Securing Communications in the Internet of Things using ID-based Cryptography and Modern Elliptic Curves

Master Thesis

Tobias Markmann

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Tobias Markmann

Securing Communications in the Internet of Things using ID-based Cryptography and Modern Elliptic Curves

Master Thesis submitted in the context of Masterprüfung

in the course Master of Science Informatik
at the Department of Computer Science
at the Faculty of Engineering and Computer Science
of Hamburg Univserity of Applied Sciences

Betreuender Prüfer: Prof. Dr. Thomas C. Schmidt
Zweitgutachter: Prof. Dr. Heike Neumann

Submitted on: August 26, 2015

Tobias Markmann

Thema der Arbeit / Title of the thesis
Securing Communications in the Internet of Things using ID-based Cryptography and Modern Elliptic Curves

Stichworte
Internet of Things, Identitätsbasierte Kryptographie, Elliptische-Kurven-Kryptografie, Ende-zu-Ende-Kryptographie, Föderierte Authentifizierung

Kurzzusammenfassung
Das Internet of Things (IoT) ist eine sich schnell entwickelnde Technologie, die Endanwendern schon zur Verfügung steht. Die Sicherheit in diesem Bereich hat sich jedoch langsamer entwickelt. Aktuelle Ansätze bleiben entweder proproietär oder folgen schwergewichtigen Internetstandards, einschließlich aufwändiger Public-Key-Infrastruktur. In dieser Arbeit stellen wir einen Ansatz für eine Sicherheitsarchitektur, aufbauend auf identitätsbasierter Kryptographie und eingebetteter kryptographischer Informationen in IPv6-Adressen, erlaubt Ende-zu-Ende-Authentifizierung in federierten IoT-Netzwerken vor. Unser Prototyp basiert auf elliptischen Twisted Edwards Kurven und erlaub den Einsatz auf schwachen Endgeräten. Die Tests zeigen, dass die Architektur sich für die Größe und der Vielfalt des IoT eignet.

Tobias Markmann

Keywords
Internet of Things, identity-based cryptography, elliptic-curve cryptography, end-to-end security, federated authentication

Abstract
The Internet of Things (IoT) is a fast evolving technology with many devices already available to end-users. However, the security of the IoT is less evolved. Current approaches either employ proprietary protocols, or use standard Internet protocols including the existing heavyweight and complex public-key infrastructure. In this thesis, we propose a security architecture built on identity-based cryptography that embeds of cryptographic information in IPv6 addresses, allowing end-to-end authentication of federated IoT networks. Our prototype using twisted Edwards curves allows deployment on constrained IoT devices. Our evaluation shows performance characteristics suitable for the scale and diversity of the IoT.

Contents

List of Tables

List of Figures

Abbreviations

AEAD Authenticated Encryption with Associated Data

CA certificate authority

CBID Crypto-based Identifier

CBOR Concise Binary Object Representation

CGA Cryptographically Generated Addresses

CGN Carrier-Grade NAT

CL-PKC certificateless public-key cryptography

CoAP Constrained Application Protocol

CRL certificate revocation list

DHCP Dynamic Host Configuration Protocol

DLP discrete logarithm problem

DoS Denial of Service

DSA Digital Signature Algorithm

DTLS Datagram Transport Layer Security

ECC elliptic curve cryptography

ECDH Elliptic Curve Diffie-Hellman

ECDLP elliptic curve discrete logarithm problem

ECDSA Elliptic Curve DSA

HPKP Public Key Pinning Extension for HTTP

HSM hardware security module

HTTP Hypertext Transfer Protocol

IBAKE Identity-Based Authenticated Key Exchange

IBC identity-based cryptography

IBE identity-based encryption

IBS identity-based signature

IDS intrusion detection system

IoT Internet of Things

ISP Internet Service Provider

KGC key-generation center

KGS key-generating server

KRL key revocation list

MANET mobile ad-hoc network

MitM man-in-the-middle

MTU maximum transmission unit

NAT network address translation

OCSP Online Certificate Status Protocol

OS operating system

PBC pairing-based cryptography

PKC public-key cryptography

PKI public-key infrastructure

REST Representational State Transfer

RTT round-trip time

SLAAC stateless address autoconfiguration

SPA simple power analysis

TA trusted authority

TLS transport layer security

TOFU trust on first use

TPM trusted platform module

WSN wireless sensor network

WWW World Wide Web

1 Introduction

The idea of the Internet of Things (IoT) combines the flexibility and ubiquity of small dedicated devices with the global addressability of the IPv6 Internet. Devices in the IoT are highly diverse in their properties, especially processing power, memory and communication abilities [1]. The same constrains are known from wireless sensor networks (WSNs), which however are not globally interconnected. The application area of the IoT ranges from domestic smart home and smart metering applications to national smart grids controlling the energy flow [1].

Security and privacy are critical requirements for many IoT application scenarios [2]. As IoT devices interact with the real world, these requirements extend to the communication between devices, commonly occurring over uncontrollable and open media. This asks for a protected network-layer, for which end-to-end authentication is an essential foundation.

Current solutions for securing communication between IoT devices include standards-based proposals which build on optimized Datagram Transport Layer Security (DTLS) implementations [3] providing end-to-end security. While the interoperability of standard technology is essential for fast adoption, the recycling of certificate-based authentication [4] brings along key management and revocation issues, known from the World Wide Web (WWW) and its public-key infrastructure (PKI).

Instead, we aim for a security system that still leverages standard network and security protocols, but provides easier key management, distributed authorities controlling their devices and scales to both the high number of endpoints on the Internet and the diversity of the IoT devices.

This work presents a security architecture for the IoT, enabling end-to-end authentication for the network layer based on federated identity-based cryptography (IBC) authorities. IBC [5] is a public key cryptography mechanism allowing use of arbitrary identifies as public keys, in our case IPv6 addresses, thereby eliminating the need to distribute them. We validate our proposed architecture using a prototype implementation for a constrained device, followed by evaluation in an interconnected local testbed with a border gateway and another endpoint.

Organization: The rest of this thesis is organized as follows. In chapter 2 we first introduce background on elliptic curve cryptography (ECC), specifically twisted Edwards curves, their security, storage complexity and computational performance.

This leads to an overview of IBC and identity-based signature (IBS) in chapter 3. After an introduction and definition of IBS, we define the specific IBS used for our architecture and discuss major key management problems related to IBC. This includes the issue of *key escrow* and *key revocation*, which need to be dealt with differently in IBC compared to

traditional asymmetric cryptography.

Chapter 4 deals with the IoT and existing proposals to provide end-to-end security for the communication between IoT devices.

The main part of this work is chapter 5. After posing the problem of end-to-end security in the IoT and declaring our objectives, we list the requirements for the proposed federated end-to-end security architecture for the IoT. Following this, the system components of our architecture are introduced in section 5.4.1 and the proposed architecture based on IBC and Crypto-based Identifier (CBID)-like IPv6 subnet addressing is described. The remainder of this chapter describes each step of the end-to-end authentication procedure in detail, including the revocation and the rollover of the trusted authority (TA).

In chapter 6 we discuss the compatibility with existing standard IPv6 and security technologies and describe deployment scenarios which benefit from our proposed end-to-end security architecture.

We describe our implementation, its software components and dependencies in chapter 7. The three interacting programs to be deployed on three different devices for evaluation are described at a high level.

Chapter 8 covers the security evaluation of the proposed federated end-to-end security architecture and evaluates the performance of a prototype implementation for two different deployment scenarios.

In the final chapter we draw our conclusion over the overall architecture and its previous evaluation for security and practical performance. Final ideas on the continuation of the proposed concept and its implementation are presented as part of the outlook.

2 Elliptic Curve Cryptography

In the 1980s, Miller [6] and Koblitz [7], both independently suggested to use elliptic curves defined over finite fields for cryptography, also known as elliptic curve cryptography (ECC). The most popular uses today are Elliptic Curve DSA (ECDSA) and Elliptic Curve Diffie-Hellman (ECDH). The primary advantage of ECC compared to classic arithmetic on prime groups like they are used by RSA [8] or Digital Signature Algorithm (DSA) [9] is the hardness of the mathematical problem behind it.

The hardness directly influences the security scalability properties of the cryptographic primitives like RSA or ECC. That is ECC key sizes and operation run times are lower compared to RSA at the same symmetric security level. In addition the key sizes and run times scale linearly for ECC with increasing security level while for RSA they scale super-linearly.

2.1 Twisted Edwards Curves

Originally the proposed form for elliptic curves in elliptic curve cryptography (ECC) were Weierstrass curves, but over the time further forms of elliptic curves have been suggested. The curve forms of particular interest are twisted Edwards curves [10]. They allow simple and fast implementations which avoid the majority of security problems known from short Weierstrass curves.

Elliptic curves can be defined over prime fields (i.e., $\mathbb{F}_p$ with prime p) or binary extension fields (i.e., $\mathbb{F}_{2^m}$). This work focuses on ECC using prime fields, because they have a stable security history compared to binary fields and extension fields. The choice of field—prime field ($\mathbb{F}_p$) or binary extension field ($\mathbb{F}_{2^m}$)—can have a crucial influence on the performance depending on hardware architecture [11]. Bilinear pairings transfer supersingular curves over $\mathbb{F}_q$ to $\mathbb{F}^*_{q^k}$, where the discrete logarithm problem (DLP) is easier to solve compared to the hard elliptic curve discrete logarithm problem (ECDLP). $\mathbb{F}_q$ is of prime ($q = p$) or prime power ($q = p^k$) order, where p is prime and k is a positive integer. This fact is used by the MOV attack [12]. Some elliptic curves over $\mathbb{F}_{q^n}$ transfer to hyperelliptic curves, where the ECDLP is easier to solve [13]. Elliptic curves over $\mathbb{F}_p$ or $\mathbb{F}_{2^p}$ where p is a prime are not vulnerable to this attack.

In 2007 Edwards [14] proposed a new form of elliptic curves over number fields that are defined by the equation $x^2 + y^2 = c^2(1 + x^2y^2)$.

All elliptic curves over non-binary finite fields are transformable into the Edwards form of elliptic curves. However this transformation sometimes requires the Edwards form to

be defined over a field extension of the original field [15].

Building on Edwards proposal, Bernstein *et al.* [15] defined an expanded formula for Edwards curves to include more curves for possible transformation to Edwards curves without change of the underlying finite field. Their formula for Edwards curves is $x^2+y^2 = c^2(1+dx^2y^2)$ where $cd\left(1-dc^4\right) \neq 0$ holds.

They propose a formula for Edwards curves to include all curves $x^2+y^2 = 1+dx^2y^2$ and proof that all elliptic curves with a point of order 4[1] are transformable to this Edwards curve formula. The addition law for Edwards curves is *unified*, meaning that it can be used for both addition and doubling. It is also *complete*, meaning it is valid for all possible inputs, including the identity element. Neither of these properties apply to classic Weierstrass curves and implementations need to handle special cases. Bernstein *et al.* also presented fast addition and doubling formulas and showed their advantage for the performance of ECC.

Having a unified and complete addition law not only enables compact implementations but also reduces the attack surface on side-channels. ECC implementations using classic Weierstrass curves commonly have a highly branched addition law handling various special cases. Since not all branches are of equal computational complexity, the implementation is subject to simple power analysis (SPA), timing attacks and other side-channel attacks.

In 2008, Bernstein *et al.* [10] proposed a generalization of the original Edwards curve proposed by Edwards [14] in 2007, the twisted Edwards curve. A twisted Edwards curve group, i.e. the group of points on a twisted Edwards curve, is defined as

$$E_{TE,a,d}(\mathbb{F}_p) : \left\{(x,y) \in {\mathbb{F}_p}^2 : ax^2+y^2 = 1+dx^2y^2\right\} \tag{2.1}$$

for curve parameters $a, d \in \mathbb{F}_p$ [10, p. 3].

If the group law on an elliptic curve is described by a single formula for point addition and doubling, it is known to be *unified*. If a formula for group arithmetic is valid for all elements of an ECC group, including the identity element, it is said to be *complete*.

Twisted Edwards curves have a *complete* and *unified* group arithmetic or group law that can be described by the following formula using affine coordinates of two points (x_1, y_1) and (x_2, y_2) on the twisted Edwards curve $E_{TE,a,d}(\mathbb{F}_p)$:

$$(x_1,y_1)+(x_2,y_2) = \left(\frac{x_1y_2+y_1x_2}{1+dx_1x_2y_1y_2}, \frac{y_1y_2-ax_1x_2}{1-dx_1x_2y_1y_2}\right) \text{ [10, p. 11]} \tag{2.2}$$

In contrast, the group law for short Weierstrass curves is not unified and complete. Its implementation comes with additional code to handle special cases like point doubling or cases where one operant is the identity element. This increases the surface of the ECC implementation for side-channel attacks like SPA and timing attacks [17].

Finally, Twisted Edwards curves are birationally equivalent to Montgomery curves [10,

[1]The order of point P is the smallest positive integer x with $x \cdot P = \mathcal{O}$, with $\mathcal{O}$ being the identity element of the elliptic curve group [16, p. 20].

p. 3], which hold the speed record for fastest Elliptic Curve Diffie-Hellman (ECDH) [18] key exchange.

2.2 Curve Point Representations

Points on an elliptic curve can be represented in different forms. Each elliptic curve point representation has its own advantage, e.g. small storage complexity ideal for long term storage or communication, or a computationally efficient group law suitable for efficient computations required for elliptic curve cryptography (ECC) algorithms and schemes.

The twisted Edwards curve formula given in equation (2.1) uses affine coordinates, i.e. x- and y-coordinates, for its point representation. However, the addition formula associated with this point format — seen in equation (2.2) — requires division or inversion in $\mathbb{F}_p$. Inversion of finite prime field elements is an expensive group operation on most architectures. Its computation commonly uses the extended Euclidean algorithm which has a high computational complexity, namely $\mathcal{O}(n^3)$ for n-bit numbers [19, p. 937].

In order to avoid expensive computations of inversions and the extended Euclidean algorithm, projective coordinates [20] have been suggested. Using projective coordinates, i.e. x-, y- and z-coordinates, expensive inversions can be postponed until an elliptic curve point in affine or compressed form is needed for storage or communication. The trick of the group law for projective coordinates, i.e. the formulas for addition and doubling of elliptic curve points, is to carry along the common denominator of the calculations on the y- and x-coordinates for all group operations in the z-coordinate of a point. With that, the projective point (x, y, z) is equal to the affine point $(\frac{x}{z}, \frac{y}{z})$. The conversion from a point in projective coordinates to a point in affine coordinate requires inversions. For a survey on methods with projective coordinates, see [21].

Building on projective coordinates, Hisil *et al.* [22] suggested to use a forth coordinate (t) for their extended twisted Edwards curve point format. The idea behind this point format is, to keep an intermediate result of the addition and doubling operations around in a fourth t coordinate, to save the computation of this intermediate value in the next operation. This results in a doubling formula for their group law which has a lower computational complexity for consecutive doublings than the doubling formula for standard projective coordinates. However, it requires specialized scalar multiplication algorithms which optimize for consecutive doublings to take advantage of this feature.

The most storage efficient point representation is the compressed point format. Twisted Edwards curves are symmetric with respect to the y-axis. This allows to represent a point on the curve with nearly half the storage complexity of the affine coordinate representation, ideal for reducing the overhead for long term storage on memory constrained devices or in communication protocols. Compressed coordinates contain a canonical, bit-by-bit, description of the y-coordinate and a sign bit of the x-coordinate. The x-coordinate can be recovered or decompressed using the curve formula for twisted Edwards curves and the sign bit. The curve formula from equation (2.1) solved by x has two results. The sign bit is used to distinguish between the two.

ECC point format	Storage Complexity	Remarks
Compressed	1 sign bit + $1 \times \mathbb{F}_p$	Ideal for storage or communication
Affine	$2 \times \mathbb{F}_p$	Slow due to inversions in group law
Projective	$3 \times \mathbb{F}_p$	Fast group law, ideal for ECC computations
Extended	$4 \times \mathbb{F}_p$	Requires special scalar multiplication for faster group law

Table 2.1: Comparison of the memory/storage complexity of elliptic curve point formats with remarks.

A comparison of the memory or storage complexity of all four elliptic curve point representations can be seen in table 2.1. The ideal point representation for storage or communication to other parties is the compressed point format. It has the lowest storage complexity among all four point formats and allows to encode an point on the curve used in this work, Ed25519, in only 256 bit. Projective coordinates are the recommended representation for computations, as it avoid expensive field inversions. While the extended coordinates allow even faster computations, the added code complexity for a dedicated scalar multiplication algorithm and higher memory requirements are undesirable for memory constrained Internet of Things (IoT) devices.

2.3 Scalar Multiplication

Scalar multiplication is the main operation that builds on the elliptic curve cryptography (ECC) group law and is used by all protocols that build upon ECC, e.g., Elliptic Curve Diffie-Hellman (ECDH) and Elliptic Curve DSA (ECDSA).

Scalar multiplication of the elliptic curve point P on the elliptic curve E by the natural number n, is defined in [16, p. 271] as follows:

Definition 2.3.1. Take $n \in \mathbb{N} \setminus \{0\}$ and let us denote *scalar multiplication* by n on E by $[n]$, or $[n]_E$ to avoid confusion. Namely,

$$[n] : E \longrightarrow E \tag{2.3}$$

$$P \longmapsto [n]P = \underbrace{P + P + \cdots + P}_{n \text{ times}}. \tag{2.4}$$

There are various algorithms to compute scalar multiplication which vary in their computational complexity, ability to take advantage of precomputations and side-channel resistance. The twisted Edwards curve implementation for RELIC [23], introduced in our previous work [24], uses the `LWNAF` algorithm with non-unified addition and doubling

formulas for projected coordinates on a twisted Edwards curve. For extended projective coordinates on twisted Edwards curves, we also added the `LWNAF_MIXED`.

While both algorithms are among the fastest scalar multiplications available in RELIC, neither is secure against timing side-channel attacks. In 2014, Benger *et al.* showed that by observing the time behavior wNAF — the same algorithm as implemented by RELIC — on a secret factor, i.e. a private key, the same factor can be recovered by a FLUSH+RELOAD [25, 26] attack. By timing access to the memory, during the wNAF computation over many runs, a secret factor can be recovered bit by bit. The access time to the memory can fluctuate on CPU architectures that have a data cache.

As the attack depends on the availability of a data cache on the architecture, not all platforms vulnerable to it. For example on of our three evaluation platforms (see table 8.1), namely the SAM R21 board, does not have any data cache for the RAM.

2.4 Security and Hardness of ECC

There exist algorithms for solving the RSA-problem or the discrete logarithm problem (DLP) in finite fields that have subexponential-time complexity [27, Chapter 2].

Definition 2.4.1. The *discrete logarithm problem* is defined as finding element x for a given a, g and p, p being a large prime, in the formula:

$$a \equiv g^x \mod p \tag{2.5}$$

The corresponding problem in elliptic curve groups is called elliptic curve discrete logarithm problem (ECDLP).

Definition 2.4.2. The *elliptic curve discrete logarithm problem* is defined as finding $n \in \mathbb{Z}$ given $P, Q \in E(\mathbb{F}_q)$ in the following function. E is describing the elliptic curve function and $\mathbb{F}_q$ the finite field used for the coordinates.

$$Q = nP \tag{2.6}$$

The index-calculus algorithm for solving DLP in $\mathbb{F}_q$ has a time complexity given in L-notation [27, p. 60] of $L_q\left[\frac{1}{2}, c\right]$ with constant $c > 0$ [27, p. 112]. The L-notation is defined as

$$L_q[\alpha, c] = \mathcal{O}\left(e^{(c+o(1))(\ln q)^{\alpha}(\ln \ln q)^{1-\alpha}}\right)$$

with the positive constant c and $0 < \alpha < 1$ [27, p. 60, Eqn. 2.3].

However, until now the best known algorithm to solve the ECDLP on elliptic curves over prime fields has exponential-time complexity. For general elliptic curves the best known algorithm to solve ECDLP is Pollard's rho algorithm for logarithms [28]. This algorithm has polynominal-time complexity of $O(\sqrt{p})$ in the size of the elliptic curve group. However, the input of complexity descriptions of algorithms is commonly defined in the number of bits. With p being a number of n bits, testing all values of p would take a time of $O(2^n)$. Thus the runtime complexity of Pollard's rho algorithm for general elliptic curves with

$p = O(2^n)$ becomes $O(\sqrt{2^n}) = O(2^{n^{\frac{1}{2}}}) = O(2^{\frac{n}{2}})$. This describes an exponential-time complexity algorithm.

Symmetric	Asymmetric (RSA / DLOG)	Asymmetric (Elliptic Curve)
64 bit	816 bit	128 bit
80 bit	1,248 bit	160 bit
112 bit	2,432 bit	224 bit
128 bit	3,248 bit	256 bit
160 bit	5,312 bit	320 bit

Table 2.2: ECRYPT II comparison of key sizes (in bits) at the same security level between symmetric, asymmetric (RSA) and elliptic curve [29].

Due to the hardness of the ECDLP, one can use smaller groups in elliptic curve cryptography (ECC)-based schemes with the same equivalent symmetric security level as compared to schemes based on RSA or DLP. This leads to smaller key sizes and smaller signatures, which is especially beneficial in low-power computing environments. A comparison of key sizes of equal security in symmetric, traditional asymmetric and the elliptic curve setting can be found in Table 2.2 and takes latest hardware advances and the best algorithms for the cryptographic problems into account.

3 Identity-based Cryptography

In 1985, Shamir [5] proposed a new asymmetric cryptographic system, identity-based cryptography (IBC) that compared to classic public-key cryptography allows arbitrary bit-strings to be used as public key. This flexibility comes with the *key escrow* constraint, meaning the key issuing party, the trusted authority (TA), knows the private key of all members of the system. The free choice of public key enable to use already existing identifiers (IDs) of communication protocols as public key, and thereby saving storage and transmission cost for additional public keys.

Like in traditional asymmetric cryptography, there are IBC-variants of the major known asymmetric schemes, e.g. identity-based signature (IBS), identity-based encryption (IBE) and ID-based key exchange protocols. The first IBS scheme was already proposed by Shamir [5] based on the RSA cryptosystem in his original proposal. However, it was not until 2001 that Boneh *et al.* [30] proposed the first specific IBE scheme based on pairing-based cryptography (PBC) introduced just shortly before in 2000 by Joux [31].

3.1 Identity-based Signatures

There are identity-based signature (IBS) designed using the RSA cryptosystem, elliptic curve cryptography (ECC) and pairing-based cryptography (PBC). On a higher level they are all provide the function interface and provide similar functionality, but with different performance, memory and storage characteristics.

Definition 3.1.1. An IBS is defined using four functions:

$$\mathit{Setup}\,(\mathit{sec\_level}) \longrightarrow (TA_{SK}, TA_{SP}) \tag{3.1}$$

$$\mathit{KeyExtract}\,(TA_{SK}, TA_{SP}, ID) \longrightarrow (ID_{key}) \tag{3.2}$$

$$\mathit{Sign}\,(TA_{SP}, ID_{key}, m) \longrightarrow (\sigma) \tag{3.3}$$

$$\mathit{Verify}\,(TA_{SP}, ID, m, \sigma) \longrightarrow \mathit{true}/\mathit{false} \tag{3.4}$$

Initially, a setup phase is run where the secret key and system parameters, including the public key (TA_{SK}, TA_{SP}) of the trusted authority (TA) is generated for a desired security level (*sec_level*).

After the setup, private keys (ID_{key}) can be extracted from bit strings essentially binding the private key to a specific identity (ID). The signatures generated using this private key are only valid signed for the specific identity used during the key extraction.

During the normal operation, the signing and verification functions are very similar to those in classic public-key infrastructure (PKI). The only difference is that instead of the public key, an arbitrary identity bit-string is used for verification.

3.2 IBS for the Internet of Things

In our previous work [32] we analyzed different identity-based signatures (IBSs), across several cryptographic primitives. The evaluation covered the following three IBSs:

SH-IBS The original IBS proposed by Shamir [5] is based on the RSA cryptosystem. The signature size is relative large compared to pure elliptic curve cryptography (ECC) based solution at a symmetric security level of 128 bit.

vBNN-IBS Cao *et al.* [33] described an IBS based on ECC. Compared to SH-IBS, it is faster and has smaller signatures at the 128 bit symmetric security level.

TSO-IBS We also evaluated an IBS based on the more recent area of pairing-based cryptography (PBC) [30, 31]. However, the IBS proposed by Tso *et al.* [34] has both a larger signature size and higher computational complexity compared to the purely ECC-based IBS.

Our analysis concluded that an IBS based on basic ECC would be best suited as security primitive for constrained devices. RSA-based cryptographic schemes show a bad scalability in storage and computational complexity as the security parameters increases.

3.2.1 vBNN-IBS (ECC-based IBS)

vBNN-IBS is a ID-based signature scheme described by Cao *et al.* as part of IMBAS [33]. It is based on BNN-IBS [35], the first provable secure identity-based signature (IBS) based on elliptic curve cryptography (ECC).

Setup

To initialize the system for security parameter k, take the following steps:

1. Chose an elliptic curve with the parameters $E/\mathbb{F}_q$, P and p satisfying the security parameter k.
2. Generate random master secret key $x \in \mathbb{Z}_p$ and set the master public key, $P_0 = xP$.
3. Define two cryptographic hash functions, H_1 and H_2:

$$H_1 : \{0,1\}^* \times \mathbb{G} \to \mathbb{Z}_p \tag{3.5}$$
$$H_2 : \{0,1\}^* \times \{0,1\}^* \times \mathbb{G} \times \mathbb{G} \to \mathbb{Z}_p \tag{3.6}$$

4. Publish public system parameters:

$$\langle E/\mathbb{F}_q, P, p, P_0, H_1, H_2 \rangle \tag{3.7}$$

Key Extraction

To generate the private key, s_{ID}, for a user with the identity *ID*, carry on with the following steps:

1. Calculate a random $r \in \mathbb{Z}_p$ and compute $R = rP$.
2. Using the master secret key x, calculate:

$$c = H_1(ID, R) \tag{3.8}$$
$$s = r + cx \tag{3.9}$$

3. The private key for the user with the identity *ID* is $s_{ID} = (R, s)$.

Signature Generation

For generating the signature σ for message $m \in \{0,1\}^*$ do:

1. Generate random $y \in \mathbb{Z}_p$ and compute $Y = yP$.
2. Compute the following:

$$h = H_2(ID, m, R, Y) \tag{3.10}$$
$$z = y + hs \tag{3.11}$$

The final signature for message m is $\sigma = (R, h, z)$.

Signature Verification

To verify if message m from a user with the identity *ID* is correctly authenticated by signature $\sigma = (R, h, z)$, proceed with:

1. Compute the following:

$$c = H_1(ID, R) \tag{3.12}$$
$$T = zP - h(R + cP_0) \tag{3.13}$$

2. To verify the signature, check whether the following equation holds:

$$h \stackrel{?}{=} H_2(ID, m, R, T) \tag{3.14}$$

Complexity Overview

Signature generation comes with the cost of one scalar multiplication in $E(\mathbb{F}_q)$ from step 1 and signature verification costs 3 scalar multiplications in $E(\mathbb{F}_q)$ from equation 26. The signature size, with $\sigma = (R, h, z)$, is one element of $E(\mathbb{F}_q)$ and two elements of $\mathbb{Z}_q$.

3.3 Key Management in ID-based Cryptosystems

There are two major issues with identity-based cryptography (IBC) which result from the inherent binding between identity and public key, and the fact that the trusted authority (TA) issues all private keys. The following sections will discuss the topics of *key revocation* and *key escrow* in IBC systems in more depth.

3.3.1 Key Escrow

In contrast to classical public-key cryptography (PKC), the private keys need to be generated by the commonly trusted third party, called trusted authority (TA), key-generation center (KGC) or key-generating server (KGS) in identity-based cryptography (IBC). This outsourcing of private key generation to a trusted party is called *key escrow* and is a critical property of simple IBC systems.

A third party knowing the private keys of other users is able to decrypt all messages in an identity-based encryption (IBE) scheme and is able to forge signatures for any message and any user in an identity-based signature (IBS) scheme. Thus IBS cannot offer real non-repudiation. In addition, this third party is an attractive target for attacks because access to it discloses all private information of the crypto system.

Key escrow has been active topic in the research community in the mid-1990s. It was a suggested method for integration in all cryptographic devices mandated by the US government, as it would provide both secure communication for users and easy access to the clear text by the government for lawful interception [36]. A survey about a broad range of issues around *key escrow* is provided in [37].

There are proposals to mitigate the *key escrow* problem in IBC systems. Boneh *et al.* [30] suggest distributing the KGC over multiple servers where each server only holds part of the master secret key. To gain access to the full master secret key multiple servers need to collude. Distributed KGCs are widely discussed and proven secure in [38]. Another proposal by Al-Riyami *et al.* is certificateless public-key cryptography (CL-PKC) [39]. In CL-PKC the final private key is generated by the user based on secret information from the KGC and secret information of the user. This way the KGC cannot forge signatures and the system is free of *key escrow*. However, the system is not ID-based anymore, because public keys are no longer derivable from IDs.

3.3.2 Key Revocation

Traditional public-key cryptography (PKC) has the key revocation phenomenon. If a private key is compromised in any way, the owner can revoke the corresponding public key at the certificate authority (CA). There are two categories in which a private key can be compromised:

1. direct local access to the storage or memory where the private key is held
2. remotely via errors in the protocol [40] or side-channels in the implementation of algorithms involving the private key [41]

Thereby the issuer can revoke the previous binding between his identity and his public key. The CA provides public access to all key revocations in its realm via two ways: certificate revocation lists (CRLs), an offline and asynchronous method that provides verifying parties with a list of revoked public keys, signed by the CA, and Online Certificate Status Protocol (OCSP), an online and synchronous method where the verifying party can ask a service at the CA, whether a certificate has been revoked or continues to be valid.

A more extensive overview about revocation in the public-key infrastructure (PKI) can be found in our previous work [32].

Revoking a public key in identity-based cryptography (IBC) also revokes the identity, because they are basically equivalent in IBC systems. Boneh *et al.* [30] proposed in 2001 to use automatic key renewal to avoid revoking the identities themselves and the maintenance cost of revocation lists. They suggest to extend the identity with temporal information, e.g. use `identity + week` as identity passed to the IBC functions.

The process demands each user to renew her private key from the trusted authority (TA) once per time period (e.g. year) and the TA will stop issuing new private keys for users which are known to be compromised. The reliability of signature verification using this identity construct depends on rough synchronization of clocks among the communicating devices. Furthermore it requires an active and online TA in the network and a secure channel between it and the devices for them to renew their keys.

Hoeper *et al.* [42] proposed an *ID* format that allows explicit revocation of keys before their expiration time. They add a version number to the identity so that the final *ID* is defined as $ID = H(identity \| time \| version)$. Their revocation scheme is designed for mobile ad-hoc networks (MANETs) and builds heavily on pairing-based IBC and non-interactive key establishment. The method proposed by Hoeper *et al.* handles explicit revocation inside the network without the help of a third party and can be summarized as follows:

1. *Neighborhood watch*: Each node monitors its one-hop neighbor nodes for suspicious behavior and handles explicit revocation requests (i.e. *harakiri* messages), distributed by other node. Examples for suspicious behavior are extreme traffic patterns or sending invalid messages. A node accumulates the accusations in an accusation matrix (AM), containing a row for each neighbor node and columns for ID, current version and accusation value.

2. *Accusation and its propagation*: Every time the AM is updates by a node, it securely propagates the new accusation value to its one-hop neighbors using accusation messages. Other nodes receiving an accusation message update their key revocation list (KRL) to reflect that the sending neighbor node accuses another node. Each node computes its KRL based on accumulated accusations and harakiri messages received from its m-hop neighbors.

3. *Revocation*: If the sum of all accusations for a public key Q_i with expiration t_i and version v_i for neighbor node with ID_i exceeds a threshold, the public key with exactly these expiration and version numbers is considered as revoked.

	Boneh *et al.* [30]	Hoeper *et al.* [42]
ID format	*identity*‖*time*	*identity*‖*time*‖*version*
revocation method	expiration	expiration, explicit revocation
regular key renewal	no	yes
pre-expiration key revocation	no	yes
communication-free revocation	yes	no
management cost for revocation	low	high

Table 3.1: Overview of key revocation concepts proposed for IBC.

4. *Key renewal*: Nodes with revoked or expired keys can request a new private key from the off-line TA, after correct authentication with an increased version number. The version number for an identity ID_i limited to a maximum value within a specific time period, to protect the network from malicious nodes that request new private keys over and over again.

Table 3.1 shows an overview of the two approaches and compares their properties.

4 The Internet of Things

4.1 Overview of the Internet of Things

Compared to the classic Internet the Internet of Things (IoT) has a higher diversity of participating devices. In particular some IoT devices are energy critical, like smart watches and wireless thermostats. There are even smaller sensor devices which are highly constrained in their computation power and memory capacity. In [43, sec. 3] the authors provide three device classes for constrained devices of which the least constrained class describing systems with 50 KiB data storage and 250 KiB code storage capacities.

With some devices being completely autonomous from the environment and communicating wireless with other devices or a router.

The special constraints of the IoT demand a different network protocol stack that is optimized for the memory, power, energy and computational properties. Table 4.1 show the classic Internet network protocol stack compared to the protocol of the IoT.

	Classic Web		Internet of Things
Application	HTTP		CoAP
Security	TLS		DTLS
Transport	TCP	$\Longrightarrow$	UDP
Network	IP		IP, RPL
Data link	Ethernet, WLAN		IEEE 802.15.4 [44]

Table 4.1: Classic protocol stack compared to the Internet of Things protocol stack.

The use of the Constrained Application Protocol (CoAP) [45] protocol instead of the Hypertext Transfer Protocol (HTTP) protocol and DTLS [46] instead of TLS are of particular interest to us. CoAP provides a Representational State Transfer (REST)-ful protocol like HTTP however with much smaller header overhead and lower paring complexity. In addition it defined DTLS to be used as optional security layer. DTLS is similar to TLS, but based on unreliable datagrams as transport method and unlike TLS it comes with support for multicasting.

4.1.1 IPv6 in the Internet of Things

Due to the huge number of devices in the Internet of Things (IoT)—multiple billions of connected computers[47]—and their communication over the Internet with other IoT

devices and backend services the network-layer protocol of choice is IPv6 [48].

A common wireless media protocol for low-power devices in the IoT is IEEE 802.15.4 [44]. With its maximum transmission unit (MTU) of 127 bytes it poses multiple challenges on IPv6 usage, like the IPv6 minimum MTU of 1280 bytes and the size of the IPv6, UDP and TCP headers.

Starting with the IEEE 802.15.4 MTU of 127 bytes and its minimum header of 25 bytes without layer 2 security, the network layer is left with 102 bytes. Further, the minimal IPv6 header alone already occupies 40 bytes, and the UDP header with 8 bytes this leaves just 54 for the application layer content.

6LoWPAN [49, 50] addresses, among others, the MTU and header size issues.

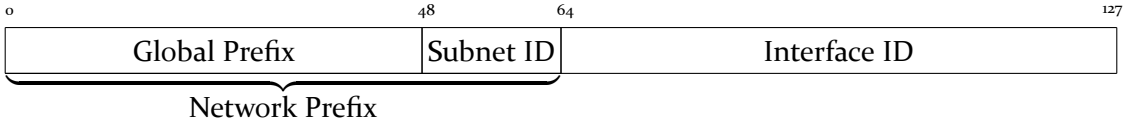


Figure 4.1: Standard IPv6 unicast address format.

Among other things, 6LoWPAN allows to compress the standard IPv6 header and the UDP [51] header. The compression works by eliding common IPv6 header options that are predefined by 6LoWPAN, local-link address that can be derived from the the 802.15.4 layer and network prefixes that are compressed to a 2 byte *context ID*. Depending on communication scenario, 6LoWPAN can reduce the standard IPv6 header of 40 bytes to 2 bytes in the best case and 20 bytes in the worst case [52, p. 7f]. As soon as 6LoWPAN packets leave their local network, e.g. at a border router, they are converted to standard IPv6 packets.

4.2 Security Protocols for the Internet of Things

In this section we will discuss relevant related work on the topic of secure communication protocols designed for the Internet of Things (IoT). This includes approaches based on standard Internet protocols, the application of identity-based cryptography (IBC) in the IoT, and enhancements for the specialties of the IoT.

4.2.1 Modifications of TLS Protocol Handshake using IBC for WSN

In [53], Mzid *et al.* propose two modified TLS handshake protocols to reduce the security management overhead of classic certificate-based solutions and the handshake latency of the traditional TLS handshake. They incorporate identity-based cryptography (IBC) for authentication to reduce the management overhead in wireless sensor network (WSN) applications and describe adapted handshake protocols based on elliptic curve cryptography (ECC), Elliptic Curve Diffie-Hellman (ECDH) and pairing-based cryptography (PBC).

Their two proposed TLS handshake adjustment work as follows:

1. This proposed TLS handshake uses still ECDH as a key exchange mechanism but skips the certificate transfers from the traditional TLS handshake. Instead each side

sends its IBC signed ECDH value along with the clear ECDH value which can then be used for authentication. The specific identity-based signature (IBS) algorithm used in their evaluation is not mentioned. Using these adjustments the TLS handshake is shortened to 10 messages, compared to the 13 messages of the traditional TLS handshake with full authentication.

2. The second proposal goes further by using an identity-based key establishment protocol instead of ECDH and Elliptic Curve DSA (ECDSA). They use an ID-based key establishment protocol with takes advantage of PBC [54]. A key establishment protocol allows to parties to establish a shared secret key without any interaction at all. This allows the TLS handshake to be further reduced to only 7 messages.

Both protocols can skip the transfer of certificates and public keys, as they use IBC with the IPv6 addresses of each end-point being their ID, i.e. their public key. From an architecture perspective their proposed system requires a single trusted 3rd party, the trusted authority (TA). This only allows application scenarios where client and server have a preset trusted third party, the TA. The TA securely distributes the private keys for all participating parties in the system in the pre-deployment phase.

The two protocols in comparison show a trade off between latency and energy efficiency. According to their evaluation, the first proposed handshake protocol provides lower latency as it does not have to execute expensive pairing operations. The pairing-based protocol however, is more energy efficient as it saves the communication for the ECDH key exchange by using ID-based key-establishment.

4.2.2 DTLS-based Security and Two-way Authentication

Kothmayr *et al.* [55, 56] proposed the first application layer end-to-end security architecture for the Internet of Things (IoT) completely based on established Internet standards. They aim for high interoperability between IoT devices and existing servers on the Internet, and propose to reuse present security infrastructure for authentication. As an application layer end-to-end solution, it can provide security in cases where there is no complete control over the underlying transmission media or in multi-hop communication scenarios with uncontrolled intermediaries.

They choose DTLS [57] for providing end-to-end security and rely on classic X.509 public-key infrastructure (PKI) certificates using RSA for authentication. DTLS can provide the same security properties known from standard TLS used to secure Internet traffic, i.e. authenticity, integrity and confidentiality. However, in comparison to TLS, DTLS can deal with message loss which is common in wireless network protocols found in the IoT.

Certificate-based authentication requires trusted certificate authority (CA) certificate to be configured in the device before deployment. Based on these trusted root CA certificates, other party's certificates can be verified for integrity and trust during the DTLS handshake.

They test and evaluate their implementation using an OPAL sensor node with a 48 MHz Atmel CPU with added hardware support for RSA operations and secure key storage in form of a trusted platform module (TPM). The software run on the Opal nodes is TinyOS

with a custom DTLS implementation. The evaluation covers the computation time, energy use and memory requirements of their DTLS handshake implementation.

4.2.3 Authenticated Node to Multi-user Communication in Sensor Networks

In [58], Oliveira *et al.* analyze and discuss the authentication of the communication between a shared sensing node and its multiple users.

They cover an application scenario, where deployed sensor nodes have sensing capabilities which are of interest to different, not directly related, users from independent authorities. By avoiding intermediary gateways, they allow direct communication of low-power sensor nodes to multiple users over the public Internet. Various signature mechanisms are compared within a setup of a Secure-TinyWebService / IP stack on two different sensor node hardware platforms.

Their design analysis includes consideration for both symmetric and asymmetric signatures. However, approaches using symmetric signatures are discarded due to the key distribution and synchronization problems associated with existing symmetric protocols, e.g. μTESLA [59], which are not well suited for a scenario of a changing set of users sharing some sensor nodes over the public Internet.

In their further analysis of asymmetric digital signatures, they decided against certificate-free schemes, including identity-based cryptography (IBC). They argument that the trusted authority (TA) in an IBC system always knows everybody's private key and can impersonate any user in the system. This concept is also known as key escrow. In single self-contained sensor networks it might be acceptable that there is one entity which escrows everybody's private key. However, in a public shared network with different and changing parties, key escrow is unacceptable.

Certificate-free schemes, i.e. IBC schemes but without having the TA escrowing users' private keys, have been proposed before [39], but are also disqualified due to their very high computational complexity by the authors.

Which leads to their conclusion that certificate-based schemes are further analyzed, where Oliveira *et al.* include not only classic Digital Signature Algorithm (DSA) [9] and its elliptic-curve counterpart Elliptic Curve DSA (ECDSA) [60] but also more recent short-signature schemes, BLS [61] and ZSS [62].

For their usage scenario, a changing set of external users accessing an optimized web service on sensor nodes, they conclude that Schnorr signatures [**s-esgsc-91**] or, if broader compatibility is needed, ECDSA is used. A Schnorr signature is digital signature scheme similar to ECDSA, however in comparison it is more efficient since it does not require expensive computations of the modular inverse in the finite field.

The measurements results Oliveira *et al.* [58, p. 392] provide, show that the additional energy required for the short-signatures using pairing-based cryptography (PBC), has little influence on the communication cost on their evaluation platforms. The additional cost for shorter signatures does not result in huge energy savings on the radio communication side. However, on platforms where radio usage is more energy heavy, e.g. underwater communications, operation lifetime improvements could be gained from using short.signatures

like BLS and ZSS.

Their recommendation to use elliptic curve cryptography (ECC) as a signature scheme for constrained platforms coincides with our choice of an ECC-based identity-based signature (IBS). Our use of IBC and its *key escrow* problem is not as critical as in their scenario, as there is not a single global TA in our architecture outlined in chapter 5.

5 Federated Authentication for Internet of Things

Objective of this work is to provide a solution to the problem of end-to-end authentication in the Internet of Things (IoT). Common with the classic Internet, the IoT is characterized as a network of computers communicating with each other in a multi-hop and multi-path fashion.

Special to the IoT is the actuators or sensing devices may also carry out network supporting tasks, acting as routers for traffic of other nearby devices. It is the exception that the whole communication path between two devices in the IoT is in complete control of a single party. The properties of *Things* are highly diverse. Examples go from constrained battery powered environment monitoring systems to smart home appliances with steady power supply.

5.1 Problem Statement

The Internet of Things (IoT) is part of the globally interconnected Internet. An authentication mechanism for the IoT needs to work on devices that are typically constrained in their computational and physical properties. They have CPUs with simpler instruction sets, lower clock frequency compared to mobile handsets and desktop systems, and are highly limited in runtime memory (RAM) as well as program memory (ROM). Furthermore, some *Things* are limited in their energy resources and communication capabilities.

A security solution for the IoT needs to follow a lightweight design with these characteristics in mind. This means it has to come with little runtime overhead in terms of additional processing time, communication, memory consumption and code size.

End-to-end secure communication on the traditional Internet is enabled by transport layer security (TLS) and external third party certificate authoritys (CAs). End-to-end security does not require control of all intermediary parties, but only the two end points require control. Many IoT scenarios interact with the real world in private and security sensitive manner, that requires protection against an invasion of the privacy of end users and theft of business critical data. Thus, a security solution for the IoT needs to include the same standard security properties like message integrity, authenticity and confidentiality. Modern security protocols are faced with a large attack surface, including global monitoring institutions, remote attackers trying to circumvent the security and local attackers.

IoT applications can collaborate with other IoT applications and classic Internet services to provide a location independent service. Finding a commonly trusted third party is not feasible, because the Internet is a distributed network of subnetworks controlled by

independent authorities. Finding an entity, equally trusted by all authorities in play, is unrealistic. This rules out the reliance on a single authority controlling the security of all devices.

A viable solution for the authentication in the IoT needs to scale to millions of devices and provide sufficiently secure authentication in the absence of commonly trusted external authorities. It needs to enable end-to-end secured communication between independently managed networks of devices that can establish a secure federated communication channel between each other.

5.2 Objectives

Our core objectives are:

Independent secure authentication We aim for a security solution that is independent of the support of intermediate routing devices on the path between devices of two different edge-level networks. The authority controlling an edge-network, controls the trust configuration of the devices within the network.

While terminating security at border gateways takes computational load from constrained Internet of Things (IoT) devices, it turns the gateway into a high-value target for attackers. Thus, the security of the designed architecture should terminate at the constrained IoT devices, with them sending and receiving authenticated messages. However, the border gateway can still provide non-critical support to the authentication process.

Localized trust management Unlike the traditional handling of trust in the public-key infrastructure (PKI) for the World Wide Web (WWW), we aim for a solution without external parties, such as certificate authoritys (CAs) that control the trust and are hard to control. Trust in the devices of a local IoT network should be managed by the authority of the local network.

Target constrained hardware IoT scenarios like home automation controlling lights and switches, or environmental monitoring need small devices with little power and computational capabilities. RFC 7222 [43] lists available, RAM, processing power, energy and accessibility as the constrained properties of devices in constrained-node networks, like the IoT.

5.3 Requirements

Given these objectives in the view of the problem statement, we identify the following requirements for our security architecture:

End-to-End Authentication The security of the authentication should ultimately depend on the network endpoints of the communicating devices. No control about intermediate devices is required to ensure security.

Multiple Independent Authorities An Internet of Things (IoT) users or providers should be the only authority over the security of their devices. A single central authority is not feasibly and would introduce a single point of failure.

Secure Trust Federation IoT devices of different users and different authorities need to be able to easily communicate with each other in a secure fashion.

Lightweight Protocol and Implementation The architecture needs to be a viable solution for small constrained devices, with little power and memory, which represent a large part of the IoT. Small protocol overhead, and small and efficient implementations are made possible by the use of lightweight cryptography, in the sense of having a low computational complexity for operation and adding a small overhead to the payload to be secured.

Identity Revocation IoT devices are rarely hardened against physical attacks as this would increase the cost per unit. For this reason the identities of compromised devices need to be revoked to limit its attack capabilities. Further, as control over compromised devices is regained and it is reset to a secure state, it can rejoin the system under a new identity.

5.4 Identity-based Security Architecture for the IoT

We propose to use identity-based cryptography (IBC) to reduce communication and management overheads of classic asymmetric public key cryptographic systems. The goal is to provide flexible trust, lightweight and secure authentication of things based on their owners. We utilize the address structure of IPv6 to enable an inherent relation between the device and its owner.

After describing the system entities and roles a short overview over the overall authentication process in the proposed architecture is given. Further, each step during the authentication process within the system is defined in detail.

5.4.1 System Components

Our architecture encompasses the following entities:

Internet of Things (IoT) service An unconstrained computer at an Internet Service Provider (ISP), providing data collection or controlling services to many constrained and distributed IoT devices.

IoT device A constrained device—functioning as actuator or sensor—that is connected to the Internet. This connection can be either direct via standard network routers and switches or via IoT specific border gateways that bridge between different data layers.

Border gateway A dedicated network gateway that, compared to the IoT device, is usually less constrained in memory, processing power and energy supply. It provides IoT devices that communicate via specialized protocols, e.g. IEEE 802.15.4 [44], access to the global Internet.

Authentication support server A piece of software that can be deployed on already existing or dedicated hardware and provides supportive functions required for federated authentication. This can include responding to lookup requests for trusted authority (TA) parameters and accusation or revocation requests.

Hardened security device A specialized security device, comparable to a hardware security module (HSM) that is hardened and secured against physical attacks. It provides critical cryptographic operations related to very private keys, which if publicly exposed, cause a huge damage.

These entities can act as the following roles:

Signature receiver A receiver of a message with a identity-based signature (IBS). If the signer belongs to the same local network only the local TA parameters are required. These are usually received alongside the private identity key during *device initialization* (section 5.4.4).
If the signer is from a federated remote network, the remote TA parameters are required to verify the signature. The TA parameters can be obtained by a TA lookup client.

Signature sender The only security critical information it holds is its private identity key. Anybody with access to this identity key can generate valid signatures for the corresponding *ID*.

Accusation and revocation server This server receives and collects accusations and handles revocations for devices in the local network. It is also the role that kicks off the TA rollover procedure.

Dynamic configuration client A client sending a secure dynamic configuration request to the *dynamic configuration server* during the device initialization phase. The request send by the client and the replied response from the server are secured using pre-shared symmetric keys and a 128 bit symmetric Authenticated Encryption with Associated Data (AEAD) cipher.

Dynamic configuration server This server handles dynamic initialization requests from a *dynamic configuration client*. This could be IoT devices and servers. It responds with the public system parameters of the local TA and the identity key for the device. These requests are sent either when new devices or servers are added to the local network, or during a TA rollover.

During the *device initialization* phase and a *TA rollover* this server securely communicates with the TA to request the new identity keys. Only for this procedure the TA has to be online. During normal operation it can remain offline.

In order to stop handing out new identity keys to revoked devices, it consults the *accusation and revocation server* about the revocation state of a requesting device.

TA The TA generates the private identity keys for all the devices in the local network. It is highly security critical and it is not required to be online during the whole time. Access to the TA is only required when a) new devices are added to the local network and an identity key needs to be generated for them, or b) a TA rollover procedure is started.

It is recommended to deploy the TA on a *hardened security device*, as it represents a high value target to attackers. Exposing the TA private key to attackers results in compromising the security of all future communications with devices of the network corresponding to the TA.

TA lookup client The role queries the TA responder of a remote federated IoT network for its TA parameters and verifies the response. Correctly verified responses are cached locally by the lookup client.

TA lookup responder This role is part of every IoT network and enables secure federated authentication with other IoT networks. It is the communication endpoint of the *TA lookup client*. As per convention, the *TA lookup responder* is always reachable under the node ID of 1 within a subnet.

Furthermore the *TA lookup responder* does not process highly security critical information, as it only need the public parameters of the local TA to function.

Pinning TA lookup cache This role optimizes the authentication process and improves the overall security. It is required for the gateway-based remote TA lookup procedure described in section 5.4.6. The caching and injecting of remote TA system parameters provides the following advantages:

1. Simply injecting *TA lookup responses* in front of incoming authenticated messages eliminates the need for a *signature receiver* to request the remote TA system parameters on its own. Thereby, it reduces the overall message verification time.
2. Caching TA system parameters on the gateway essentially pins the system parameters from a TA lookup reply — only protected by the 64 bit hash embedded in the subnet prefix, to the full 112 bit subnet prefix. This way the risk of man-in-the-middle (MitM) attacks in the Internet on further TA lookups for pinned remote networks is eliminated for all IoT devices behind the gateway.

Depending on the deployment environment and the application scenarios some entities can play multiple roles in the architecture. For example, in a constrained IoT scenario the *border gateway* can take the roles of *accusation and revocation server*, *dynamic configuration server* and *TA lookup client* and *TA lookup responder*. In a server environment at an ISP, the role of the *TA lookup client* is played by the *server* itself which verifies the signatures. In this case a dedicated *authentication support server* is deployed in the local

network for the roles of *accusation and revocation server*, *dynamic configuration server* and *TA lookup responder*.

5.4.2 System Architecture Overview

This section the continuation of our work on "Federated End-to-End Authentication for the Constrained Internet of Things using IBC and ECC" presented in [63]. In order to satisfy the requirements set out in section 5.3, our architecture combines identity-based cryptography (IBC) and Crypto-based Identifiers (CBIDs).

An administrative domain in the Internet of Things (IoT) is an edge-network of the public Internet. With the help of IBC each domain can run an independent trusted authority (TA) and have it generate private identity keys for constrained IoT devices in the edge-network. The entire IPv6 address of a device is also its ID used for the IBC operations. The ID and the identity key is securely delivered to a device with a dedicated secure dynamic device initialization protocol. Using IBC and the entire IPv6 address as ID, allows us to reduce the system management overhead for public key and certificate distribution present in classic public key architectures.

However, communication in the IoT is not limited to a single administrative domain. Authenticating messages from a device of a remote domain, requires the public parameters and the public key of the TA of the remote domain. The system of the remote system TA parameters can be requested from a service running in the remote edge-network. These system parameters are sent unencrypted and could be modified in transit by an attacker. Utilizing lightweight CBIDs in form of subnet IDs equal to a 64 bit cryptographic hash of the public TA parameters, modification of the exchanged public TA parameters can be detected by the receiver. However, as a 64 bit cryptographic hash does not provide a collision resistance of current standard Internet security (256 bit), the use of key pinning is highly recommended, to elevate the lightweight trust from the embedded CBID to a strong trust relationship between subnet prefix and its public system parameters.

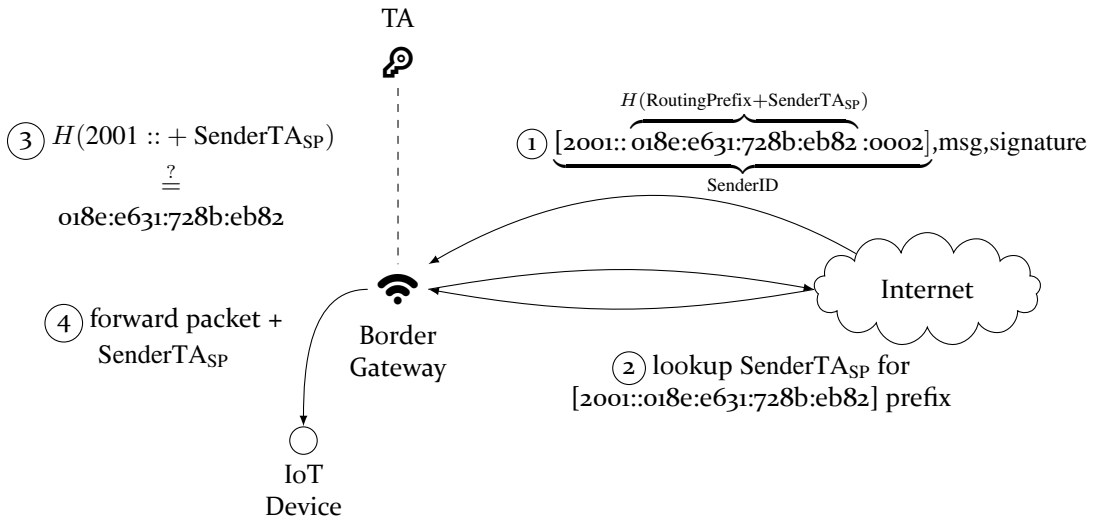


Figure 5.1: Authentication process of IBC-based federated message authentication minimal IoT deployment.

Figure 5.1 shows a minimal deployment scenario describing the process of a federated end-to-end authentication for messages coming from remote administrative domains.

Starting with the IoT deployment domain, i.e. an edge-level network, with a single authority, an IBC-based subnet can be configured. Using the global routing prefix and a compact hash of the system parameters of the TA of the domain, an edge-level network can be configured. This embedding of the hash, shown at (1) in figure 5.1, essentially binds the explicit TA to the subnet and allows independent secure verification of TA parameters that are associated to a network prefix. The security of this autonomous verification depends on the size of the embedded hash, i.e. 64 bit as shown in figure 5.1.

A cryptographic hash of 64 bit is rather small for an architecture designed for 128 bit symmetric security level. This increases the risk of an attacker finding a TA with parameters colliding with the 64 bit embedded hash the attacker aims to compromise. However, this risk only exists during the TA lookup procedure described in section 5.4.6. Once remote TA parameters are looked up and correctly verified against the embedded hash, the complete parameters are pinned to the full 112 bit subnet prefix used during the lookup by the endpoint, i.e. a *border gateway*, *server* or *IoT device*. This risk is further discussed in section 8.1.

The concept of using cryptographically hashed public key material as building block for communication identifiers has already been demonstrated with CBIDs [64, 65] and Cryptographically Generated Addressess (CGAs) [66].

Devices of the user are then assigned node IDs in the subnet of the border gateway. The *dynamic configuration server* — with the help of the TA — generates the identity keys for the devices and securely delivers them during *device initialization* (see section 5.4.4). These identity keys are IBC private keys bound to the identity, i.e., the IP address, of the device.

This allows a verifier to build a secure trust verification chain from the identity-based signature (IBS) of a received message to the corresponding TA parameters that issued the private identity key of the sender. The TA parameters can be verified against the compact hash within the prefix on first contact.

After the first contact verification, a *signature verifier* and a *TA lookup client* can cache the complete TA public parameters for a prefix an essentially strongly pin this association. This is similar to Public Key Pinning Extension for HTTP (HPKP) [67] and prevents attacks from third parties that could inject false TA public parameters for a subnet prefix that create a hash collision to the compact hash in embedded in the prefix.

Figure 5.1 shows the abstract protocol flow of sending an authenticated message to an IoT device, connected to the Internet via a *border gateway*. In this figure, the *border gateway* plays the *TA lookup client* and *signature receiver* roles.

As soon as the *border gateway* receives an IBC signed message over the Internet targeted at a local device, it will attempt to lookup the TA public system parameters corresponding to the prefix in the local cache of pinned TAs. In case of a cache miss, a TA lookup is initiated by requesting the public TA system parameters at the remote *TA lookup responder*, shown as (2) in figure 5.1. The response of remote *TA lookup responder* is verified by the *TA lookup client*, which initiated the request, see (3) in figure 5.1. This, and an alternative client only lookup method, are detailed in section 5.4.6.

The original signed message and the now available public TA system parameters are then forwarded to the final routing endpoint, see (4).

The remaining sections of this chapter go into detail of each step of the authentication process, including key management tasks like *key revocation* and *TA rollover*, to assure continuous security and trust of the system.

5.4.3 System Initialization

Before the administrator of an edge-level network can add devices to the Internet of Things (IoT) network, the cryptographic system parameters need to be initialized and system components need to be configured. The following steps are required:

First, the trusted authority (TA) is initialized using the $Setup(sec\_level)$ function of the chosen identity-based signature (IBS) scheme, which generates the system secret key (TA_{SK}), the public system parameters (TA_{SP}) for the specified security level. The public system parameters are required for signing and verification and need to be made available to the local devices within the edge-network. Furthermore, they need to be distributed to remote networks on demand that need to verify signatures from local IoT devices.

The TA_{SK} is securely stored by the TA and only required for generating an ID_{key} for devices added to the local network. A compromise of the system wide private key means that communication related to this key is no longer secure and can be imitated by an attacker.

After cryptographic initialization has been completed, the subnet ID can be generated and the IoT border gateway can be initialized. For this, the border gateway assigns itself the IP with the node ID of 1 according the format described in figure 5.2 and configures its network interface with this IP address. This allows a *TA lookup responder* to run on the border gateway and as per convention all *TA lookup responders* shall be reachable at the node ID of 1.

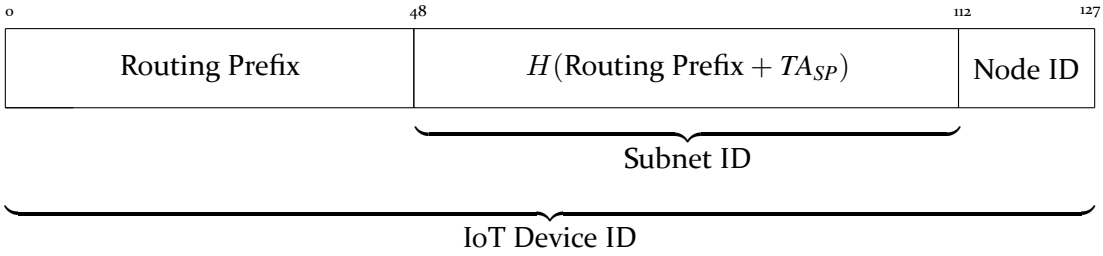


Figure 5.2: IPv6 address format with embedded TA hash

The H-function in figure 5.2 is a cryptographic hash function which outputs a 64 bit hash value. We require H to be *preimage resistant* [27, § 9.2.2], meaning it is computationally infeasible to find any input that hashes to a given hash value.

Possible candidates for the hash function H are the standard SHA2-256 [68] or BLAKE2 [69, 70], trimmed to the first 64 bit. BLAKE2 is a particular interesting option for constrained IoT environments, because it comes in a variant optimized for 32-bit architectures common for IoT devices.

The corresponding private key for the gateway is generated by the TA using the *KeyExtract* function of the specified IBS. From now on, the gateway can verify messages received from other local parties and sign messages.

After the border gateway has been configured for with the subnet cryptographically anchored to the TA, IoT devices IoT devices or server can be regularly added on the IP level. The secure distribution of the ID_{key} for additional devices is described in more detail in section 5.4.4. As a last step of system initialization secret shared keys can be loaded on the *dynamic configuration server* to enable secure online device configuration.

Verification of signatures requires the complete canonical identity of the *signature sender*. In constrained IoT networks the application of 6LoWPAN is common to space optimize protocol headers to increase available application data per packet. It allows to elide known prefixes between two routing points which share a predefined context. IPv6 addresses that are used as identities in this architecture, need to be decompressed before they can be used as input for signature verification.

5.4.4 Device Initialization

For signing messages, devices need to be equipped with an ID_{key}. The ID_{key} is implicitly bound to the ID of the device, its IP address.

Conceptionally there are two strategies to deliver the ID_{key} to a device:

1. Static Initialization (Offline): This strategy configures the network address and the associated ID_{key} during the flashing time of a new Internet of Things (IoT) device. The network parameters, the public trusted authority (TA) parameters and the ID_{key} are written to the device storage.

   This allows the device to immediately start signing messages using identity-based cryptography (IBC) and enable secured end-to-end communication. However, static initialization requires knowledge of the current configuration of the target deployment network and binds the device to the current state of the TA before deployment.

2. Dynamic Initialization (Online): In this case, the final network and security initialization is delayed to the time of device deployment. The finalization of the network and security configuration is secured by symmetric encryption and two one-time keys, $K_{ConfReq}$ and $K_{ConfRsp}$.

   The pairs of securely random generated one-time keys, i.e. $(K_{ConfReq}, K_{ConfRsp})$, are loaded in the configuration of the border gateway during system initialization. When the administrator of an IoT network wants to add a device, one $(K_{ConfReq}, K_{ConfRsp})$ pair is loaded on the device. Afterwards, the device can be deployed into the field.

   During deployment the two keys are used to dynamically and securely setup network and cryptographic long term key material.

The protocol for dynamic initialization, based on the two symmetric keys, $K_{ConfReq}$ and $K_{ConfRsp}$, uses authenticated encryption to request the key material and network configuration from the gateway in a secure manner. The detailed protocol flow for dynamic initialization or dynamic device configuration is shown in figure 5.3 and works as follows.

First, the nonce N is generated and the request for key material and network configuration, i.e. the string "REQ", is encrypted using an Authenticated Encryption with Associated Data (AEAD) [71] symmetric cipher scheme, resulting in Req_C. Req_C alongside with the nonce N forms the request message ($ReqMsg$) and is send over to the border gateway, playing the role of the *dynamic configuration server*.

Definition 5.4.1. RFC 5116 [72, sec. 2.3] defines the high-level interface for a symmetric AEAD cipher scheme as follows:

$$Enc\,(Key, Nonce, Plaintext, Associated\ data) \longrightarrow Ciphertext \tag{5.1}$$

$$Dec\,(Key, Nonce, Ciphertext, Associated\ data) \longrightarrow Plaintext\ \textbf{or}\ FAIL \tag{5.2}$$

After the gateway received the request message from a device, it splits it up into nonce and Req_C and then decrypts it with the $K_{ConfReq}$. This can be looked up in a database using

the layer 2 address of the device (*DevMAC*). An attacker sending device initialization requests from a false layer 2 address also has to know the corresponding $K_{ConfReq}$ and $K_{ConfRsp}$ to successfully acquire an identity key.

After verifying the decrypted message is equal to the string "REQ", a local *NodeID* is generated and the full IPv6 address, i.e. *DevID*, is constructed. Using the *KeyExtract* function of the chosen identity-based signature (IBS) scheme, the IBC private key is generated. This requires access to the security critical master secret of the TA, the TA_{SK}. With that, the plaintext response packet, Rsp_P, is constructed, containing the IPv6 address for the device (*DevID*), the private identity key for the device (ID_{key}) and the system parameters of the TA (TA_{SP}). Finally, the packet is encrypted with the AEAD cipher with the symmetric key $K_{ConfRsp}$ corresponding to the request key and send back to the device.

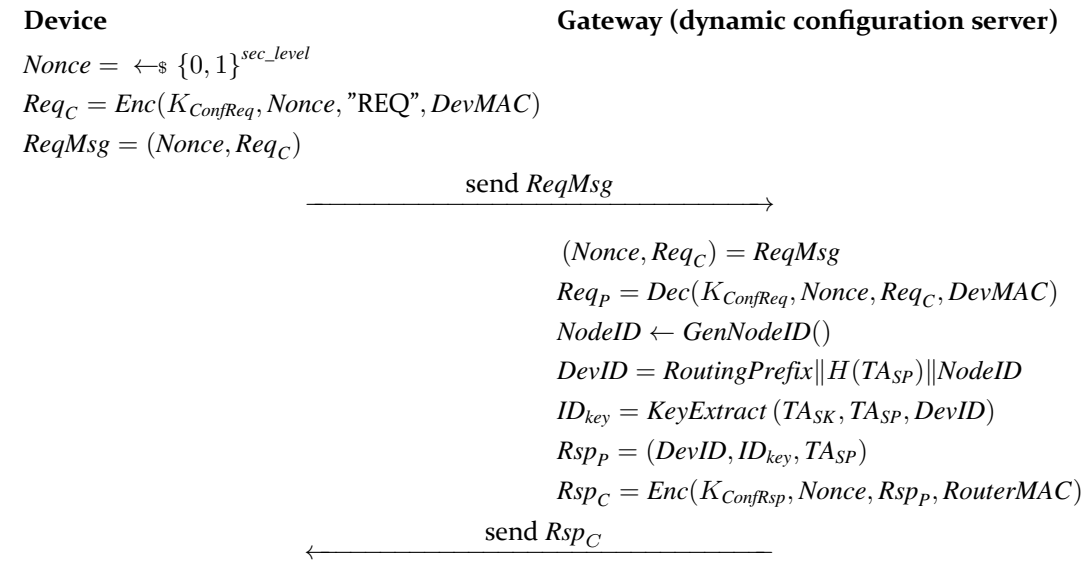


Figure 5.3: Protocol flow for dynamic network initialization between an IoT device and the border router.

The device can then decrypt the response and configure its network interface accordingly.

After a device has dynamically and securely obtained the network and IBC configuration from the gateway as shown in figure 5.3, both sides can clear the symmetric keys $K_{ConfReq}$ and $K_{ConfRsp}$. Using the network configuration the device can then start signing new packets using an IBS. Note that device validation and error checking of the decryption method are left out in figure 5.3 for clarity.

Possible AEAD ciphers for the dynamic initialization protocol include AES-GCM [73], an AEAD construction from ChaCha20 and Poly1305 [74, sec. 2.8], and NORX [75]. NORX

is particularly interesting for our IoT application environment, because of its flexibility to adjust to low power 32-bit devices [76]. This flexibility comes in form of *instances*, the term used by the NORX authors to describe different configuration of the abstract algorithm [77, § 2.3]. An *instance* configuration is parameterized by a word size, a number of rounds and a parallelism degree. Furthermore, NORX performs well in performance and code size benchmarks for low power and inexpensive devices, compared to the classic AES-GCM cipher [78].

5.4.5 Authenticated Two-way Message Exchange

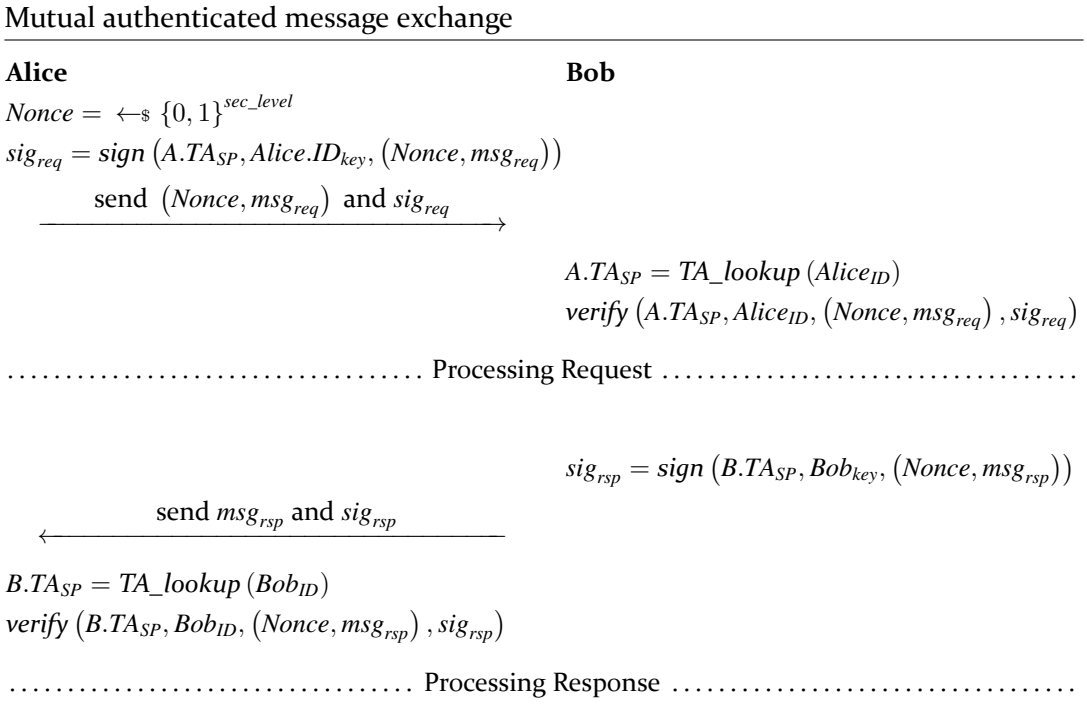


Figure 5.4: Authenticated two-way message exchange diagram

In Figure 5.4, we describe an authenticated message exchange between an Internet of Things (IoT) device and a server. Starting with the public system parameters (TA_{SP}), Alice's ID ($Alice_{ID}$), her private key ($Alice_{key}$) and a request message (msg_{req}), Alice can generate a signature to sign her request and send it over to Bob.

As soon as Bob receives the request, he can look up the correct public system parameters of the trusted authority (TA), TA_{SP}, based on Alice's ID. Afterwards he verifies the request and, provided the verification succeeds, can start processing the request. A potential response can then be signed and send back to Alice, who can lookup the public system

parameters of Bob's TA, TA_{SP} and then verify and process the response. The requests and responses can be anything, but in the case of the IoT likely they are Constrained Application Protocol (CoAP) messages.

While this protocol is very simple and straightforward, the *TA_loopup* function has yet to be defined. The definition of the *TA_lookup* function, is a major contribution of this work and is discussed in detail in the following section. Furthermore, the described protocol, as an authentication protocol, does not provide confidentiality nor perfect forward secrecy.

However, it can be used as a foundation for more secure protocols like:

1. Elliptic Curve Diffie-Hellman (ECDH): identity-based signatures (IBSs) can be used to sign the two messages of an ECDH key exchange and help securely establish a symmetric shared secret between two parties. Using this shared secret and symmetric cryptography forward secrecy is enabled, i.e. even a future compromise of a private identity key (ID_{key}) will not expose messages secured using the shared secret established in past.

2. Identity-Based Authenticated Key Exchange (IBAKE): Depending on the identity-based cryptography (IBC) system that is used, combined algorithms can be available that provide an authenticated key exchange using IBC in one go. Ideally, the IBAKE algorithm would be more efficient than the plain ECDH based construction described before.

   There exist IBAKE schemes which are designed for secret-key agreement between user from different domains and authorities [79], which would be a good match for the federating system architecture we propose.

3. Datagram Transport Layer Security (DTLS): Regardless of whether plain ECDH or an IBAKE is used, it is possible to integrate the established shared secret into the DTLS authentication step. This allows to build on the security and more heavily tested DTLS protocol and implementation. The fact that a common application level protocol for the IoT, CoAP, builds on DTLS to secure request and response messages, supports this point.

5.4.6 Authority Lookup and Verification Methods

When both communication partners are from the same subnetwork, they have private identity keys issued from the same trusted authority (TA). This means the lookup for the corresponding authority is trivial, because the devices already have the public system parameters of their own local TA.

In the case of communication partners from different networks, the corresponding border gateway—playing the role of a *TA lookup responder*—needs to be consulted for the remote public system parameters.

The following sections describe two different authority lookup methods, a self-contained independent client-based approach that can be done by an unconstrained Internet of

Things (IoT) service itself and a gateway supported method that offloads heavy work from the constrained devices to an unconstrained border gateway.

Client-based Authority Lookup

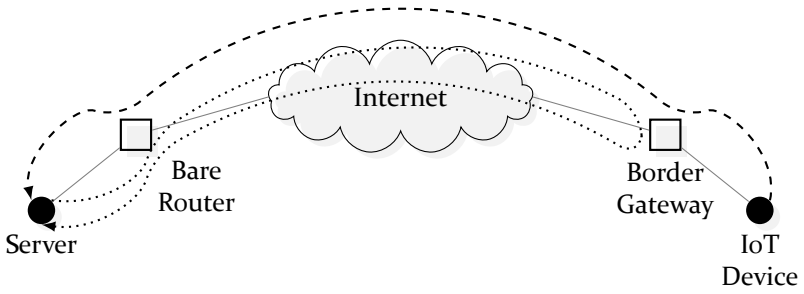


Figure 5.5: Client-based TA system parameter lookup and verification.

Figure 5.5 shows the abstract process of the lookup of public system parameters by a standalone powerful endpoint. After the server received a signed message from a device in a remote network (marked as - - - - in figure 5.5), it attempts to lookup the pubic system parameters of the TA corresponding to the remote network in its local trust database. If the lookup in the local trust database fails, the server endpoint will start the lookup process (·····) for the remote system parameters, which works as follows:

1. The requesting endpoint sends the system parameters request to the remote border gateway, conventionally located at the $NodeID =: 1$.
2. The remote border gateway responds with a signed message, containing the system parameters TA_{SP}.
3. On received of the gateway response from *GatewayID* the requesting endpoint validates the message and the contained public system parameters in the following way:
   a) Hashing the received public system parameters of the remote system, i.e. $H_{Rsp} = H(TA_{SP})$.
   b) Verify that H_{Rsp} is equal to the hash value of the subnet prefix of the from IPv6 address, i.e. the remote border gateway.
   c) If the verification succeeds and the subnet prefix of the IPv6 from address is of the subnet looked up in step 1, the trust association $(\text{Prefix}(GatewayID), TA_{SP})$ is stored in the local trust database.
   d) If the verification fails, the error is reported back to the calling code.

This lookup procedure combines a cryptographic address approach for the system parameters of a TA with a *leap of faith* or trust on first use (TOFU) trust approach. The

cryptographic address approach provides a medium protection against man-in-the-middle (MitM) attackers during the initial lookup of remote system parameters. The TOFU approach protects against MitM attackers at a future time.

Gateway-based Authority Lookup

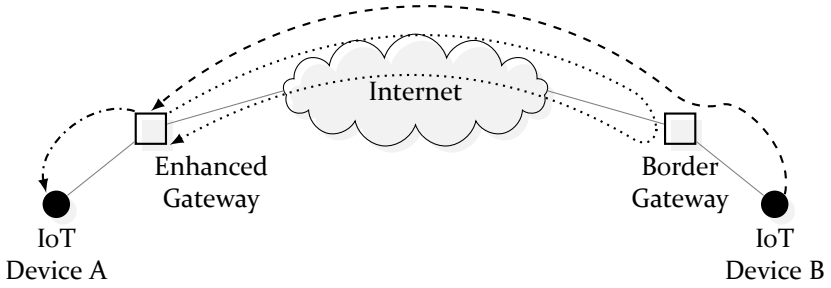


Figure 5.6: Gateway-based TA system parameter lookup and verification.

This scenario requires an enhanced gateway which connects IoT devices in a subnet using a specialized protocol to the worldwide routed Internet. Simple network gateways are already present in many IoT scenarios in form of IEEE 802.15.4 [44] border routers and can be extended with additional security features to help during the authentication process.

This means that all communication to and from the IoT devices goes through the *border gateway*. The border gateway can support the hardware constrained IoT devices, by acquiring and verifying the public system parameters of remote TAs. Verified TA parameters are then forwarded to the endpoints, i.e. constrained IoT devices, which can immediately verify the signatures of messages.

Figure 5.6 shows the abstract acquisition of the public system parameters of a remote TA by the enhanced gateway. The process is described as follows:

1. As soon as the border gateway detects a message from an unknown network to one of its clients (marked as - - - - in figure 5.6), it delays routing of that message.

2. Before forwarding the message, the border gateway requests the corresponding TA public key from the gateway (·····). The verification procedure is the same as in the client based method described in the previous section.

3. On successful verification, the original message and the acquired and verified system parameters of the remote network are forwarded to the constrained IoT device (- · - · -).

5.4.7 Device Revocation

Revocation of identities or their associated private key in identity-based cryptography (IBC) is not a trivial task, as already discussed in section 3.3.2. Existing approaches to the revocation problem in IBC discussed there either only support revocation by expiration or follow

a completely decentralized approach without the goal to scale up to the size of the Internet.

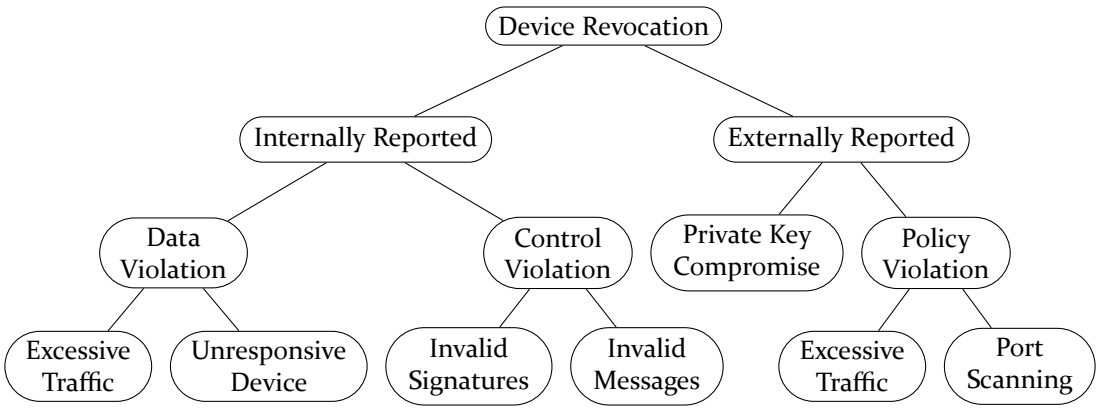


Figure 5.7: Device revocation categories.

We categorize revocation scenarios for as shown in figure 5.7. A device revocation can be either reported internally, by neighbor devices in the same network or externally by users noticing bad behavior. If a device notices a huge amount of invalid signatures or traffic in general from a specific node, it accuses the node of malicious behavior and can notify its gateway.

Internally Reported Accusations

If a device in the local network notices malicious behavior by one of its neighboring devices, it can send an official accusation message to the border gateway. This accusation message is constructed as shown in figure 5.8.

$$1: \quad AccMsg = (\text{"}Acc\text{"}, AccusedNodeID, AccusedMAC)$$
$$2: \quad AccPacket = (AccMsg, Sign\,(SP, ID_{key}, AccMsg))$$

Figure 5.8: Construction of an accusation message.

The packet *AccPacket* is then send to the border gateway of local network of the reporting device.

All accusations for malicious behavior or explicit revocation requests are validated and managed by the border gateway on receive. If the accusation message passes the validation, row is added to the device revocation table shown in table 5.1. The accusation is saved with the sender ID of the accusation message and the *NodeID* and MAC address of the accused device. The table contains only one row per $(AccuserID, AccusingID, AccusingMAC)$ triple.

AccuserID	*AccusingID*	*AccusingMAC*
...	...	...

Table 5.1: Device accusation table.

As soon as there are more than *AccusationThreshold* rows for a specific *AccusingID* in the device accusation entry, the device is revoked. This means a row for the revoked device is added to the revocation table shown in table 5.2.

RevokedID	*RevokedMAC*
...	...

Table 5.2: Revocation table.

The revocation table serves two purposes:

1. The IP address of a revoked identity, i.e. *RevokedID*, can immediately added to a network firewall. This firewall can be located on the border gateway and block traffic for the revoked identities.
2. It is checked against during the trusted authority (TA) rollover procedure, in which devices dynamically renew their ID and private key material. Revoked devices will not receive a new ID and key material during that process.

If there are more than *RevocationThreshold* revoked devices in the revocation table, a TA rollover is initiation, described in detail in the following section.

Externally Reported Accusations

Accusations and revocation requests from external parties need to be reviewed more carefully to prevent Denial of Service (DoS) attacks where remote parties try to revoke identities they never communicated with. The reports can be caused by human users or automatically be remote intrusion detection system (IDS). Correctly validated and reviewed external reports are added to the accusation table, or in case of severe reports directly added to the revocation table.

5.4.8 Rollover of the Trusted Authority

Once the decision is due, a secure rollover of the trusted authority (TA) is initiated. An appropriate *RevocationThreshold* depends on the application scenario and its potential attackers and needs to be carefully evaluated.

A high *RevocationThreshold* means that more storage is required for the accusation table and the revocation table. Network protection systems like firewalls have to manage and evaluate more rules per packet. However, the constrained Internet of Things (IoT) devices in the local network remain unprotected against the revoked devices until a TA rollover

successfully completed. This is because the revocation information is not distributed to all local devices as they are highly constrained in their memory capacity.

Lower values for the *RevocationThreshold* will result in more frequent TA rollovers as devices are revoked. TA rollovers are associated with a communication and computation cost for the distribution of new keys and reconfiguration of potential routing tables due to change of subnet prefix. These costs affect all but the revoked devices in the local network.

The procedure to rollover to a new TA works as follows:

1. Generate new TA, including system parameters TA_{SP} and secret key TA_{SK} as described in section 5.4.3.

2. Generate new IP address and corresponding identity key for the gateway.
   $ID^{new} = RoutingPrefix \| H\left(TA_{SP}^{new}\right) \| :1$
   $ID_{key}^{new} = KeyExtract\left(TA_{SK}^{new}, TA_{SP}^{new}, ID^{new}\right)$

3. Assign new IP address, i.e. the generated ID^{new}, as additional address to the network interface.

4. Notify other IoT networks about the network identity transition. The other IoT networks are discovered by the gateway by monitoring the IP level metadata of the forwarded traffic. This allows to transition the identities of devices of the old TA securely to their new identities in the new TA. It also actively updates the TA lookup cache of bound public TA public parameters. This saves a future superfluous TA lookup for the new prefix and TA.

   The notification message send to remote networks is constructed as follows:

   1 : $TARolloverMsg = (\text{"TA\_Rollover"}, H\left(TA_{SP}^{new}\right))$
   2 : $TARolloverPacket = \left(TARolloverMsg, Sign\left(TA_{SP}^{old}, ID_{key}^{old}, TARolloverMsg\right)\right)$

   Figure 5.9: TA rollover notification to federated IoT networks.

5. The local router or gateway broadcasts a signed TA rollover notification to their devices.

6. On receive of the notification broadcast from the previous step, the devices securely request a new ID and key in the new network via Elliptic Curve Diffie-Hellman (ECDH) and Authenticated Encryption with Associated Data (AEAD).

Key renewal

Device (Dev)		**Gateway (Ga)**
$q \leftarrow \#E(\mathbb{F}_p)$		$q \leftarrow \#E(\mathbb{F}_p)$
$Dev.private \leftarrow_\$ \mathbb{Z}_q$		
$Dev.public \leftarrow P_0 \cdot Dev.private$		
$ReqMsg \leftarrow$ ("KRReq", $Dev.public$)		
$ReqSig \leftarrow Sign\,(TA_{SP}, ID_{key}, ReqMsg)$		
$ReqPacket \leftarrow (ReqMsg, ReqSig)$		
	send $ReqPacket$ $\longrightarrow$	
		$(ReqMsg, ReqSig) \leftarrow ReqPacket$
		$Verify\,(TA_{SP}, Dev.ID, ReqMsg, ReqSig) = true$
		Check that device is not revoked
		("KRReq", $Dev.public$) $\leftarrow ReqMsg$
		$Ga.private \leftarrow_\$ \mathbb{Z}_q$
		$Ga.public \leftarrow P_0 \cdot Ga.private$
		$RenewKey \leftarrow H\,(ReqPacket \| (Dev.public \cdot Ga.private))$
		Generate new ID in the new network
		$NewDevID \leftarrow GenDevID()$
		$NewID_{key} \leftarrow KeyExtract\,(New.TA_{SK}, New.TA_{SP}, NewDevID)$
		$RspSecretMsg \leftarrow Enc(RenewKey, Nonce, NewID_{key}, RouterMAC)$
		$RspMsg \leftarrow$ ("KRRes", $Ga.public, DevID, New.TA_{SP}, Nonce, RspSecretMsg$)
		$RspSig \leftarrow Sign\,(TA_{SP}, ID_{key}, RspMsg)$
		$RspPacket \leftarrow (RspMsg, RspSig)$
	$\longleftarrow$ send $RspPacket$	
$(RspMsg, RspSig) \leftarrow RspPacket$		
$Verify\,(TA_{SP}, Gateway.ID, RspMsg, RspSig) = true$		
("KRRes", $Ga.public, DevID, New.TA_{SP}, Nonce, RspSecretMsg$) $\leftarrow RspMsg$		
$RenewKey \leftarrow H\,(ReqPacket \| (Ga.public \cdot Dev.private))$		
$NewID_{key} \leftarrow Dec\,(RenewKey, Nonce, RenewKey, RouterMAC)$		

Figure 5.10: Authenticated device key renewal during TA rollover.

7. Remove old TA and network routing after a grace period. From this point on revoked devices should be unable to communicate in the local and global networks.

6 Protocol Compatibility and Deployment Considerations

6.1 Compatibility with Existing Internet Protocols

Our proposed security architecture changes the semantic of the IPv6 address, by embedding of an cryptographic hash. The length of the embedded hash is 64 bit and requires to increase the subnet ID accordingly. The remaining 16 bit are used as node ID.

The usage of IPv6 addresses as cryptographic identities and public keys in an identity-based cryptography (IBC) system, and the short node IDs result in compatibility issues which are discussed in the following sections.

6.1.1 Large IPv6 Prefixes and Stateless Address Autoconfiguration

In legacy IPv4 networks, the Dynamic Host Configuration Protocol (DHCP) [80] is commonly used for automatic address configuration. In DHCP a new host requests network configuration including a IPv4 address from a DHCP server.

IPv6 takes a completely different approach. It uses stateless address autoconfiguration (SLAAC) [81], a stateless configuration method which generates link-local and global addresses independently. The global addresses are generated by appending an interface identifier derived from the link-layer address, e.g. an EUI-64 ID, to the routing prefix.

However, our proposal works with 16 bit node IDs and a 112 bit prefix length which is incompatible with standard SLAAC.

An alternative protocol for address configuration in IPv6 networks is DHCPv6 [82], similar to DHCP in IPv4 networks. While it allows address configuration for smaller subnets which do not provide large enough node IDs for stateless autoconfiguration, it is an insecure protocol.

Instead our architecture uses the dynamic device initialization protocol, outlined in section 5.4.4. It extends standard network configuration with cryptographic configuration by also providing private keys to the configuring device.

6.1.2 Compliance to Global and Local IPv6 Routing

The general format for IPv6 global unicast addresses [83, § 2.5.4] allocates the first 64 bit of the IP address to the *network prefix* which is split up into the *global routing prefix* at front, followed by the *subnet ID*, as shown in figure 4.1. Our proposed architecture changes the structure of the IPv6 address. The first 48 bit are still allocated to the *global routing prefix*, but the size of the *subnet ID* has increased to 64 bit to include a cryptographic hash.

However, compliance to standard IPv6 global addresses is still achieved, as long as the first 16 bit of the 64 bit subnet ID are locally unique. The first 16 bit of the subnet ID are equivalent to the complete to the total subnet ID in traditional IPv6 global unicast addressing.

This uniqueness is given, as long as the following requirements are met:

- The random number generator used for generating trusted authoritys (TAs) provides high-quality cryptographic randomness [84]. This reduces the chance of generating two identical TAs under a single *global routing prefix*, as identical TAs would have identical cryptographic hashes as *subnet ID*.
- The output of the cryptographic hash function approximates a uniform random variable. This property is typically present in cryptographic hash functions [27, § 9.7.1].
- The number of TAs generated for a *global routing prefix* is limited. The theoretical number of subnets is limited by the size of the subnet ID, i.e. 2^{64} for our 64 bit subnet IDs. However, due to the *birthday paradox* [27, § 2.1.5], after generating 2^{32} TAs, there is one colliding TA on average.

These requirements are easily met in our proposed architecture, meaning that the subnets used by our systems and the full IPv6 addresses are compliant to standard IPv6 global unicast addresses, resulting in interoperability on the Internet layer.

6.1.3 Incompatibility with IPv6 Transition Mechanism

To enable connectivity between the coexisting IPv4 and IPv6 Internet, multiple transition mechanisms have been proposed. The security architecture proposed in this work requires the use of IPv6 addresses as identifiers. The main transition mechanism interesting to this proposal is allowing IPv6-only enabled Internet of Things (IoT) devices to access IPv4 servers in the Internet.

NAT64 [85, 86], as part of Carrier-Grade NAT (CGN) [87] deployments, allows IPv6-only hosts to access IPv4-only hosts with the help of a protocol translation gateway. The gateway provides its own IPv6 network prefix and maps IPv4 hosts to an IPv6 address within the prefix, by embedding the 32 bit IPv4 address in the IPv6 address.

However, with this translation the IPv4-only host has no control over the IPv6 subnet prefix the network address translation (NAT) gateway from the provider will use, and can not generate a trusted authority (TA) and IDs for the IoT devices. While the end-user devices in the Internet Service Provider (ISP) network are able to communicate with other IPv6 hosts on the global Internet, they can not generate IPv6 packets which are authenticatable using our proposed architecture and protocol, due to missing control over the IPv6 subnet prefix.

Another IPv4/IPv6 transition technology in deployment is Dual-Stack Lite [88]. Here, the end-user devices receive globally routed IPv6 addresses in addition to private IPv4

addresses from the ISP. The private addresses in IPv4 packets are translated via NAT in a gateway supplied by the ISP to IPv6 packets with a shared IPv6 address. Our proposed federated end-to-end security architecture is compatible with this transition technology, as the IPv6 part of the stack is native and end-users are assigned globally routed subnets without NAT.

6.1.4 Compatibility with Internet of Things Application Layer Protocols

Constrained Application Protocol (CoAP) [45] and MQTT [89] are popular choices used by IoT solutions as application layer protocols. They both focus on low transport overhead and easy processing for constrained devices.

CoAP recommends to use Datagram Transport Layer Security (DTLS) [46] to provide security to the protocol. MQTT specifies to use standard transport layer security (TLS) [90] for securing the communication channel [89, sec. 5]. In section 5.4.5 we outlined the idea on how identity-based signatures (IBSs) or Identity-Based Authenticated Key Exchange (IBAKE) algorithms can be integrated in TLS and DTLS to integrate our proposed authentication architecture in these existing security protocols.

Part of the DTLS handshake is a Diffie-Hellman key exchange, where the messages are signed by each party. The public keys used for signing the key exchange are commonly transfered to each other as X.509 [91] certificates. One possible way for integration of IBS and DTLS is to modify this key exchange procedure. The X.509 certificates are not needed in identity-based cryptography (IBC) and can be omitted. Instead the key exchange messages are signed using an IBS and the IPv6 address as the identity.

6.2 Deployment Scenarios

This section discusses possible Internet of Things (IoT) deployment scenarios and how our identity-based cryptography (IBC) security architecture can add usable security to the scenarios.

6.2.1 Federated Smart Metering with Independent Electricity Providers

Smart meters measure the power usage of a household and digitally transfer the collected measurement data to the electricity provider. This eliminates the traditional manual process of reporting the power usage by the customer.

The electricity retailing market is open to competition which results in regular prices changes by the providers to outbid competitors. Economic customers will watch for these changes and switch to a less expensive provider if possible.

Power usage measurements are very private data, as it can be used to deduce behavioral patterns about the members of a household, e.g. working hours or times at which heavy energy consumers like washer and dryer are used. Current proposals for secure smart metering do not consider the fact of the customer changing electricity providers.

So *et al.* [92], propose an identity-based cryptography (IBC) based solution for secure smart metering with little configuration overhead. However, they also envision a central power supplier being the sole authority on the security of the system.

In a secure smart metering scenario based on our proposed system architecture for secure federated end-to-end authentication, both the electricity provider and each user would run publicly accessible Internet of Things (IoT), with the associated trusted authority (TA) and border gateway.

For smart metering based on standardized protocol, the use end user would just setup the smart meter in his IoT network and run the dynamic device configuration procedure. After configuring the receiving end-point of the electricity provider, end-to-end authenticated messages can be exchanged and metering measurements can be reported over a highly secure Datagram Transport Layer Security (DTLS). When the household changes the electricity provider, it only has to change the provider endpoint in their smart meter device.

If the power company uses proprietary smart meters, the protocol can still take advantage of the already present end-to-end authentication infrastructure and build their custom protocol on top of this.

This allows flexible integration of secure smart metering in the IoT networks in households, which keep the authority over the security and trust in their local network while still being able to communicate securely with the rest of the Internet.

6.2.2 Environmental Monitoring with Sensor Sharing

In section 4.2.3 we discussed the proposal of Oliveira *et al.* [58] to use public-key cryptography to authenticating traffic from a constraint data collection node to multiple users. This section focuses on the deployment scenario described by them and how our proposed federated authentication architecture works in this scenario.

Oliveira *et al.* describe a scenario, where a sensor node provides valuable data, interesting to different parties. For instance, a smart meter installed by the power company in the household collects power usage and reports it back to the power company for smart metering and accounting purposes. Other parties, e.g. the household members or the landlord, could also be interested in the metering data.

Applying our proposed architecture to this scenario, the user would install a border gateway with a trusted authority (TA) for her Internet of Things (IoT) devices. A smart metering device can simply be added by supplying symmetric keys for dynamic device configuration. It would connect to the power provider and authenticate the packets using identity-based signature (IBS) to report power usage. Other interested parties could do the same to access the data via authenticated messages, if they are authorized.

Another example is a research institute which deploys IoT devices to collect environmen-

tal measurements like temperature, humidity and barometric pressure. Probably there are many researchers who are interested in the data beside the deploying institute.

The devices are simple shipped with new symmetric keys for dynamic configuration and are then deployed in the wild. A single IoT border gateway, for TA lookup services and dynamic device configuration, which is independent from battery power and has access to IPv6 can then be used to connect many environmental monitoring devices to the global IPv6 Internet. With mesh routing protocols even larger areas can be covered, while the end-to-end security of authenticated packet stays intact. After deployment users in remote networks with the federated end-to-end security architecture set up, can simply query the devices and get access to the monitored data via authenticated messages.

7 Implementation

In order to practically evaluate the proposed architecture we implement the components required for an evaluation scenario. We implement the proposed architecture in C/C++ using Raspbian[1] and RIOT [93] operating systems (OSs) and the RELIC [23] library for cryptographic routines.

The communication protocol UDP is used, and messages for any data exchange and stored data are encoded using the Concise Binary Object Representation (CBOR) [94]. CBORs allows schema-free and compact encoding of object data with the availability of lightweight encoding and decoding development libraries. RIOT already includes a CBOR parsing and serialization implementation. The border gateway and the Internet of Things (IoT) service component use the open source and MIT licensed `libcbor`[2] library. Thus it fits the requirements for our constrained application domain, the IoT.

Our implementation and the following evaluation covers the high-level functionality and interactions described in section 5.4, except for the device revocation in section 5.4.7 and the trusted authority (TA) rollover in section 5.4.8.

7.1 Dependencies

The implementation created to evaluate our federated end-to-end authentication architecture depends on several open-source software projects. All the dependencies are C or C++ libraries and applications freely available on the Internet. The exact version and sources of the dependencies are listed in table 7.1.

7.2 Software Components

The following three sections go into the details of the software components implemented to evaluate the proposed secure Internet of Things (IoT) architecture. These are the implementation for the IoT sensing node, the IoT border gateway and an IoT service.

The source code of all three software components can be found online in a GitHub repository [3].

[1] `https://www.raspbian.org/`

[2] `http://libcbor.org/`

[3] For the source code of our implementation of the three described software components in this work, see `https://github.com/tfar/ibce2eiot`

Software	Version	Description	Website
RIOT	git:b68213a	embedded OS	http://riot-os.org/
RELIC	git:7d6f795	cryptography	https://github.com/relic-toolkit/
libcbor	git:3371385	CBOR encoding	http://libcbor.org/
libtins	git:af71a4e	network packet encoding	http://libtins.github.io/
boost	1.50	general purpose	http://www.boost.org/
NORX	git:36209b8	AEAD cipher	https://norx.io/
easyloggingpp	9.80	logging	https://github.com/easylogging/

Table 7.1: Software dependencies, version and sources used for the implementation.

7.2.1 Internet of Things sensing node

The sensing node for our evaluation scenario is implemented using RIOT [93]. RIOT allows to develop constrained embedded applications as standard ANSI C applications and features cooperative multi-tasking between different threads.

Our IoT sensing node provides secure access to sensing data collected by the node. As the SAM R21 board does not feature any on board sensor, in our scenario the sensing node servers randomly generated data.

Our sensing node covers the following roles introduced in section 5.4.1:

- Dynamic configuration client
- Signature sender
- Signature receiver
- Trusted authority (TA) lookup client

After the initialization phase of RIOT, our main code is executed which initializes the cryptographic library RELIC, sets up the 802.15.4 radio channel, registers our networking thread for UDP packets and starts the dynamic device initialization protocol described in section 5.4.4.

Following the successful transmission of the dynamic device initialization packet, the sensing node waits for incoming packets.

On receive of a UDP packet, it is analyzed by port number and it can fall in one of the following three cases:

Device initialization reply When a reply to our dynamic device initialization request is received, it is decrypted via NORX [75] Authenticated Encryption with Associated Data (AEAD) cipher. If the decryption succeeds the reply is authentic and the own IPv6 address, TA parameters and private identity key are set.

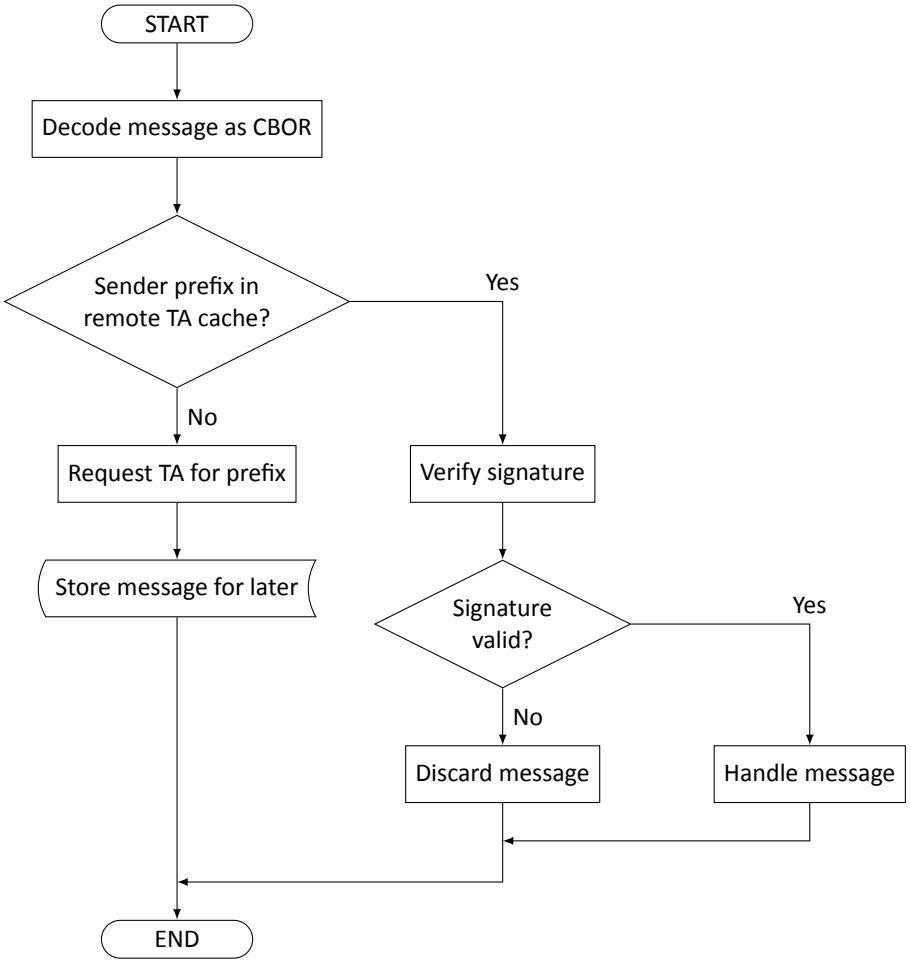


Figure 7.1: Flow chart of the authentication procedure on the embedded node.

TA lookup reply To authenticate signatures from devices of remote IoT edge-networks, their TA system parameters are required. If the IoT devices does not have the TA system parameters for the remote network present in its cache, it sends a request to the remote *TA lookup responder*. The system parameters in the reply are verified against the hash embedded in the remote subnet prefix and validated. For elliptic curve cryptography (ECC)-based identity-based cryptography (IBC) systems this validation is done by assuring that elliptic curve points are on the elliptic curve defined by the system. If the system parameters pass the verification, they are added to the cache; else they are discarded.

In a gateway supported TA lookup setup, described in section 5.4.6, *TA lookup replies* can be received from the gateway just before it forwards the original remote message to the IoT device. For security remarks regarding this gateway-injected TA lookup response, see section 7.2.2.

Due to the memory constraints of IoT devices, this cache is highly limited. In our implementation the cache only contains a single subnet prefix/TA system parameter pair.

Signed message On receive of a signed message it is validated and handled in a multi stage process, described in figure 7.1. If the IPv6 subnet prefix of the sender address of the message matches a prefix in the local TA cache, it can be authenticated using its signature and the *verify* function of the vBNN-IBS implemented in RELIC and described in section 3.2.1. If the verification proofs the signed message to be correctly authenticated by the sender, the message is processed. In our case this means a reply with random data is generated, signed with a vBNN-IBS signature, CBOR encoded and sent back to requesting peer.

If the IPv6 subnet prefix of the sender address is not in the local TA cache, a TA lookup request is initiated as described in section 5.4.6. In this case the original signed message is cached and authentication is delayed until the required TA parameters are acquired.

7.2.2 Internet of Things border gateway

The border gateway is implemented as a standard C++ application running on the Raspbian, a Debian-based Linux distribution for the Raspberry Pi.

The roles supported by the IoT border gateway are as follows:

- Dynamic configuration server
- Trusted authority (TA)
- TA lookup responder
- TA lookup requester
- TA lookup cache

After initialization of RELIC library, the TA component of the gateway program is initialized. For this the program checks for existing TA system parameters and TA keys in a file in the current working directory and loads them if present. Otherwise it will generate new TA keys as described in section 3.2.1 from random data. The deployment of the TA on the gateway device and storing the TA keys on the same device is not recommended for real world secure deployments because of cost and complexity of hardening the Raspberry Pi and its software stack. Instead, a specialized *hardened security device*, e.g. a hardware security module (HSM) operating the TA and storing the high value master secret key, is recommended for deployment in non-testing scenarios.

Following, the network components for the *dynamic configuration server*, the TA lookup responder, and, if requested, the TA lookup cache roles are started. In addition, the network device is configured for the IPv6 subnet with the embedded hash of the TA system

parameters and assigns itself an address with the node ID 1.

Part of the IoT border gateway is the *authentication support server* that plays the role of the *dynamic configuration server*. The symmetric keys for the initial device authentication during the dynamic device initialization are stored build into the program. After start of the gateway component the *authentication support server* joins a predefined IPv6 link-local multicast group, where it awaits dynamic configuration requests from IoT devices.

On receive of a dynamic configuration request, the request is decoded from Concise Binary Object Representation (CBOR), decrypted using the Authenticated Encryption with Associated Data (AEAD) cipher NORX, and the nonce supplied by the requester and the $K_{ConfReq}$ symmetric key. The cleartext is then tested against the fixed string "REQ". If the request is valid, a node ID within the subnet and the corresponding identity key ID_{key} are generated, encoded to a CBOR message and encrypted using the same algorithm and nonce but with the $K_{ConfRsp}$ key.

The prototype implementation holds the keys in static memory inside the code. A security hardened implementation would store the key externally from the code and securely remove them from the storage media. This would prevent attackers, who gain control of the machine, from the ability to reuse keys for authentication during the dynamic device initialization phase.

The *TA lookup responder* is another essential component of the IoT border gateway implementation. During the start of the gateway and after initialization of the TA it is started with the public system parameters of the TA and listening on a predefined port for incoming TA parameter lookup requests from remote IoT networks.

The Linux kernel and its 6LoWPAN stack[4] take care of the gateway functionality, translating and forwarding packets between IPv6 and 6LoWPAN networks.

If requested at start by specifying the `-piggyback=1` parameter, the gateway will also play the role of a *TA lookup cache*. The component for this role depends on the Linux *ip6tables* firewall, which is installed on the Raspberry Pi. In addition, it uses the Linux *netfilter queue* feature to register itself for packet handling of incoming packets authenticated with an identity-based signature (IBS) signature.

The Linux *netfilter queue*[5] is a feature that allows to setup firewall rules in the *ip6tables* kernel-space network firewall, that forward packets to special in kernel queue. Using the API of the *netfilter queue* library, user-space programs like our IoT gateway component, can connect to a *netfilter queue* and read the network packets. The same API also provides a back channel for the user-space program to signal the *ip6tables* firewall different packet handling decisions like dropping a packet or letting it path through.

[4] For more information on the 6LoWPAN implementation in Linux and its configuration, see `http://wpan.cakelab.org/`.

[5] Further information about user-space low-level network handling with *netfitlter queue* on Linux, see `http://www.netfilter.org/projects/libnetfilter_queue/`.

As soon as *ip6tables* adds an incoming authenticated network packet to the *netfilter queue*, our IoT gateway component is notified. It reads the packet from the queue and tells the firewall to drop it. With the authenticated IPv6 packet in user-space, it is parsed using *libtins*, a high-level, multiplatform C++ network packet sniffing and crafting library. The *TA lookup cache* will check the if TA system parameters for the 112 bit prefix of the IPv6 source address of the packet are in the cache and act as following:

1. If the subnet prefix of the IPv6 source address is unknown to the TA parameter cache, a TA lookup is initiated. On successful validation of the TA system parameters replied from the remote *TA lookup responder*, the TA system parameters will added to the cache alongside the corresponding 112 bit prefix. If the validation fails, the original authenticated message will be discarded.

   Afterwards, the cached TA lookup response for the remote subnet and the original identity-based cryptography (IBC) authenticated message are injected back in the network stack using *libtins* to be forwarded to the destination of the authenticated message.

2. If the subnet prefix of the IPv6 source address is already in the TA parameter cache, the gateway will construct a TA lookup response for the cached system parameters and inject it and the original authenticated message back into the network stack to be forwarded to the destination.

As currently implemented, the TA lookup responses created by the TA lookup cache are not further authenticated. An additional signature by the IoT border gateway would add further security to these lookup responses, but add an extra validation burden to the constrained IoT devices. To improve the security further the gateway should never forward TA lookup responses without validation against the embedded hash and against the pinned remote TA system parameters.

7.2.3 Internet of Things service

The IoT service is an example for a compact implementation a service provider could use as a self contained application that contains all the required components of the architecture to sign messages and verify authenticated messages received from remote networks. It serves as initiator of the request/reply protocol used for testing in the evaluation chapter.

The roles supported by the IoT service are as follows:

- Dynamic configuration server
- Trusted authority (TA)
- TA lookup responder
- TA lookup requester
- Signature sender

- Signature receiver

It is implemented in C++, reusing most code of the IoT border gateway implementation. This means the same roles are implemented which the border gateway component already provides. In addition, a signature sender which sends a predefined message with a vBNN-IBS signature, and a signature receiver, which queries the TA lookup requester for the public system parameters of the remote TA, and when present verifies the message against the vBNN-IBS signature, are implemented.

7.3 Interoperability Issues

The 6LoWPAN implementation for Linux, i.e. wpan-next[6], and RIOT are both under active development. At the time of writing, RIOT did not support 6LoWPAN UDP header compression [50, sec. 2.3]. However, it is still possible to setup interoperable 6LoWPAN communication between Linux and RIOT, by disabling UDP header compression on Linux. This can be done by running the following command on the Linux host: `rmmod nhc_ud`.

[6] See `https://github.com/linux-wpan/linux-wpan-next`.

8 Evaluation

8.1 Security Evaluation

In this section, we will show which threats and risks relevant to the architecture have been considered in the design and how our architecture protects against potential attackers. Further, we describe how the authentication and validation procedures of the identity-based signature (IBS) and Crypto-based Identifiers (CBIDs) reduce to known hard problems in computer science.

8.1.1 Attacker Model

The following attack model describes the abilities of a potential adversaries, that are considered in during further security analysis and under which our proposed architecture is secure. We consider an attacker unable to do the following:

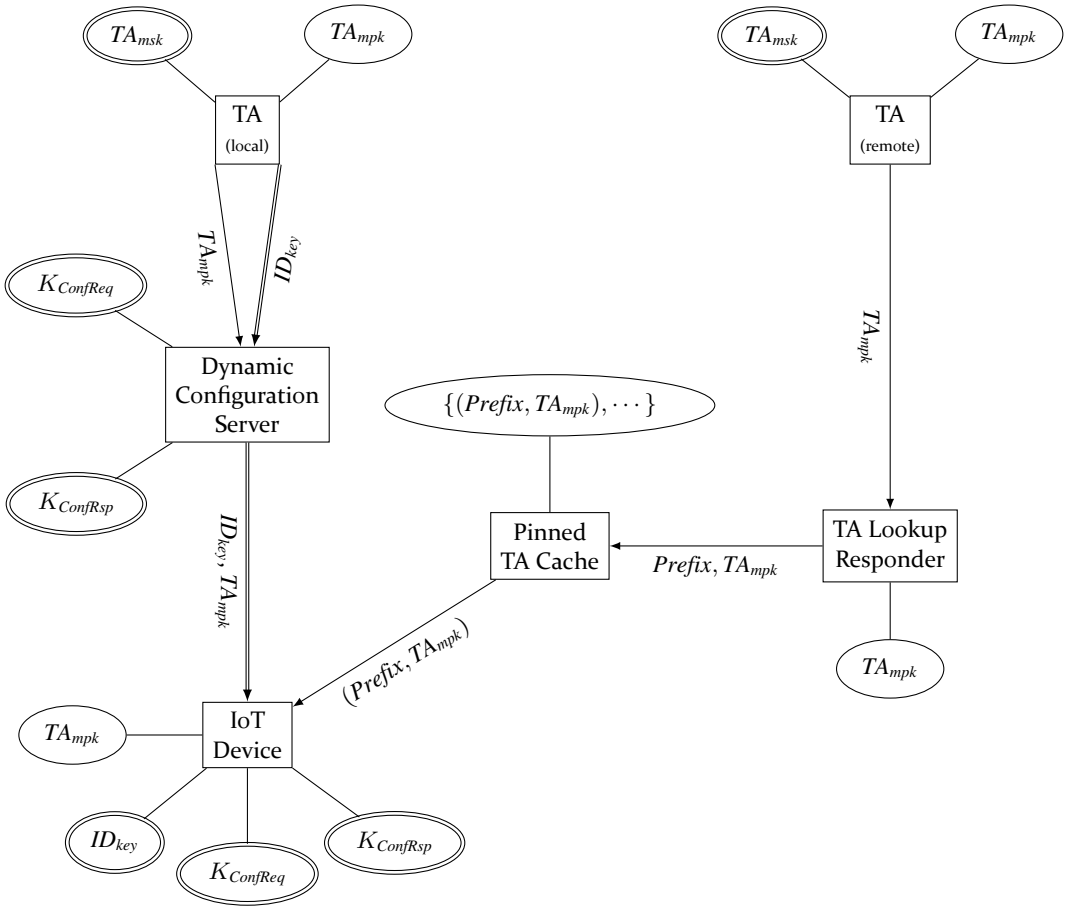
- Break into the device carrying the trusted authority (TA) master private key. This can be achieved by running the TA on a *hardened security device*, e.g. a hardware security module (HSM).
- Break the symmetric Authenticated Encryption with Associated Data (AEAD) cipher used for dynamic device configuration.
- Build algorithms for solving the elliptic curve discrete logarithm problem (ECDLP) more efficient than Pollard's rho [28].

In contrast, an attacker is assumed to have the following abilities:

- Compromise the security of Internet of Things (IoT) devices to gain access to their private identity keys.
- Inject and modify messages within the local IoT networks and the path between the two federated IoT networks.
- Monitoring global traffic containing authenticated messages and local traffic within the wireless network interconnecting constrained IoT networks.

8.1.2 Threat Model

To gain an overview on possible threats to the security of our federated end-to-end authentication architecture for the Internet of Things (IoT), we start by identifying the data



Note: Private security relevant data is double circled and public data single circled. Communication of private data uses double lined arrows and of public data simple arrows.

Figure 8.1: Data flow graph of security critical secret and public data between a local IoT network (left half) and a remote IoT network (right half).

critical to the security of the system, its flow between the different system components, and how an attacker can gain access to security critical data.

Figure 8.1 shows the security relevant data flow between the components of our architecture during the *system initialization*, *dynamic initialization* and *trusted authority (TA) lookup*. Based on the security data and its flow, we extract the security assets and describe their access limitation with regard to other system entities and possible attackers.

TA_{msk} The *master secret key* is the most valuable asset in an administrative IoT network in the system architecture. With access to the TA_{msk}, you can generate new and trusted identity keys for the devices in the IoT network. Due to the inherent *key escrow* problem in identity-based cryptography (IBC) systems, access to the TA_{msk} also allows you to generate identity keys for identities the TA already issued identity

keys to.

Thus, access to it needs to be strongly protected and only the TA is allowed access, during the *dynamic device initialization* phase.

TA_{mpk} Everybody is allowed access to the *master public key* of the TA. On its own, the TA_{mpk} is not required to be authenticated. However, except for signing messages, the TA_{mpk} is used together with the 112 bit network prefix associated to it. This association is lightly protected by the embedded hash in the prefix as shown in figure 5.2 and strongly protected by the *pinning TA lookup cache*.

ID_{key} Only the device with the *ID* to which the ID_{key} is bound is allowed access to it. During the *dynamic device configuration* phase, the *dynamic configuration server* also gains short time access to it. For maximum security it securely clears the key out of memory as soon as it is securely delivered to the IoT device.

$K_{ConfReq}, K_{ConfRsp}$ The symmetric key-pair used for *dynamic device initialization* is also private information that needs to protected from everybody else except the IoT device originally deployed with them and the *dynamic configuration server*.

$(Prefix, TA_{mpk})$ The association of the public system parameters of a TA and the prefix belonging to it are public information accessible to everybody. However, by default this association is only protected by the 64 bit embedded in the subnet prefix.

$\{(Prefix, TA_{mpk}), \cdots\}$ The is the list of pinned associations between a 112 bit prefix and corresponding TA system parameters, i.e. TA_{mpk}. While being public information, it is required to protect this list strongly against malicious modification. Only the *pinning TA lookup cache* is allowed to modify it and only add an association, if the prefix is not present yet in the cache and the TA system parameters have been validated against the embedded hash of the prefix.

This list of assets shows that the TA with its TA_{msk} is most valuable and compromising its secrecy would have disastrous consequences. To provide a more complete overview, we continue our threat analysis by describing the risk of failing to protect each security asset, with the attacker ability that gives access, and the consequences of compromising the access restrictions.

TA_{msk} Compromising the secrecy of the *master secret key* is impossible in our described *attacker model*. However, to show how valuable the *master secret key* is, we explain the consequences for failing to protect it.

An attacker gaining access to the TA_{msk} is able to generate new identity keys and reproduce identity keys which have already been issued. With access to the secret identity key can also be seen as an *identity theft* of the identity associated to an identity key.

With access to the identity key, an attack can produce valid signatures for the corresponding identity. The ability to forge messages for an identity automatically results

in total loss of non-repudiation for the signatures of that identity, because other users, apart from the legitimate user, can also be the signer.

To minimize this risk, it is recommended to either deploy the TA on a *hardened security device* like a hardware security module (HSM), or use an offline TA. An offline TA complicates *dynamic device initialization* and TA rollover, and would require some modifications to the security architecture which are out of scope of this work.

TA_{mpk} This asset already has widest access permission as public information. The only risk, is the extraction of the corresponding TA_{msk} based on only the TA_{mpk}. The TA_{mpk} is generated by elliptic curve cryptography (ECC) scalar multiplication of the ECC base point point with TA_{msk}, as described in section 3.2.1. Calculating the factor TA_{msk} from TA_{mpk} requires solving the elliptic curve discrete logarithm problem (ECDLP), which is considered a hard mathematical problem (see section 2.4).

ID_{key} Failure to protect the identity key against physical attackers on the constrained IoT device or the *dynamic configuration server*, results in *identity theft* of the identity that is bound to the ID_{key}. An attacker with access to ID_{key} can forge signatures for arbitrary messages, for the specific identity.

If noticed, the compromised identity needs to be revoked as described in section 5.4.7 to limit the possibilities of the attacker and its damage. This is done by devices sending authenticated *revocation messages* to the *authentication support server*, which can be deployed on the *border gateway*.

The *border gateway* can modify firewall rules to limit communication to the local network. As soon as enough revocations are collected, the *border gateway* initiates a TA rollover as described in section 5.4.8. During this process devices are authenticated again using their current identities and are issued new identity keys. Revoked devices will not receive a new key.

After this process is done and no legitimate device is left using the old TA, the old TA and the associated network can be discarded. Thereby the previously revoke devices have been excluded from the new network and the revocation list can start fresh.

$K_{ConfReq}, K_{ConfRsp}$ The consequence of compromising the secrecy of the symmetric key-pair used for *dynamic device initialization* depends on the time when access to them is gained. If they are recovered after the original device has already been dynamically configured and the *dynamic configuration server* has cleared the keys from its memory, the attacker can not use them to request a valid identity key. However, if recovered before the original device has been dynamically configured, the attacker can request an identity key from the *dynamic configuration server* instead, leaving the original device without any way to successfully configure within the network.

This risk can be mitigated, by binding the symmetric key-pair to the layer 2 address of the true device.

$(Prefix, TA_{mpk})$ An attacker that can inject arbitrary network packets on the path between a *TA lookup client* and a *TA lookup responder* is able to suppress the response. A sufficiently powerful attacker can generate a colliding TA, i.e. a TA which system parameters resulting in the same embedded hash, for the targeted subnet prefix if allowed sufficient time. If successful, it can mislead the *TA lookup client* to bind the colliding TA parameters to the prefix, resulting in a Denial of Service (DoS) for the original TA and the associated network. The authenticated messages send by devices from the original network are invalidated as devices in the receiving network use falsely pinned TA parameters to validate them.

This risk of this attack can be limited by reducing the amount of TA parameter lookups from devices. Increasing the local TA cache on constrained devices and utilizing gateway-based TA parameter lookup mitigates this risk effectively.

$\{(Prefix, TA_{mpk}), \cdots\}$ Strong security and trust is enabled by the *pinning TA lookup cache*. Failing to protect its pinning store, i.e. the list of network prefix and TA system parameters associations, results in attackers being able to introduce falsely generated TAs for subnets, essentially preventing communication to the original networks behind the subnets.

Identity theft is the major threat in our system architecture. We address this by providing a secure mechanism to report accusations on malicious devices and the secure rollover to a new TA, excluding revoked devices. Bhargav-Spantzel *et al.* [95] discuss the problem of *identity theft* specifically for federated security systems.

8.1.3 Security Proof

In order to proof the security of our federated end-to-end authentication architecture, we need show that only the true devices that originally received an identity key from their respective trusted authoritys (TAs) are able to generate valid signatures that are verified as correct by the device in the opposite network.

The security of the architecture requires two parts of the system to be secure:

1. vBNN-IBS, the identity-based signature (IBS) used to generate and verify asymmetric signatures on messages in our system.

   In [33, § 5], Cao *et al.* prove the security of the IBS used by our architecture, vBNN-IBS, based on the security of BNN-IBS. They proof that if BNN-IBS is *existential unforgeable*, then vBNN-IBS is also existential unforgeable. The former is proven in by Bellare *et al.* in [35].

2. The dynamic device initialization protocol.

   Its security is based on the NORX [75] Authenticated Encryption with Associated Data (AEAD) cipher, a candidate in the current CAESAR[1] competition on AEAD ciphers.

[1] `http://competitions.cr.yp.to/caesar-submissions.html`

3. The remote TA system parameter lookup method.

   During the *system initialization* phase (see section 5.4.3), a TA is generated and the hash value of $H(globalprefix + TA_{mpk})$ is appended to the global IPv6 routing prefix to generate the subnet for Internet of Things (IoT) devices. Remote receivers of TA system parameters, i.e. the TA_{mpk}, can validate the received parameters against the embedded hash.

   In order to send valid messages from subnet S, an adversary in form of a man-in-the-middle (MitM) has to either (a) find the corresponding TA_{msk} to the TA_{mpk} of the subnet or (b) generate a new TA, $A.TA$, with a $A.TA_{mpk}$ that produces the same hash value as TA_{mpk}. We further discuss the feasibility and hardness of these two options:

   a) Finding the *master secret key* based on a specific *master public key* of a TA requires to solve the elliptic curve discrete logarithm problem (ECDLP) which is considered to be hard mathematical problem according current knowledge and research of the cryptographic community. For more details on this subject see section 2.4.

   b) Since the cryptographic hash function used for generating the embedded hashes is preimage resistant, the easiest attack an adversary can run to find a TA, with the hashed TA_{mpk} matching that of the attacked TA, is a brute force attack. The size of the hash value used in our architecture is 64 bit. This means on average, after generating 2^64 random TAs, the adversary finds has a colliding embedded hash with the attacking network.

8.2 Practical Evaluation

We conduct our practical evaluation by deploying the critical parts of our proposed architecture in a test network and executing two evaluation scenarios. We record the network traffic on the border gateway using Wireshark and the logging output of the Internet of Things (IoT) device, the border component and the IoT service.

For a performance evaluation of the asymmetric signature algorithm used in the prototype of our implementation, i.e. vBNN-IBS [33], and the twisted Edwards curve implementation in RELIC, we refer to our recent project report [24].

8.2.1 Evaluation Metrics

The logging output and packet captures recorded during the execution of our evaluation scenarios are analyzed for the following evaluation metrics. In addition, we analyze the ROM image of the Internet of Things (IoT) device.

Protocol overhead The layer 2 protocol 802.15.4 — common in IoT networks — leaves around 100 B for the upper layers. 6LoWPAN takes 2 B to 20 B of the following layer. Sending large packets over 6LoWPAN networks leads to fragmentation which increases the chance of the whole packet getting lost in transmission.

We analyze the message size of the packets send in our two evaluation scenarios described in section 8.2.3, and compare the Concise Binary Object Representation (CBOR) encoded packets to the plain payload contained in them, determining the encoding and security overhead as part of the total protocol overhead of our authentication protocol.

Timing Time is an important factor when it comes to wireless communication between IoT devices. For request and reply scenarios, it is important to reduce the overall minimal time of the communication between the devices. The total time for a request/reply exchange is mainly influenced by the processing speed for each endpoint and the network round-trip time (RTT). After receiving a request message and sending the reply back, the CPU and radio of the device can fall back to low-power modes increase the overall lifetime of battery-based IoT devices.

Time measurements also provide an indirect indication for the energy use of the operations on the IoT device.

Static program size One of the critical constraints of IoT devices is the available ROM/flash storage on the devices. The SAM R21 used in our evaluation has 256 KiB of available flash. IoT devices with 128 KiB to 1,024 KiB are among the list of devices supported by RIOT [2].

Analysis for the static program size and a breakdown in its components give an insight into how much our proposed end-to-end authentication layer contributes to a software IoT stack.

8.2.2 Evaluation Platforms

The evaluate our implementation of the proposed architecture for message sizes and performance, we run the three system components on the platforms described in table 8.1.

We compile the source code of the implementation using *arm-none-eabi-gcc (GNU Tools for ARM Embedded Processors) 4.9.3 20141119* for the Internet of Things (IoT) device, *arm-linux-gnueabihf-g++ (crosstool-NG linaro-1.13.1-4.9-2014.05 - Linaro GCC 4.9-2014.05) 4.9.1 20140505* for the border gateway, and *Apple LLVM version 6.0* for the IoT service.

For the execution of the evaluation scenario, the IoT device uses the RIOT version listed in table 8.1. The border gateway uses *Linux raspberrypi 4.1.2+ #1 PREEMPT Thu Jul 23 16:39:12 CEST 2015 armv6l GNU/Linux* and the IoT service uses *OS X 10.9.5 (13F1077)*.

The Raspberry Pi does not have any on-board 802.15.4 radio as compared to the SAM R21. Instead, connectivity to 6LoWPAN over 802.15.4 is provided by the openlabs Raspberry Pi 802.15.4 radio[3]. This enables a wireless low-power communication channel between the Raspberry Pi and the SAM R21 board.

[2] `https://github.com/RIOT-OS/RIOT/wiki/RIOT-Platforms`
[3] `http://openlabs.co/store/Raspberry-Pi-802.15.4-radio`

		IoT Device	Border Gateway	IoT Service
Model	Device	SAM R21 [96]	Raspberry Pi (2011/Model B)	MacBook Pro
	Vendor	Atmel	Raspberry Pi Foundation	Apple
Processor	Architecture	ARM	ARM	Intel
	CPU	Cortex-M0+	ARM1176	Core i7
	Clock speed	48 MHz	800 MHz	2 GHz
	Word size	32 bit	32 bit	64 bit
	CPU cores	1	1	4
	L1 cache	N/A	32 KiB	64 KiB per core
	L2 cache	N/A	(256 KiB)	256 KiB per core
	L3 cache	N/A	N/A	6 MiB shared
Memory	ROM	256 KiB	N/A	N/A
	RAM	32 KiB	512 MiB	8 GiB
Network	802.15.4 radio	On-board	Add-on module	No
	Ethernet	No	Yes	Yes

Note: The L2 Cache is used by the GPU on the Raspberry Pi and therefore is not available to the CPU.

Table 8.1: Overview of the platforms used for the practical evaluation of our proposed security architecture.

8.2.3 Evaluation Scenarios

In the two scenario we aim to evaluate the performance, in particular the computation time and message size overhead, and behavior of the three implemented components in a local test network. Both scenarios will cover initializing all components, the dynamic device initialization of the Internet of Things (IoT) device and the IoT service sending a query to the IoT device and it responding with an authenticated reply, as described in section 5.4.5.

For the first scenario, the deployment is configured to use client-based authority lookup. The border gateway will only support dynamic device initialization and forwarding network traffic to the IoT end-points The second scenario will employ gateway-based authority lookup.

We use the same approach for evaluating both scenarios. The network capturing tool Wireshark[4] is used on the Raspberry Pi to capture all network traffic, including the Ethernet link between the Raspberry Pi and the laptop, and the network link between the Raspberry Pi and the SAM R21 IoT device before and after the 6LoWPAN conversion.

In addition, time-stamped log messages have been inserted into interesting code parts, to deduce the duration for relevant cryptographic and management computations. While log messages can be cached, delayed and have the overhead of formatting and terminal output, they provide a good estimate on the computational complexity of the parts of the evaluation. A detailed and more precise evaluation of the identity-based signature (IBS) operations can be found in our recent project report [24].

From the Wireshark network captures and the log messages of all three programs, we produce message communication flow and computation diagrams, shown in figure 8.2, figure 8.3 and figure 8.4.

Note that the blocks for the computations are not proportional to the duration of the computation, to save vertical space. In addition, we leave out Concise Binary Object Representation (CBOR) encoding and decoding, as it takes minimal time on all devices and would further increase the size of the diagrams.

Client-based Authority Lookup Scenario

The diagram shown in figure 8.2 displays the major computations and all network communication for a fresh authenticated message exchange between an IoT service deployed on a laptop and a constrained shows the runtimes, delay and communication between the three components. It is reproduced from Wireshark captures and the log messages of all three programs. In following, we will go into detail and analyze the diagram for the first two evaluation metrics, namely *message size* and *timing*.

Message	Content	Security	CBOR	Total
Dynamic configuration request	3 B	ciphertext: 20 B nonce: 16 B	3 B	39 B
Dynamic configuration response	ID: 16 B TA_{mpk}: 33 B ID_{key}: 65 B	16 B	20 B	150 B
TA lookup request	3 B	0 B	1 B	4 B
TA lookup response	33 B	0 B	2 B	35 B
Authenticated request	3 B	100 B	17 B	117 B
Authenticated response	10 B	100 B	17 B	124 B

Table 8.2: Overview of the communication messages and their overhead distribution.

[4]See `https://www.wireshark.org/` for more information on Wireshark.

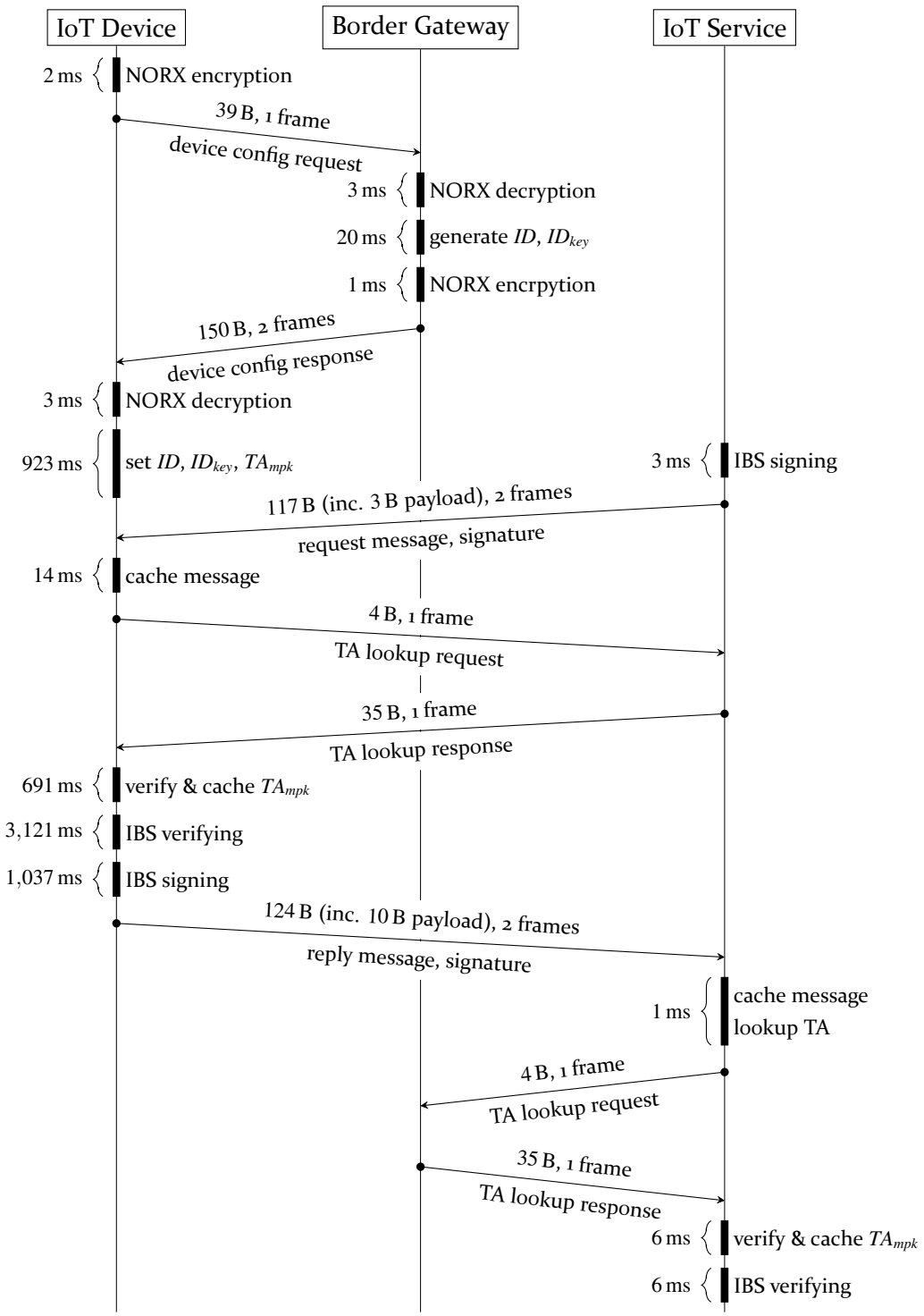


Figure 8.2: Communication and computation sequence diagram for the client-based authority lookup scenario.

Protocol overhead Using the communication flow diagram shown in figure 8.2 , the log messages of our components, and the Wireshark network traffic captures during the run of the evaluation scenario, we compiled an overview on the overhead of the security and CBOR [94] encoding used by the proposed protocol. This overview is shown in table 8.2.

The dynamic configuration request message is build by encrypting the string "REQ" with a random nonce and the predefined symmetric keys $K_{ConfReq}$. The reply is encrypted using the predefined key $K_{ConfRsp}$. Due to NORX being a block cipher, the plaintext is expanded from 3 B to a multiple of the block size, resulting in 20 B ciphertext. The CBOR encoding adds little overhead and the resulting request message totals at 39 B and fits in one 6LoWPAN frame.

The corresponding response however requires two 6LoWPAN frames to transmit back to the IoT device. Its total size of 150 B is mainly caused by the amount of cryptographic information that is transmitted back to the device. This includes not only public information like *ID* and TA_{mpk}, but also the secret ID_{key}. Together these keys and identity constitute 114 B to the reply. Additionally, due to use of more verbose CBOR primitives, i.e. maps with byte string keys, the CBOR overhead increased highly compared to the request.

In comparison, the trusted authority (TA) lookup request and reply are compact. The request, simply made up from a CBOR encoded fixed string "TAL", only needs 4 B. The response consists of a compress elliptic curve cryptography (ECC) point, encoded as CBOR.

CBOR maps are also used for encoding the authenticated request and authenticated response. The large size of the response and request messages are due to the security overhead of 100 B, the vBNN-IBS signature. The storage complexity of the vBNN-IBS, explained at the end of section 3.2.1, is one ECC point and two 32 B integers.

Timing Due to the fact that RIOT [93] currently does not support low energy modes for the CPU of the SAM R21 board, the CPU runs at full speed at all times and the runtime measured is proportional to the energy consumption.

The runtime for the Authenticated Encryption with Associated Data (AEAD) cipher is minimal with only a couple milliseconds. This confirms, that NORX [75] is very suitable for constrained devices in the IoT.

The computational cost of the vBNN-IBS verification is roughly three times higher than the vBNN-IBS signing operation. This is the same ratio we have seen in our more detailed practical evaluation of IBS mechanisms [32] and our evaluation of twisted Edwards curves [24]. In addition, this timing difference is also explained by the theoretical computational complexity of the vBNN-IBS algorithm, shown at the end of section 3.2.1.

Generating an ID and the corresponding private identity key has roughly the same computational complexity as the vBNN-IBS signing operation. However, the Raspberry Pi is more powerful resulting only in a 20 ms runtime. Border gateways are commonly less energy constrained and are connected to the power grid or high energy batteries.

For small messages, e.g. 3 B and 10 B payload, our authentication protocol with IBS shows an overhead of 100 B. This already requires two 6LoWPAN frames to transmit. Due to our use of IBS, and the possibility to use IPv6 addresses as ID and public key, no addition public keys or certificates need to be transfered, provided that the TA is already known. In contrast, traditional certificate-based authentication does have smaller signatures but also has the overhead of certificate transmission for the public key and its binding to an identity. According to Gupta *et al.* [97], certificates exchanged during the Datagram Transport Layer Security (DTLS) handshake require at least 220 B for storage. This is more than double of our security overhead for an authenticated message.

Considering that asymmetric cryptography is primarily used for securely negotiating a shared symmetric secret key to use with symmetric cryptography — the same protocol that transport layer security (TLS) [90] and DTLS [46] follow — messages authenticated using asymmetric cryptography, e.g. IBS are exchanged rarely between two endpoints.

Overall, vBNN-IBS accounts for the majority of the overall computational overhead of our architecture and presents a key architectural component to optimize.

Gateway-based Authority Lookup Scenario

The gateway-based authority lookup scenario aims at evaluating the performance, and demonstrating the advantage, of a deployment scenario where the border gateway is used to support the authentication process. It is composed of two authenticated message requests from the IoT service to the constrained IoT device.

The computations times for each operation are similar to the times measured in the previous evaluation. They are not discussed further here. Therefore, we will focus on the procedure of this evaluation scenario and its specialties.

Figure 8.3 shows the message exchanges and computations after dynamic device configuration up to the end of the verification of the first message. At this time, the TA lookup cache in the IoT service and in the IoT border gateway are still cold, i.e. they do not contain any mappings from subnet prefix to TA system parameters yet.

After the IoT service signs the initial query message, it sends it to the remote IoT device. On its way, it passes through the IoT border gateway. Here it is intercepted using *ip6tables* firewall and passed along to the IoT border gateway program described in section 7.2.2.

The IoT gateway tries to lookup the corresponding TA system parameters for the received query message in its pinning TA system parameter cache. It then caches the original query message and starts the TA system parameter lookup procedure, because the cache is empty at the beginning. After receiving the TA system parameters from the remote *TA lookup*

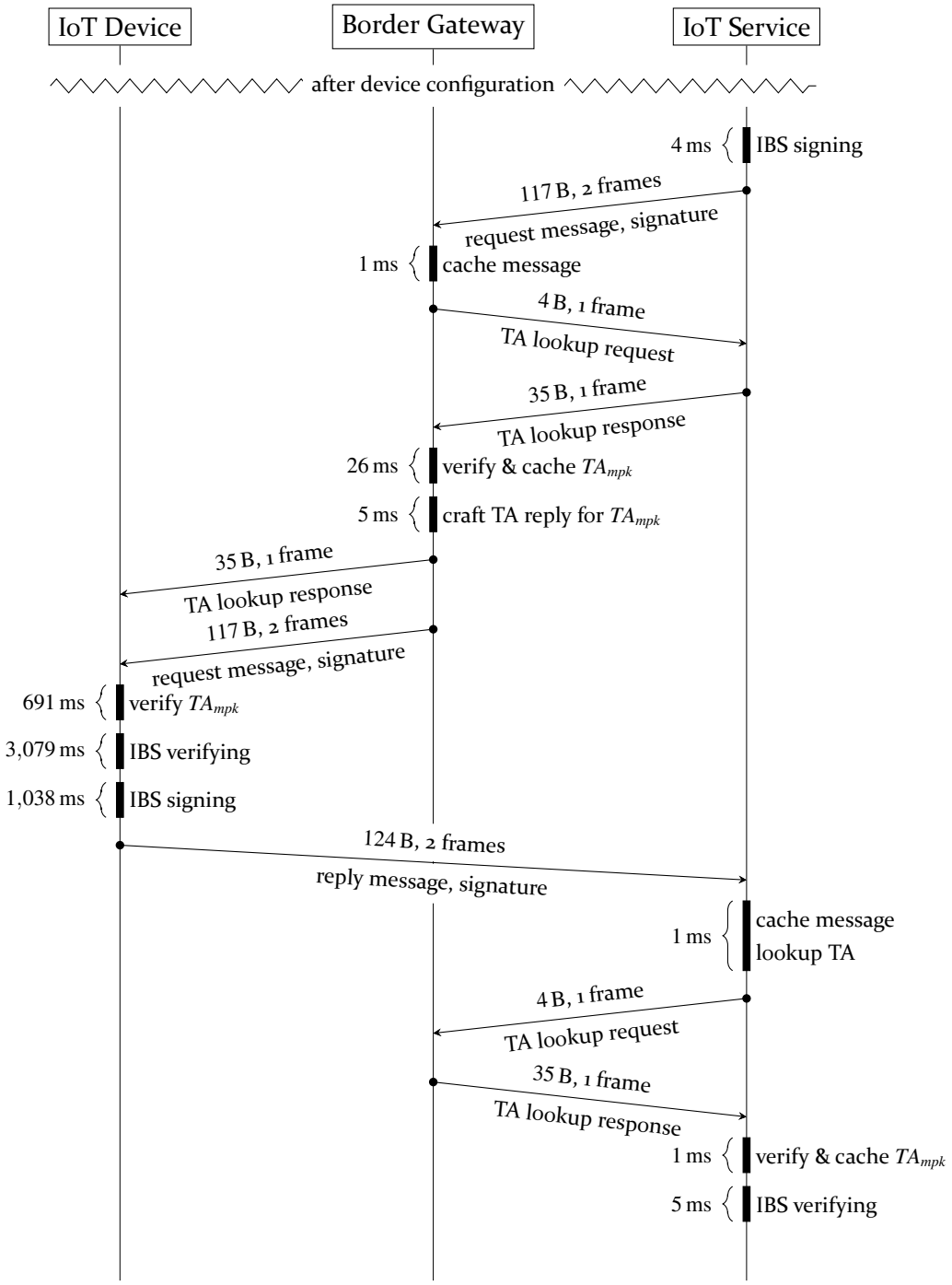


Figure 8.3: Communication and computation sequence diagram for the gateway-based authority lookup scenario after successful device configuration with cold caches.

responder, the border gateway verifies the received system parameters against the 64 bit hash which is embedded in the IPv6 address.

If the verification succeeds, the TA system parameters are stored in the cache with the associated 112 bit subnet prefix. This essentially pins the TA system parameters to the prefix and protects against potential man-in-the-middle (MitM) attacks, which have computed a colliding TA with the same 112 bit subnet prefix. Afterwards a TA lookup response is crafted by the border gateway and it, and the original message to be forwarded, are send to the destination address of the initial query message.

In case the validation fails, both the original message and the TA lookup response from the remote network are discarded.

The following reception of the TA lookup parameters, the original query message and the verification of the end-to-end IBS signature attached to it, works the same as described in the client-based authority lookup evaluation scenario. The signed reply message is send back to the IoT service via the IoT border gateway, which simply forwards outgoing messages in the current implementation. Reception, TA lookup and verification of signature by the IoT service are handled like in the first evaluation scenario.

Directly afterwards, another authenticated query is send by the IoT service to the constrained IoT device. The computations and communications for this continuation of the gateway-based authority lookup scenario are shown in figure 8.4.

At this point the pinning TA cache in the border gateway and the IoT service are warm, as they contain the subnet prefix/TA parameter mapping from the previous request. This means that after receiving the second authenticated query message from the IoT service, it can directly lookup the TA system parameters, corresponding to the IPv6 subnet prefix of the message, in the pinning TA system parameter cache. With that a TA lookup response can be synthesized and send to the IoT device, followed by the original message. The lookup and synthesizing takes minimal time, no longer than 1 ms.

Afterwards, the IoT does the verification, reply generation and signing computations and sends the reply back. The receiving IoT service can directly verify it, as the TA system parameters are still cached from the last query/response exchange.

This scenario shows the advantage of caching TA parameters on the border gateway. In deployment scenarios, where an IoT device needs to verify requests from many networks in the Internet, the border gateway can cache and pin the TA system parameters required for verification. A scenario like this, is the sharing of sensor devices which provide data interesting to different parties [58].

This caching is possible, because the border gateway has more memory available compared to the constrained IoT device. For example, our current implementation for the IoT device can only store a single master public key from a remote TA due to the constraints on the runtime memory.

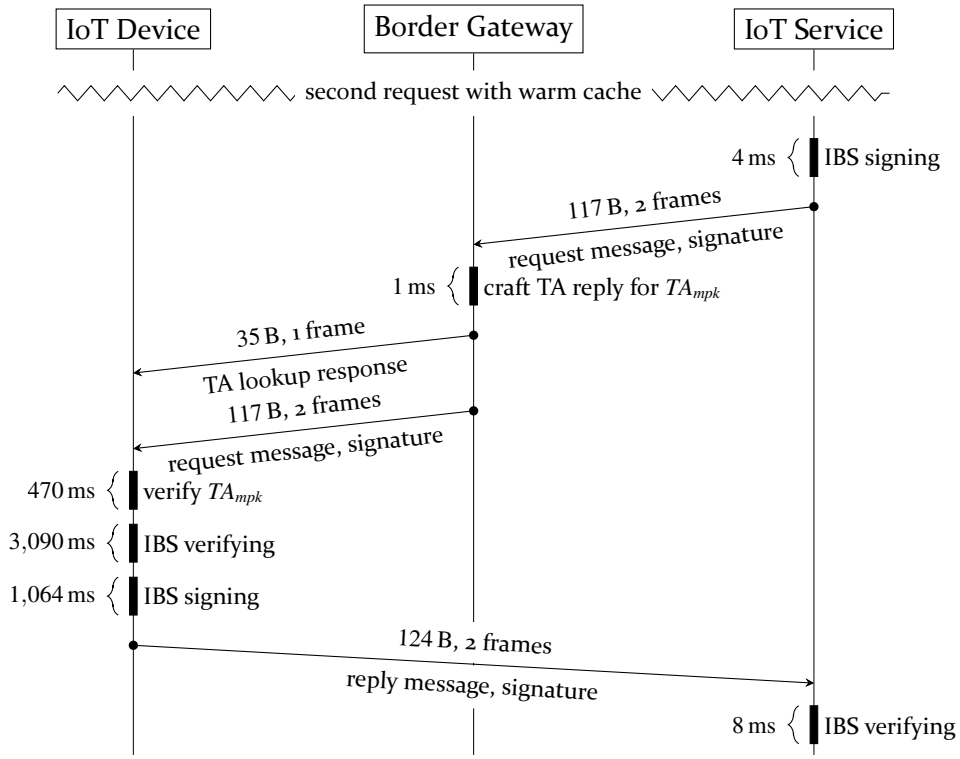


Figure 8.4: Communication and computation sequence diagram for the gateway-based authority lookup scenario after successful device configuration with warm caches.

8.2.4 Program Size Evaluation

ROM is a critical resource for constrained Internet of Things (IoT) devices, as it is highly limited compared to flash sizes of other devices on the Internet like smart phones, embedded computers like desktop computers. Due to this special property of constrained IoT devices it is important to keep the program size small enough with respect to the available ROM of a device.

To gain an insight over the distribution of the program code size across the various components, the link map, the cross reference table and information about removed dead code are analyzed. This information is provided by the GNU `ld` linker program with the help of the following program flags: `-Map=program.map -cref -print-gc-sections`. Finally, this output is analyzed by a Python script which counts symbol sizes per application component.

Table 8.3 shows the breakdown of the program image which is flashed on the ROM of the SAM R21 board, our IoT evaluation platform.

RIOT accounts for 53.5 % of the overall image size. This includes code and data for multi-threading, 802.15.4 radio driver, network stack, virtual timers, Concise Binary Object Representation (CBOR) encoding/decoding and more. The runtime for the C programming language, which covers low-level functions that emulate required instruction missing on

Component	Size	Share
RIOT	56,123 B	53.5 %
RELIC	30,072 B	28.6 %
Application	10,082 B	9.6 %
C runtime	6,833 B	6.5 %
NORX	1,864 B	1.8 %
Total	104,974 B	100 %

Table 8.3: Program size distribution across program components.

the target CPU and functions from the C standard library, accounts for 6.5 % of the total size.

Our application, including its specific cryptographic dependencies RELIC and NORX, takes up the remaining space of the code image. RELIC, accounts for 28.6 % of the total image size. RELIC includes and implementation of vBNN-IBS and the lower level cryptographic primitives required by it. This are implementations for finite prime fields, twisted Edwards curves, cryptographic hash functions and random number generation.

Similar ROM requirements for RELIC have been measured by Oliveira *et al.* [58, p. 392], who evaluated the memory overhead of a RELIC-based secure web service on 8 bit AVR and 16 bit MSP platforms. However, compared to the Datagram Transport Layer Security (DTLS)-based security and two-way authentication proposal for the IoT by Kothmayr *et al.* [56, p. 2717], our implementation requires nearly double the ROM as their implementation for a Cortex-M3 platform. They count 10,838 B of ROM for the cryptography and 67,315 B for the total ROM image. Instead of RELIC [23] and RIOT [93], they use CyaSSL and TinyOS.

The Authenticated Encryption with Associated Data (AEAD) cipher, NORX, fits in less than 2 kB. This shows its suitability to provide a high-level of authenticity, integrity and confidentiality for communication with constrained devices in the IoT. The application logic for handling the roles of the IoT sensing node listed in section 7.2.1 account for the remaining 10,082 B.

Bormann *et al.* [43, § 3] define three classes of constrained IoT devices. The most constrained class describes devices with a ROM size smaller than 100 KiB and the least constrained class devices with a ROM size of about 250 KiB. With a total of 104,974 B for our application it just fits on class 1 devices, with have about 100 KiB of ROM available. Noting that while our application is optimized for size by the compiler, our code base has not been highly optimized for code size. Both our application code and RIOT include conditional error checks with debug output, which comes with a non-negligible code overhead. It is conceivable that with further optimization for code size, our IoT application will easily fit on the middle and least constrained classes of devices.

9 Summary

In this final chapter we will close our work by drawing conclusion from our implementation and the insights from the evaluation, followed by an outlook into future work.

9.1 Conclusion

We have presented a new system architecture for federated end-to-end authentication in the Internet of Things (IoT). It is build from identity-based signature (IBS), which allow to eliminate public key management and key distribution overhead known from classic public-key cryptography (PKC), leading to a smaller size of authenticated messages.

With the proposed system, independently administrated IoT networks in the IPv6 Internet can exchange identity-based cryptography (IBC) authenticated messages. These messages can be verified by a federated authentication process which is secured by embedding cryptographic information in the IPv6 network prefix, an idea similar to Cryptographically Generated Addressess (CGAs) and Crypto-based Identifiers (CBIDs). The proposed architecture includes means to deal with compromised devices in the network and provides a mitigation strategy using revocation combined with a secure rollover of the trusted authority (TA).

Our system implementation uses an IBS based on modern elliptic curve cryptography (ECC), specifically twisted Edwards curves. This allows the usually computationally complex IBS to be used on constrained IoT devices which are have limited physical resources, e.g. memory, processing speed and energy.

We further discussed the interoperability of our proposed architecture and protocol with existing standards of the Internet and the IoT, specifically standard IPv6 addressing and 6LoWPAN. The proposed system architecture and procedures are proven to be secure under a specified attacker model.

Our implementation and practical evaluation on a constrained IoT device, i.e. the SAM R21 controller, demonstrates the implementability of our system architecture and its suitability for the constraints of the IoT. Utilizing additional resources on border gateways to support the authentication process, enables good scalability regarding the number of participating IoT networks and devices for the federated end-to-end authentication architecture.

9.2 Outlook

A popular application layer protocol specifically with designed for constrained Internet of Things (IoT) devices is Constrained Application Protocol (CoAP) [45, 98]. CoAP recommends to use Datagram Transport Layer Security (DTLS) to provide transport security and this combination has been focus for optimization for constrained devices [99]. Therefore, an implementation of the integration of our secure federated end-to-end authentication architecture with DTLS, as described in section 6.1.4, is an ideal continuation of this work.

In section 5.4.7 and section 5.4.8 we described the revocation of compromised and malicious devices and the resulting trusted authority (TA) rollover as a countermeasure. While these procedures are part of the specification and description of our distributed system architecture for federated end-to-end authentication, they have not been implemented and practically evaluated.

The detection of malicious behavior and compromised devices is non-trivial. However, it an important topic for unmaintained and constrained IoT devices and deserves further evaluation and analysis.

We concentrated on limited communication scenarios during our evaluation of the proposed architecture. To provide a more sound validation of the proposal, an evaluation on the public IPv6 Internet with more intercommunicating devices from different networks would be ideal.

This would also provide an insight in the cache performance of the gateway-based authentication strategy.

Further, our implementations of twisted Edwards curves for the C library RELIC could benefit from algorithm and implementation improvements. Currently, it is not protected against side-channel attacks on the prime field and elliptic curve cryptography (ECC) layers. The abstractions it uses to allow flexibility and easy implementation of new features come with an associated performance penalty.

There is room for improvements in the area of performance and side-channel security. Compared to the state-of-the-art, side-channel protected, implementation of Curve25519 in assembler by Düll *et al.* [100], our twisted Edwards curve implementation requires five times the computations for the same ECC scalar multiplication operation on equal hardware.

Bibliography

[1] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, Amsterdam, Netherlands: Elsevier, 2013.

[2] S. Sicari, A. Rizzardi, L. A. Grieco, and A. Coen-Porisini, "Security, privacy and trust in Internet of Things: The road ahead," *Computer Networks*, vol. 76, pp. 146–164, Amsterdam, Netherlands: Elsevier, 2015.

[3] S. L. Keoh, S. S. Kumar, and H. Tschofenig, "Securing the Internet of Things: A Standardization Perspective," *Internet of Things Journal*, vol. 1, no. 3, pp. 265–275, Piscataway, NJ, USA: IEEE, Jun. 2014.

[4] R. Hummen, J. H. Ziegeldorf, H. Shafagh, S. Raza, and K. Wehrle, "Towards Viable Certificate-based Authentication for the Internet of Things," in *Proceedings of the 2nd ACM Workshop on Hot Topics on Wireless Network Security and Privacy*, ser. HotWiSec '13, Budapest, Hungary: ACM, 2013, pp. 37–42.

[5] A. Shamir, "Identity-Based Cryptosystems and Signature Schemes," in *Advances in Cryptology — CRYPTO 1984*, ser. Lecture Notes in Computer Science, G. R. Blakley and D. Chaum, Eds., vol. 196, Santa Barbara, California, USA: Springer, Aug. 1985, pp. 47–53.

[6] V. S. Miller, "Use of Elliptic Curves in Cryptography," in *Advances in Cryptology — CRYPTO 1985*, ser. Lecture Notes in Computer Science, H. C. Williams, Ed., vol. 218, Berlin, Heidelberg, Germany: Springer Berlin Heidelberg, 1986, pp. 417–426.

[7] N. Koblitz, "Elliptic Curve Cryptosystems," *Mathematics of Computation*, vol. 48, no. 177, pp. 203–209, Providence, RI, USA: American Mathematical Society, 1987.

[8] R. L. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-key Cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, New York, NY, USA: ACM, Feb. 1978.

[9] NIST, "Digital Signature Standard," Federal Information Processing Standards 186–4, Jul. 2013.

[10] D. J. Bernstein, P. Birkner, M. Joye, T. Lange, and C. Peters, "Twisted Edwards Curves," in *Progress in Cryptology — AFRICACRYPT 2008*, ser. Lecture Notes in Computer Science, S. Vaudenay, Ed., vol. 5023, Berlin, Heidelberg, Germany: Springer, 2008, pp. 389–405.

[11] N. P. Smart, "A Comparison of Different Finite Fields for Elliptic Curve Cryptosystems," *Computers & Mathematics with Applications*, vol. 42, no. 1–2, pp. 91–100, Amsterdam, The Netherlands: Elsevier, 2001.

[12] A. Menezes, T. Okamoto, and S. Vanstone, "Reducing Elliptic Curve Logarithms to Logarithms in a Finite Field," *Information Theory, IEEE Transactions on*, vol. 39, no. 5, pp. 1639–1646, Piscataway, NJ, USA: IEEE, Sep. 1993.

[13] P. Gaudry, F. Hess, and N. P. Smart, "Constructive and Destructive Facets of Weil Descent on Elliptic Curves," *Journal of Cryptology*, vol. 15, no. 1, pp. 19–46, Berlin, Heidelberg, Germany: Springer, 2002.

[14] H. M. Edwards, "A normal form for elliptic curves," *Bulletin of the American Mathematical Society*, vol. 44, no. 3, pp. 393–422, Providence, RI, USA: American Mathematical Society, 2007.

[15] D. J. Bernstein and T. Lange, "Faster Addition and Doubling on Elliptic Curves," in *Advances in Cryptology — ASIACRYPT 2007*, ser. Lecture Notes in Computer Science, K. Kurosawa, Ed., vol. 4833, Berlin, Heidelberg, Germany: Springer, 2007, pp. 29–50.

[16] H. Cohen, G. Frey, R. Avanzi, C. Doche, T. Lange, K. Nguyen, and F. Vercauteren, *Handbook of Elliptic and Hyperelliptic Curve Cryptography*, H. Cohen, G. Frey, and C. Doche, Eds., ser. Discrete Mathematics and Its Applications. Abingdon, United Kingdom: Taylor & Francis, 2005.

[17] N. Benger, J. van de Pol, N. P. Smart, and Y. Yarom, ""Ooh Aah... Just a Little Bit": A Small Amount of Side Channel Can Go a Long Way," in *Cryptographic Hardware and Embedded Systems — CHES 2014*, ser. Lecture Notes in Computer Science, L. Batina and M. Robshaw, Eds., vol. 8731, Berlin, Heidelberg, Germany: Springer-Verlag, 2014, pp. 75–92.

[18] E. Barker, D. Johnson, and M. Smid, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography," National Institute of Standards and Technology, Gaithersburg, MD, USA, Tech. Rep. NIST Special Publication 800-56A, Mar. 2007. [Online]. Available: `http://csrc.nist.gov/publications/nistpubs/800-56A/SP800-56A_Revision1_Mar08-2007.pdf`.

[19] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed. Cambridge, Massachusetts, USA: MIT Press, 2009.

[20] J. H. Silverman, "The Arithmetic of Elliptic Curves," *Graduate Texts in Mathematics*, vol. 106, Berlin, Heidelberg, Germany: Springer Berlin Heidelberg, 1986.

[21] H. Cohen, A. Miyaji, and T. Ono, "Efficient Elliptic Curve Exponentiation Using Mixed Coordinates," in *Advances in Cryptology — ASIACRYPT 1998*, ser. Lecture Notes in Computer Science, K. Ohta and D. Pei, Eds., vol. 1514, Berlin, Heidelberg, Germany: Springer, 1998, pp. 51–65.

[22] H. Hisil, K. K.-H. Wong, G. Carter, and E. Dawson, "Twisted Edwards Curves Revisited," in *Advances in Cryptology — ASIACRYPT 2008*, ser. Lecture Notes in Computer Science, J. Pieprzyk, Ed., vol. 5350, Berlin, Heidelberg, Germany: Springer, 2008, pp. 326–343.

[23] D. F. Aranha and C. P. L. Gouvêa, *RELIC is an Efficient LIbrary for Cryptography*, https://github.com/relic-toolkit/relic.

[24] T. Markmann, "Modern Elliptic Curve Cryptography for Constrained Devices," Project Report, May 2015. [Online]. Available: http://inet.cpt.haw-hamburg.de/teaching/ws-2014-15/project-class/tobias-markmann-modern-elliptic-curve-cryptography-for-constrained-devices.

[25] Y. Yarom and K. Falkner, *Flush+Reload: a High Resolution, Low Noise, L3 Cache Side-Channel Attack*, Cryptology ePrint Archive, Report 2013/448, 2013. [Online]. Available: http://eprint.iacr.org/2013/448.

[26] Y. Yarom and N. Benger, *Recovering OpenSSL ECDSA Nonces Using the FLUSH+RELOAD Cache Side-channel Attack*, Cryptology ePrint Archive, Report 2014/140, 2014. [Online]. Available: http://eprint.iacr.org/2014/140.

[27] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*, 1st. Boca Raton, Florida, USA: CRC Press, 1996.

[28] J. M. Pollard, "Monte Carlo methods for index computation $\pmod{p}$," *Mathematics of Computation*, vol. 32, no. 143, pp. 918–924, Providence, RI, USA: American Mathematical Society, Jul. 1978.

[29] ECRYPT II, "ECRYPT II Yearly Report on Algorithms and Keysizes," European Network of Excellence in Cryptology II, Tech. Rep., Sep. 2012, http://www.ecrypt.eu.org/documents/D.SPA.20.pdf.

[30] D. Boneh and M. Franklin, "Identity-Based Encryption from the Weil Pairing," in *Advances in Cryptology — CRYPTO 2001*, ser. Lecture Notes in Computer Science, J. Kilian, Ed., vol. 2139, Berlin, Heidelberg, Germany: Springer, 2001, pp. 213–229.

[31] A. Joux, "A One Round Protocol for Tripartite Diffie—Hellman," in *Algorithmic Number Theory*, ser. Lecture Notes in Computer Science, W. Bosma, Ed., vol. 1838, Berlin, Heidelberg, Germany: Springer, 2000, pp. 385–393.

[32] T. Markmann, "Performance Analysis of Identity-based Signatures," Project Report, Aug. 2014. [Online]. Available: http://inet.cpt.haw-hamburg.de/teaching/ss-2014/master-projekt/tobias_markmann_prj1.pdf.

[33] X. Cao, W. Kou, L. Dang, and B. Zhao, "IMBAS: Identity-based multi-user broadcast authentication in wireless sensor networks," *Computer Communications*, vol. 31, no. 4, pp. 659–667, Amsterdam, Netherlands: Elsevier, 2008.

[34] R. Tso, C. Gu, T. Okamoto, and E. Okamoto, "Efficient ID-Based Digital Signatures with Message Recovery," in *Cryptology and Network Security*, ser. Lecture Notes in Computer Science, F. Bao, S. Ling, T. Okamoto, H. Wang, and C. Xing, Eds., vol. 4856, Berlin, Heidelberg, Germany: Springer, 2007, pp. 47–59.

[35] M. Bellare, C. Namprempre, and G. Neven, "Security Proofs for Identity-Based Identification and Signature Schemes," in *Advances in Cryptology — EUROCRYPT 2004*, ser. Lecture Notes in Computer Science, C. Cachin and J. L. Camenisch, Eds., vol. 3027, Berlin, Heidelberg, Germany: Springer, 2004, pp. 268–286.

[36] M. Blaze, "Key Escrow from a Safe Distance: Looking Back at the Clipper Chip," in *Proceedings of the 27th Annual Computer Security Applications Conference*, ser. ACSAC '11, Orlando, Florida, USA: ACM, 2011, pp. 317–321.

[37] H. Abelson, R. Anderson, S. M. Bellovin, J. Benalob, M. Blaze, W. Diffie, J. Gilmore, P. G. Neumann, R. L. Rivest, J. I. Schiller, and B. Schneier, "The Risks of Key Recovery, Key Escrow, and Trusted Third-party Encryption," *Columbia University Academic Commons*, New York, NY, USA: Columbia University, 1997. [Online]. Available: `http://academiccommons.columbia.edu/catalog/ac:127127`.

[38] A. Kate and I. Goldberg, "Distributed Private-Key Generators for Identity-Based Cryptography," in *Security and Cryptography for Networks*, ser. Lecture Notes in Computer Science, J. A. Garay and R. Prisco, Eds., vol. 6280, Berlin, Heidelberg, Germany: Springer, 2010, pp. 436–453.

[39] S. S. Al-Riyami and K. G. Paterson, "Certificateless Public Key Cryptography," in *Advances in Cryptology — ASIACRYPT 2003*, ser. Lecture Notes in Computer Science, C.-S. Laih, Ed., vol. 2894, Berlin, Heidelberg, Germany: Springer, 2003, pp. 452–473.

[40] Common Vulnerabilities and Exposures, *CVE-2014-0160*, `http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-0160`, 2014.

[41] B. B. Brumley and N. Tuveri, "Remote Timing Attacks Are Still Practical," in *Computer Security — ESORICS 2011*, ser. Lecture Notes in Computer Science, V. Atluri and C. Diaz, Eds., vol. 6879, Berlin, Heidelberg, Germany: Springer, 2011, pp. 355–371.

[42] K. Hoeper and G. Gong, "Key Revocation for Identity-Based Schemes in Mobile Ad Hoc Networks," in *Ad-Hoc, Mobile, and Wireless Networks*, ser. Lecture Notes in Computer Science, T. Kunz and S. Ravi, Eds., vol. 4104, Berlin, Heidelberg, Germany: Springer, 2006, pp. 224–237.

[43] C. Bormann, M. Ersue, and A. Keranen, "Terminology for Constrained-Node Networks," IETF, RFC 7228, May 2014.

[44] IEEE 802.15 Working Group, "IEEE Standard for Local and metropolitan area networks—Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs)," IEEE, New York, NY, USA, Tech. Rep. IEEE Std 802.15.4™–2011, Sep. 2011.

[45] Z. Shelby, K. Hartke, and C. Bormann, "The Constrained Application Protocol (CoAP)," IETF, RFC 7252, Jun. 2014.

[46] E. Rescorla and N. Modadugu, "Datagram Transport Layer Security Version 1.2," IETF, RFC 6347, Jan. 2012.

[47] D. Evans, "The Internet of Things: How the Next Evolution of the Internet is Changing Everything," Cisco, White Paper, Apr. 2011. [Online]. Available: `https://www.cisco.com/web/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf`.

[48] S. Deering and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification," IETF, RFC 2460, Dec. 1998.

[49] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks," IETF, RFC 4944, Sep. 2007.

[50] J. Hui and P. Thubert, "Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks," IETF, RFC 6282, Sep. 2011.

[51] J. Postel, "User Datagram Protocol," IETF, RFC 768, Aug. 1980.

[52] J. Olsson, "6LoWPAN demystified," Texas Instruments, Dallas, Texas, USA, Tech. Rep., Oct. 2014. [Online]. Available: `http://www.ti.com/lit/wp/swry013/swry013.pdf`.

[53] R. Mzid, M. Boujelben, H. Youssef, and M. Abid, "Adapting TLS Handshake Protocol for Heterogenous IP-Based WSN using Identity Based Cryptography," in *2010 International Conference on Communication in Wireless Environments and Ubiquitous Systems: New Challenges (ICWUS)*, Piscataway, NJ, USA: IEEE, Oct. 2010, pp. 1–8.

[54] M. Boujelben, O. Cheikhrouhou, H. Youssef, and M. Abid, "A Pairing Identity based Key Management Protocol for Heterogeneous Wireless Sensor Networks," in *N2S '09. International Conference on Network and Service Security*, Piscataway, NJ, USA: IEEE, Jun. 2009, pp. 1–5.

[55] T. Kothmayr, C. Schmitt, W. Hu, M. Brünig, and G. Carle, "A DTLS based end-to-end security architecture for the Internet of Things with two-way authentication," in *2012 IEEE 37th Conference on Local Computer Networks Workshops (LCN Workshops)*, Clearwater, FL, USA: IEEE, Oct. 2012, pp. 956–963.

[56] ——, "DTLS based security and two-way authentication for the Internet of Things," *Ad Hoc Networks*, vol. 11, no. 8, pp. 2710–2723, Amsterdam, Netherlands: Elsevier, 2013.

[57] E. Rescorla and N. Modadugu, "Datagram Transport Layer Security," IETF, RFC 4347, Apr. 2006.

[58] L. B. Oliveira, A. Kansal, C. P. Gouvêa, D. F. Aranha, J. López, B. Priyantha, M. Goraczko, and F. Zhao, "Secure-TWS: Authenticating Node to Multi-user Communication in Shared Sensor Networks," *The Computer Journal*, vol. 55, no. 4, J. Hu, Ed., pp. 384–396, Oxford, UK: Oxford University Press, 2012.

[59] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. E. Culler, "SPINS: Security Protocols for Sensor Networks," *Wireless Networks*, vol. 8, no. 5, pp. 521–534, Berlin, Heidelberg, Germany: Springer, Sep. 2002.

[60] D. Johnson, A. Menezes, and S. Vanstone, "The Elliptic Curve Digital Signature Algorithm (ECDSA)," *International Journal of Information Security*, vol. 1, no. 1, pp. 36–63, Berlin, Heidelberg, Germany: Springer, 2001.

[61] D. Boneh, B. Lynn, and H. Shacham, "Short Signatures from the Weil Pairing," in *Advances in Cryptology — ASIACRYPT 2001*, ser. Lecture Notes in Computer Science, C. Boyd, Ed., vol. 2248, Berlin, Heidelberg, Germany: Springer, 2001, pp. 514–532.

[62] F. Zhang, R. Safavi-Naini, and W. Susilo, "An Efficient Signature Scheme from Bilinear Pairings and Its Applications," in *Public Key Cryptography — PKC 2004*, ser. Lecture Notes in Computer Science, F. Bao, R. Deng, and J. Zhou, Eds., vol. 2947, Berlin, Heidelberg, Germany: Springer-Verlag, 2004, pp. 277–290. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-24632-9_20.

[63] T. Markmann, T. C. Schmidt, and M. Wählisch, "Federated End-to-End Authentication for the Constrained Internet of Things using IBC and ECC," in *Proceedings of ACM SIGCOMM 2015, Poster Session*, London, UK: ACM, Aug. 2015, pp. 603–604.

[64] C. Castelluccia and G. Montenegro, "Statistically Unique and Cryptographically Verifiable (SUCV) Identifiers and Addresses," in *9th Annual Network and Distributed System Security Symposium (NDSS'02)*, Internet Society, Feb. 2002, pp. 87–99.

[65] G. Montenegro and C. Castelluccia, "Crypto-based Identifiers (CBIDs): Concepts and Applications," *ACM Trans. Inf. Syst. Secur.*, vol. 7, no. 1, pp. 97–127, New York, NY, USA: ACM, 2004.

[66] T. Aura, "Cryptographically Generated Addresses (CGA)," IETF, RFC 3972, Mar. 2005.

[67] C. Evans, C. Palmer, and R. Sleevi, "Public Key Pinning Extension for HTTP," IETF, RFC 7469, Apr. 2015.

[68] National Institute of Standards and Technology, "FIPS 180–3, Secure Hash Standard, Federal Information Processing Standard (FIPS), Publication 180-3," Department of Commerce, Gaithersburg, MD, USA, Tech. Rep., Oct. 2008.

[69] J.-P. Aumasson, S. Neves, Z. Wilcox-O'Hearn, and C. Winnerlein, "BLAKE2: Simpler, Smaller, Fast as MD5," in *Applied Cryptography and Network Security*, ser. Lecture Notes in Computer Science, M. Jacobson, M. Locasto, P. Mohassel, and R. Safavi-Naini, Eds., vol. 7954, Berlin, Heidelberg, Germany: Springer, 2013, pp. 119–135.

[70] M.-J. Saarinen and J.-P. Aumasson, "The BLAKE2 Cryptographic Hash and MAC," IETF, Internet-Draft – work in progress 04, Jun. 2015.

[71] P. Rogaway, "Authenticated-encryption with Associated-data," in *Proceedings of the 9th ACM Conference on Computer and Communications Security*, ser. CCS '02, Washington, DC, USA: ACM, 2002, pp. 98–107.

[72] D. McGrew, "An Interface and Algorithms for Authenticated Encryption," IETF, RFC 5116, Jan. 2008.

[73] J. Salowey, A. Choudhury, and D. McGrew, "AES Galois Counter Mode (GCM) Cipher Suites for TLS," IETF, RFC 5288, Aug. 2008.

[74] Y. Nir and A. Langley, "ChaCha20 and Poly1305 for IETF Protocols," IETF, RFC 7539, May 2015.

[75] J.-P. Aumasson, P. Jovanovic, and S. Neves, "NORX: Parallel and Scalable AEAD," in *Computer Security — ESORICS 2014*, ser. Lecture Notes in Computer Science, M. Kutyłowski and J. Vaidya, Eds., vol. 8713, Berlin, Heidelberg, Germany: Springer, 2014, pp. 19–36.

[76] ——, "NORX8 and NORX16: Authenticated Encryption for Low-End Systems," in *TRUDEVICE 2015 — 3rd Workshop on Trustworthy Manufacturing and Utilization of Secure Devices*, Grenoble, France, Mar. 2015. [Online]. Available: http://cryptomaths.com/data/papers/2015-AJN-norx8-and-norx16.pdf.

[77] ——, *NORX*, Version 1.1, Specification, Jul. 2014. [Online]. Available: https://norx.io/data/norx.pdf.

[78] J. Birr-Pixton, *Benchmarking Modern Authenticated Encryption on 1€ Devices*, Jun. 2015. [Online]. Available: http://jbp.io/2015/06/01/modern-authenticated-encryption-for-1-euro/.

[79] C. Schridde, M. Smith, and B. Freisleben, "An Identity-Based Key Agreement Protocol for the Network Layer," in *Security and Cryptography for Networks*, ser. Lecture Notes in Computer Science, R. Ostrovsky, R. De Prisco, and I. Visconti, Eds., vol. 5229, Berlin, Heidelberg, Germany: Springer, Sep. 2008, pp. 409–422.

[80] R. Droms, "Dynamic Host Configuration Protocol," IETF, RFC 2131, Mar. 1997.

[81] S. Thomson, T. Narten, and T. Jinmei, "IPv6 Stateless Address Autoconfiguration," IETF, RFC 4862, Sep. 2007.

[82] J. Jeong, "IPv6 Host Configuration of DNS Server Information Approaches," IETF, RFC 4339, Feb. 2006.

[83] R. Hinden and S. Deering, "IP Version 6 Addressing Architecture," IETF, RFC 4291, Feb. 2006.

[84] D. E. 3rd, J. Schiller, and S. Crocker, "Randomness Requirements for Security," IETF, RFC 4086, Jun. 2005.

[85] C. Bao, C. Huitema, M. Bagnulo, M. Boucadair, and X. Li, "IPv6 Addressing of IPv4/IPv6 Translators," IETF, RFC 6052, Oct. 2010.

[86] M. Bagnulo, P. Matthews, and I. van Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers," IETF, RFC 6146, Apr. 2011.

[87] S. Jiang, D. Guo, and B. Carpenter, "An Incremental Carrier-Grade NAT (CGN) for IPv6 Transition," IETF, RFC 6264, Jun. 2011.

[88] A. Durand, R. Droms, J. Woodyatt, and Y. Lee, "Dual-Stack Lite Broadband Deployments Following IPv4 Exhaustion," IETF, RFC 6333, Aug. 2011.

[89] A. Banks and R. Gupta, "MQTT Version 3.1.1," Organization for the Advancement of Structured Information Standards (OASIS), OASIS Standard, Oct. 2014. [Online]. Available: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html.

[90] T. Dierks and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2," IETF, RFC 5246, Aug. 2008.

[91] 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," IETF, RFC 5280, May 2008.

[92] H. K. H. So, S. H. M. Kwok, E. Y. Lam, and K.-S. Lui, "Zero-Configuration Identity-Based Signcryption Scheme for Smart Grid," in *First IEEE International Conference on Smart Grid Communications (SmartGridComm) 2010*, Gaithersburg, MD, USA: IEEE, Oct. 2010, pp. 321–326.

[93] E. Baccelli, O. Hahm, M. Günes, M. Wählisch, and T. C. Schmidt, "RIOT OS: Towards an OS for the Internet of Things," in *Proc. of the 32nd IEEE INFOCOM. Poster*, Piscataway, NJ, USA: IEEE Press, 2013.

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

[95] A. Bhargav-Spantzel, A. C. Squicciarini, and E. Bertino, "Establishing and Protecting Digital Identity in Federation Systems," in *Proceedings of the 2005 Workshop on Digital Identity Management*, ser. DIM '05, Fairfax, VA, USA: ACM, Oct. 2005, pp. 11–19.

[96] Atmel, *Atmel SAM R21 — SMART ARM-Based Wireless Microcontroller*, Datasheet, 2015. [Online]. Available: `http://www.atmel.com/images/atmel-42223-sam-r21_datasheet.pdf`.

[97] V. Gupta, M. Wurm, Y. Zhu, M. Millard, S. Fung, N. Gura, H. Eberle, and S. C. Shantz, "Sizzle: A standards-based end-to-end security architecture for the embedded Internet," *Pervasive and Mobile Computing*, vol. 1, no. 4, pp. 425–445, Amsterdam, Netherlands: Elsevier, Dec. 2005, Special Issue on PerCom 2005.

[98] C. Bormann, A. P. Castellani, and Z. Shelby, "CoAP: An Application Protocol for Billions of Tiny Internet Nodes," *IEEE Internet Computing*, vol. 16, no. 2, pp. 62–67, Los Alamitos, CA, USA: IEEE, 2012.

[99] S. Raza, H. Shafagh, K. Hewage, R. Hummen, and T. Voigt, "Lithe: Lightweight Secure CoAP for the Internet of Things," *Sensors Journal*, vol. 13, no. 10, pp. 3711–3720, Piscataway, NJ, USA: IEEE, Oct. 2013.

[100] M. Düll, B. Haase, G. Hinterwälder, M. Hutter, C. Paar, A. Sánchez, and P. Schwabe, "High-speed Curve25519 on 8-bit, 16-bit, and 32-bit microcontrollers," *Designs, Codes and Cryptography*, pp. 1–22, Berlin, Heidelberg, Germany: Springer, 2015.

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, August 26, 2015 Tobias Markmann