known to all recipients, an adversary with control over any device would be capable of generating fraudulent tokens

and PoP keys. This is problematic, although PSK networks are already subjected to similar risks. Any adversary in possession of a PSK could cause significant damage and disruption, even without the ability to issue firmware updates. If a symmetric PoP key is used and the token is signed by the AS, as in Section V-B, non-repudiation is only guaranteed for the token, but not necessarily the manifest. The UA must encipher the PoP key for each recipient, so if any of the recipients are controlled by an adversary, that adversary would then be in possession of a valid token and the associated PoP key. The use of symmetric PoP keys also breaks end-to-end security between the author and recipients, because the key is known to the AS. The analysis presented in Section VI demonstrated that asymmetric PoP keys with Key Wrap has a similar overhead but without the risks, making that approach clearly preferable.

B. Key Agreement

The manifest exchange between the author and recipients is one-way, i.e., there is no nonce exchange or handshake like in DTLS or EDHOC. The manifest's CEK is either wrapped (Options B and D) or derived directly (Option C), as described in Section V from the author and recipients' key pairs. In COSE, ECDH key derivation comes in two types: Static-Static (SS) or Ephemeral-Static (ES). In the former case, the author of the COSE object declares that the CEK is either wrapped or derived from the key pairs bound to the author and recipient. In the latter case, the author of the COSE object provides an ephemeral key pair generated for a single encryption operation. ECDH-ES is generally safer to use, because even if an adversary obtains the author's private key, it is not usable for decryption of other manifests or impersonation of the author. It is therefore preferable for UAs to request access tokens bound to an ephemeral public key, not the public key found in their certificate.

C. Firmware Image Digests

Firmware manifests are only linked to firmware images via the inclusion of a secure message digest. If a weak algorithm with the possibility of a hash collision is used for this purpose, such as SHA-1, devices may be exposed to fraudulent images referenced by authentic manifests.

VIII. Conclusion

In this work we have presented an architecture based on existing standards, which can address the urgent need for secure firmware updates in the IoT. We have described the challenges and limitations of access control in constrained environments, and why a token-based framework, such as ACE, is a promising candidate solution. In addition, we have proposed encoding schemes for firmware manifests using the CBOR and COSE standards, and detailed how these would work in conjunction with CWT to provide authorized updates. Examples of these objects were encoded and the result totaled no more than 600 bytes for the firmware manifest data, including authentication and authorization.

Acknowledgments

This research is partially funded by the Swedish Foundation for Strategic Research (SSF) Institute PhD grant, the SSF aSSIsT project and by the H2020 CONCORDIA (Grant ID: 830927) project.

References

[1] B. Schneier, "The Internet of Things is Wildly Insecure–And Often Unpatchable," January 2014. [Online]. Available: https://www.wired.com/2014/01/theres-no-good-way-to-patch-the-internet-of-things-and-thats-a-huge-problem/

[2] E. Stenberg. (2017, September) Key Considerations for Software Updates for Embedded Linux and IoT. [Online]. Available: https://www.linuxjournal.com/content/key-considerations-software-updates-embedded-linux-and-iot

[3] J. P. Vasseur and A. Dunkels, *Interconnecting Smart Objects with IP: The Next Internet*. Morgan Kaufmann, 2010.

[4] N. Lethaby, "A more secure and reliable OTA update architecture for IoT devices," Texas Instruments, Tech. Rep., 2018. [Online]. Available: http://www.ti.com/lit/wp/sway021/sway021.pdf

[5] G. Montenegro, J. Hui, D. Culler, and N. Kushalnagar, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks," RFC 4944, Sep. 2007.

[6] "Datagram Transport Layer Security Version 1.2," RFC 6347, Jan. 2012.

[7] Z. Shelby, K. Hartke, and C. Bormann, "The Constrained Application Protocol (CoAP)," RFC 7252, Jun. 2014.

[8] G. Selander, J. Mattsson, F. Palombini, and L. Seitz, "Object Security for Constrained RESTful Environments (OSCORE)," RFC 8613, RFC Editor, Tech. Rep. 8613, Jul. 2019. [Online]. Available: https://rfc-editor.org/rfc/rfc8613.txt

[9] H. Tschofenig and S. Farrell, "Report from the Internet of Things Software Update (IoTSU) Workshop 2016," RFC 8240, RFC Editor, Tech. Rep. 8240, September 2017. [Online]. Available: https://tools.ietf.org/html/rfc8240

[10] B. Moran, M. Meriac, H. Tschofenig, and D. Brown, "A Firmware Update Architecture for Internet of Things Devices," Internet Engineering Task Force, Internet-Draft draft-ietf-suit-architecture-05, Apr. 2019, work in Progress.

[11] C. Bormann and P. Hoffman, "Concise Binary Object Representation (CBOR)," Internet Requests for Comments, RFC Editor, RFC 7049, October 2013.

[12] J. Schaad, "CBOR Object Signing and Encryption (COSE)," RFC 8152, RFC Editor, Tech. Rep. 8152, Jul. 2017. [Online]. Available: https://rfc-editor.org/rfc/rfc8152.txt

[13] P. van der Stok, P. Kampanakis, M. Richardson, and S. Raza, "EST-coaps: Enrollment over Secure Transport with the Secure Constrained Application Protocol," Internet Requests for Comments, RFC Editor, RFC 9148, April 2022.

[14] G. Selander, S. Raza, M. Furuhed, M. Vučinić, and T. Claeys, "Protecting est payloads with oscore," Working Draft, IETF Secretariat, Internet-Draft draft-selander-ace-coap-est-oscore-05, May 2021. [Online]. Available: https://www.ietf.org/archive/id/draft-selander-ace-coap-est-oscore-05.txt

[15] Z. He, M. Furuhed, and S. Raza, "Indraj: Certificate Enrollment for Battery-powered Wireless Devices," in *Proceedings of the 12th ACM Conference on Security and Privacy in Wireless and Mobile Networks*. ACM, 2019.

[16] J. Höglund, S. Lindemer, M. Furuhed, and S. Raza, "PKI4IoT: Towards public key infrastructure for the Internet of Things," *Computers & Security*, vol. 89, 2020.

[17] L. Seitz, G. Selander, E. Wahlstroem, S. Erdtman, and H. Tschofenig, "Authentication and authorization for constrained environments (ace) using the oauth 2.0 framework (ace-oauth)," Working Draft, IETF Secretariat, Internet-Draft draft-ietf-ace-oauth-authz-46, November 2021.

[18] S. Gerdes, O. Bergmann, C. Bormann, G. Selander, and L. Seitz, "Datagram transport layer security (dtls) profile for authentication and authorization for constrained environments (ace)," Working Draft, IETF Secretariat, Internet-Draft draft-ietf-ace-dtls-authorize-18, June 2021. [Online]. Available: https://www.ietf.org/archive/id/draft-ietf-ace-dtls-authorize-18.txt

Copyright (c) IARIA, 2022. ISBN: 978-1-68558-007-0

[19] C. Sengul and A. Kirby, "Message queuing telemetry transport (mqtt)-tls profile of authentication and authorization for constrained environments (ace) framework," Working Draft, IETF Secretariat, Internet-Draft draft-ietf-ace-mqtt-tls-profile-17, March 2022. [Online]. Available: https://www.ietf.org/archive/id/draft-ietf-ace-mqtt-tls-profile-17.txt

[20] M. Jones, E. Wahlstroem, S. Erdtman, and H. Tschofenig, "CBOR Web Token (CWT)," RFC 8392, May 2018.

[21] M. Jones, L. Seitz, G. Selander, S. Erdtman, and H. Tschofenig, "Proof-of-possession key semantics for cbor web tokens (cwts)," Internet Requests for Comments, RFC Editor, RFC 8747, March 2020.

[22] L. Seitz, "Additional oauth parameters for authorization in constrained environments (ace)," Working Draft, IETF Secretariat, Internet-Draft draft-ietf-ace-oauth-params-16, September 2021. [Online]. Available: https://www.ietf.org/archive/id/draft-ietf-ace-oauth-params-16.txt

[23] D. Dolev and A. Yao, "On the security of public key protocols," *IEEE Transactions on Information Theory*, vol. 29, no. 2, pp. 198–208, 1983.

[24] MCUboot contributors, "MCUboot," https://github.com/mcu-tools/mcuboot, 2022.

[25] A. Dunkels, N. Finne, J. Eriksson, and T. Voigt, "Run-time dynamic linking for reprogramming wireless sensor networks," in *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems*, ser. SenSys '06. New York, NY, USA: ACM, 2006, pp. 15–28. [Online]. Available: http://doi.acm.org/10.1145/1182807.1182810

[26] P. Ruckebusch, E. De Poorter, C. Fortuna, and I. Moerman, "Gitar," *Ad Hoc Netw.*, vol. 36, no. P1, pp. 127–151, Jan. 2016. [Online]. Available: https://doi.org/10.1016/j.adhoc.2015.05.017

[27] Jaein Jeong and D. Culler, "Incremental network programming for wireless sensors," in *2004 First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks, 2004. IEEE SECON 2004.*, Oct 2004, pp. 25–33.

[28] M. Stolikj, P. J. L. Cuijpers, and J. J. Lukkien, "Efficient reprogramming of wireless sensor networks using incremental updates," in *2013 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, March 2013, pp. 584–589.

[29] J. Samuel, N. Mathewson, J. Cappos, and R. Dingledine, "Survivable key compromise in software update systems," in *Proceedings of the 17th ACM Conference on Computer and Communications Security*, ser. CCS '10. New York, NY, USA: ACM, 2010, pp. 61–72. [Online]. Available: http://doi.acm.org/10.1145/1866307.1866315

[30] N. Asokan, T. Nyman, N. Rattanavipanon, A. Sadeghi, and G. Tsudik, "Assured: Architecture for secure software update of realistic embedded devices," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 11, pp. 2290–2300, Nov 2018.

[31] N. Xue, D. Guo, J. Zhang, J. Xin, Z. Li, and X. Huang, "Openfunction for software defined iot," in *2021 International Symposium on Networks, Computers and Communications (ISNCC)*, 2021, pp. 1–8.

[32] Y. Jararweh, M. Al-Ayyoub, A. Darabseh, E. Benkhelifa, M. Vouk, and A. Rindos, "Sdiot: A software defined based internet of things framework," *Journal of Ambient Intelligence and Humanized Computing*, vol. 1, pp. 453–461, 08 2015.

[33] B. Moran, H. Tschofenig, D. Brown, and M. Meriac, "A firmware update architecture for internet of things," Internet Requests for Comments, RFC Editor, RFC 9019, April 2021.

[34] B. Moran, H. Tschofenig, and H. Birkholz, "A manifest information model for firmware updates in internet of things (iot) devices," Internet Requests for Comments, RFC Editor, RFC 9124, January 2022.

[35] B. Moran, H. Tschofenig, H. Birkholz, and K. Zandberg, "A concise binary object representation (cbor)-based serialization format for the software updates for internet of things (suit) manifest," Working Draft, IETF Secretariat, Internet-Draft draft-ietf-suit-manifest-17, April 2022.

[36] K. Zandberg, K. Schleiser, F. Acosta, H. Tschofenig, and E. Baccelli, "Secure firmware updates for constrained IoT devices using open standards: A reality check," *IEEE Access*, vol. 7, pp. 71 907–71 920, 2019.

[37] S. El Jaouhari and E. Bouvet, "Secure firmware over-the-air updates for iot: Survey, challenges, and discussions," *Internet of Things*, vol. 18, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2542660522000142

[38] J. L. Hernández-Ramos, G. Baldini, S. N. Matheu, and A. Skarmeta, "Updating iot devices: challenges and potential approaches," in *2020 Global Internet of Things Summit (GIoTS)*, 2020, pp. 1–5.

[39] "Lightweight Machine to Machine Technical Specification 1.0.2," Open Mobile Alliance, Tech. Rep. OMA-TS-LightweightM2M-V1_0_2-20180209-A, February 2018.

[40] IETF. Ietf hackathon: Software / firmware updates for iot devices. IETF. [Online]. Available: https://datatracker.ietf.org/meeting/111/materials/slides-111-suit-suit-hackathon-report-00

[41] P. J. Leach, R. Salz, and M. H. Mealling, "A Universally Unique IDentifier (UUID) URN Namespace," RFC 4122, Jul. 2005. [Online]. Available: https://rfc-editor.org/rfc/rfc4122.txt

Copyright (c) IARIA, 2022. ISBN: 978-1-68558-007-0