International Journal of Information Security
https://doi.org/10.1007/s10207-024-00825-z

REGULAR CONTRIBUTION

AutoPKI: public key infrastructure for IoT with automated trust transfer

Joel Höglund[1] · Simon Bouget[1] · Martin Furuhed[2] · John Preuß Mattsson[3] · Göran Selander[3] · Shahid Raza[1]

Abstract
IoT deployments grow in numbers and size, which makes questions of long-term 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-term maintainability, and risk vendor lock-in. The manual overhead must be kept low for large-scale IoT installations to be economically feasible. 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 IoT · PKI · Digital certificates · Enrollment · Embedded systems

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-term robustness and secure management.

Security solutions, such as secure communication and authentication, have been adapted to suit 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 standardized. 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-term support and maintainability of IoT deployment. This area 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 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 and inclusions of existing protocols together with new proposals where there are currently 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 of manual intervention, 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

✉ Joel Höglund
joel.hoglund@ri.se

Simon Bouget
simon.bouget@ri.se

Martin Furuhed
martin.furuhed@nexusgroup.com

John Preuß Mattsson
john.mattsson@ericsson.com

Göran Selander
goran.selander@ericsson.com

Shahid Raza
shahid.raza@ri.se

1 RISE Research Institutes of Sweden, Stockholm, Sweden

2 Nexus Group, Stockholm, Sweden

3 Ericsson, Stockholm, Sweden

Published online: 02 March 2024

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 work presented here continues and extends our work presented in [1], with the formal theorem prover analysis being the most significant addition, together with more in-depth coverage of compact certificates.

The rest of this paper is organized as follows: Sect. 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 (10). In Sect. 11 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

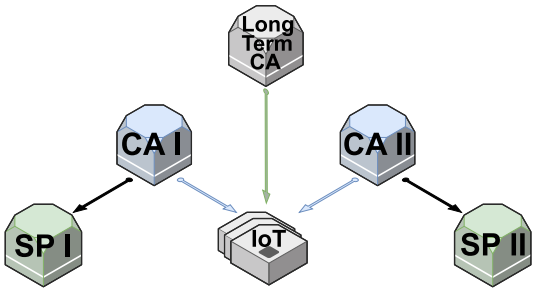
A cornerstone of PKIs is authentication through publicly available keys, keys encapsulated in certificates signed by a certificate authority, CA. The certificate authorities are identified by their certificates, which can be either self-signed or signed by another CA. This system of signed certificates 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 [2].

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]

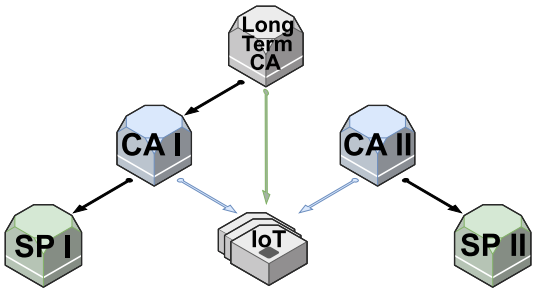
The IoT devices act as leaf nodes at the lowest layers of the CA hierarchies, together with the service provider servers, with which the devices need to communicate. Figure 1 illustrates some alternatives for CA hierarchies, depending on the trust relations between the entities, involving two different 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 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 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

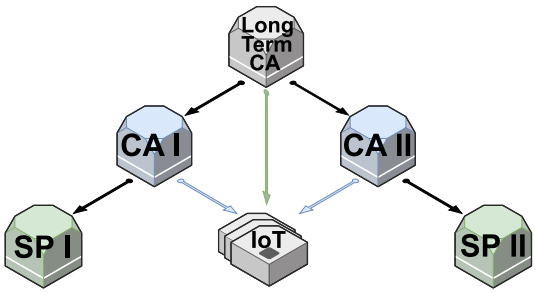
[1] Advancements in both IoT capabilities and solutions targeting IoT have made PKI enrollment solutions feasible for IoT [3, 4].



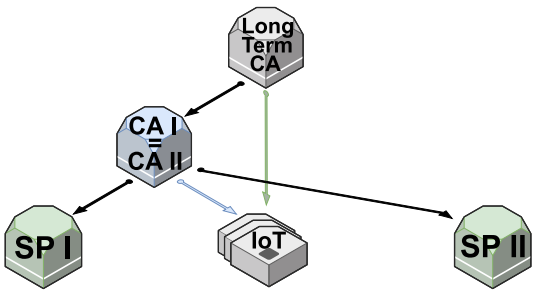
(a) Three separate trust roots



(b) Two separate trust roots



(c) One trust root, different operational CAs



(d) One trust root, one operational CA

Fig. 1 Different options for CA hierarchies. All arrows represent certificate issuing, green arrows for factory certificates, blue arrows for operational IoT certificates (colour figure online)

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 [5].

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 [2]. 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 *organizational 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 [6]. 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 [7] 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 [8] 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.

Identity based encryption and certificateless cryptography: Identity based encryption, ID, where public keys for an entity can derived using only relatively short and publicly known identifiers, offers a more lightweight alternative to standard PKI, with respect to the PKI certificate handling. These solutions rely on a completely trusted third-party

Private Key Generator, PKG, to provide the private keys, introducing a single point of failure for the entire system. In addition, they have difficulties handling revocation since this requires a full withdrawal of the publicly known identity. To overcome the key escrow problem where the PKG has full knowledge of all private keys, certificateless cryptography solutions have been proposed, where also the user takes part in the key generation process. This on the other hand reintroduces the need to distribute the public keys, as they can no longer be computed only from the public information. In addition the offered security will depend on the underlying schema, with no widely accepted models available [9, 10].

Certificateless cryptography is an active area of research, with new models and mechanisms being proposed, including attempts to limit resource needs [11]. For IoT deployments with known organizational boundaries and no outside interoperability needs, they might offer an efficient alternative. The overall conclusion is that the certificateless solutions are currently not capable of providing building blocks for large-scale interoperable key management services.

Further PKI alternatives: For scenarios with IoT devices that are incapable of running PKI mechanisms at all, several 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 [12]. For devices with more capabilities [13] 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 standardized. 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.

A protocol is needed for the interactions involved in the handover to be both secure and efficient in terms of minimal manual efforts. In the following, we present an enabling PKI environment, details on the required interactions, and how our proposed protocol fulfils 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 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 often communicate 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 [14], 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 (Sect. 8.3) and our formal analysis, where we also consider the case where SP1 acts as an attacker (Sect. 11).

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.

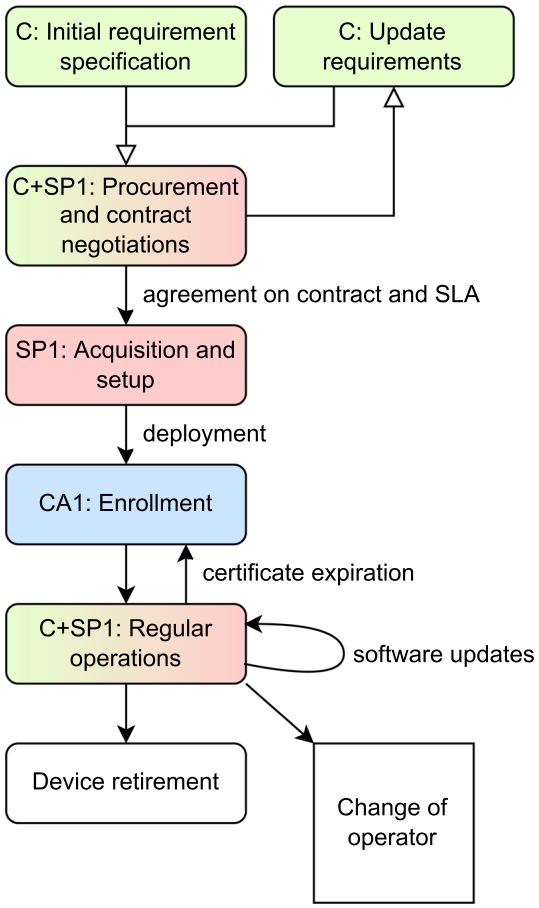


Fig. 2 The IoT life cycle

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. 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 Fig. 2. A more detailed diagram of the initial life cycle phases is given in Fig. 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 that need to be treated separately and 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 Sect. 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

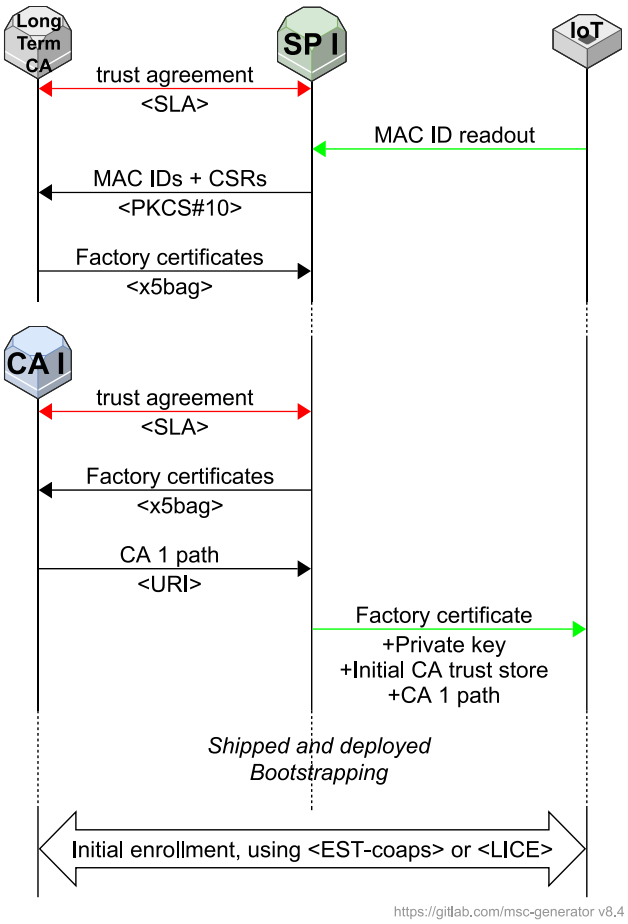


Fig. 3 The IoT device's initial life cycle stages, showing the standards used for setup and enrollment. Red arrows correspond to operations where manual intervention is expected. Green arrows are deployment-specific, while black arrows are standard-based and fully automated (colour figure online)

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 [15]. 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 [16] considered cloud environments, while newer works also address IoT scenarios. For example, in [17] 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 [18].

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 Fig. 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 [19]. 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 also offer 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-term 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 [20]. 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 [21] (see also Sect. 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 emphasizing 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 [22]. 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 automatized, 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 Fig. 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 [21]. 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 C509 work has been expanded beyond an initial IoT profile coverage (corresponding to RFC 7925 [23]) to specify identifiers 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 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.

The overall protocol design is agnostic to the certificate format used, as long as viable protocols for enrollment, revocation and secure communication establishment in general exist. This means that other compact formats such as implicit ECQV certificates could be used instead, if the issues regarding certificate management are resolved [24].

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.

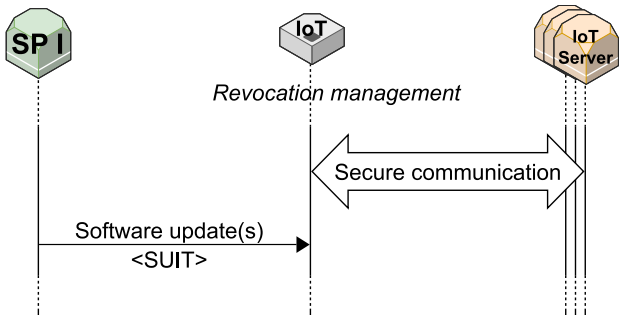


Fig. 4 The IoT interactions during normal operations

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 [25]. 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 [21], 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 [26] or the proposed more flexible EDHOC-based enrollment protocol [4]. If the device is using the proposed compact enrollment protocol the enrollment message exchange can be encapsulated already inside an EDHOC handshake.

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 [27]. Before the operational certificate expires, the device will do re-enrollment with the CA1. The normal operations are illustrated in Fig. 4.

8 IoT trust transfer

8.1 Introduction and problem formulation

A motivating high-level use case was given in Sect. 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 Fig. 5, and the pseudocode for the main actors is listed in the three procedures below.

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
```

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.

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 mem-

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
```

ory areas. There is active research in the area [28] as well as a large ongoing IETF standardization effort [29].

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 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:

Listing 2 UpdateInfoList

```
UpdateInfoList = [* DeviceUpdateInfo]
DeviceUpdateInfo = (
  factoryCertificate:   TBSCertificate,
  updateTimeNotBefore:  Time,
  updateTimeNotAfter:   Time,
  versionInfo:          (suit-manifest-seq-number,
                         suit-reference-uri),
)
```

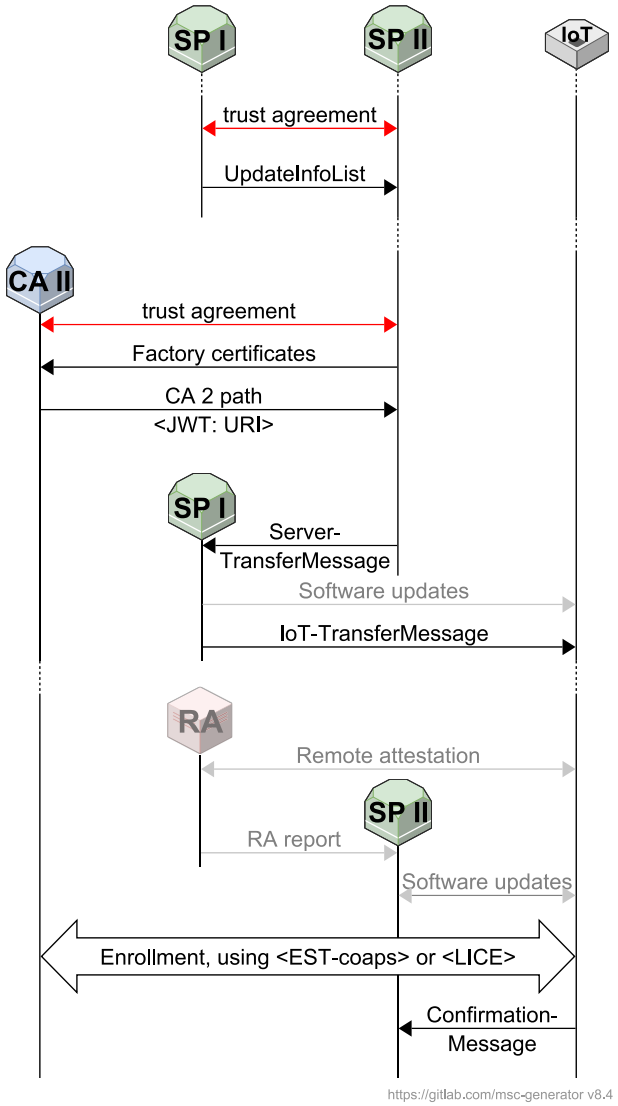


Fig. 5 AutoPKI, Operator change. Automated operations in black, optional operations in gray

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 Fig. 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 is 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. 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 certifi-

cate, 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 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 [30].

9 AutoPKI: implementation and evaluation

Based on the protocol design goals, to target resource-constrained devices, the critical target is to show that the protocol overhead is sufficiently small to match IoT device capabilities.

9.1 Implementation

In the following, we validate the proposed building blocks in terms of messaging, computational and memory overhead using a Contiki-NG OS-based prototype implementation in C for constrained IoT platforms.

A complete AutoPKI system involves both the IoT-specific components as well as a number of server-side components. In our test setup, the server-side components are represented by relatively basic test modules, which have served to validate that the server side functionality does not pose any performance bottlenecks. A full setup includes, for the CAs: functionality for enrollment and revocation, which could be based either around the legacy X.509 format, with a small component for C509 conversion or a more modern setup working directly with C509 in native format. The enrollment could be done over DTLS, OSCORE or integrated with EDHOC [3, 4]. For both the CAs and the service providers: components for bulk handling of factory certificates, and for communicating URI metadata in a web-token format. For the service providers: components for processing of Transfer- and Confirmation messages. Optionally and finally, components for remote attestation and software updates.

9.2 Evaluation

This section reports the micro benchmarks corresponding to the critical protocol operations from the constrained IoT device point of view. 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. It should be noted that the required cryptographic operations are hardware agnostic. Most of the underlying functionality has also been demonstrated to work on even more constrained platforms, such as the Zolertia Firefly [3, 31]. While the required crypto operations can be performed on the older generation of IoT platforms, the available memory becomes a limiting factor for fitting a complete system while still having space available for normal operations. With this perspective, the current mid-range IoT devices, as represented by the selected nRF platform, form a

Table 1 Protocol message size in bytes, lower bounds for compact certificates

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

suitable target for IoT deployments capable of updates and long-time support.

9.2.1 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 shown 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.

9.2.2 IoT computational and time 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.

9.2.3 IoT 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 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 [32], and could therefore coexist with the required PKI components on more capable devices such as nRF52840.

10 Non-functional requirement compliance

The functional requirements are assessed below in Sect. 11. 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 Fig. 1 which need to be supported. The requirements, together with the needed volumes, all affect the resulting price offer.[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 [33]. This price range is infeasible for large-scale IoT deployments. Depending on the deployment scenario, companies might benefit from running their own CA service. In general, self-signed certificates are not automatically recognized and trusted by external parties, and for a non-CA company to be verified and trusted as a globally trusted sub-CA comes with a significant effort. While the solution could lower the cost per individual certificate, the total cost will depend on the cost of the CA software, and the maintenance costs stemming from keeping

[2] Nexus company policies.

the complex software and related infrastructure trusted and up to date.

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 standardization 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.

11 AutoPKI: security assessment

The security assessment of the protocol builds upon the derivations done in the SIGMA paper [34]. 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.

11.1 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.

The above-presented capabilities together with previous positive experience with the tool made Tamarin a suitable choice for our modelling tasks. Other theorem provers could potentially also have been used.

11.2 Modelling choices

The Tamarin prover by default has a Dolev-Yao adversary model, introduced in Sect. 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 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.

[3] Available online at https://tamarin-prover.github.io/.

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-term 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.

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.

11.3 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 Sect. 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 Sect. 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.

11.4 Further considerations

While the solutions proposed in this paper fill 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 standardized.

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.

When IoT deployments become more common and grow in size, issues of long-term maintenance and the scalability of the security services become critical. Making use of proposed and available PKI solutions suitable for IoT we have proposed a lightweight protocol for the transfer of control of IoT deployments, with a minimal manual overhead. The solution ensures the possibility of long-term support for IoT deployments, preventing vendor lock-in. We have shown that given the integrity of the secure building blocks, the protocol maintains the desired security properties.

12 Conclusion

In the context of expanding Internet of Things (IoT) deployments, the challenges associated with long-term maintenance and the scalability of security services assume paramount importance. This necessitates viable Public Key Infrastructure (PKI) solutions tailored specifically for IoT environments. In this study, we propose a protocol designed for the seamless transfer of control within IoT deployments, accomplished through the transference of trust from one operational domain to another. A fundamental objective of this protocol is to minimize manual intervention while ensuring the integrity and security of the IoT ecosystem.

The proposed solution addresses the critical concerns of establishing enduring support mechanisms for IoT deployments while mitigating the risks of vendor lock-in. To validate our protocol, formal verification tools are employed. The results of our analysis demonstrate that assuming the integrity of the foundational secure components, the protocol preserves the desired security properties. Through this work, we contribute to the body of knowledge aimed at increasing the robustness and reliability of IoT systems.

Acknowledgements This research is partially funded by the Swedish SSF Institute PhD grant and by the EU H2020 projects ARCADIAN-IoT (Grant ID. 101020259) and VEDLIoT (Grant ID: 957197).

Author Contributions J.H. wrote the initial manuscript text, S.B. provided input on the formal modelling sections. J.H., M.F. and S.R. have worked together on developing solutions for enrollment and revocation. J.H., J.M., G.S. and S.R. have worked together on developing the compact certificates. All authors reviewed the manuscript.

Funding Open access funding provided by RISE Research Institutes of Sweden.

Data availibility The full data from the feasibility study are available from the corresponding author upon reasonable request.

Declarations

Conflict of interest The authors have no conflicts of interest relevant to the content of this article and the work covered does not involve any human participants or animal usage.

References

1. Höglund, J., Raza, S., Furuhed, M.: In *2022 IEEE International Conference on Public Key Infrastructure and its Applications*

(PKIA) (2022), pp. 1–8. https://doi.org/10.1109/PKIA56009.2022.9952223
2. Housley, R., Ford, W., Polk, T., Solo, D.: Internet X.509 Public Key Infrastructure Certificate and CRL Profile. RFC 2459, RFC Editor (1999)
3. Höglund, J., Lindemer, S., Furuhed, M., Raza, S.: PKI4IoT: towards public key infrastructure for the Internet of Things. Comput. Secur. **89** (2020). https://doi.org/10.1016/j.cose.2019.101658
4. Höglund, J., Raza, S.: In: *IEEE Conference on Communications and Network Security, CNS 2021, Tempe, AZ, USA, October 4–6, 2021* (IEEE, 2021). https://doi.org/10.1109/CNS53000.2021.9705036
5. Selander, G., Mattsson, J., Palombini, F.: Ephemeral Diffie–Hellman over cose (edhoc). Internet-Draft draft-ietf-lake-edhoc-03, IETF Secretariat (2020)
6. Schoorman, F.D., Mayer, R.C., Davis, J.H.: An integrative model of organizational trust: Past, present, and future. Acad. Manag. Rev. **32**(2), 344 (2007)
7. Khan, M.S.N., Marchal, S., Buchegger, S., Asokan, N.: In: Privacy and Identity Management. Fairness, Accountability, and Transparency in the Age of Big Data, vol. 547, pp. 205–221 (2018). https://doi.org/10.1007/978-3-030-16744-8_14
8. Gunnarsson, M., Gehrmann, C.: In: Proceedings of the 6th International Conference on Information Systems Security and Privacy, vol. 1, ed. by S. Furnell, P. Mori, E. Weippl, O. Camp (SciTePress, 2020), vol. 1, pp. 33–44. https://doi.org/10.5220/0008928300330044
9. Dent, A.W.: Certificateless Cryptography (Springer US, Boston, MA, 2011), pp. 192–193. https://doi.org/10.1007/978-1-4419-5906-5_314
10. Dent, A.W.: In: Public Key Infrastructures, Services and Applications, ed. by F. Martinelli, B. Preneel (Springer Berlin Heidelberg, Berlin, Heidelberg, 2010), pp. 1–16
11. Karati, A., Islam, S.H., Karuppiah, M.: Provably secure and lightweight certificateless signature scheme for IIoT environments. IEEE Trans. Ind. Inf. **14**(8), 3701 (2018). https://doi.org/10.1109/TII.2018.2794991
12. Safkhani, M., Rostampour, S., Bendavid, Y., Sadeghi, S., Bagheri, N.: Improving RFID/IoT-based generalized ultra-lightweight mutual authentication protocols. J. Inf. Secur. Appl. **67**, 103194 (2022) https://doi.org/10.1016/j.jisa.2022.103194
13. AbuAlghanam, O., Qatawneh, M., Almobaideen, W., Saadeh, M.: A new hierarchical architecture and protocol for key distribution in the context of IoT-based smart cities. J. Inf. Secur. Appl. **67**, 103173 (2022). https://doi.org/10.1016/j.jisa.2022.103173
14. Dolev, D., Yao, A.: On the security of public key protocols. IEEE Trans. Inf. Theory **29**(2), 198 (1983). https://doi.org/10.1109/TIT.1983.1056650
15. Müller, C., Gutierrez, A.M., Fernandez, P., Martín-Díaz, O., Resinas, M., Ruiz-Cortés, A.: Automated validation of compensable SLAs. IEEE Trans. Serv. Comput. **14**(5), 1306 (2021). https://doi.org/10.1109/TSC.2018.2885766
16. Uriarte, R.B., de Nicola, R., Kritikos, K.: In: 2018 IEEE International Conference on Cloud Computing Technology and Science (CloudCom) (2018), pp. 266–271. https://doi.org/10.1109/CloudCom2018.2018.00059
17. Alzubaidi, A., Mitra, K., Solaiman, E.: In: 2021 IEEE International Conference on Smart Internet of Things (SmartIoT) (2021), pp. 74–81. https://doi.org/10.1109/SmartIoT52359.2021.00021
18. Beniiche, A.: ArXiv arXiv:2004.07140 (2020)
19. IEEE Std 802.1AR-2018 pp. 1–73 (2018). https://doi.org/10.1109/IEEESTD.2018.8423794
20. Nystrom, M., Kaliski, B.: PKCS #10: Certification Request Syntax Specification Version 1.7. RFC 2986, RFC Editor (2000)
21. Mattsson, J.P., Selander, G., Raza, S., Höglund, J., Furuhed, M.: CBOR Encoded X.509 Certificates (C509 Certificates). Internet-Draft draft-ietf-cose-cbor-encoded-cert-03, IETF Secretariat (2022)
22. Schaad, J.: CBOR Object Signing and Encryption (COSE): Header parameters for carrying and referencing X.509 certificates. Internet-Draft draft-ietf-cose-x509-08, IETF Secretariat (2020)
23. Tschofenig, H., Fossati, T.: Transport layer security (tls) / datagram transport layer security (dtls) profiles for the internet of things. RFC 7925, RFC Editor (2016)
24. Ha, D.A., Nguyen, K.T., Zao, J.K.: In: Proceedings of the 7th Symposium on Information and Communication Technology (Association for Computing Machinery, New York, NY, USA, 2016), SoICT '16, pp. 173–179. https://doi.org/10.1145/3011077.3011108
25. Pritikin, M., Richardson, M., Eckert, T., Behringer, M., Watsen, K.: Bootstrapping remote secure key infrastructure (brski). RFC 8995, RFC Editor (2021)
26. van der Stok, P., Kampanakis, P., Richardson, M., Raza, S.: EST-coaps: Enrollment over Secure Transport with the Secure Constrained Application Protocol. RFC 9148, RFC Editor (2022)
27. Moran, B., Tschofenig, H., Brown, D., Meriac, M.: A Firmware Update Architecture for Internet of Things. RFC 9019, RFC Editor (2021)
28. Ankergård, S.F.J.J., Dushku, E., Dragoni, N.: State-of-the-art software-based remote attestation: opportunities and open issues for Internet of Things. Sensors **21**(5) (2021). https://doi.org/10.3390/s21051598
29. Birkholz, H., Thaler, D., Richardson, M., Smith, N., Pan, W.: Remote attestation procedures architecture. Internet-Draft draft-ietf-rats-architecture-15, IETF Secretariat (2022)
30. Höglund, J., Furuhed, M., Raza, S.: Lightweight certificate revocation for low-power IoT with end-to-end security. J. Inf. Secur. Appl. **73** (2023). https://doi.org/10.1016/j.jisa.2023.103424
31. Höglund, J., Raza, S.: In: 2022 IEEE International Conference on Cyber Security and Resilience (CSR) (2022), pp. 253–260. https://doi.org/10.1109/CSR54599.2022.9850290
32. Dushku, E., Rabbani, M.M., Conti, M., Mancini, L.V., Ranise, S.: SARA: Secure asynchronous remote attestation for IoT systems. IEEE Trans. Inf. Forensics Secur. **15** (2020). https://doi.org/10.1109/TIFS.2020.2983282
33. ComodoSSLstore. Comodo positive ssl certificate. https://web.archive.org/web/20220420135513/https://comodosslstore.com/positivessl.aspx (2022)
34. Krawczyk, H.: In: Advances. In: Boneh, D. (ed.) Cryptology—CRYPTO 2003, pp. 400–425. Springer, Berlin (2003)