INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Masterarbeit

Identity-based source authentication in constrained networks

Sophia Grundner-Culemann

INSTITUT FÜR INFORMATIK
DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Masterarbeit

Identity-based source authentication in constrained networks

Sophia Grundner-Culemann

Aufgabensteller:	Prof. Dr. Dieter Kranzlmüller
Betreuer:	Tobias Guggemos Dr. Nils Otto vor dem gentschen Felde
Abgabetermin:	30. Oktober 2017

Hiermit versichere ich, dass ich die vorliegende Masterarbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den 30. Oktober 2017

. . . .

(Unterschrift des Kandidaten)

Abstract

Resource constrained devices communicating among themselves and exchanging data (e.g. in automated homes) are becoming ever more prevalent. Unfortunately, so are reports of them having security flaws. This is at least in part due to the difficulty that arise from the strict limitations on power, memory, and energy use the items are subject to. Classical methods often rely on computationally challenging methods or large keys for security, especially in Public Key Cryptography. However, to identify individual devices in a network symmetric keys as used to ensure confidentiality within the group are not sufficient. Adi Shamir's proposal to use the identity itself as public key (identity-based cryptography) is therefore a useful idea in constrained environments. The implications of using it in group communication are hardly addressed in the literature, however. Especially the issue of key revocation deserves close attention. In order to evaluate the suitability of identity-based signatures in a constrained group setting, the thesis proposes a taxonomy that allows the comparison of different schemes with varying characteristics. To facilitate its use, a comprehensible introduction to the most important concepts related to identity-based signatures is provided, along with some guidance on details of an IBS scheme beyond the taxonomy's reach.

Contents

1. Introduction

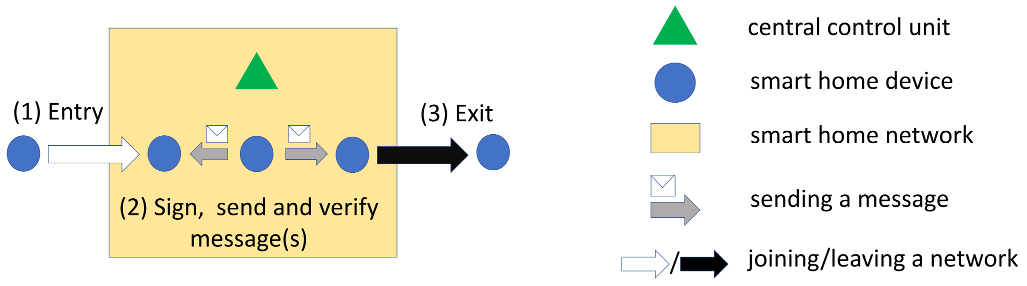
In October of 2016, a DDos attack made headlines by taking down several web services like Netflix, Twitter, PayPal and others over the course of a day [37]. Throughout the media [52], security experts were quoted describing the attack as based on the Internet of Things (IoT for short). This term was coined for devices whose primary function lies outside the computerized world which are nevertheless connected to the internet. This includes everyday objects like toasters, TVs, thermostats, security cameras etc., even Christmas Trees and slippers. They can, for example, be found in smart homes, public infrastructure or industrial environments. In spite of their increasingly extended use [10], many IoT devices are poorly secured and can easily be compromised, even when the security of a home may depend on a wirelessly connected alarm system, for example. Allegedly, the aforementioned DDoS attack exploited those weaknesses [52]. Even more dangerous than single devices being unsafe is a badly secured communicating IoT-group. In this scenario, a successful attacker may gain control over many items at once by targeting only the weakest, rendering the other devices' security useless. The lack of proper security measures is not (only) due to the ignorance or carelessness of manufacturers and end-users, but also lies in the difficulty to subject those needs to the devices' constraints. Often, IoT items need to be very light, cheap, and energy-efficient all at once. Thus, there are two somewhat contrary aspects to be considered: On the one hand, the computation and the result need to be very frugal in terms of the needed space, power, and energy. IoT devices often depend on a battery for their power. To maximize their life span, the reduction of energy cost for computing and transferring packages is important. On the other hand, communicating in a group requires an extra high level of security, since compromising one device means compromising many devices at once. Therefore, the benefit of breaking the counter-attack measures may be higher than for the communication of just two entities.

To lessen the benefit of breaking into a single device, individual source authentication is important for the group communication. If only the group membership but not the specific source of a message can be verified, all senders must be treated equally though they might not be equally safe. With individual authentication, roles and authorization can be managed in the group [50].

1.1. Scenario

The usecase employed in this thesis is a so-called smart home, a term coined for households in which everyday appliances are automated or controlled via a digital network and can even communicate with or control each other. Often there is a central management instance involved, e.g. to control access to the system from outside. For example, a smart front door lock may be remotely controlled by a central unit and interact with lighting and heating devices in the house to automatically set up a comfortable environment upon the arrival of an inhabitant. This constitutes a group of devices communicating with each other. It can be extended (e.g. to include a surveillance camera) or reduced (e.g. if devices break or

Figure 1.1.: The basic events in a smart home network as discussed in this thesis



are no longer needed). An attacker might try to connect her own device with the group to gain access to the home, masquerading as a new device or impersonating an existing door lock, for example. It is therefore important to detect undesired messages by putting an authentication system in place that allows all legitimate members to prove their identity to the group.

The following basic events for smart homes (depicted in figure 1.1) will be discussed in this thesis: (1) A new device entering an existing group (for example to add a new feature in the smart home), (2) signing and verification of a message within the group for authentication (e.g. when devices share data with each other), (3) the exit of a device from the system (e.g. to be replaced with a newer instance), as well as the setup of a group and its dissolution.

Upon joining the group, a device is equipped with means to create a credible signature that authenticates them to be the sender of the corresponding message, and with the ability to prove themselves as legitimate members of the group.

Signing a message on behalf of another member should be impossible without the consent of the member itself, even if several devices cooperate. When leaving the group, it has to be ensured that the former participant can no longer credibly pose as a group member. The creation and verification of signatures may still be possible, but all legitimate recipients must be clear that the device's membership in the group has expired.

Similar scenarios can be described in the context of computerized cars, ad-hoc networks for military operations, or the so-called Industry 4.0.

1.2. Prerequisites

The scenario rests on three main pillars: There are constrained devices, who form a dynamic group and want to communicate securely. Those prerequisites might have conflicting implications, so the following analysis is meant to shine light on the requirements the model will have to meet.

1.2.1. Constrained devices

With respect to security, there are three central aspects to be considered for constrained devices: the disk storage available to store information on the group, keys, and algorithms, the computing power (specifically processors and memory) that can be used for calculation of information, and the energy supply.

Low storage capacity has implications for the cryptography in the model, because security can usually be increased with key size - but with little space to retain them, it becomes more important to provide high security with relatively short keys. It also restricts the possibilities in the model for handling other members' data, because devices can not be expected to retain much (if any) information on their peers.

The computing power limits the ability of the device to perform elaborate operations and algorithms with large parameters. Even simple computations like RSA [40] can therefore pose a challenge if the memory is too small to handle the necessary information.

The energy supply becomes the bottleneck for communication among the group because all transmission is very costly - sending 1 bit corresponds to the execution of 10, sometimes up to 100 CPU instructions in terms of energy cost [21]. But smart devices need to be very light and cheap, and often depend on batteries for power. This heavily restricts the model, because it implies that for the most part, communication should be replaced by computation, unless the latter is so complex that the device is unable to perform it or significantly profits from delegating it.

1.2.2. Dynamic groups

In this thesis, a group is understood to be a set of devices where each one is uniquely identified by some identity ID. The focus lies on dynamic groups, as described in the scenario. To coordinate entries and exits within them, the role of a group manager needs to be introduced. They oversee admittance, revoke membership, and keep track of the devices' changeable membership status. As devices join and leave the group, other participants need to be informed if a sender is a rightful member. Also, if a public key was included for authentication (as it will be in the model, following the reasoning of subsection 1.2.3), they need to verify that the key does indeed belong to the sender. The possibilities to achieve both goals are threefold: Either the recipient checks with the group manager and queries the sender's status and public key; or it is regularly notified with respect to the current members and their credentials in order to store the information herself. The third option is a mathematical connection between the sender's identity and their corresponding key pair, as is the idea of Identity-Based Cryptography ([44], 3). This way, any recipient can verify the legality and validity of the signature on their own and with mathematical certainty. It is important to note, however, that in this case any entity able to compute a valid key pair for one identity can do so for any identity. No device can therefore be given the power to compute their own credentials alone. It is necessary that some information unknown to any individual group member (master secret) be included in the computation of all key pairs. The secret can either be distributed among all members, who then need to cooperate to compute any and all keys. Or, a trusted third party acts as key escrow (i.e. an entity different from the sender who legitimately knows the sender's secret keys): As such, that party alone knows the master secret and is able to compute valid key pairs. Because the group is dynamic, key revocation is an important matter as well. It can not happen without a change of the master secret, because apart from that factor, the validity of keys is modeled as a mathematical truth. Hence, the secret information is subject to change, and with it every identity-based key pair. In other words: While identities remain fixed, the corresponding keys may vary. It is also noteworthy that group membership is implicitly proven by the signature - the public key depends on the membership in the group, no outsider can acquire a valid key pair. It would not be possible to verify a signature with the sender's public key using the master public key if the sender has not received a private key dependent on the master secret.

1.2.3. Secure group communication

For secure communication, authenticity is an important building block. Successfully enforcing confidentiality for a message implies its authenticity to a certain degree, because in a non-corrupted scenario a message could only have been correctly encrypted by someone who shares the secret (in the symmetric case) or is in possession of the private key associated to the public key that was officially recognized to belong to the alleged sender. However, using symmetric keys does not authenticate individual participants in group communication, and asymmetrically encrypting a message with the sender's private key does not provide confidentiality, so additional encryption is necessary in any case. Integrity is usually achieved when authenticity is established by signing the message, because tampering with the message would cause the signature-verification to fail as well. Non-repudiation is given in a public key crypto-system, though a trusted third party that acts as key escrow weakens this notion. A symmetric setting does not warrant non-repudiation, since the recipient can forge the signature, and for the same reason only authenticates the sender as one of the parties in possession of the shared secret.

As laid out in the scenario, the thesis focuses on dynamic groups where membership can change. The goal is individual authentication within the group instead of group authentication. That means signatures should be specific to individual members of the group and not only prove that the sender is a member of the group.

Approaching this task with symmetric keys means introducing a separate key for every communication relation in the group, which in the worst case equals the number of distinct pairs of members in the group, $\sum_{k=1}^{n-1} k = \frac{(n-1)*(n-2)}{2}$. Only in small groups or for very few communication partners it is possible to store this amount of keys on a constrained device, let alone negotiate them. Additionally, they come at the cost of a diminished sense of authentication - after all, a third party can not be convinced of a signature's authenticity, as either communication partner could have produced it.

Instead, the model will employ asymmetric keys to individually authenticate members. Thus, only one set of keys is necessary for each member of the group. Asymmetric keys necessitate the introduction of a trusted third party (TTP). It is tasked with independently confirming the link between a group member and their key pair, since there is no other way for a recipient to ensure that the public key does indeed belong to the alleged sender (seeing as neither of them are human in our scenario, so checking in person is not an option).

1.2.4. Summary

To provide maximum security for dynamic groups with limited resources, the following requirements are proposed for an authentication model: All devices must be equipped with a unique identity. To facilitate authentication, the model should employ asymmetric keys. In order to minimize the storage and energy cost, Identity-Based Signatures are chosen for the scheme. This approach implicitly ensures that authentic senders are authorized to communicate in the group. It also necessitates the existence of a master secret, which can be distributed among the members or kept with a third party performing key escrow for the group. Since the group is dynamic, the master secret must be replaceable, which means the key pairs are exchangeable as well and usually have a relatively short life span. To manage entries and exits, the role of a group manager needs to be introduced. Also, a trusted third party is required to make public key cryptography possible, as well as the trusted handling of the master secret.

The authentication system that is thereby created should meet the following demands:

- Allow for a sender's public key to be derived from some individually identifying information that was included anyway.
- Prove that the sender is a member of the group.
- Allow for the group communication to rely on symmetric keys for confidentiality.
- At the same time, employ asymmetric keys to individually authenticate the sender of a message.
- Prevent outsiders (including former members) from creating new valid signatures and verifying signatures created after their membership expiration.
- Prevent outsiders (including former members) from credibly claiming to be a group member.

The demands are understood to be met within the scope of control provided by a signature scheme. For example, safe storage of private information so that it can not be used by unauthorized parties is fundamental to authentication, but lies outside of what an IBS scheme offers for security. To define the boundaries of what it needs to cover, several assumptions are made for the thesis:

unique ID It is assumed that all group members are equipped with an identity and that it is unique.

legitimate ID It is assumed that some form of group management exists that can reliably check whether an applicant is allowed to join the group and prevent illegitimate applicants from contacting the TTP.

trust in TTP It is assumed that the Trusted Third Party is benign and that all devices have established a trustful relationship with it.

secure channel It is assumed that there exists a separate secure channel with every device such that the Trusted Third party can communicate confidential information to each of them individually.

secure storage of sensitive information It is assumed that no private information is lost before it expires and that it is not accessible to outsiders or other group members.

safety of TTP and member devices It is assumed that the TTP and the group members are available at any given time and the communication is reliable.

1.3. Objective

Many existing authentication schemes are not suited for this specific task, since they either lack efficiency (due to large keys or inefficient computations) or work with symmetric keys which do not prove individual authorship (unless they are negotiated separately for each 1:1-conversation, which in general is inefficient as well).

As mentioned in section 1.2.3, identity-based signatures (the idea that a public key should be directly derivable from the identity they belong to) seem to be especially useful in constrained networks. This thesis will therefore provide an analysis of the concepts related to IBS and their impact on whether a certain scheme is applicable in a constrained context. Furthermore, a comprehensive taxonomy will be developed to judge and compare IBS schemes by several criteria including computational expense, the amount of information that needs to be transferred at certain stages, and known attacks. Several IBS-schemes will be compared using this tool. As proof of concept, one scheme is implemented in a group setting and evaluated with respect to the demands given above. The framework Charm [1] provides a solid basis for cryptographic programming and contain some of the schemes and concepts considered in this thesis. A relatively secure authentication process can therefore be established and tested in a constrained test bed.

1.4. Outline

The thesis will be structured as follows: First, the theoretical background on groups, mathematically hard problems, Elliptic Curve Cryptography and pairings is given. Chapter 3 introduces the basics of IBS and related research. It also presents six identity-based signature-schemes and their respective characteristics. In the next chapter, a comprehensive taxonomy to judge the suitability of a scheme for use in constrained networks is developed. All exemplary schemes are evaluated with respect to the taxonomy and it is used to select the best option for the smart home scenario presented above. The following chapter presents the quantitative and formal results of implementing the second best scheme by the taxonomy rating. Chapter 6 concludes and discusses future prospects.

2. Theoretical foundation

To understand the IBS schemes presented in this thesis, a fair amount of mathematical concepts needs to be discussed. First, an introduction to groups and cyclic groups is provided. Next, the notion of hard problems is introduced and all relevant problems for this thesis are defined. Section 2.3.3 gives an overview on elliptic curves and their use in cryptography, followed by an introduction to pairings and pairing-based cryptography in section 2.4.1.

2.1. Groups

All definitions comply with [35].

Definition 2.1 (Group)**.** Let G be a set and $\circ : G \times G \to G$ a binary operation. $(G, \circ)$ is a group if, and only if, three properties are fulfilled:

1. There exists an identity element $\mathbb{1}$: $\exists \mathbb{1} \in G$, s.t. $\forall g \in G : \mathbb{1} \circ g = g = g \circ \mathbb{1}$
2. There is an inverse for every element: $\forall g \in G \, \exists h \in G : g \circ h = \mathbb{1} = h \circ g$
3. The operation is associative: $\forall g, h, j \in G : (g \circ h) \circ j = g \circ (h \circ j)$

The **order ord($\mathbb{G}$) of a group** is defined as the number of its elements.
The **order ord(g) of a group element** refers to the smallest number $n \in \mathbb{N}$, s.t. $g^n = \mathbb{1}$. A group element is said to have infinite order if no such number exists.

An example for groups that are commonly used in cryptography are the groups $(\mathbb{Z}_p, +)$. They consist of all numbers $i \in \{0, ..., p-1\}$ and the group operation $+$ is defined as $a + b = (a + b) \bmod p$, where the right side is the usual addition and modulo operation in $\mathbb{Z}$. One can easily check that this group is well-defined and all properties hold. The order of this group is p.

Remark. In this chapter, groups will be denoted multiplicatively, i.e. $g \circ g = g^2$ etc., except for groups defined on elliptic curves (following the standards in the related literature) or if clearly stated otherwise (like for $(\mathbb{Z}_p, +)$).

A special kind of groups are the so called cyclic groups. They have the additional property that there exists at least one element, called **generator**, such that any element of the group can be expressed as a power of that element.

Definition 2.2 (Cyclic group)**.** Let $\mathbb{G} = (G, \circ)$ be a group. $\mathbb{G}$ is called cyclic if, and only if $\exists g \in G \, \forall h \in G \, \exists x \in \mathbb{Z} : g^x = h$. g is called a generator of $\mathbb{G}$ and $\mathbb{G}$ is sometimes denoted as $< g >$.

The group $(\mathbb{Z}_p, +)$ defined above is cyclic for all $p > 1$. There may exist multiple generators. For example, for $(\mathbb{Z}_3, +)$ $(\mathbb{Z}_3 = \{0, 1, 2\})$, both 1 and 2 are generators: $1 + 1 = 2$, $1 + 1 + 1 = 3 = 0 \bmod 3$, so $\mathbb{Z}_3 = \{3 * 1, 1, 2 * 1\} =< 1 >$. (Indeed, 1 is always a generator). Equally, $2 + 2 = 4 = 1 \bmod 3$, $2 + 2 + 2 = 6 = 0 \bmod 3$, so $\mathbb{Z}_3 = \{3 * 2, 2 * 2, 2\} =< 2 >$. 0 is the identity element of $\mathbb{Z}_3$, which can never be a generator.

Fields

Similar to groups, fields are sets with two operations and a few additional properties. A field is finite if it contains only finitely many elements. For example, the integer numbers $\mathbb{Z}$ together with the commonly known operations + and $\cdot$ are a field. So is $(\mathbb{Z}_p, +, \cdot$ if p is prime. A more abstract notation for fields with p elements is $\mathbb{F}_p$.

2.2. Hard problems

A mathematical problem is called **(computationally) hard** (relative to today's non-quantum computers) if there is no known algorithm to solve it in polynomial time or less [1]. Comparatives such as "harder than" or "at least as hard as" refer to whether a solving algorithm for one problem would also solve the other.

Some hard problems are the bases for cryptographic schemes:

2.2.1. Integer factorization

RSA is a cryptosystem (named after Ron Rivest, Adi Shamir, and Leonard Adleman) [41] that uses asymmetric keys and is based on the problem of integer factorization, i.e. finding all prime factors of a given integer.
This problem is most difficult for numbers that are the product of two similar sized primes[2], which is the case for RSA. The most commonly used algorithm for large numbers is the generel numer field sieve, it was used in 2009 to factorize RSA-768, which is the largest RSA-number of the RSA-challenge that has been factorized. [25] Currently it is not known whether this problem has a polynomial solution, so it is hard according to the previously given definition.
At the time of this writing, 3072-bit numbers are recommended for the medium-term secure use of RSA. [47] Due to this large key size, RSA is better suited for non-constrained environments. An IBS scheme based on the difficulty of integer factorization is presented in [44], the original IBS paper.

2.2.2. Discrete logarithm

A discrete logarithm is the smallest number $x \in \mathbb{Z}_p$ such that for $a, m \in \mathbb{N}$ and prime $p \in \mathbb{N} : a^x = m \, mod \, p$. [47] It is defined as the inverse of the discrete exponentiation, which is a so-called one-way function. One-way functions are easily computed but reversed only with great difficulty, i.e. for a given function $f : X \to Y$, $f(x)$ (where $x \in X$) is efficiently computable, but finding $x \in X$ such that $y = f(x)$ for a given $y \in Y$ is difficult. [27]

[1] This is not to say that no such algorithm exists, it just may not have been discovered yet.

[2] These products are commonly called semiprimes.

DLog Given the generator g of a cyclic group and g^a $(a \in \mathbb{Z})$ computing a is known as the Discrete Logarithm (DLog) problem. This problem is hard for suitably chosen groups.

In 1976, Whitfield Diffie and Martin Hellman developed a Key Exchange Protocol allowing for two parties to acquire a shared secret communicating over an insecure channel. It is based on the DLog problem and entails the following procedure: Two parties, Alice and Bob, who want to exchange a shared secret, each choose a secret integer $a \in \mathbb{Z}$ and $b \in \mathbb{Z}$ respectively. For the generator g of a cyclic group they agreed on before (not necessarily over a secure channel), they each compute their part of the secret g^a and g^b, respectively, which they can openly send to the other party. By combining their counterpart's information and their own secret, Bob and Alice will end up with the same key g^{ab} without ever explicitly communicating it: $(g^a)^b = g^{(a*b)} = g^{(b*a)} = (g^b)^a$.

CDH For a cyclic group $\mathbb{G}$ with generator g, the problem of computing g^{ab} given the tuple (g, g^a, g^b) $(a, b \in \mathbb{Z})$ is known as Computational Diffie Hellman problem (CDH). Its hardness relies on that of the DLog problem.

DDH The problem of deciding for a given integer z whether $g^z = g^{ab}$ given (g, g^a, g^b, g^z) (where g is the generator of a cyclic group $\mathbb{G}$) is known as the Decisional Diffie-Hellman problem (DDH).

Lemma 2.3 (Hardness relation between the DLog-based problems)**.** *Given a cyclic group, the DLog-problem on this group is at least as hard as the corresponding CDH-problem, which in turn is at least as hard as the corresponding DDH-problem.*

Proof: Let $\mathbb{G}$ be a cyclic group, let g be a generator of $\mathbb{G}$.

- DLog is at least as hard as CDH: For arbitrary $a \in \mathbb{Z}$, let there be an algorithm $\mathcal{A}$ to solve the DLog-problem in polynomial time,i.e. to compute a given g^a. Then the CDH-problem can be solved in polynomial time by obtaining $a, b \in \mathbb{Z}$ from g^a and g^b using $\mathcal{A}$ and computing g^{ab}. Therefore, the CDH-problem is not harder than the DLog-problem.

- CDH is at least as hard as DDH: For arbitrary $a, b \in \mathbb{Z}$, let there be an algorithm $\mathcal{B}$ to solve the CDH-problem in polynomial time, i.e. compute g^{ab} given g, g^a, and g^b. Then the DDH-problem can be solved in polynomial time by computing g^{ab} from g^a and g^b using $\mathcal{B}$ and comparing it to g^z. Therefore, the DDH-problem is not harder than the CDH-problem.

Remark. Although it is clear that the Decisional Diffie-Hellman problem is no harder than the Computational Diffie-Hellman problem, the reverse is not true: There are groups in which the DDH problem is easy, but the CDH problem is hard. Such groups are called **Gap Diffie-Hellman (GDH) groups**. [5] They will be discussed some more in Section 2.4.1 in relation to pairings.

2.3. Elliptic Curves

In 1985, the two mathematicians Neal Koblitz [26] and Vincent S. Miller [33] independently proposed the idea of using elliptic curves for cryptographic purposes [23]. At that time, multiplicative groups of finite fields and the related DLog Problem were the basis for many protocols in Public Key Cryptography (introduced some ten years earlier by Diffie and Hellman as mentioned in section 2.2.2). The hardness of the DLog Problem in a concrete case naturally depends on the underlying group - in $(\mathbb{Z}_2, +)$, there is no difficulty at all: $\mathbb{Z}_2 = \{0, 1\}$ has only two elements, so finding the logarithm is equivalent to computing two discrete exponentiations and checking which yields the expected result. Choosing suitable groups is therefore essential. For $\mathbb{F}_q^*$, specific attacks against schemes relying on the Discrete Logarithm problem are known that lower the security level. As by a recent recommendation by [47], q is required to be at least a 1024 bit number for cryptographic purposes. Cyclic groups for the DLog problem can also be found on elliptic curves defined over finite fields. Their value is rooted in the fact that most of them are not susceptible to the specific attack to which finite field subgroups are vulnerable. For the corresponding problem (called Elliptic Curve DLog Problem, ECDLP) the underlying field therefore can be smaller, which allows for smaller groups and shorter keys to reach the same security level.

An elliptic curve (EC) is a special curve that can be used as the underlying set for a group whose group laws are derived geometrically. Defining them over different fields such as $\mathbb{R}$, $\mathbb{C}$, and $\mathbb{F}_q$ ($q \in \mathbb{Z}$ prime[3]) yields different curves.

A standard representation for its definition is the so-called **short Weierstrass equation**[4], as follows:

Definition 2.4 (Elliptic Curve over a finite field)**.** Let $\mathbb{F}_q$ be a finite field of prime characteristic $q \in \mathbb{Z}, q > 3$, $a, b \in \mathbb{F}_q$ some constants and $+$ the additive operation in $\mathbb{F}_q$. Then the elliptic curve over $\mathbb{F}_q$ consists of all points $(x, y) \in \mathbb{F}_q \times \mathbb{F}_q$ that fulfill

$$y^2 = x^3 + ax + b \; mod \; q$$

$$4a^3 + 27b^2 \neq 0$$

The second condition ensures that the polynomial has no multiple roots, i.e. the curve is non-singular (refer to [14] for a more detailed explanation). A point 'at infinity', usually denoted by $\mathcal{O}$ or ∞ is added to the set. The elliptic curve is then given by

$$E(\mathbb{F}_q) = \{(x, y) \in \mathbb{F}_q \times \mathbb{F}_q \mid y^2 = x^3 + ax + b \; mod \; q \wedge 4a^3 + 27b^2 \neq 0\} \cup \{\mathcal{O}\}$$

2.3.1. Group definition on elliptic curves

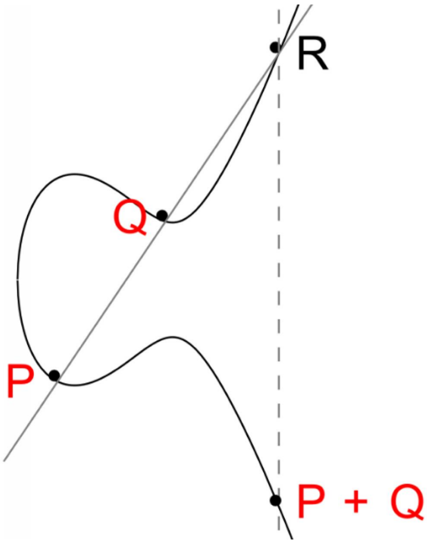
To make use of the Discrete Logarithm problem, a definition of (cyclic) groups on elliptic curves is necessary. Therefore, a group law, an identity element, and an inverse (relative to the group law) need to be defined. The group law must also be associative.

Identity element In order to fulfill the group requirements, a point 'at infinity', usually denoted by $\mathcal{O}$ or ∞ is added to the set as identity element.

[3] Please note that the DLog problem in a group $\mathbb{G}$ is not harder than in the greatest subgroup of prime order. Hence, without loss of generality it suffices to consider groups of prime order.

[4] It is only suitable to define elliptic curves over fields with characteristic greater than 3.[23]

Figure 2.1.: The addition $P \oplus Q$ of two distinct points on an elliptic curve [12]



Inverse All elliptic curves are symmetric about the x-axis since for all input x, the equation $y^2 = x^3 \oplus ax \oplus b$ yields two solutions $\pm y$. Therefore, the inverse of a point $P = (p_x, p_y)$ can be defined as $-P = (p_x, -p_y)$. Drawing the straight line $\overline{-PP}$ then yields a parallel to the y-axis ($y = p_x$), so $P \oplus -P = \mathcal{O} = -P \oplus P$. The inverse is thus well-defined.

Group law A binary group operation (group law) has to be provided. The idea is to draw straight lines between two points that intersect the curve at a third point (with the identity element as auxiliary result if there is no intersection) and reflect it w.r.t. the x-axis. This is visualized in figure 2.1.

Specifically, an addition $P \oplus Q$ for $P, Q \in E(\mathbb{F}_q)$ can be envisioned as follows:

1. Draw the straight line $\overline{PQ}$. For $P = Q$, draw the tangent line instead.[5]
2. Let $R \in E(\mathbb{F}_q)$ be the intersection R of $\overline{PQ}$ and $E(\mathbb{F}_q)$. [6]
3. Reflect $R = (r_x, r_y)$ w.r.t. the x-axis and get $S = -R = (r_x, -r_y)$.
4. $S \coloneqq P \oplus Q$

Note: In the case $Q = -P$, the straight line is parallel to the y-axis and there is no third intersection with $E(\mathbb{F}_q)$ because for all $x \in \mathbb{F}_q$ there exist at most two y that solve $y^2 = x^3 \oplus ax \oplus b$ and naturally, when connecting P and $-P$ the two solutions for p_x are already covered. Therefore, the result of $P \oplus (-P)$ is defined as $\mathcal{O}$.

Similarly, $P \oplus \mathcal{O} \coloneqq P$.

Closure Since for any two points P and Q (where $P \neq Q$ and $P, Q \neq \mathcal{O}$) there always exists an intersection R of $\overline{PQ}$ and $E(\mathbb{F}_q)$ and reflecting it yields a point on the curve as well (due to the symmetry about the x-axis), the group is closed under the addition $\oplus$.

[5] This is always possible because of second condition in the definition.

[6] This intersection point always exists because the equation has degree three [46].

Associativity The proof that $\oplus$ is associative is too elaborate for this thesis and will be omitted. Interested readers please refer to [46] for an algebraic proof and to [16] for a geometric proof.

There are two implications of this definition:

Lemma 2.5 (Commutativity). *The group $(E(\mathbb{F}_q), \oplus)$ is commutative, i.e.:*

$$\forall P, Q \in E(\mathbb{F}_q): \; P \oplus Q = Q \oplus P$$

.

Proof: As the straight lines $\overline{PQ}$ and $\overline{QP}$ are the same and therefore the intersection with $E(\mathbb{F}_q)$ and its reflection are the same, $P \oplus Q = Q \oplus P$.

Lemma 2.6. *When defined over a finite field, $(E(\mathbb{F}_q), \oplus)$ is either cyclic or the product of two cyclic groups.*

Remark. This is important, because the DLog-Problem and the corresponding ECDLP are defined on cyclic groups.

Proof: See chapter 13.1.3 in the "Handbook of Elliptic and Hyperelliptic Curve Cryptography" [9] for proof.

2.3.2. Operations on elliptic curves

The following computation rules can be derived from the previous section 2.3.1 (given $P = (p_x, y_p), Q, R \in E(\mathbb{F}_q)$):

$$-P = (p_x, -p_y) \tag{2.1}$$

$$P \oplus (-P) = P - P = \mathcal{O} \tag{2.2}$$

$$P \oplus \mathcal{O} = P = \mathcal{O} \oplus P \tag{2.3}$$

$$P \oplus Q = Q \oplus P \tag{2.4}$$

$$(P \oplus Q) \oplus R = P \oplus (Q \oplus R) \tag{2.5}$$

Scalar multiplication by an integer $n \in \mathbb{N}, n > 0$ can be defined recursively:

$$1 \cdot P = P \tag{2.6}$$

$$n \cdot P = P \oplus (n-1) \cdot P \tag{2.7}$$

It follows:

$$(-n) \cdot P = -(n \cdot P) = -P - (n-1) \cdot P \tag{2.8}$$

$$0 \cdot P = (n-n)P = nP - nP = \mathcal{O} \tag{2.9}$$

Double and Add Method When computing nP $(n \in \mathbb{Z}, P \in E(\mathbb{F}_q))$, instead of $(n-1)$ additions on the curve, a method called 'double and add' is usually used for the calculation. It is based on the binary representation of n. Starting from the left and iterating to the right, two steps are taken for every digit but the last: First, if the digit is 1, add one P to the result, otherwise proceed with the second step and double the current result. At the last digit, stop after the addition step. This reduces the number of necessary operations drastically, especially for large n.

Table 2.1.: Key-size in bits: comparison for various concepts (given in [47]

Symmetric keys	ECDLP in $E(\mathbb{F}_q)$	DLP in $\mathbb{F}_q$
80	160	1024
128	256	3072
256	512	15360

2.3.3. Elliptic curve cryptography

Usually, an identity-based signature using elliptic curves is made up of one or more group elements and elements of $\mathbb{F}_q$. Elements of $\mathbb{F}_q$ have size q. The size elliptic curve group elements equals the number of elements $\#E(\mathbb{F}_q)$. Especially for random curves it can be hard to determine it's exact size, but for a quick estimate the following lemma is useful:

Lemma 2.7 (Hasse's Theorem [14])**.** *Let $E(\mathbb{F}_q)$ be an elliptic curve defined over a finite field $\mathbb{F}_q$. Then*

$$q + 1 - 2\sqrt{q} \leq \#E(\mathbb{F}_q) \leq q + 1 + 2\sqrt{q}$$

[14] concludes: "(..) $|E(\mathbb{F}_q)| \approx q$, i.e. q and $E(\mathbb{F}_q)$ are of same order of magnitude."[7]

In general, for solving the ECDLP no algorithm faster than exponential runtime is known (although some classes of curves are vulnerable to attacks). There are subexponential algorithms known for the DLog problems on finite fields $\mathbb{F}_q$, so to achieve a certain security level the corresponding keys need to be relatively larger than in elliptic-curve-based schemes to provide secure use. As can be noted in table 2.1 where the respective key-sizes are compared to those in symmetric settings, not only are the keys smaller, they also scale linearly for ECDLP with increasing security level while for the DLog they scale super-linearly. In fact, this also holds for the respective operation run times. [32]

The European Union Agency for Network and Information Security, enisa, [47] recommends a key-length of 256 bit/ 3072 bit in ECDLP/DLog based schemes, respectively, for medium term use in future systems.

Commonly, cryptographic attacks exploit patterns and regularities. The same holds for elliptic curve cryptography: Some classes of curves are more easily and efficiently computable but also more vulnerable to attacks. An important example are **supersingular curves** [8] [9] They can be defined in several equivalent manners; for example, a curve is supersingular if $|E(\mathbb{F}_q| = 1 \bmod q$. Their properties are beneficial for implementation and computation [51]. However, the ECDL problem on supersingular curves is reduceable to the DLog problem on the multiplicative group of a finite field (see 2.4.1, where it can be solved in subexponential time [46]. Therefore, either the corresponding parameters have to be chosen large enough to make DLog hard in the reduced setting - defeating the purpose of using elliptic curves in the first place - or avoid supersingular curves altogether. Maas[30] argues that elliptic curves are best chosen at random and then tested for security issues, because the probability of picking a supersingular curve is very low. However, non-supersingular curves are not guaranteed to

[7] Curves with the property $E(\mathbb{F}_q) = q$ are called **anomalous**. Please note that using them is highly discouraged because the DLog problem on an anomalous curve can easily be reduced to a finite field of the same magnitude.

[8] Non-supersingular curves are called **ordinary**.

[9] This should not be confused with the requirement that elliptic curves be non-singular. The terms are unrelated ([46], Remark 3.2.2.)

be more secure and can even be weaker [15]
Another option is following recommendations by established authors, e.g. from the brainpool group [28] or the SECG consortium [43]. This approach is also proposed by the German Federal Office for Information Security [13], advising the use of provably randomized parameters with a comprehensibly documented construction that have been thoroughly checked for security issues.

2.4. Pairings

As mentioned in the section on hard problems 2.2.2, "Gap Diffie Hellman (GDH) groups" is a term coined for groups where the CDH is hard, but the DDH is easy. This echoes the notion of a signature: computing it should not be possible without a secret ingredient, but deciding whether it is valid should be easy. GDH groups are therefore very useful for signature schemes (see for example [6])

Pairings are special functions on cylic groups which make the construction of GDH groups possible.

Definition 2.8 (Pairings[10])**.** Let $\mathbb{G}_1, \mathbb{G}_2$, and $\mathbb{G}_3$ be cyclic groups and let $\mathbb{1}_i$ be the identity element of $\mathbb{G}_i$ $(i = 1, 2, 3)$. A bilinear pairing is a function $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$ with the following properties:

1. Bilinearity: $\forall (g, h) \in \mathbb{G}_1 \times \mathbb{G}_2 \; \forall n, m \in \mathbb{Z} : e(g^n, h^m) = e(g, h)^{nm}$
2. Non-degeneracy: $\forall (g, h) \in \mathbb{G}_1 \times \mathbb{G}_2 : e(g, h) = \mathbb{1}_3 \Leftrightarrow g = \mathbb{1}_1 \vee h = \mathbb{1}_2$
3. Computability: $\forall (g, h) \in \mathbb{G}_1 \times \mathbb{G}_2$, there is an efficient algorithm to compute $e(g, h)$.

Remark. If $\mathbb{G}_1 = \mathbb{G}_2$, the pairing is called symmetric, otherwise antisymmetric. [30] [11] In the symmetric case, the non-degeneracy-requirement can be equivalently written as: For g generator of $\mathbb{G}_1$: $e(g, g) \neq \mathbb{1}_3$. This is called **strong non-degeneracy**. In this case $e(g, g)$ generates $\mathbb{G}_3$ [30].
In the literature, three types of pairings are distinguished, where Type 1 are symmetric pairings. Type 2 and Type 3 both refer to asymmetric pairings, but for Type 2 a certain kind of map $\phi : \mathbb{G}_2 \to \mathbb{G}_{\nVdash}$ can be found. [12] Refer to [17] for details.

The most common instantiations of this definition are the **Weil** and the **Tate** pairings on elliptic curves over finite fields. (It is therefore proven that such maps exist.) All practical implementations of pairings are based on them[17] For implementation, the Tate pairing is usually better and therefore more frequently used in practice [5]

2.4.1. Pairing-based Cryptography

Pairings are a difficult chapter in the history of elliptic curves. They were initially introduced to break them by reducing the ECDLP to a regular DLog problem in a finite field. By

[10] Sometimes pairings are referred to as **bilinear maps**.

[11] Please note that a pairing is also called symmetric if $\forall x, y \in \mathbb{G}_1 e(x, y) = e(y, x)$. It can only be symmetric in this sense if it is also symmetric in the other sense.

[12] Those 3 types will not be referenced in the remainder of the thesis, but are mentioned for further reading.

mapping into large finite fields where the DLog is infeasible, pairings also became useful for Elliptic Curve Cryptography.
The group $\mathbb{G}_1$ in 2.4 is a Gap-Diffie-Hellman group. Let $\mathbb{G}_1 = \mathbb{G}_2$ be a cyclic group where the CDH problem (i.e. calculating g^{ab} from g^a and g^b) is hard. Given a group element g^z, with $z \in \mathbb{Z}$ unknown, and a pairing $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_3$, it is easy to decide whether g^z equals g^{ab} by transferring it to $\mathbb{G}_3$ via the pairing:

$$e(g^a, g^b) = e(g, g)^{ab} = e(g, g^{ab}) \overset{?}{=} e(g, g^z)$$

The DDH problem is therefore easily solved in $\mathbb{G}_1$. The solution is very similar in the asymmetric case.

Remark. It is essential that the CDH problem be hard in $\mathbb{G}_3$. Assume it was not hard. Let g be the generator of $\mathbb{G}_1$, $h \in \mathbb{G}_2$. To compute $a \in \mathbb{Z}$ from g^a, it would then suffice to transfer the problem to $\mathbb{G}_3$: $e(g^a, h) = e(g, h)^a$, from which a can be easily computed by the assumption. Therefore, CDH is hard no harder in $\mathbb{G}_1$ than in $\mathbb{G}_3$.

It would be very useful to have pairings with $\mathbb{G}_1 = \mathbb{G}_2 = \mathbb{G}_3$, because then it would be clear that CDH is hard in both $\mathbb{G}_1$ and $\mathbb{G}_3$. However, no such pairings are known.
In general, if $\mathbb{G}_1$ is an elliptic curve of order n, then $\mathbb{G}_3$ is a multiplicative subgroup of a so-called extension field $\mathbb{F}_{q^k}$ [23] (i.e. the set of $(f_1, .., f_k) \in (\mathbb{F}_q)^k$) of size $k * q$, where k is the **embedding degree** of the curve. The DLog problem is subexponential in finite fields like $\mathbb{F}_{q^k}$, so if k is small, applying the pairing also breaks CDH in $\mathbb{G}_1$, as mentioned above. Therefore, a pairing-based scheme is only as secure as the DLog problem is difficult in $F^*_{q^k}$. For example, while the ECDLP on a 256 bit curve usually provides security matching the DLog in a 3072 bit field, a pairing-based scheme using a 256 bit curve with embedding degree $k = 6$ is only as secure as the DLog in a $256 * 6 = 1536$ bit field. Indeed, 512 bit would be necessary to reach the 3072 bit DLog level. The embedding degree of most supersingular curves is too small (≤ 6) [23], which explains why using them is discouraged in practice (as mentioned in subsection 2.3.3). On the other hand, k should not be too large to allow for the pairings to be computed efficiently. [15] provides some insight on so-called **pairing-friendly curves** (e.g. Barreto-Naehrig curves of degree 12, on which pairing operations can be done quite efficiently).
Symmetric pairings are barely used anymore. Barreto et al. [2] state: "All modern pairing libraries are built on ordinary[13] elliptic curves in the asymmetric setting."
Finding appropriate groups is not easy: they must be suitable for defining a pairing on them, but if they have certain unfortunate properties, the pairing will make the DLog problem too easy to solve [14]. For example, anomalous curves are not suitable for cryptographic purposes. Similarly, as mentioned above, supersingular curves are hardly used anymore, because the groups have to be quite large for them to be secure.
As a result, while pairing-based schemes provide smaller key material than integer-based ones like RSA, its size grows at a similar rate. This also makes computations slower on the curves. At the moment, 256-bit Barreto-Naehrig curves[4] (which have a embedding degree of 12) are considered ideal for providing a 128bit security level with respect to the balance of key size and computational efficiency. However, their fitness for cryptographic appliances could already be showing cracks. So pairings remain a difficult topic.

[13] i.e. non-supersingular
[14] Of course, curves on which pairings apply also need to be cautiously handled in a non-pairing-based context.

3. Identity-based Signatures

This chapter provides a more detailed introduction to Identity-based Signatures. First, the general structure and a short note on key escrow are given together with a quick overview on the related work relevant to the thesis. Then the different phases in an IBS scheme are discussed more elaborately. Following this, four centralized IBS schemes are presented. Next, the notion of hierarchical IBS schemes is explained, together with two more exemplary schemes. A short introduction to threshold IBS is followed by a discussion of quantifying the operations and parameters used in IBS schemes.

In total, six schemes are presented in this chapter, each based on a different research paper. Four of them (sometimes referred to as *regular*) have a centralized structure, the other two employ hierarchical IBS.

Identity-based cryptography (IBC) was introduced by the cryptographer Adi Shamir in 1984 [44] and rests on the idea that the public key should be mathematically connected to the identity of the identified object or person itself. This helps reduce overhead because it allows group members to authenticate even unknown senders without having to look up their public key. Hence, there is no need for the communication participants to keep track of the group members. This proves useful specifically in a dynamic environment, where the group changes fast and unforeseeably.

Unlike other public key schemes, the private key is not extracted arbitrarily, but computed from the public key of a participant; instead, the public key can be chosen freely and it is therefore possible to instantiate it with the identity (or to let it be computable given the identity). However, this implies that anybody able to compute the secret key for one identity can do the same for any identity. IBS schemes therefore contain the notion of a trusted third party or key generation center (referred to as TTP or kgc, respectively) which computes the secret keys on behalf of participants using its own randomly chosen secret (master secret).

For group communication, the scheme has the following structure:

Setup With a security parameter λ as input for this step, the TTP chooses appropriate parameters, randomly generates a master secret (msk) and computes the associated public key (mpk).

Key Generation Given an identity ID of an applicant (and therefore their public key), the TTP computes the respective secret key (usk) using the msk.

Sign The signer computes a signature sig using the message m and her usk.

Verify Given the sender's identity ID, the mpk, and m, the message recipient can verify the signature s. Note that only (former)[1] group members are in possession of the mpk and can compute the current upk for a given ID.

[1] As a key can not be mathematically invalidated w.r.t. the underlying master key pair, only a re-key can prevent former members from verifying signatures computed after their exit from the group.

Re-key The TTP randomly generates a new *msk* and computes the corresponding *mpk*. It performs the Key Generation step for all current group members once again with the new *msk*.

Remark. The re-key step is not explicitly specified in other literature but for dynamic groups it is of great interest for the evaluation of costs in the IBS schemes, because memberhip in the group is de facto controlled through key management; thus, key revocation is realized by re-keying all other members. The connection between a *usk* and the corresponding *msk* can not be mathematically invalidated. As long as a device knows the current master public key and possesses a matching *usk*, it can compute *upk*s, and compute and verify signatures. Other than with new credentials for all other devices, a membership can not be revoked.

The trusted third party is a significant point of criticism against the use of IBS. It acts as a key escrow entity, knowing the secret keys of all users, and constitutes a single point of failure. With the TTP in place, non-repudiation can not be offered from an IBS scheme. However, Kiltz and Neven argue that the same holds true for Certificate Authorities (CA), which can forge certificates for arbitrary public keys and impersonate any user, claiming that they registered two keys [24].

In the original paper, Shamir proposes a signature scheme based on RSA. Since then, instantiations that use Elliptic Curve Cryptography (ECC), for example [18], and Lattice Cryptography [42] have become known. (Refer to chapter 2.3.3 for insight on ECC).

3.1. Related work

In his defining paper [44], Shamir proposed an RSA-based scheme for Identity-based signatures. Identity-based encryption can not be based on RSA, however (see [44] for details), so Shamir could not present a concrete implementation. Only in 2001 did two independent parties discover IBE schemes based on bilinear pairings 2.4.1 that proved to be secure and efficient ([6], [8]. Since then, pairing-based cryptography has played an important role for the development of IBS schemes, too (see [5] for an overview). In 2002, Craig Gentry and Alice Silverberg published a paper on "Hierarchical ID-based Cryptography"[19], prompting the development of similar schemes. Apart from Elliptic Curve Cryptography, IBS schemes have also been proposed for Lattice Based Cryptography [42]. Usually, the focus is on the mode of operation in the signing process. To the best of the authors knowledge, the topic of key-revocation has not yet been addressed at significant length, even though there are some interesting implications. A comprehensive guide to a classification and standardized evaluation of IBS scheme like the taxonomy presented in the next chapter is also unheard of.

3.2. Phases of IBS

Setup

To set up an IBS authentication scheme, the TTP needs to choose the setting: the elliptic curve on which it will be based, the corresponding finite field and the concrete functions to instantiate pairings, hashes etc. From the elliptic curve group one element that generates

the group has to be determined.
Additionally, the master public and secret keys (*mpk* and *msk*, respectively) need to be computed. They are the basis for every group member's key pair and therefore different for every group of devices to which the TTP attends. In the re-keying phase they will be re-computed. Parts of the information generated and computed in this phase constitutes the *public parameters* and needs to be disclosed to each member when they join group. This phase has to be entered only once per group of devices [2].

Extract

Once a new device wants to join the group, the TTP needs to compute the secret key corresponding to the applicant's identity (*user secret key* or *usk* for short).[3] Therefore, this step needs to be executed separately for every member. The *usk* can then be sent to the new member together with the *public parameters*. Note that unlike them, the secret information needs to be transmitted via a secure channel.

Re-key

To re-key in a group of n members, the *mpk*, *msk*, and all n *usk* need to be re-computed. All steps of the extraction phase and some of the setup are therefore repeated. This phase is usually not considered in the literature as its importance is specific to scenarios with a group of communicating devices. The EC group and finite fields, the pairing and hash functions, and the group generator chosen in the setup phase may stay the same. [4] Since some of the information to be sent out is the same for all members, that part may be distributed in multicast mode. The secret information obviously must be sent to each device individually over a secure channel. The phase tables display the unicast setting; what data is universal can be easily deduced by consulting the *send* line of the extraction table . Everything that is not included there is the same for all members.

Sign

To sign a message m, a device always uses its secret key (received after *Extract* from the TTP), usually an integer or group element. Depending on the scheme, additional information like the device's ID may be necessary. The resulting signature is unique for every message and device and should not be forgeable. Of all phases, *Sign* is ideally executed far more often than *Setup*, *Extract*, and *Re-key.* After all, the goal is secure communication, not managing scarcely used keys. Therefore, the size of the signature and the related computations are the most important criteria for the use in constrained networks. Also, *Setup* and *Verify* are the two phases which are mainly processed on the constrained devices - all other phases only affect what information the devices receive. Of course, the message itself must be to attached to the signature (or rather the other way around).

[2] Although re-keying is technically the same as forming a new group under the same setup, it will be treated as a process within the same group.

[3] Deciding whether or not a device is allowed to enter is not part of this procedure. For the purposes of this thesis it is assumed that only benign devices apply.

[4] It is advisable to consider the life span of those basic parameters, though: The master secret key is usually chosen at random, so as the probability of choosing the same *msk* a second time grows, a whole new setup with different curves, pairings and hash functions should be considered gradually more urgent.

Verify

In the last phase of the authentication process, all computation is aimed at answering a simple yes/no-question: Is the signature valid, i.e. does it authenticate the alleged sender of the underlying message? So no data needs to be distributed at the end. The sign-line is therefore omitted in the respective table.

When pairings are used it is this phase in which their advantage come into operation, i.e. making the DDH easier on the elliptic curve to facilitate verification (see 2.4.1). So in the respective schemes one or more pairing operations are expected to be carried out during verification.

Also, the signer's public key is necessary for verification. The verifying device computes it in this phase from the signer's ID, but since that is the same in all schemes by virtue of being identity-based, this computation is also not taken into account in the tables.

3.3. Examples for centralized IBS

Of the centralized schemes presented in this subsection, two are based on pairings and two on plain elliptic curve cryptography. There are two of each because that way it becomes clearer which differences are rooted in the choice between pairing and non-pairing based setting and which aspects are distinguishing for a specific scheme.

They are explained and analysed with respect to their use in a group setting.

The rules of notation as described in table A.1 apply.

Hess02 and BLMQ05

Florian Hess proposed an IBS scheme in his 2002 research paper "Efficient Identity Based Signature Schemes Based on Pairings" [22], denoted as "Hess02" in this thesis. "BLMQ05" refers to a scheme introduced in 2005 by Messrs. Barreto, Libert, McCullagh, and Quisquater. They are among the most prominent examples for pairing-based IBS.

The specific steps taken in each phase are put in direct comparison in Table 3.1 to highlight subtle differences.

The setup phase is identical for both schemes: A random number is chosen as master secret key msk and its product with P produces the master public key mpk. The extraction of the user secret key (usk) shows the first difference: Hess02 features the most basic solutions to combining the msk and the device's identity to get a usk: Hashing the ID in $\mathbb{G}$ and multiplying it with the msk. BLMQ05 does not feature a hash function that hashes into $\mathbb{G}$, because when pairings are involved, doing so implies some restrictions with respect to the choice of $\mathbb{G}$ (see [17] for details).

Signing is also slightly different in the two schemes. For example, Hess02 always requires a pairing operation $e(Q, P)$ at that stage, while BLMQ05 could precompute the value $g = e(P, P)$ which never changes after the setup (not even through re-keying) and therefore only needs to be computed once. It is noteworthy that in both schemes the signature consists of one elliptic curve (EC) group element and one integer - the size of the transmitted data is therefore equal unless some outside factors restrict the curve that can be chosen more strictly for one of the schemes. As [3] points out, their scheme is "faster at verification than previously known IBS schemes"(pg. 2), because it requires only one pairing computation. As the sizes of usk and signature are equal, and because pairing operations are very costly

Table 3.1.: The computations by phases in Hess02 and BLMQ05

Phase	Hess02	BLMQ05
Setup	$msk \leftarrow \mathbb{Z}_p$ $mpk = msk * P$	$msk \leftarrow \mathbb{Z}_p$ $mpk = msk * P$
Extract	$usk = msk * H_1(ID)$	$usk = \frac{1}{msk+h_1(ID)}P$
Sign	1. $x \leftarrow \mathbb{Z}_p^*$, $Q \leftarrow \mathbb{G}^*$ 2. $r = e(Q,P)^x$ 3. $h = h_2(M,r)$ 4. $S = h * usk \oplus x * Q$ $sig = (h,S)$	1. $x \leftarrow \mathbb{Z}_p^*$ 2. $r = e(P,P)^x$ 3. $h = h_2(M,r)$ 4. $S = (x+h) * usk$ $sig = (h,S)$
Verify	$\widetilde{r} = e(S,P)e(H_1(ID), -mpk)^h$ $h \stackrel{?}{=} h_2(M,\widetilde{r})$	$\widetilde{r} =$ $e(S, h_1(ID) * P \oplus mpk) \cdot e(P,P)^{-h}$ $h \stackrel{?}{=} h_2(M,\widetilde{r})$

this is also the key difference between Hess02 and BLMQ05. Please note that the possibility of precomputing values is taken into account for the quantifying tables discussed in 3.6. In the taxonomy developed in the next chapter, as very small devices could have problems storing them.)
The correctness of the verification in Hess02 is based on the equality $r = \widetilde{r}$:

$$\begin{aligned} \widetilde{r} &= e(S,P)\,e(H(ID),-mpk)^h && = \\ &= e(h * usk \oplus x * Q, P)\,e(H(ID),-tP)^h && = \\ &= e(h * t * H(ID), P)\,e(x * Q, P)\,e(H(ID),-tP)^h && = \\ &= e(H(ID),P)^{ht}\,e(Q,P)^x\,e(H(ID),P)^{-ht} && = \\ &= e(Q,P)^x && = \\ &= r \end{aligned}$$

The correctness of BLMQ05 can be proven similarly.

GG09 and vBNN-IBS

In 2009, David Galindo and Flavio Garcia proposed "A Schnorr-like Lightweight Identity-Based Signature Scheme" [18], denoted as "GG09" in this thesis. A year earlier, the research paper "IMBAS: Identity-based multi-user broadcast authentication in wireless sensor networks" [7] by Mssrs Cao, Kou, Dang, and Zhao had been published. Neither scheme requires pairing. The computational cost that is thereby saved comes at the prize of larger signatures and a slightly more complex procedure.
Like Hess02 and BLMQ05, the two schemes are quite similar. The main difference lies in the composition of the signature and the subsequent verification process. While the signature in GG09 is comprised of two group elements and one integer, it is the other way around for vBNN-IBS. If the size of the integers is only a little smaller than the size of the group elements (as often happens in reality) that can make a decisive difference in some cases. See chapter 4.4 for a concrete example.

3.4. Hierarchical IBS

The Trusted Third Party which computes all private keys is a great concern in many group communication settings. Although the devices do not have to confer with it to verify certificates or manage revocation lists, it is tasked with the challenge of admitting members and coordinating their exit. In IBS for groups, the master secret needs to be changed every time a device leaves the group, because their key pair can not be mathematically rendered useless with respect to the existing msk, so some or all devices need to be notified of the change and equipped with new user secret keys. Therefore, the TTP can quickly become a bottleneck in large dynamic networks.
Hierarchical schemes help distribute the work load to several TTP instances. Given a tree structure with the TTP at the root, the master secret is employed only to generate private keys for the TTP's direct children. All inner nodes (i.e. all recursive children of the TTP except the leaves) now serve as Trusted Third Parties themselves and distribute private keys to their respective children. Therefore, the root TTP is relieved of the burden to manage all members' keys sigle-handedly.

Table 3.2.: The computations by phases in GG09 and vBNN-IBS

Phase	GG09	vBNN-IBS
Setup	$msk \leftarrow \mathbb{Z}_p$ $mpk = msk * P$	$msk \leftarrow \mathbb{Z}_p$ $mpk = msk * P$
Extract	$r \leftarrow \mathbb{Z}_p^*$ $R = rP$ $u = r + msk \cdot h_1(R, ID)$ $usk = (u, R)$	$r \leftarrow \mathbb{Z}_p^*$ $R = rP$ $u = r + msk \cdot h_1(R, ID)$ $usk = (u, R)$
Sign	$x \leftarrow \mathbb{Z}_p^*$ $X = xP$ $h = h_2(ID, M, X)$ $s = x + h \cdot u$ $sig = (s, R, X)$	$x \leftarrow \mathbb{Z}_p^*$ $X = xP$ $h = h_2(ID, M, R, X)$ $s = x + h \cdot u$ $sig = (s, R, h)$
Verify	$c = h_1(R, ID)$ $d = h_2(ID, M, X)$ $s * P \overset{?}{=} X \oplus d * (R \oplus c * mpk)$	$c = h_1(R, ID)$ $\widetilde{X} = s * P \ominus h * (R \oplus c * mpk)$ $h \overset{?}{=} h_2(ID, M, R, \widetilde{X})$

Another desired effect of using a hierarchical IBS scheme[5] would be to restrict the impact of re-keying on the group members. If every member has to be equipped with new keys at every re-key, this process sets a natural limit to the applicability of IBS in large and highly dynamic groups. When a device is frequently affected by a re-key because of the high fluctuation, even though it only communicates with a fraction of its peers, it might become more prudent to look up public keys for signature verification instead of having to keep up with the continuous changing of the msk, mpk, and own usk. So a desirable feature of HIBS would be to restrict the effect of a member leaving the group to a subset and re-key only those devices with the same parent as the ex-member.
However, that is far from simple. A TTP's public key is notably not derivable from its identity. This holds true for all inner-node TTP's as well. So to communicate across branches, not only is it necessary to know the other party's location in the tree, but the public keys of all its parents needs to be explicitly known for verification. It can not simply be computed from their identity.
To the author's best knowledge, no HIBS scheme has yet been discovered where the re-key for the purpose of excluding an ex-member[6] only affects a part of the leaves. However, by adjusting one of the exemplary schemes it is possible to reduce the impact, such that not all members always need to process a complete re-key that changes their usk, but where a large part of the devices only needs to update the public information of some inner nodes. A more detailed explanation is given below.

There are two ways of implementing HIBS: The inner nodes can either be part of the TTP or group members. The latter approach delegates the TTP's function partly to the devices themselves. The group then needs to be managed as a tree. That tree does not necessarily need to reflect any actual hierarchy in the group. However, some defining criterion for the role allocation could be sensible: In a group with constantly changing members, long-term members should be preferred for the role of inner nodes, short-term members for leaves, for example. Adding the task of private key generation and management to the devices' functions takes pressure off the TTP, yet it contradicts the need of constrained devices to be spared as many computations and as much communication as possible. It therefore seems more prudent to divide the trusted third party in several instances and hierarchically organize them. The devices are then strictly assigned the role of leaves of the tree. They still need to make their position known to their communication partners because the verification depends on their parents' public keys.

3.4.1. Examples for hierarchical IBS schemes

Both hierarchical schemes presented in this subsection are pairing-based. However, only one allows for a flexible number of levels, whereas in the other, the number of levels has to be fixed at the setup. This is related to the signature size: When the number of levels is variable, so is the length of the signature, otherwise it is constant.

The first scheme (called GS02) originates in a research paper published in 2002 by Craig Gentry and Alice Silverberg [19], which was very influential and the first to explore hierarchical IBS (HIBS) [31]. It allows for the hierarchy to be exteneded also after the setup, at the price of the key and signature size increasing with every level.

[5] Also referred to as HIBS [19]
[6] I.e. for the purpose of rendering the ex-members key information useless

Table 3.3.: The computations by phases in GS02 and ALYW06

Phase	GS02	ALYW06
Setup	**root:** $msk_0 =\leftarrow \mathbb{Z}_p$ $mpk = msk_0 * P$ **lower level t:** $msk_t \leftarrow \mathbb{Z}_p$ $Q_t = msk_t * P$	**root:** $msk \leftarrow \mathbb{Z}_p$ $mpk = msk_0 * P$ $P_1, .., P_\ell \leftarrow \mathbb{G}$ $U_i = msk * P_i \ (i = 2, .., \ell)$ **no lower level setup**
Extract for ID_k	$R_k = H_1(ID_1, ..., ID_k)$ $S_k = S_{k-1} \oplus msk_{k-1} * R_k$ $(S_0 = \mathbb{1} \in \mathbb{G})$ $usk_k = (S_k, Q_1, .., Q_{k-1})$	**root:** $r_1 \leftarrow \mathbb{Z}_p$ $A_1 = \frac{1}{msk - h_1(ID_1)} * (P_1 \ominus r_1 * P)$ $usk_1 = (A_1, r_1)$ **lower level:** $r_k \leftarrow \mathbb{Z}_p$ $h = h_1(ID_1)$ $F_j = U_j \ominus h_1(ID_j) * P_j$ $(j \in \{2, .., \ell\})$ $\tilde{F} = \sum_{j=2}^{k}(h * F_j)$ $A_k = A_{k-1} \oplus h * C_{k-1} \oplus r_k * \tilde{F}$ $B_k = B_{k-1} \oplus r_k * (mpk \ominus h * P)$ $C_{k,j} = C_{k-1,j} \oplus r_k * F_j$ $(j \in \{k+1, .., \ell\})$ $usk_k = (A_k, B_k, C_{k,k+1}, .., C_{k,\ell}, r_1)$ **where** $B_0 = C_0 = \mathcal{O}$
Sign at level k	$T_M = H_3(Id_1, .., ID_k, M)$ $Q_k = msk_k * P$ $S = S_k \oplus msk_k * T_M$ $sig = (S, Q_1, .., Q_k)$	$s \leftarrow \mathbb{Z}_p$ $h = h_1(ID_1)$ $F_j = U_j \ominus h_1(ID_j) * P_j$ $(j \in \{2, .., k+1\})$ $\tilde{F} = \sum_{j=2}^{k}(h * F_j)$ $S_1 = A_k \oplus h_2(M) * C_{k,k+1} \oplus s *$ $[h_2(M) * F_{k+1} \oplus \tilde{F}]$ $S_2 = B_k \oplus s * mpk \ominus h * P$ $sig = (S1, S2, r_1)$
Verify	$R_i = H_1(ID_1, .., ID_i) \quad i = 2, .., k$ $T_M = H_3(ID_1, .., ID_k, M)$ $g = e(mpk, R_1)$ $v = \prod_{i=2}^{k} e(Q_{i-1}, R_i)$ $e(P, S) \overset{?}{=} g \cdot e(Q_k, T_M) \cdot v$	$h = h_1(ID_1)$ $F_j = U_j \ominus h_1(ID_j) * P_j$ $(j \in \{2, .., k+1\})$ $\tilde{F} = \sum_{j=2}^{k}(h * F_j)$ $T = e(P, P_1) \cdot e(mpk, mpk)^{-r_1} \cdot$ $e(S_2, F_k * h_2(M) \oplus \tilde{F})$ $T \overset{?}{=} e(mpk \ominus h * P, S_1)$

Mssrs. Au, Liu, Yuen, and Wong propose "Practical Hierarchical Identity Based Encyrption and Signature schemes Without Random Oracles" in their 2006 publication with the same title [31]. In this scheme, not only is the size of the signature the same for all levels, its also independent of the number of levels in the hierarchy. However, the hierarchy can not be extended flexibly. In table 3.3, a step-by-step comparison of their HIBS scheme (called ALYW06 in this thesis) to GS02 is given.
The constant size of the signature comes at the prize of a larger master public key and more complex computations. However, the private key size grows in inverse proportion to the corresponding level, i.e. the deeper the device is located on the tree, the smaller is its private key. In a setting where all but the lowest level are part of the Trusted Third Party, this is the best case. There is also no harm in the number of levels required to be static if the devices in the group are always added on the same (lowest) level as all others and all higher levels are restricted to the TTP.
The constant signature size also allows for the number of pairings needed for verification to be independent of the number of levels, whereas the computational cost significantly grows with every level in GS02.

Unfortunately, only GS02 can be adjusted in a manner that allows a re-key without replacing the *usk*s of all group members. In this scheme, every inner node possesses a master secret of their own on which each child's private information depends. To verify a signature, the master public keys of all parents of the sender (including the root) are used. Therefore, if one of the parents' secret key changes, so does their public key and all their childrens' private keys. The devices in the corresponding subtree are therefore aware of the change and would not accept a signature using old key material.
However, GS02 leave it to the devices themselves to communicate their parents' public keys with the signature. How many of them need to be included then depends on the recipient's location in the tree. A device with the same parent as the sender does obviously not require the public information to be communicated with the signature to be able to verify it. They already have that information. A device in another part of the key would require the public keys of all the sender's parents up to the lowest common parent to be sent to them.
Of course in this setting nothing prevents an ex-member from sending messages to devices that don't share the same parents on all levels. It can simply enclose old key material. If the public keys are sent with the signature, then the only way to effectively exclude a member is re-keying at the root and therefore changing the private information of all members. As explained above, this is impractical in large groups, which is what hierarchical schemes are targeted at. Luckily, a quite obvious adjustment reduces the impact of re-keying: The inner nodes' public keys are not included in the signature but communicated to all group members by the TTP. Thus, the devices can not cheat, but on average, only a few devices need to process a full re-key, whereas for the majority it suffices to update a public parameter.
Of course, there are still drawbacks to this approach; a re-key still needs to be communicated to every member and the cost of storing all the inner nodes' information may weigh heavily for devices with constrained memory. On the upside, the size of signature is constant in this setting, and on top of it quite small. For devices with sufficient memory, this adjusted scheme is therefore quite beneficial. The adjusted scheme will be referred to as GS02*.

ALYW06 does not leave room for any such improvement: there is no lower-level setup in this scheme, therefore the inner nodes do not have their own master secrets. A parent chooses individual secrets s for all its children. Of course, when a child computes its own children's keys, those depend on s. To re-key the children of a certain node, it is therefore necessary to re-key one parent who then computes new keys for its descendants. However, there seems to be no possibility of excluding a group member without changing the master secret and therefore changing every member's keys.

Remark. The pairings in both papers are defined as symmetric, however it remains unclear to the author of this thesis whether this is a necessary requirement or if it is a soft requirement. For example, in case none had been known at the time, more recent advances of research on pairing-based cryptography may have discovered antisymmetric bilinear maps that work as well.

3.5. Threshold IBS

A single msk to compute all usk means that all of them are compromised if an attacker knows the msk.[7] To eliminate this single point of failure and additionally facilitate non-repudiation, a distributed TTP can be set up. This means distributing the master secret key to several devices such that neither can know or compute the msk on their own, but needs a minimum number of its peers to cooperate in order to generate a secret key for identity ID. Even if one of the devices is then corrupted or not available, secret keys for users can safely be issued. In a (k,n)-threshold system, k out of n participants can jointly generate usks, but any group of less than k devices will not be able to gather enough information to compute the master secret. [11] However, the computation is more costly, because more communication has to take place. If devices directly communicate with the different parts of the TTP (as they should in order to retain the distributed model), the communication can become expensive quickly; also the devices need to keep an index of all TTP-agents.

[7] This is true for the hierarchical setting as well. In that setting it is however possible to compromise only parts of the tree by obtaining a lower-level msk.

3.6. Phase-by-phase quantifying overview of process details in comparative tables

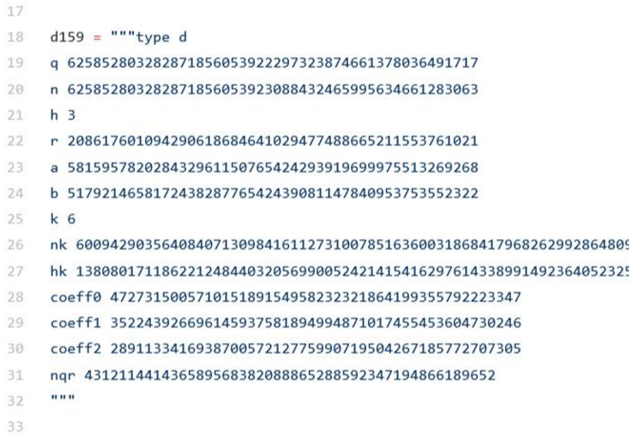
In order to understand the specific effects of using a certain scheme, a quantifying analysis of the respective operations and data transmissions is advisable. To this end, comparative tables of every phase (i.e. *Setup, Extract, Re-key, Sign, Verify*) are included in the thesis. There is one table for every phase comparing all centralized schemes and another comparing the hierarchical ones. For compactness, only the table for signing in the hierarchical schemes is introduced in this chapter. The complete set can be found in appendix A. [8]
As pointed out in 1.2.1, three aspects need to be considered for constrained devices: storage, computing power, and energy supply. Therefore, in the analysis of schemes that may be employed in a constrained environment, there are three factors to take into account for every stage: How much and what kind of information needs to be available at that phase? What operations need to be computed and how costly are they? What information of what size needs to be sent to other devices? In the tables, those three questions are referred to by the respective keywords **know**, **compute**, and **send**. Please note that in the setup phase (depicted in tables B.1 andB.2), **know** is substituted by **generate**, because no information is available yet, but rather needs to be chosen or computed. Similarly, in the verification phase no information needs to be sent, so **send** is omitted (see tables B.9 andB.10).
Information that is always necessary independent of the scheme is also left out. For example, the message and sometimes the own ID need to be known for the signing process. But since the message is always necessary, it is not included in the tables. The ID is included when it is not obviously necessary, e.g. if it is used for signing. In the verification, re-key and extraction phases it is always needed, so it is not included in the comparison there.

Know and Send

The information that needs to be known or sent can be quantified in bits - and more abstractly in terms of group elements, integers and bit strings.
The devices also need to be informed about the parameters of the fields, curves, pairings, and hash functions that are used. This information is much harder to quantify, however. For example, an elliptic curve group can be described by the elliptic curve parameters $q \in \mathbb{Z}$, $a, b \in \mathbb{F}_q$ and the generator P for the cyclic subgroup that is used. Then the device may be able compute the relevant information itself. If more than a minimal amount of memory is available in the devices, simply storing several curve, hash or pairing descriptions and sending a denominator for each one with the setup information seems more prudent. Figure 3.1 shows an examplary description of an elliptic curve in the Charm framework.
A more detailed examination would be necessary to determine the best options and quantify this information. Therefore the manner of communicating curve parameters and the like will not be addressed in the quantitative analysis.

[8] The phases *Setup*, *Extract*, and *Re-key* are executed by the TTP, which is not necessarily constrained. Analyzing the respective data of each phase still has its merit: On the one hand, the TTP might very well be constrained, too. On the other hand, what it sends to the devices is important in any case; so at least the "send"- line should be considered carefully, as it affects the group members as well (which have to receive and process the data)

Figure 3.1.: Description of an elliptic curve for a pairing-based scheme in the Charm [1] framework



```

d159 = """type d
q 625852803282871856053922297323874661378036491717
n 625852803282871856053923088432465995634661283063
h 3
r 208617601094290618684641029477488665211553761021
a 581595782028432961150765424293919699975513269268
b 517921465817243828776542439081147840953753552322
k 6
nk 60094290356408407130984161127310078516360031868417968262992864809
hk 1380801711862212484403205699005242141541629761433899149236405232
coeff0 472731500571015189154958232321864199355792223347
coeff1 352243926696145937581894994871017455453604730246
coeff2 289113341693870057212775990719504267185772707305
nqr 431211441436589568382088865288592347194866189652
"""

```

Table 3.4.: List of all relevant operations in the schemes by category

pairing	basic EC operations	integer operations	misc
P Pairing	**GE** exponentiation	**ID** division	**H** hash computation
	GI inversion	**IM** multiplication	
	GO operation	**IA** addition	
	RG extraction of a random group element	**modInv** modular inversion	
		mod mod comp.	
		RI extr. of random int	

Compute

The operations in the schemes can be divided in four categories: computation of a pairing, basic operations on elliptic curve group elements, integer operations, and computations that do not fall into either category (misc).

According to [3], pairings are up to about 21 times as expensive as group exponentiation. Of course, this depends on the choice of the specific parameters and the state of the art regarding efficient pairing cryptography.

In general, operations on group are far more costly than integer operations. Among them, exponentiation of a group element is clearly more expensive than even a single group operation, because even with the Double and Add Method 2.3.2 it involves computing the group operation several times. [9] The cost required to compute a hash depends on the significance of this operation: If hashing is used merely to harmonize certain numbers, a relatively weak and cheap function can be chosen. If it is used to increase security, it may also cost significantly more. Table 3.6 displays all relevant operations by category.

[9] Of course, a group operation $g * g$ is a group exponentiation with power 2. So, for the term "group exponentiation", the implicit assumption is an exponent $>> 2$.

In the quantifying tables, all operations are displayed regardless of the cost so that they may be used for an exact analysis.

Table 3.5.: Signing phase at level k for hierarchical IBS schemes

Paper	GS02	ALYW06 (with ℓ levels)
know	2 group elem.	$2 + l + k$ group elem. 1 int
compute	1 H 1 GO 1 GE	$2k + 4$ GE $2k + 4$ GO $k + 1$ GI $k + 2$ H 1 RI
send	$k + 1$ group elem.	2 group elem. 1 int

Table 3.6 shows the quantitative data of the signing phases of the two hierarchical IBS schemes. The value are given with respect to the level k of the instance that is signing the message, and the overall number of levels ℓ since it must be fixed at setup in the ALYW06 scheme. Note that even though the signature is always computed with respect to a message (which is assumed to be a bit string), this value does not appear in the table, as it is same for all possible schemes. It is assumed that no values are pre-computed. Such a distinction could be easily incorporated however.

4. A taxonomy for IBS schemes

To judge the suitability of an IBS scheme for a constrained environment various factors have to be taken into account. Depending on the specific scenario, the devices might be restricted in different manners and therefore different schemes might be suitable. The following chapter provides a taxonomy that makes it easier to compare schemes and evaluate their appropriateness in a given setting. The criteria include size of the signature, size of the information sent with every re-key, the added computational cost for signing and verifying, the number of group members affected by a re-key, the memory requirements for permanent information (like private keys), and a classification by the efficiency of known attacks on the underlying mathematical concepts. The taxonomy will be given in form of a so-called spider chart (also called radar diagram), where several criteria are plotted on corresponding axes of a two-dimensional diagram, all starting at the same point. Connecting the respective values on each axis yields a graphic resembling a spider web.
First, all evaluation factors are laid out in greater detail, then the taxonomy is applied to all exemplary schemes discussed in chapter 3.3 and every plot explained briefly. It is then demonstrated how a suitable scheme for the proposed smart home scenario can be selected by using the taxonomy, and some general instructions and recommendations for applying it are given. The chapter ends with a remark on the difficulty of handling pairings in the taxonomy.

4.1. Composition of a quantifying spider chart

The following attributes are quantified in the taxonomy:

Signature Size The size of the signature is measured in the number of group elements needed to match it. (In pairing-based schemes, this refers to elements of $\mathbb{G}_1$.) Often, the signature contains not only EC group elements, but also integers. In that case, its size is often given in relation to the group elements as well, so it is easy to account for the integers as well. In the evaluation of the spider chart some caveats with respect to the relation of theoretical and practical size of the signature are discussed. In the exemplary schemes, the smallest signature has a size of about 2 group elements, and sometimes there is no explicit limit to the size. Therefore the axis shows 1, 2, 3, and ≥ 4 as marks.

Size of re-keying data The re-keying data for a single device contains the part of the public parameters that changes during re-key and its secret key usk. Same as the signature it is measured in the number of group elements to match it. In the exemplary schemes, the smallest unit of re-key data has a size of 1 group element; to match the measurement for the signature size, the axis shows 1, 2, 3, and ≥ 4 as marks.

Memory requirement This axis specifies the size of the fixed input data needed for signing and verification. In EC-based schemes this may for example contain the generator

of the group, the *mpk* and the *usk*. Precomputable values which are independent from the sender are not included in the assessment. The signature, sender-ID, and message are needed as input for verification, but they are not permanently stored in the memory, so they are not considered for measuring the memory requirement. Neither are computation tools like hash functions or pairings. Although they require memory, the respective size depends very much on the implementation and specific details and is therefore hard to measure.
Same as the signature the memory need is measured in the number of group elements needed to match it. In the exemplary schemes, the smallest amount of required memory equals 3, therefore the axis shows 2, 3, 4, and ≥ 5 as marks.

Of all operations shown in 3.6, P (= computing a pairing) and GE (= computing an exponentiation in a group) are by far the most expensive ones. Because not all schemes feature pairing, GE is chosen as a unit to measure the cost of computations. In doing so, 1P is converted to 21GE as estimated by [3]. All other operations are disregarded for the purposes of the spider chart on the assumption that their impact is comparatively very small.

Computing signature and verification The added cost of the most frequent events (signing and verifying) of a group member is used as an indicator of its workload. In the exemplary schemes, the cost ranges from 4 GE to 156 GE. The axis is treated as a continuous scale and labeled with the marks for 10GE, 45GE, 80GE, 115GE, and $\geq$ 150GE.

Minimal number of devices affected by re-key This axis shows the roughly estimated number of devices (including the TTP) that is affected in the event of a re-key. It depends on n, the number of members in the group. Therefore, no concrete numbers are given on this axis, but the number relative to n. Also, the values are not plotted continuously but categorized in the following classes of magnitude:

$\frac{n}{m}$ Only a fraction of the group members is affected by the re-key. As long as no scheme is discovered where this is realistic, this value can also be interpreted to show that only a fraction of the group members have to process a full re-key, while the rest merely needs to update a part of the re-key data.

n All group members are (to some degree) affected by the re-key and need to process information to be up to date. As long as no scheme is known where a partial re-key is possible, this value can be interpreted to show that all members need to process the full re-key.

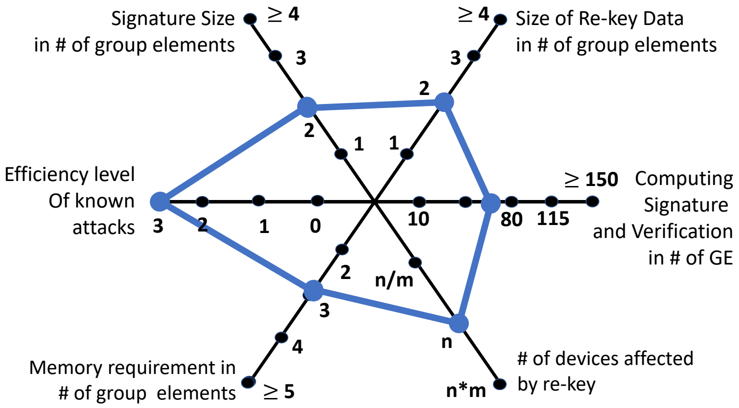
$n * k$ This value slightly deviates from the interpretation of the axis. There is hardly a scenario imaginable where more devices are affected by the re-key than are group members. Therefore, this value is taken to mean that a member needs to process more than one message for re-key. When using threshold IBS, a re-key may mean receiving a message from each of the parts of the TTP; then this value could express the overhead compared to a regular scheme. (If there are threshold schemes where a re-key can be conducted by changing only part of the msk, the value n could be chosen to express that each device has to process only one message.)

Efficiency level of known attacks Apart from procedural weaknesses that may render a scheme insecure, the underlying mathematical problems limit its security, e.g. a signature scheme can at most be as secure as the problem it is based on. IBS schemes are not confined to the realm of elliptic curves. They can be defined on the basis of RSA [44] or lattices [42] as well, and other applications may follow. In the same way, concepts like pairings or hash-functions that are employed in the schemes pose limits to their security. It is therefore useful to rate an IBS scheme with respect to the level of security it provides given a sensible choice of parameters. For this thesis, those levels are defined as follows:

Level 0 There are no known attacks.
Level 1 In theory, an attack is known that could be carried out efficiently by a quantum computer.
Level 2 In theory, an attack is known that may in the medium term future become feasible on classical computers.
Level 3 A practically feasible attack is known.

Remark. Given a set of options where the value on this axis differs, it is important to know that the attack efficiency level is crucial for the size of keys and signatures. The reason for not using RSA with IBS in constrained devices is that there exist relatively efficient methods to factorize integers (on which RSA is based). To securely use it, the corresponding keys need to have at least 1024 bits , and the key size grows superlinearly in proportion to the security level. So for a scheme with value 3 on the attack efficiency axis, not only are probably large parameters needed to begin with, it only gets worse for increasing security. Even if the signature consists of only one group element, it may still be larger then a signature of 3 group elements in a level 2 elliptic curve based scheme. Therefore, the possible impact of this axis on the other factors needs to be taken into consideration when choosing a scheme. Of course the relation is not always directly proportional: A lower efficiency of known attacks does not necessarily imply smaller keys. For example, lattice based cryptography [42] is attracting ever more interest because it is believed to be quantum-secure. The corresponding parameters are however quite large by nature of the underlying problem. So even though the attacks are less efficient than for solving the ECDLP, the keys are still larger. To sum up, the value on the attack efficiency axis can give some insight on the concrete sizes of signature and keys, but not the relative size. Similarly, the axes for plotting signature size could be designed to show absolute key size, but this would not convey much, if any information about the efficiency of attacks on the schemes' underlying mathematics. Therefore, both

Figure 4.1.: Hess02 represented in a spider chart



axes are necessary even though their relation needs to be taken into account for evaluation of the spider charts.

4.2. Representing schemes in the chart

4.2.1. Hess02

Signature Size The signature is composed of one group element $P \in E(\mathbb{F}_q)$ and one integer $i \in \mathbb{Z}_p$. By Hasse's Theorem, $\#E(\mathbb{F}_q) \approx q$ holds and in the scheme, $p \approx q$ is requested [22]. Therefore, i is about the same size as one group element. That means the signature size is approximately 2 group elements.

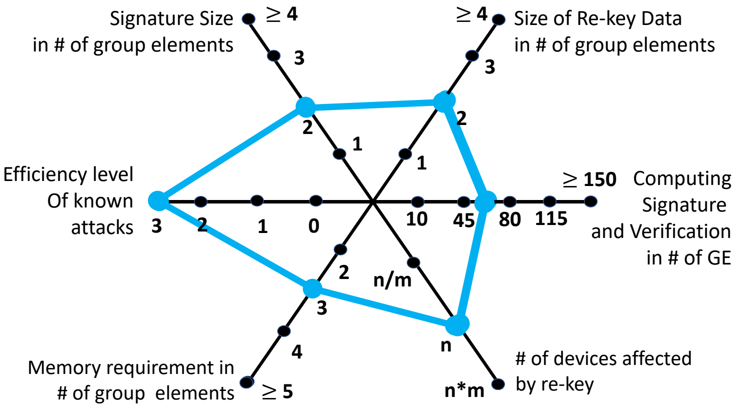
Size of re-keying data For re-keying, the master public key mpk and the user secret key usk, each consisting of 1 group element, are sent to the member. Therefore the re-keying data's size also matches 2 group elements.

Computing signature and verification The computation of the signature requires 3 group exponentiations (GE) and 1 pairing (P). By the conversion rate of $1P \approx 21\text{GE}$, this adds up to 24GE. Verifying the signature takes another 2P and 1GE, i.e. $(2*21+1)$GE $= 45$GE. In total, 69GE are required.

Minimal number of devices affected by re-key All members need to be fully re-keyed when a device is excluded from the group. Therefore the number affected by a re-key is in the magnitude of n.

Memory requirement The values permanently stored on a group member device for the Hess02 scheme consist of the generator P, the mpk, and the usk. All of those values are elements of $E(\mathbb{F}_q)$, so the sum of their sizes comes up to 3 group elements.

Figure 4.2.: BLMQ05 represented in a spider chart



Efficiency level of known attacks In practice, even though pairings are defined on Elliptic Curves, they are only as secure as the DLog problem is hard. If the DLog problem could be efficiently solved, pairing-based schemes as used at the moment could not be relied on anymore. Therefore, pairing-based schemes do not exceed level 3 on this axis.

4.2.2. BLMQ05

Signature Size As in the Hess02 scheme, the signature is composed of one group element $P \in E(\mathbb{F}_q)$ and one integer $i\mathbb{Z}_p$. Therefore, the signature size approximately matches 2 group elements as well.

Size of re-keying data For re-keying, the master public key mpk and the user secret key usk, each consisting of 1 group element, are sent to the member. Therefore, the re-keying data's size is also 2.

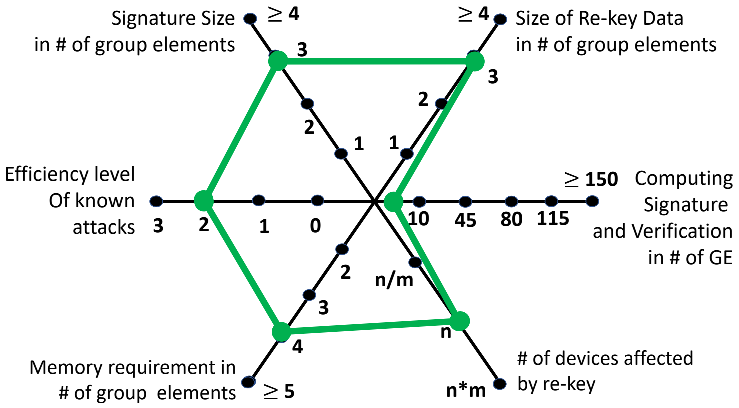
Computing signature and verification The computation of the signature requires 2GE and 1P. Another 2GE and 2P go into the verification. With 1P = 21GE, this adds up to $3 * 21 + 4 = 67$ GE.

Minimal number of devices affected by re-key All members need to be fully re-keyed when a device is excluded from the group. Therefore the number affected by a re-key is in the magnitude of n.

Memory requirement The values permanently stored on a group member device consist of the generator P, the mpk, and the usk. As before, all of those values are elements of $E(\mathbb{F}_q)$, so the sum of their sizes comes up to 3 group elements.

Efficiency level of known attacks In practice, even though pairings are defined on Elliptic Curves, they are only as secure as the DLog problem is hard. If the DLog problem could be efficiently solved, pairing-based schemes as used at the moment could not be

Figure 4.3.: GG09 represented in a spider chart



relied on anymore. Therefore, pairing-based schemes do not exceed level 3 on the scale defined above.

4.2.3. GG09

Signature Size The signature is composed of two group elements $P, Q \in E(\mathbb{F}_q)$ and one integer $i \in \mathbb{Z}_p$. According to the reasoning introduced in the Hess02 representation, this amounts to approximately 3 group elements.

Size of re-keying data For re-keying, the master public key *mpk* (one group element) and the user secret key *usk* (consisting of one group element and 1 integer) are sent to the member. Therefore, the re-keying data's size is also 3 group elements.

Computing signature and verification The computation of the signature requires 2GE, as does the verification. So the total cost are 4GE.

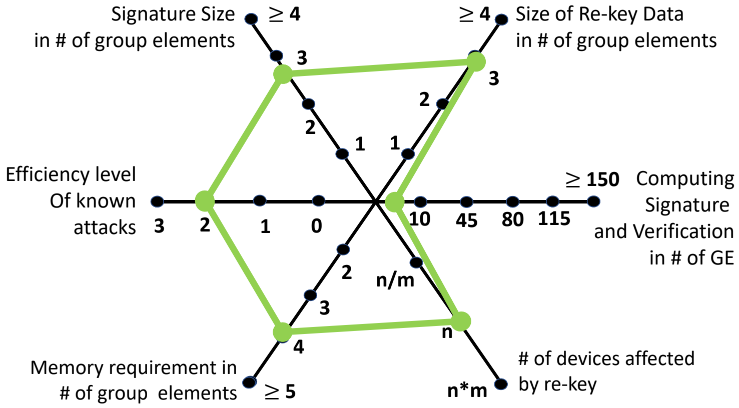
Minimal number of devices affected by re-key All members need to be fully re-keyed when a device is excluded from the group. Therefore the number affected by a re-key is in the magnitude of n.

Memory requirement The values permanently stored on a group member device consist of the generator P, the *mpk*, and the *usk*. P and *mpk* are elements of $E(\mathbb{F}_q)$, and the *usk* contains one group element and one integer, which is about as large as one group element. So the sum of their sizes comes up to 4 group elements.

Efficiency level of known attacks There exists an algorithm[1] that renders the ECDLP easy on quantum computers (which at the moment are still hypothetical constructs). Keeping the curve parameters secret is a possible counter-measure [23], the practicality of this is doubtful however. Therefore the realization of quantum computers threatens

[1] Known as "Shor's algorithm", refer to [45] for details.

Figure 4.4.: vBNN-IBS represented in a spider chart



the security of elliptic curves and the security of schemes based on the ECDLP does not exceed level 2.

4.2.4. vBNN-IBS

Signature Size The signature is composed of two group elements $P, Q \in E(\mathbb{F}_q)$ and one integer $i \in \mathbb{Z}_p$. According to the reasoning introduced in the Hess02 representation, this amounts to approximately 3 group elements.

Size of re-keying data For re-keying, the master public key *mpk* (one group element) and the user secret key *usk* (consisting of one group element and 1 integer) are sent to the member. Therefore, the re-keying data's size is also 3 group elements.

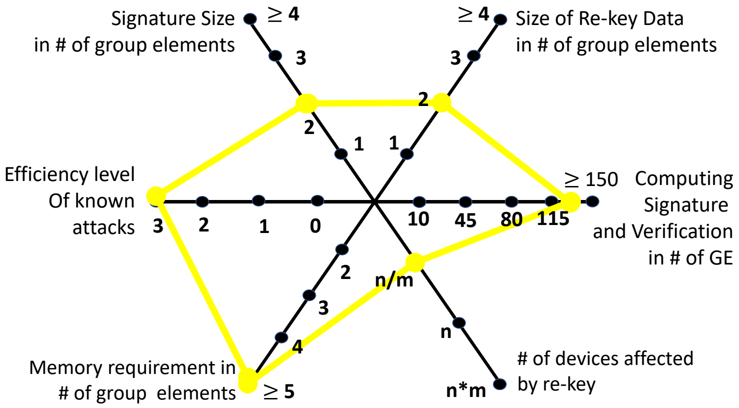
Computing signature and verification The computation of the signature requires 2GE, as does the verification. So the total cost are 4GE.

Minimal number of devices affected by re-key All members need to be fully re-keyed when a device is excluded from the group. Therefore the number affected by a re-key is in the magnitude of n.

Memory requirement The values permanently stored on a group member device consist of the generator P, the *mpk*, and the *usk*. P and *mpk* are elements of $E(\mathbb{F}_q)$, and the *usk* contains one group element and one integer, which is about as large as one group element. So the sum of their sizes comes up to 4 group elements.

Efficiency level of known attacks For the same reasons as in the GG09 scheme representation above, the security of vBNN-IBS matches level 2 on the scale defined in this thesis.

Figure 4.5.: GS02* represented in a spider chart



4.2.5. GS02*

In hierarchical schemes, some parameters depend on the number of levels ℓ. For the sake of comparability to the other schemes, this number is fixed at $\ell = 3$, the smallest number such that the TTP has more than one level. It is assumed that the gorup members are all located at level 3. For the re-key, it is assumed that it is always conducted by a direct parent of the device.

Signature Size In the adjusted version of the scheme, the signature is composed of 2 group elements $(S, Q_k) \in E(\mathbb{F}_q)$ at every level k, which are two group elements.

Size of re-keying data For re-keying, the lower-level master public key mpk (one group element) and the user secret key usk (also one group element) are sent to the member. Therefore, the re-keying data's size is $2k$.

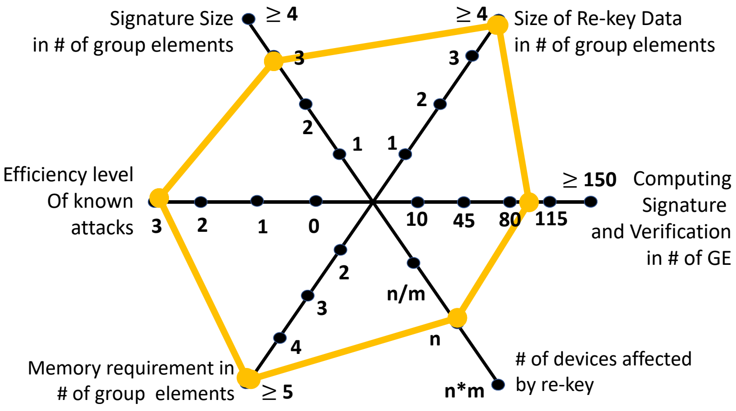
Computing signature and verification The computation of the signature requires $1GE$. The verification adds $(k+3)P = 6P$. So the total cost are $(21*6+1)GE = 127GE$.

Minimal number of devices affected by re-key Only direct children of a re-keying parent need to process a full re-key. For other members it suffices to update the nodes public key. Therefore the number affected by a full re-key is in the magnitude of $\frac{n}{m}$.

Memory requirement The values permanently stored on a group member device consist of the public parameters and the usk. The public parameters are the generator P and the mpk of the root and all inner nodes. They are all group elements. The usk on every level is comprised of one element in GS02* (instead of ℓ in GS02). In a binary hierarchy of three levels, there are 2 inner nodes. So the sum of the parameter sizes comes up to 6 group elements, which in the taxonomy is depicted with the value ≥ 5.

Efficiency level of known attacks For the same reasons as in the Hess02 scheme representation above, the efficiency of known attacks on this scheme matches level 3 on the scale defined in this thesis.

Figure 4.6.: ALYW06 represented in a spider chart



4.2.6. ALYW06

Signature Size The signature size is constant with two group elements and one integer, so adds up to ≈ 3 group elements, irrespective of the number of levels.

Size of rekeying data In case of a re-key to exclude a device from the group, a large part of the public parameters and the whole *usk* need to be replaced. The public parameters amount to $2\ell = 6$ group elements and the *usk* consists of 3 group elements and one integer. The approximate size of the re-key data is therefore 7 group elements.

Computing signature and verification For $\ell = 3$, the comptuational effort for the signatures comes up to 9 GE. For the verification, 4 pairings (=84 GE) and $6GE$ need to be executed, therefore the total cost equals $99GE$.

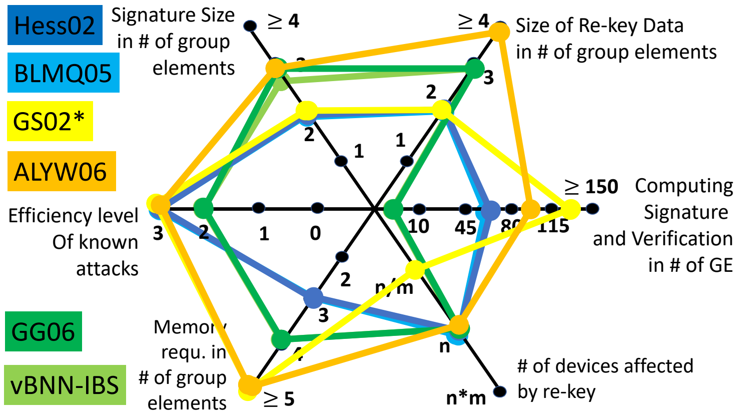
Minimal number of devices affected by re-key All members need to be fully re-keyed when a device is excluded from the group. Therefore the number affected by a re-key is in the magnitude of n.

Memory requirement The values permanently stored on a group member device consist of the public parameters and the *usk*. The public parameters are the generator P, the mpk, ℓ values P_i and $\ell - 1$ values U_i, all of which are group elements. The *usk* on level ℓ is comprised of three group elements and one integer [2]. So the sum of their sizes comes up to 11 group elements, which in the taxonomy is depicted with the value ≥ 5.

Efficiency level of known attacks For the same reasons as in the Hess02 scheme representation above, the efficiency of known attacks on this scheme matches level 3 on the scale defined in this thesis.

[2] Note that the *usk* therefore has a constant size independent of the number of levels for the units at the leaves.

Figure 4.7.: All exemplary schemes represented in a single spider chart



4.3. A suitable IBS scheme

All axes are oriented towards the center, therefore an optimal scheme would form the smallest possible ring in the spider chart. Of course a scheme like that is hardly realistic. Usually, a trade-off between security, parameter size, and computational efficiency is inevitable. In that case, the axes need to be weighted according to the specific scenario for which an IBS scheme is desired. For example, in a highly dynamic group, the number of devices affected by a re-key and the total size of the renewed data might be deemed more significant than the size of the signature and cost of computing it.

In the proposed smart home scenario, it is plausible that many devices in the group rely on a battery for power supply, either because they need to be flexibly located (e.g. a thermostat) or to be independent of the home's power supply, as may be vital for fire alarms [21]. Therefore energy consumption is a crucial criterion in the selection of a scheme. With a low-power radio (less than 10 mW), which has a higher networking efficiency than can be plausibly assumed for the devices in the smart home group, sending one bit costs about 100-1000 nJ according to [49], whereas one instruction on an energy-efficient MSP430 microprocessor consumes an estimated 0.5 nJ [48]. Guggemos [21] concludes that the transfer of information is 10-100 times more expensive than computing it. Therefore, computation is clearly favored over networking in this setting, and in the spider chart, the "signature size" axis is given a higher weight for the rating of the schemes. The re-key data size may also be considered important for the same reason. However, since the scenario suggests that the group is only moderately dynamic, the size of the signatures is ultimately more important. Only the non-pairing-based schemes considered in this thesis exceed level 3. Whether or not this axis is decisive for choosing a scheme depends on the security level required fot the scneario. For the sake of discussing the other axes more, the efficiency of known attacks will not play a role in choosing the scheme. With those considerations in mind, BLMQ05 and Hess02 remain as the best candidates: They are equally well or better suited than either hierarchical scheme apart from re-keying related axes, but as the group is not very dynamic, re-keying is not important enough to make a difference. They also outperform the two non-pairing based centralized schemes except for computational cost, which is also considered less important than signature size. With respect to memory requirements, Hess02 and BLMQ05 are at least as well suited as the other schemes, too. Therefore, those two schemes remain. They differ only in the computational cost for signing and verifying, where BLMQ05 is less intensive by order of 2GE. However, in BLMQ05 there is room for improvement given a device that is not heavily constrained with respect to memory:

If the value $e(P, P)$ in BLMQ05 can be precomputed and stored, it is superior to Hess02 with respect to the amount of group exponentiations (25 instead of 67, as two pairing computations (one in sign, one in verify)) are saved. However, there is no clear-cut answer whether all devices in a smart home have a sufficiently powerful chip to store an additional group element[3]. If this is not the case, then BLMQ05 may not feasible even on a 80-bit security level, and Hess02 is the superior choice by the process of elimination.

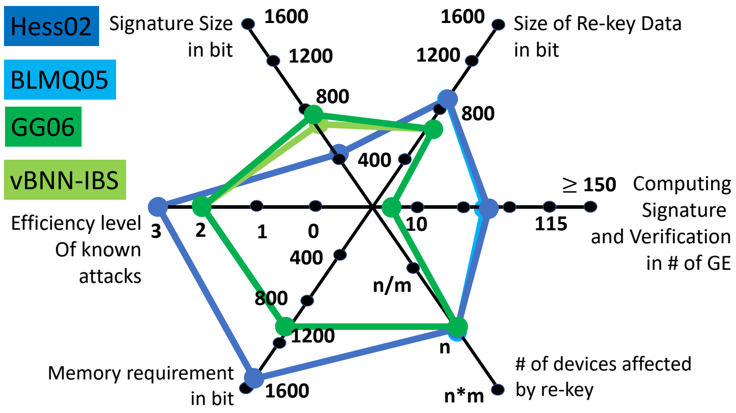
Even though identity-based signatures are still a relatively new field of research, quite a large number of different schemes have emerged since the renewed interest in identity-base cryptography. The taxonomy developed in this thesis is therefore a useful tool to get a quick and practical overview of the available schemes and selecting one for a given setting.
It is not unlikely, however, that some schemes are so similar in terms of the taxonomy that a small set of remaining candidates needs to be examined in more detail. This might concern trade-offs between the axes where slight adjustments in the schemes are possible (e.g. by precomputing values that would normally increase the computational cost of signing). The taxonomy may also be ill-suited to differentiate between optimized versions of a scheme. GG09 and vBNN-IBS for example are so close, that the respective spider chart plots are identical. As mentioned before, a closer look reveals that in vBNN-IBS signatures one group element is replaced by an integer of roughly the same size. Although that saves a few bit in signature size, the taxonomy will not illustrate the difference. Other factors to consider include additional features of the scheme (e.g. whether they offer threshold IBS) or whether older, more established schemes might be preferred to new ones that have yet to be tested in practice despite a less favorable spider chart.

4.4. Handling pairings in the taxonomy

Unfortunately for the taxonomy presented in this thesis, there are many variables involved when it comes to pairings. Symmetric pairings are hardly used anymore for security as well as efficiency reasons. Asymmetric pairings on ordinary curves are thus predominantly used today, i.e. bilinear maps of the form $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$ with a suitably chosen elliptic curve group of size q and embedding degree k. Then the size G_2 of $\mathbb{G}_2$ can be estimated to be around $\frac{k}{2}q \geq G_2 \geq \frac{k}{6}q$. So-called (quadratic/quartic/sextic) twists are used to reduce it to this range.
The difficulty for the taxonomy arises from the following: Some values are required to be elements of $\mathbb{G}_2$, e.g. the generator and the master public key in Hess02 (see chapter 5 in [22]). The size of $\mathbb{G}_2$ depends very much on the pairing and associated curve that are used when applying the scheme. Incorporating the size of the curve, the embedding degree, and the reduction factor of a twist in an abstract manner would hardly yield a comprehensible and user-friendly taxonomy. Hence, to account for this variability, there are three possibilities:

[3] Even more so as it is an element of $\mathbb{G}_3$ in the pairing, realistically by far the largest group.

Figure 4.8.: Comparison of the centralized schemes assuming the use of MNT159



- Display the schemes under the ideal and unrealistic assumption that an efficient pairing with $\mathbb{G}_1 = \mathbb{G}_2 = \mathbb{G}_3$ exists. In the case that the schemes are only mildly affected by the issues discussed above, this retains the higher level of abstractness while not significantly distorting the result. Otherwise, it renders the taxonomy quite a bit harder to read and necessitates detailed evaluation.

- Creating additional axes to measure the effects of pairing-based schemes. Of course, that approach probably means over-complicating the taxonomy solely to accommodate a fraction of the possible schemes.

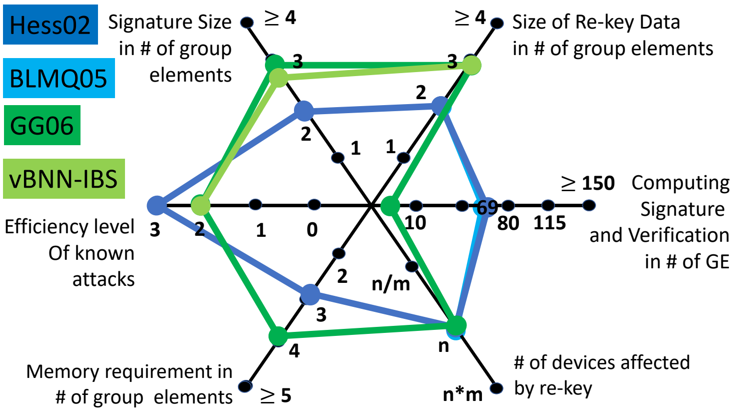
- Resort to instantiating the curves and the pairing with suitable parameters. That way, more realistic values of employing a scheme are obtained. The only drawback is a loss of abstraction that might be desired depending on the nature of the comparison.

To present the taxonomy in an abstract fashion that depends on as few concrete values as possible, the figures in the this chapter show all schemes for an ideal pairing where $\mathbb{G}_1 = \mathbb{G}_2 = \mathbb{G}_3$. This does not disturb the result very much, as only the size of the re-key data and the memory requirements are affected. Usually, the third solution is likely to be more prudent in practice.

Take, for example, MNT224 from a class of curves called MNT-curves, which was discovered Miyaji, Nakabayashi and Takano in their work [34]. They define an asymmetric pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$. The base field size is 224 bit. The curves embedding degree is $k = 6$. Therefore, $|\mathbb{G}_1| = 224$, $|\mathbb{G}_2| \approx 224 \text{ bit} * \frac{6}{2} = 672$ bit, and $|\mathbb{G}_3| \approx 224 \text{ bit} * 6 = 1344$ bit. Integers have size 216 bit.

In both pairing-based schemes, the *mpk* and the generator P are elements of $\mathbb{G}_2$, while *usk* and one part of the signature are elements of $\mathbb{G}_1$. The second part of both signatures is an integer.
All group elements used in GG09 and vBNN-IBS can, of course, be taken from $\mathbb{G}_1$ as they

Figure 4.9.: Comparison of the centralized schemes assuming an ideal pairing where $\mathbb{G}_1 = \mathbb{G}_2 = \mathbb{G}_3$



require no pairing. [4] Then the following measurements emerge[5]

Scheme	Hess02	BLMQ05	GG09	vBNN-IBS
memory requirement in bit	440	440	704	676
signature size in bit	1588	1588	948	948
size of re-key data in bit	896	896	704	704

The comparison of figure 4.8 and the more abstract spider chart (see figure 4.9) shows that the representation of schemes assuming ideal pairings is of limited value. For example, while it appears in the abstract scheme that GG09 and vBNN-IBS have higher demands in terms of available memory and re-key data than the pairing-based schemes, in reality those values are smaller than for Hess02 and BLMQ05.

However, the ratio of the signature sizes is remarkably even more in favor of the pairing-based schemes. Since the signature size is often the decisive factor for the use in constrained environments, the evaluation on the concrete chart may still result in a pairing-based scheme being favored. This is not by chance, either: as signature size is so important, the pairing based schemes are designed to be competitive in this aspect. Therefore, the abstract scheme is quite useful despite its restrictions.

[4] In practice, MNT244 would probably not be chosen for non-pairing based schemes because it is optimized for pairings. Therefore it may not be as efficient as a more freely chosen curve.

[5] Please note that the computational effort of computing the signature are not concretized in this example for simplicity, even though it is affected by the difficulties posed by pairings,ecause b larger parameters also imply more costly operations.

5. Implementation

To prove that identity-based signatures are suitable for source authentication in a constrained network, the implementation of on of the schemes in a test bed representing a group of devices and a Trusted Third Party is presented here. First, the structure and technical details of the test bed are described. In the second section, the libraries used for the implementation and the different curves for which the scheme was tested are discussed. The measurements are discussed in the subsequent section, along with the formal results regarding the requirements formulated in the Introduction.

5.1. Test bed

The test bed consists of four constrained devices, one of which fills the role of the Trusted Third Party, with the other three representing members of a group communication in all phases of membership. The setting has a centralized structure, so no hierarchy exists.

Four small computers produced by the Raspberry Pi Foundation are used, specifically three Raspberry Pi Model B Full Production Boards [38] to represent the group members and one Raspberry Pi 3 Model B [36] which is less constrained, thereby suited to be used as the Trusted Third Party. Please refer to table 5.1 for the basic technical data. The computers were equipped with the Raspbian Operating System (specifically "Raspbian Stretch with desktop" for the TTP and "Raspbian Stretch Lite" for the group members) [39].
The TTP was connected to the local network via WLAN, the group members via LAN using a switch. The local IP address was used as the unique identifier ID, as this information is contained in the IP-Header of all messages.

5.2. Cryptographic library

Charm [1] is a Python and C-based cryptography framework. Intended for the rapid prototyping of cryptosystems, it provides, among other things, a crypto library (e.g. for hash functions) and supports elliptic curves as well as pairings. In general, mathematical operations which are performance critical are implemented in native C modules, whereas the higher-level parts of the cryptosystems can be written in Python. Especially for pairing-based cryptography, it relies on PBC [29], another library, for implementation, which also provides some documentation and more insights on pairings. Apart from the means for prototyping, implementations of some well-known schemes are provided to serve as examples and for further use. Hess02 is one of them, so the implementation is based on this existing code. It had to be slightly adjusted to match the original paper.

Although the Charm framework is frequently recommended for cryptographic prototyping and seems to be rather established, using it can be a bit difficult at times. For example, a custom type is defined for elements of pairing-friendly curves, but no comprehensive guide

Table 5.1.: Technical specifications of the Raspberry Pi devices

Model	Raspberry PI Model B+	Raspberry PI 3 Model B
CPU-Speed	700 MHz single-core	1.2 GHz quad-core
Memory	512 MB (shared with GPU)	1 GB (shared with GPU)
On-board network	10/100 Mbit/s Ethernet	10/100 Mbit/s Ethernet 802.11n wireless Bluetooth 4.1
Recommended PSU current capacity	1.2A	2.5A

for using it nor a basic description could be found. Even tracing its origin was more difficult than expected.

5.3. Evaluation

5.3.1. Quantitative evaluation

The quantitative evaluation is restricted to the size of the data transmitted for authentication. For devices where the processing power is limited, testing the computational cost of signing and verifying would also provide valuable insights. However, this aspect was given a low priority (and eventually omitted) for the following reasons: First of all, throughout the thesis, of all characteristics of the IBS schemes the main focus was on signature size. Since networking is very expensive and signatures are plausibly communicated with high frequency, this factor is believed to be decisive in the application of IBS to constrained networks like a smart home. Second of all, the results depends heavily on the specific implementation, both of the mathematical tools (like the elliptic curve operations) and the scheme itself. Lastly, quite a bit of literature is available that is concerned with the efficient implementation of IBS schemes in constrained devices and discusses the resulting cost of computation and energy requirements. The parameter sizes on the other hand are rarely ever mentioned independently (if at all), let alone compared between different curves, even though they are not trivial (especially in the context of pairings). Thus, for insight on computational cost and implementation details, the following sources are recommended: "Software Implementation of Pairing-Based Cryptography on Sensor Networks Using the MSP430 Microcontroller" by C. GouvÃªa and J. Lopez [20] and "Securing Communications in the Internet of Things using ID-based Cryptography and Modern Elliptic Curves" [32]

The implementation is tested with three different curves, referred to by the denominator in Charm:

SS512 SS512 defines a supersingular curve with a symmetric pairing $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_3$. The base field size is 512 bit. Its embedding degree is k =2. Therefore, $|\mathbb{G}_1| = 512\text{bit}$[1] and $|\mathbb{G}_3| \approx 512 \text{ bit}*k = 1024$ bit.

MNT159 MNT159 belongs to the MNT-curves mentioned in section 4.4. They define an asymmetric pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_3$. The base field size is 159 bit. Its embedding degree is $k = 6$. Therefore, $|\mathbb{G}_1| = 159\text{bit}$, $|\mathbb{G}_2| \approx 159 \text{ bit}*\frac{k}{2} = 477$ bit, and $|\mathbb{G}_3| \approx 159 \text{ bit}*k = 954$ bit.

MNT224 MNT224 is also a MNT-curve, so the corresponding pairing is asymmetric, with a 224 bit base field and embedding degree 6. Therefore, $|\mathbb{G}_1| = 224$ bit, $|\mathbb{G}_2| \approx 224 \text{ bit}*\frac{6}{2} = 672$ bit, and $|\mathbb{G}_3| \approx 224 \text{ bit}*6 = 1344$ bit.

The signature is composed of one element of $\mathbb{G}_1$ and an integer, which by [22] is requested to have approximately $|\mathbb{G}_1|$ bit. The size of the generator P, the *msk*, the *usk* are also considered. Of those values, P is only sent to the device at setup, while *mpk* and *usk* are needed for every re-key. P and *mpk* are elements of $\mathbb{G}_2$, the *usk* of $\mathbb{G}_1$.
The recorded values are shown in the following table:

Table 5.2.: Implementation: Sizes of transferred data

pairing curve	P	*mpk*	*usk*	signature
SS512	512 bit	512 bit	512 bit	664 bit (= 152+512 bit)
MNT159	480 bit	480 bit	160 bit	312 bit (= 152+160 bit)
MNT224	672 bit	672 bit	224 bit	440 bit (= 216+224 bit)

Round-up errors of up to 8 bit can be attributed to the fact that the bit size was computed from the byte-size of the transferred values as no more precise function for measuring was found. With that in mind, all recorded data matches the expected size.

5.3.2. Evaluation with respect to formal requirements

For the initialization of TTP and clients, it is assumed that the curve information, the pairing information, the prime field, and the necessary hash functions are already agreed on and known on both sides. The EC group generator and *msk* are therefore the only parameters sent to new members upon joining the group.

In the test bed, at first the following chain of events was successfully carried out:

[1] This is a colloquial standard of expressing that $|\mathbb{G}_1|$ is in the order of magnitude of 512 bit

> The TTP sets up a group by choosing and generating all necessary public parameters. Upon receiving an application message from a client, it computes the user secret key *usk* corresponding to the sender's IP address, which in the test bed equals the sender's identity and which was not part of the message, but included in the IP header. All three members are admitted to the group in this manner. Each device then sends a signed message to both other members without specifying their identity or a public key.

All six signatures could be verified. Thus, the basic identity-based signature scheme was set up correctly and evidently worked on the constrained devices. The events specific to group communication will be explained in more detail now.

Excluding a device from the group

A special concern of applying IBS in a group environment is the re-keying process. What makes IBS interesting is the possibility to mathematically bind the ID to the public key through a master secret. As explained before, this mathematical connection can not be invalidated. So revoking the public key of a device D is the same as re-keying all other members of the group. The master secret changes and by not receiving a corresponding private key, D's key pair is rendered useless and the device is effectively excluded from the group.

Therefore, to revoke the membership and keys of a device, the Trusted Third Party needs to **update the member record** and **re-key the remaining members**, i.e. choose a new *msk*, computing the *mpk* and new *usk*'s and informing the remaining members about the changes.

To test that the two steps were successfully carried out, the following tests were carried out after excluding one of the three group members:

1. The ex-member signed a message and sent it to either remaining group member. The verification correctly failed, as the signature was invalid.

2. The ex-member was sent a signed message by one of the remaining group members, but could not prove its authenticity.

3. The remaining group members sent each other signed messages which were correctly verified to be authentic on both sides.

So the ex-member was successfully removed from the group without corrupting the communication for the remaining members. Therefore, all but one of the demands for a IBS scheme in a group setting, formulated in the introduction, were met:

✓ *"Allow for a sender's public key to be derived from some individually identifying information that was included anyway."* The sender's public key was, when necessary, derived from the meta-information about the communication, namely the IP header of the UDP packet.

✓ *"Prove that the sender is a member of the group."* As only group members can hold a valid key to sign, successful verification of the signature implies the sender's authority to send the message.

(✓) *"Allow for the group communication to rely on symmetric keys for confidentiality."* The implementation lacks a test to prove this point. However, there is no reason to doubt that a more extended implementation could easily achieve it.

✓ *"At the same time, employ asymmetric keys to individually authenticate the sender of a message."* Hess02 is based on asymmetric keys for signing, as are all identity-based schemes, so this holds.

✓ *"Prevent outsiders (including former members) from creating new valid signatures and verifying signatures created after their membership expiration."* It was successfully tested that former members can neither sign nor verify after a re-key, which implies the same for all outsiders.

✓ *"Prevent outsiders (including former members) from credibly claiming to be a group member."* As only members of the group can sign messages and no other proof of membership exists, the inability to sign equals the inability to claim membership. Therefore, after a re-key no former members can credibly do so, and neither can outsiders.

6. Conclusion and outlook

6.1. IBS in constrained devices

Identity-based signatures work well in constrained networks, because they avoid the overhead that comes from managing the loose relationship between the alleged sender of a message and the public key that is supposed to prove its authenticity. Even though a device needs to communicate more with a Trusted Third Party, signatures and their verification are easier to handle in identity-based schemes than for example with a certificate-based approach. Using IBS in group communication also provides a simple yet effective mechanism to prove the membership of a device as it is firmly tied to its ability to credibly communicate. In very large and dynamic groups, the fact that revoking one device's keys implies a re-key for all remaining members limits the applicability of the scheme. Even the hierarchical schemes examined in this thesis can not reduce the impact this has on every device significantly (if at all).

A significant drawback of the approach is the handling of the keys by a third party. Because of this, identity-based signatures are not non-repudiable (although arguably that is also true for certificate-based authentication). Threshold IBS schemes allow the master secret to be split such that no entity posseses all information to compute secret keys on its own. They can therefore be used to reduce the necessary trust in the TTP. How much this increases the overhead for re-key is not studied in this thesis but would necessarily be part of an analysis of threshold IBS in constrained networks.

The basic characteristics of IBS schemes can be illustrated by the taxonomy developed in this thesis. Even in its abstract form it is a practical tool for the evaluation and comparison of identity-based schemes. It also accommodates hierarchical IBS schemes and is specifically designed to give insight about a schemes suitability for constrained networks. It focuses on the characteristics that have the greatest impact on clients with limited resources, namely computational power, memory requirements and transmission cost.

The successful implementation of one of the exemplary schemes in a small group of three devices not only confirms that identity-based authentication is possible and requires no separate public keys to be communicated. It also proves that re-keying all other members is an effective measure to exclude a device from the group without disturbing the group communication. As the implications of using IBS in a group have hardly (if ever) been addressed before, this is an important result despite its simplicity.

6.2. Outlook and future work

This thesis can be used as groundwork for future endeavours in a number of directions. First of all, many assumptions were made with respect to group and identity management, secure data transmission, and secure key storage that have technically not yet been proven to work in such an environment. Even tough it is not unreasonable to believe that they are justified,

testing the setup in a much more concrete setting would hopefully remove all doubt.
Also conducting more research, for example on the impact of using multicast communication could increase the practical value quite a bit, as would a closer look at the possible incorporation of other mathematical concepts in the taxonomy, for example lattices [42]. Furthermore, it may be impossible to find a hierarchical based IBS scheme that allows a re-key which does not affect every group member. After all, a common denominator in the computation of the user secrets is the key to conducting identity-based authentication. The related literature on HIBS does not address this question however, so adjusting the existing schemes to make them more network-friendly would be worthwhile. Proving or disproving the existence of a HIBS scheme with strictly partial re-key could help in the search for an optimal scheme.
Lastly, although the proposed taxonomy is already very useful to compare, evaluate and classify identity based signature schemes for cryptographic purposes, many improvements are possible. The computational effort to compute and verify signatures could, for example, be divided into separate axes, because verification is very likely to happen significantly more often than signing. In addition, some adjustments might be needed to incorporate threshold schemes. Maybe it would even be worthwhile to extend the taxonomy to non-IBS schemes.

A. Notation

Table A.1.: Abbreviations and Notation used in this chapter and beyond

msk and mpk	The master secret key and master public key, respectively, i.e. the TTP's key pair used to compute all members' secret information.
usk	A member's secret key
ID	A member's identity
$\oplus$ and $*$	The elliptic curve group operation and the respective multiplication (with additive notation for EC groups)
$+$ and $\cdot$	Addition and multiplication of integers, respectively
h_i	A hash function $h_i : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$ that maps a bit string to a prime field. $h_i(x, y)$ means the hash of the concatenated bitstring representations of x and y.
H_i	A hash function $H_i : \{0,1\}^* \rightarrow \mathbb{G}_j$ that maps a bit string to a (elliptic curve) group. $H_i(x, y)$ means the hash of the concatenated bitstring representations of x and y.
P	The generator of the underlying elliptic curve group $\mathbb{G}$
$\mathbb{G}$	The underlying elliptic curve group
$e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_3$	A symmetric pairing Operations in $\mathbb{G}_3$ are denoted multiplicatively, because in practice pairings usually map into finite fields
M	The message that is signed and sent to another device
$g \leftarrow \mathbb{A}$	Drawing a random element from a set $\mathbb{A}$ and assigning to it the name g
case sensitivity	When no fixed meaning is given here, capital letters in general denote elliptic curve group elements and lower-level letters are used for integers.

B. Quantifying overview of process details of all schemes in comparative tables

B.1. Setup

Table B.1.: Setup phase in the centralized IBS schemes

Paper	Hess02	BLMQ05	GG09	vBNN-IBS
generate	1 group elem.	1 group elem.	1 group elem.	1 group elem.
compute	1 RI 1 GE	1 RI 1 GE	1 RI 1 GE	1 RI 1 GE
send	2 group elem.	3 group elem.	2 group elem.	2 group elem. 1 int

Table B.2.: Setup phase (both root and lower level

Paper	GS02	ALYW06 (with ℓ levels)
generate	1 group elem.	1 group elem.
compute	**root:** 1 RI 1 GE **lower level:** 1 RI	**root:** 1 RI l GE l RG **lower level:** **no setup**
send	2 group elem.	$2l+1$ group elem.

B.2. Extract

Table B.3.: Phase *Extract* in the centralized IBS schemes

Paper	Hess02	BLMQ05	GG09	vBNN-IBS
know	1 int	1 group elem.	1 group elem.	1 group elem.
		1 int	1 int	
compute	1 H	1 H	1 H	1 H
	1 GE	1 GE	1 GE	1 GE
		1 IA	1 IA	1 IA
			1 IM	1 IM
			1 RI	1 RI
		1 mod	1 mod	1 mod
		1 modInv		
send	1 group elem.	1 group elem.	1 group elem.	1 group elem.
			1 int	1 int

Table B.4.: Extract phase at root or level $k-1$ for a level k instance

Paper		GS02	ALYW06 (with ℓ levels)
root TTP ($k = 0$)	know	1 identity tupel $k+1$ group el. 1 int	2 group elem. 1 int
	compute	2 GE 1 GO 1 H	2 GE 1 GI 1 GO 1 IA 1 ID 1 RI 1 modInv 1 H
	send	t group elem.	1 group elem. 1 int
level $k-1$ TTP	know	(= Extract)	1 group 1 prime field 1 ID tuple $l-t+2$ group elem.
	compute		$2\ell+2$ GE ℓ GI $2\ell+2$ GO ℓ H 1 RI
	send		$\ell-t+1$ group elem. 1 int

B.3. Re-key

Table B.5.: Rekeying for $n \in \mathbb{N}$ group members

Paper	Hess02	BLMQ05	GG09	vBNN-IBS
know	1 group elem.	1 group elem.	1 group elem.	1 group elem.
compute	1 RI $n+1$ GE n H	1 RI $n+1$ GE n H n IA n modInv	$n+1$ RI $n+1$ GE n H n IA n IM n mod	$n+1$ RI $n+1$ GE n H n IA n IM n mod
send	n packets of: 2 group elem.	n packets of: 2 group elem.	n packets of: 2 group elem. 1 int	n packets of: 2 group elem. 1 int

Table B.6.: Rekeying at level $k-1$ for b k-level instances

Paper	GS02	ALYW06 (with ℓ levels)
compute	1 RI b+1 GE bH **unless $k=0$:** b GO	**once at root:** ℓ GE 1 RI **b times at root:** 2 GE 1 GI 1 GO 1 IA 1 ID 1 modInv 1 H **lower level:** $2\ell+2$ GE $2\ell+2$ GO ℓ GI ℓ H 1 RI
send	b packets of: if $k-1=0$: 1 group elem. if $k>0$: 2 group elem.	b packets of: **to level 2:** $\ell+1$ group el. 1 int **to level $k>2$** $2\ell-k+1$ group el. 1 int

B.4. Sign

Table B.7.: Signing phase in the centralized IBS schemes

Paper	Hess02	BLMQ05	GG09	vBNN-IBS
know	2 group elem.	2 group elem.	1 group elem.	2 group elem.
			1 int	1 int
			1 ID	1 ID
compute	1 RI	1 RI	1 RI	1 RI
	1 H	1 H	1 H	1 H
	3 GE	2 GE	1 GE	1 GE
	1 GO			
	1 RG	1 IA	1 IA	1 IA
	1 P		1 IM	1 IM
			1 mod	1 mod
send	1 group elem.	1 group elem.	2 group elem.	1 group elem.
	1 int	1 int	1 int	2 int

Table B.8.: Signing phase at level k for hierarchical IBS schemes

Paper	GS02	ALYW06 (with ℓ levels)
know	2 group elem.	$2 + l + k$ group elem. 1 int
compute	1 H 1 GO 1 GE	$2k + 4$ GE $2k + 4$ GO $k + 1$ GI $k + 2$ H 1 RI
send	$k + 1$ group elem.	2 group elem. 1 int

B.5. Verify

Table B.9.: Verification Phase in the centralized IBS schemes

Paper	Hess02	BLMQ05	GG09	vBNN-IBS
know	1 group	1 group	1 group	1 group
	1 prime field	1 prime field		1 prime field
	2 hash fct	2 hash fct	2 hash fct	2 hash fct
	1 pairing	1 pairing	1 int	
	3 group elem.	4 group elem.	4 group elem.	3 group elem.
	1 int	1 int		1 int
compute	2 P	1 P		
	1 GE	2 GE	3 GE	3 GE
	1 GO	2 GO	2 GO	2 GO
	1 GI			1 GI
	2 H	2 H	2 H	2 H
		1 IA		

Table B.10.: Verification Phase in hierarchical IBS schemes of k-level signatures

Paper	GS02	ALYW06 (with ℓ levels)
know	$k + 2$ group elem.	$3 + 2k$ group elem.
compute	k H $k + 3$ P (partly reusable) k GO	4 P $2k + 1$ GE $2k + 1$ GO $k + 1$ GI $k + 2$ H

List of Figures

Bibliography

[1] Joseph A. Akinyele, Christina Garman, Ian Miers, Matthew W. Pagano, Michael Rushanan, Matthew Green, and Aviel D. Rubin. Charm: A framework for rapidly prototyping cryptosystems. *Journal of Cryptographic Engineering*, 3(2):111–128, 2013.

[2] Paulo S. L. M. Barreto, Craig Costello, Rafael Misoczki, Michael Naehrig, Geovandro C. C. F. Pereira, and Gustavo Zanon. Subgroup Security in Pairing-Based Cryptography. In Kristin Lauter and Francisco Rodríguez-Henríquez, editors, *Progress in Cryptology – LATINCRYPT 2015: 4th International Conference on Cryptology and Information Security in Latin America, Guadalajara, Mexico, August 23-26, 2015, Proceedings*, pages 245–265. Springer International Publishing, Cham, 2015. `https://doi.org/10.1007/978-3-319-22174-8_14`.

[3] Paulo S. L. M. Barreto, Benoît Libert, Noel McCullagh, and Jean-Jacques Quisquater. Efficient and Provably-Secure Identity-Based Signatures and Signcryption from Bilinear Maps. In David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Dough Tygar, Moshe Y. Vardi, Gerhard Weikum, and Bimal Roy, editors, *Advances in Cryptology - ASIACRYPT 2005*, volume 3788 of *Lecture Notes in Computer Science*, pages 515–532. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[4] Paulo SLM Barreto and Michael Naehrig. Pairing-friendly elliptic curves of prime order. In *International Workshop on Selected Areas in Cryptography*, pages 319–331, 2005.

[5] Ian F. Blake, G. Seroussi, and Nigel P. Smart. *Advances in elliptic curve cryptography*, volume 317 of *London Mathematical Society lecture note series.* Cambridge University Press, Cambridge, 2005. `http://www.cambridge.org/BLDSS`.

[6] Dan Boneh and Matt Franklin. Identity-Based Encryption from the Weil Pairing. In Joe Kilian, editor, *Advances in cryptology, CRYPTO 2001*, volume 2139 of *Lecture notes in computer science, 0302-9743*, pages 213–229. Springer, Berlin and London, 2001.

[7] Xuefei Cao, Weidong Kou, Lanjun Dang, and Bin Zhao. IMBAS: Identity-based multi-user broadcast authentication in wireless sensor networks. *Computer Communications*, 31(4):659–667, 2008.

[8] Clifford Cocks. An Identity Based Encryption Scheme Based on Quadratic Residues. In Gerhard Goos, Juris Hartmanis, Jan van Leeuwen, and Bahram Honary, editors, *Cryptography and Coding*, volume 2260 of *Lecture Notes in Computer Science*, pages 360–363. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001.

[9] Henri Cohen, Gerhard Frey, Roberto Avanzi, Christophe Doche, Tanja Lange, Kim Nguyen, and Frederik Vercauteren. *Handbook of elliptic and hyperelliptic curve cryptography.* CRC press, 2005.

[10] Louis Columbus. Roundup Of Internet Of Things Forecasts And Market Estimates. `https://www.forbes.com/sites/louiscolumbus/2016/11/27/roundup-of-internet-of-things-forecasts-and-market-estimates-2016/#7ccb3fdc292d`.

[11] Hongmei Deng, Anindo Mukherjee, and Dharma P. Agrawal. Threshold and identity-based key management and authentication for wireless ad hoc networks. In *Information Technology: Coding and Computing, 2004. Proceedings. ITCC 2004. International Conference on*, volume 1, pages 107–111, 2004.

[12] Dr Jennifer Balakrishnan. E is for Elliptic Curves, 2016. `https://www.maths.ox.ac.uk/about-us/life-oxford-mathematics/oxford-mathematics-alphabet/e-elliptic-curves`.

[13] Federal Office for Information Security. Kryptographische Verfahren: Empfehlungen und Schlüssellängen: Version 2017-01: Technical Guide TR-02102-1, February 2017. `https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR02102/BSI-TR-02102.pdf?__blob=publicationFile&v=4`.

[14] Federal Office for Information Security. Elliptic Curve Cryptography: Version 2.0: Technical Guide TR-03111, June 2012. `https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TR03111/BSI-TR-03111_pdf.html`.

[15] David Freeman, Michael Scott, and Edlyn Teske. A Taxonomy of Pairing-Friendly Elliptic Curves. *Journal of Cryptology*, 23(2):224–280, 2010. `https://doi.org/10.1007/s00145-009-9048-z`.

[16] William Fulton and Richard Weiss. *Algebraic curves: An introduction to algebraic geometry*. Advanced book classics. Addison-Wesley Pub. Co. Advanced Book Program, Redwood City Calif., 1989.

[17] Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16):3113–3121, 2008.

[18] David Galindo and Flavio D. Garcia. A Schnorr-Like Lightweight Identity-Based Signature Scheme. In Bart Preneel, editor, *Progress in Cryptology – AFRICACRYPT 2009: Second International Conference on Cryptology in Africa, Gammarth, Tunisia, June 21-25, 2009. Proceedings*, pages 135–148. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. `https://doi.org/10.1007/978-3-642-02384-2_9`.

[19] Craig Gentry and Alice Silverberg. Hierarchical ID-Based Cryptography. In Yuliang Zheng, editor, *Advances in cryptology - ASIACRYPT 2002*, volume 2501 of *Lecture notes in computer science, 0302-9743*, pages 548–566. Springer, Berlin and London, 2002.

[20] Conrado Porto Lopes Gouvêa and Julio López. Software Implementation of Pairing-Based Cryptography on Sensor Networks Using the MSP430 Microcontroller. In David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern,

John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Doug Tygar, Moshe Y. Vardi, Gerhard Weikum, Bimal Roy, and Nicolas Sendrier, editors, *Progress in Cryptology - INDOCRYPT 2009*, volume 5922 of *Lecture Notes in Computer Science*, pages 248–262. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[21] Tobias Guggemos. *Diet-ESP: Applying IP-Layer Security in Constrained Environments.* Master thesis, Ludwig–Maximilians–Universität München, August 2014. http://mnm-team.org/pub/Diplomarbeiten/gugg14/.

[22] Florian Hess. Efficient Identity Based Signature Schemes Based on Pairings. In Kaisa Nyberg and Howard Heys, editors, *Selected Areas in Cryptography: 9th Annual International Workshop, SAC 2002 St. John's, Newfoundland, Canada, August 15–16, 2002 Revised Papers*, pages 310–324. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003. https://doi.org/10.1007/3-540-36492-7_20.

[23] Florian Hess, Andreas Stein, Sandra Stein, and Manfred Lochter. The Magic of Elliptic Curves and Public-Key Cryptography. *Jahresbericht der Deutschen Mathematiker-Vereinigung*, 114(2):59–88, 2012.

[24] Kiltz, E. and Neven, G. Identity-Based Signatures. In M. Joye and G. Neven, editor, *Identity-based cryptography*, Cryptology and Information Security Series, pages 31–44. IOS Press, 2008.

[25] Thorsten Kleinjung et al. Factorization of a 768-bit RSA modulus (version 1.4).

[26] Neal Koblitz. Elliptic Curve Cryptosystems. *Mathematics of Computation*, 48(177):203, 1987.

[27] Hugo Krawczyk. *Advances in cryptology - CRYPTO '98: 18th Annual International Cryptology Conference, Santa Barbara, California, USA August 23-27, 1998 proceedings / Hugh Krawczyk (Ed.)*, volume 1462 of *Lecture notes in computer science, 0302-9743.* Springer, Berlin and London, 1998.

[28] M. Lochter and J. Merkle. *Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation.* RFC Editor, 2010.

[29] Ben Lynn et al. The pairing-based cryptography library. http://crypto.stanford.edu/pb.

[30] Martijn Maas. *Pairing-Based Cryptography.* PhD thesis, Technische Universiteit Eindhoven, January 2004. https://www.win.tue.nl/~bdeweger/downloads/MT%20Martijn%20Maas.pdf.

[31] Man Ho Au, Joseph K. Liu, Tsz Hon Yuen, Duncan S. Wong. Practical Hierarchical Identity Based Encryption and Signature schemes Without Random Oracles. https://eprint.iacr.org/2006/368.pdf.

[32] Tobias Markmann. *Securing Communications in the Internet of Things using ID-based Cryptography and Modern Elliptic Curves.* Master thesis, Hamburg University of Applied Sciences, August 2015. https://inet.haw-hamburg.de/thesis/completed/tobias-markmann/view.

[33] Victor S. Miller. Use of Elliptic Curves in Cryptography. In Hugh C. Williams, editor, *Advances in Cryptology — CRYPTO '85 Proceedings*, volume 218 of *Lecture Notes in Computer Science*, pages 417–426. Springer Berlin Heidelberg, Berlin, Heidelberg, 1986.

[34] Atsuko Miyaji, Masaki Nakabayashi, and Shunzo Takano. Characterization of Elliptic Curve Traces Under FR-Reduction. In Dongho Won, editor, *Information Security and Cryptology — ICISC 2000: Third International Conference Seoul, Korea, December 8–9, 2000 Proceedings*, pages 90–108. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001. `https://doi.org/10.1007/3-540-45247-8_8`.

[35] Stefan Müller-Stach and Jens Piontkowski. *Elementare und algebraische Zahlentheorie: Ein moderner Zugang zu klassischen Themen*. Studium. Vieweg & Teubner, Wiesbaden, 2., erw. aufl. edition, 2011.

[36] RaspberryPi org. RASPBERRY PI 3 MODEL B. `https://www.raspberrypi.org/products/raspberry-pi-3-model-b/`.

[37] Patrick Beuth and Veronika Völlinger. Der nächste große Angriff aus dem Internet der Dinge. *Zeit Online*, 22. October 2016. `http://www.zeit.de/digital/internet/2016-10/dyn-internetdienstleister-hacker-angriff-twitter-spotify`.

[38] RaspberryPi.org. RASPBERRY PI 1 MODEL B+. `https://www.raspberrypi.org/products/raspberry-pi-1-model-b/`.

[39] RaspberryPi.org. Raspbian Stretch Lite: Minimal image based on Debian Stretch. `https://www.raspberrypi.org/downloads/raspbian/`.

[40] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[41] R. L. Rivest, Adi Shamir, and L. Adleman. *On Digital Signatures and public key cryptosystems*, volume 82 of *Technical Memo*. Laboratory for Computer Science, S.l., 1977.

[42] Markus Rückert. Strongly Unforgeable Signatures and Hierarchical Identity-Based Signatures from Lattices without Random Oracles. In David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Doug Tygar, Moshe Y. Vardi, Gerhard Weikum, and Nicolas Sendrier, editors, *Post-Quantum Cryptography*, volume 6061 of *Lecture Notes in Computer Science*, pages 182–200. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[43] SECG. Recommended Elliptic Curve Domain Parameters: SEC 2, 2000. `http://www.secg.org/SEC2-Ver-1.0.pdf`.

[44] Adi Shamir. Identity-Based Cryptosystems and Signature Schemes. In George Robert Blakley and David Chaum, editors, *Advances in Cryptology*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53. Springer Berlin Heidelberg, Berlin, Heidelberg, 1985.

[45] Peter W. Shor. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM Review*, 41(2):303–332, 1999.

[46] Joseph H. Silverman. *The arithmetic of elliptic curves*, volume 106 of *Graduate texts in mathematics*. Springer, Dordrecht and London, 2nd ed. edition, 2009. `http://www.springer.com/gb/BLDSS`.

[47] Nigel P. Smart. *Algorithms, key size and parameters: Report - 2014*. ENISA, Heraklion, 2013.

[48] Texas Instruments, Incorporated [SLAS256,D]. MSP430F11x Mixed Signal Microcontrollers (Rev. D). `http://www.ti.com/lit/ds/slas256d/slas256d.pdf`.

[49] Texas Instruments, Incorporated [SWRS046,F]. Single-Chip Low Power RF Transceiver for Narrowband Systems (Rev. F), 29.10.2017. `http://www.ti.com/lit/ds/symlink/cc1020.pdf.`

[50] Jin Wang, Daxing Li, Qiang Li, and Bai Xi. Constructing Role-Based Access Control and Delegation Based on Hierarchical IBS. In *2007 IFIP International Conference on Network and Parallel Computing Workshops (NPC 2007)*, pages 112–118. IEEE, 2007.

[51] Lawrence C. Washington. *Elliptic curves: Number theory and cryptography / Lawrence C. Washington*. Discrete mathematics and its applications. Chapman & Hall/CRC, Boca Raton, FL, 2nd ed. edition, 2008.

[52] Nicky Woolf. DDoS attack that disrupted internet was largest of its kind in history, experts say. *The Guardian*, 26. October 2016. `https://www.theguardian.com/technology/2016/oct/26/ddos-attack-dyn-mirai-botnet`.